TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Softwarová podpora komunitní agendy

Bakalářská práce

Studijní program: B0714A270001 Mechatronika
Autor práce: **Samuel Pospíšil**
Vedoucí práce: Ing. Jan Kolaja, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Softwarová podpora komunitní agendy

Jméno a příjmení: **Samuel Pospíšil**
Osobní číslo: M19000095
Studijní program: B0714A270001 Mechatronika
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Proveďte rešerši měřidel (plynu, vody, elektřiny), které umožňují vzdálený odečet.
2. Proveďte rešerši open source softwaru pro správu společenství vlastníků a možných řešení pro generování pdf souborů s rozúčtováním.
3. Vytvořte webovou aplikaci, která bude umožňovat rozúčtování energií dle podružných měřidel v domě.
4. Aplikace nechť umožňuje přehlednou správu jednotlivých společenství vlastníků, kde každé společenství má své členy, resp. bytové jednotky, hlavní měřidla, podružná měřidla, pravidelné neměřené platby.
5. Aplikace nechť umožňuje zadávání hodnot hlavních a podružných měřidel případně napojení na výstupy měřidel, které umožňují vzdálený odečet.
6. Aplikace nechť generuje souhrny a jednotlivá rozúčtování ve formátu html pro e-mail.
7. Aplikaci umístěte na veřejný server a otestujte.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] POKORNÝ, Jaroslav a Michal VALENTA. Databázové systémy: vysokoškolská učebnice. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

[2] PAVLÍČEK, Antonín. Nová média a sociální sítě. Praha: Oeconomica, 2010. ISBN 978-80-245-1742-1.

Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

15. května 2022 Samuel Pospíšil

Softwarová podpora komunitní agendy

Abstrakt

Tato práce se zabývá vytvářením webové aplikace pro zjednodušení správu bytových jednotek. Výsledkem je klientská strana aplikace, která umožňuje přehled informací týkajících se naměřených hodnot energií, vlastníků a bytových jednotek. Aplikace umožňuje tyto data vkládat, zobrazovat v tabulkách, které je možné filtrovat a z příslušných hodnot umožňuje vytvořit rozúčtování, které je možné vytisknout pomocí funkce tisku z webového prohlížeče.

Aplikace je vytvořena v programovacím jazyce Typescript a frameworku ReactJS. Při programování jsou vytvářeny znovu využitelné komponenty tak, aby program bylo možné jednoduše rozšiřovat v případě potřeby a přidat další stránky pro editaci a zobrazení nových dat. UI aplikace je vytvořen v českém i anglickém jazyce a umožňuje jednoduché přidání dalších překladů. Práce se zabývá také komunikací mezi klientem a serverem. Součástí práce je autentizace, které je řešena pomocí JWT tokenu a generování entit ze serveru pro klienta pomocí openAPI specifikace.

Na začátku je rešerše chytrých měřidel, které umožňují vzdálený odpočet a rešerše existujících aplikací pro správu bytových jednotek. Dále práce obsahuje popis jednotlivých technologií využitých ve vývoji aplikace, popis samotného vytváření aplikace a v poslední části je aplikace umístěna pro testování na veřejný server.

Klíčová slova: Společenství vlastníků, Bytové jednotky, React, Typescript, REST API

SW Support of Comunity Agenda

Abstract

This thesis is about development of a web application to simplify the management of housing units. The result is a client part of the application, which enables to view a summary of information about the measured values, owners and housing units. This application allows you to enter the data, display them in tables that can be filtered and from the appropriate values it creates bills that can be printed using the print function from a web browser.

The application is build using Typescript programming language and the ReactJS framework. Reusable components are created during programming so that the program can be easily expanded if necessary and enables to add additional pages for editing and displaying new data. UI of the application is created in both Czech and English language and allows to easily add additional translations. This thesis also deals with communication between client and server. Part of this thesis is authorization, which is solved using a JWT token and generating entities from the server for the client using the openAPI specification.

At the beginning of the thesis, there is a research relating to smart meters that allows remote reading and research for existing applications for housing management. Next the thesis contains a description of individual technologies used in the development of the application, a description of the actual development of the application and in the last part the application is deployed for testing on a public server.

Keywords: community of owners, housing units, React, Typescript, REST API

Poděkování

Děkuji vedoucímu práce Ing. Janu Kolajovi, Ph.D. za zadání tématu této bakalářské práce a panu Ing. Janu Pěničkovi za cenné rady při vytváření webové aplikace.

Obsah

Seznam obrázků 8
Seznam zdrojových kódů 9
Seznam zkratek 9

Úvod **10**

1 Rešerše měřidel **11**
1.1 Meter Bus 12

2 Rešerše softwaru **13**
2.1 Architektura 14
2.2 Klient 14
2.3 Server 15
2.4 Generování pdf 15

3 Použté techhnologie **16**
3.1 Architektura Klient-Server 16
3.2 ReactJS 16
3.2.1 Komponenty 16
3.2.2 Hooky 17
3.2.3 Kontext 18
3.3 Typescript 18
3.3.1 Datové typy 18
3.3.2 Generické datové typy 19

3.4 Komunikace klient - server . . . 19
3.4.1 REST API . . . 19
3.4.2 Generování entit . . . 20
3.4.3 Autorizace - Autentizace . . . 20
3.5 Internactionalizace a lokalizace . . . 20
3.6 Bootstrap . . . 21
3.7 Yarn . . . 21

4 Vývoj aplikace **22**
4.1 Klient . . . 22
4.1.1 Struktura . . . 22
4.1.2 Generování entit . . . 23
4.1.3 Import a export komponent . . . 24
4.1.4 Autentizace . . . 25
4.1.5 Internacionalizace . . . 25
4.1.6 Boostrap . . . 26
4.1.7 Navigace . . . 27
4.1.8 Typy stránek . . . 27
4.1.9 Grafy . . . 31
4.2 Server . . . 32
4.2.1 Kontrolery . . . 32
4.2.2 Generování entit . . . 33
4.3 Testování . . . 33

5 Závěr **35**

Použitá literatura **37**

Příloha I - zdrojové kódy **38**

Seznam obrázků

1.1 Příklad architektury pro chytrá měřidla [2] 11

2.1 Příklad Softwaru pro správu bytů [4] 13
2.2 Popularita frontend frameworků [6] 14

3.1 Příklad struktury reaktové aplikace 17

4.1 Struktura projektu 23
4.2 Zobrazení webové stránky na mobilním zařízení 26
4.3 Stránka pro přidání jednotky 28
4.4 Vygenerované rozúčtování 30
4.5 Stránka pro Přihlašování 31
4.6 Sloupcový graf 32
4.7 Ukázka webové stránky na testovacím serveru 34

Seznam zdrojových kódů

3.1 Ukázka hooku useEffect 17

4.1 Hook pro GET dotaz 23

4.2 Ukázky využití souboru index.ts 24

4.3 Odkaz pro navigaci 27

4.4 Tabulka jednotkových proměnných 29

4.5 Kontroler pro administrátorské akce 32

Seznam zkratek

HTML	Textový značkovací jazyk (Hypertext Markup Language)
CSS	Kaskádové styly (Cascading Style Sheets)
i18n	Internacionalizace
l10n	Lokalizace
JSX	Rozšířená syntaxe JavaScriptu (JavaScript Syntax Extension)
JWT	JSON Web Token
API	Aplikační programovací rozhraní (Application Programming Interface)
UI	Uživatelské rozhraní (User Interface)
GUI	Grafické uživatelské rozhraní (Graphical User Interface)
IoT	Internet věcí (Internet of Things)
P2P	Komunikace se sobě rovnými (peer-to-peer)

Úvod

Vzdálený odečet měřidel energií v bytových jednotkách je dnes běžná záležitost. Po odečtení dat je třeba jejich odeslání na server, uložení do databáze a vhodné zpracování a zobrazení pro správce bytových jednotek. Tato práce se zabývá poslední částí tohoto procesu, a to je přehledné zobrazení dat pro zjednodušení správy bytových jednotek. Cílem této práce je vytvořit webovou aplikaci, která umožňuje přehledné zobrazení a možnost vložení informací o jednotlivých bytových jednotkách a jejich vlastnících. Aplikace má také umožňovat zobrazení hodnot z měřičů s možností dálkového odečtu. Předpokládá se, že data z měřidel s dálkovým odečtem jsou odeslány na server, kde jsou uloženy do databáze a klientská strana aplikace dostává tato data ze serveru. Další součástí této aplikace je generování ročního rozúčtování pro jednotlivé bytové jednotky. Tyto rozúčtování mají být upraveny tak, ať homolu být odeslány přes e-mail.

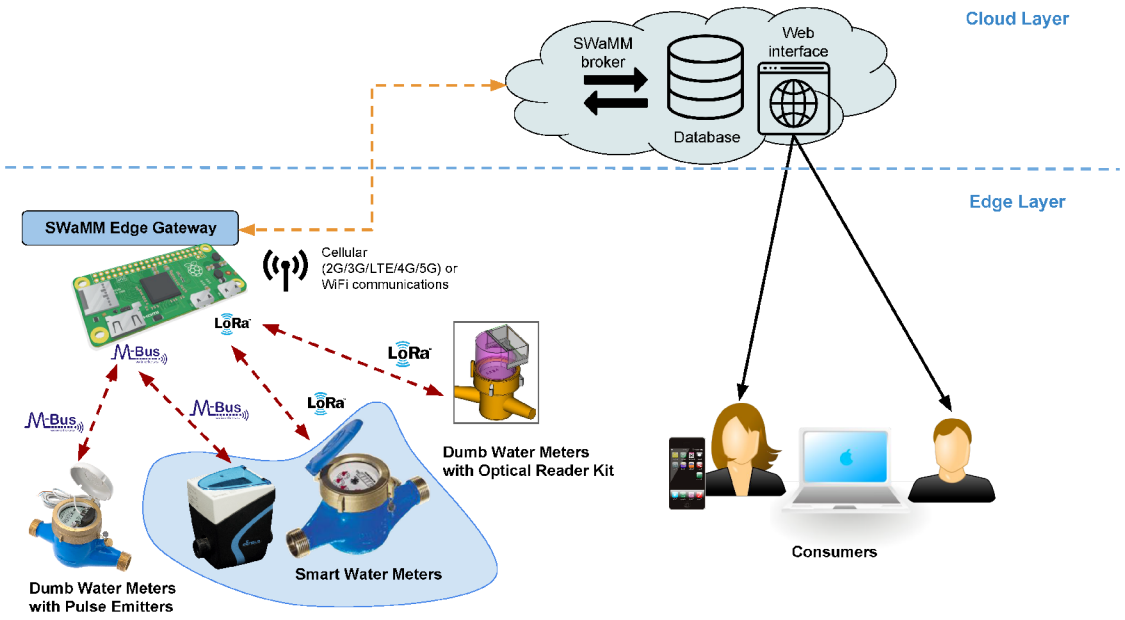
Při vývoji této aplikace je brán ohled na použitelnost pro počítač i mobilní zařízení s možností výběru různých jazyků, ale především na přehledné UI, které umožní všechny základní akce, které jsou nutné pro správu dat bytových jednotek. Kromě samotného UI pro zobrazování dat a umožnění jejich vložení, se práce zabývá komunikací se serverem, který má přístup k databázi. To zahrnuje přihlašování, API žádosti a ošetření odpovědí ze serveru.

Řešení vývoje aplikace je zaměřeno především na možnost budoucího rozšíření pro další stránky umožňující přidávání, zobrazování a editaci dat a rozšíření pro další jazyky, spíše než vytvoření kompletní aplikace.

Před začátkem vývoje byla udělána rešerše pro chytrá měřidla umožňující dálkový odečet a existující open-source aplikace pro správu bytových jednotek. Další část bakalářské práce se zabývá popisem technologií, obvykle knihoven, které jsou použité pro vývoj aplikace. V další části je popsáno vytváření základní šablony, struktury aplikace a jednotlivých stránek pro zobrazování a vkládání dat, přihlašování a případné další stránky pro zjednodušení správy bytových jednotek. Poslední část popisuje vložení aplikace na veřejný server.

1 Rešerše měřidel

Chytrá měřidla jsou měřiče vody, elektřiny, plynu atd., které umožňují připojení k IoT, díky čemu lze odečítat jejich hodnoty vzdáleně. Komunikace mezi chytrými měřidly a serverem, na který se data odesílají může být provedena několika způsoby. Jednou z možností je pomocí architektury P2P, kde měřidlo komunikuje přímo se serverem. Tato architektura vytváří větší zátěž pro infrastrukturu a lze použít jen při menším počtu měřidel. Další možnost architektury pro komunikaci je s prostředníkem, který buď zprostředkovává informace (Gateway) 1.1 nebo ukládá data, která přeposílá dále (Data Concentrator). Komunikace mezi měřidlem a prostředníkem nebo měřidlem a serverem může být uskutečněno několika možnými způsoby. Pro komunikaci přímo s měřidly se používá nejčastěji sběrnice M-Bus nebo Power Line Communication (PLC). [1]



Obrázek 1.1: Příklad architektury pro chytrá měřidla [2]

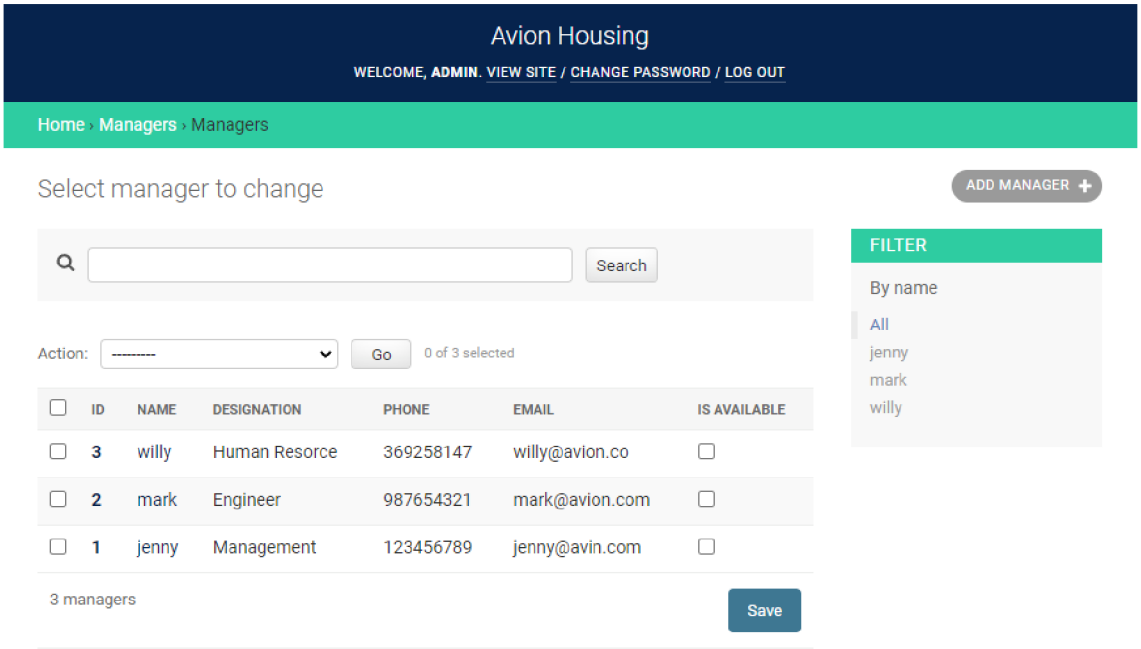
1.1 Meter Bus

Meter Bus nebo M-Bus je evropský standart vytvořený pro dálkový sběr dat z měřičů. Může být také použit pro čtení dat ze senzorů a řízení motorů. Je vytvořen se zaměřením na velký počet připojených zařízení, zabezpečení proti chybám, minimální cenu a minimální spotřebu elektřiny. Jedná se o master-slave komunikaci pomocí dvouvodičového kabelu. Informace do měřičů jsou posílány pomocí změny napětí, které se pohybuje mezi 24 V a 36 V. Měřiče posílají data pomocí změny proudu, který je v rozsahu 1,5 mA a 20 mA. [3]

2 Rešerše softwaru

Open source software byl vyhledán na webové stránce GitHub pomocí klíčových slov ”House Management”, ”Housing System” a ”House Community agenda”. Z vyhledaných projektů byly vybrány ty, které nejvíce odpovídají požadavkům ze zadání této práce.

Mezi funkce, které tyto projekty umožňují patří možnost přihlášení a autorizace, zobrazení vlastníků jednotek, zobrazení informací jednotek samotných a přinejmenším základní rozúčtování. Další data, která tyto softwary umožňují zobrazovat se výrazně liší mezi sebou. Patří mezi ně například rozloha bytů, údaje o zaměstnancích, výpisy jednotlivých výdajů a hlášení problémů v bytech. Všechna tato data jsou zobrazována v tabulkách, které je obvykle možné filtrovat 2.1. Některé aplikace umožňují zobrazení určitých dat v grafech.



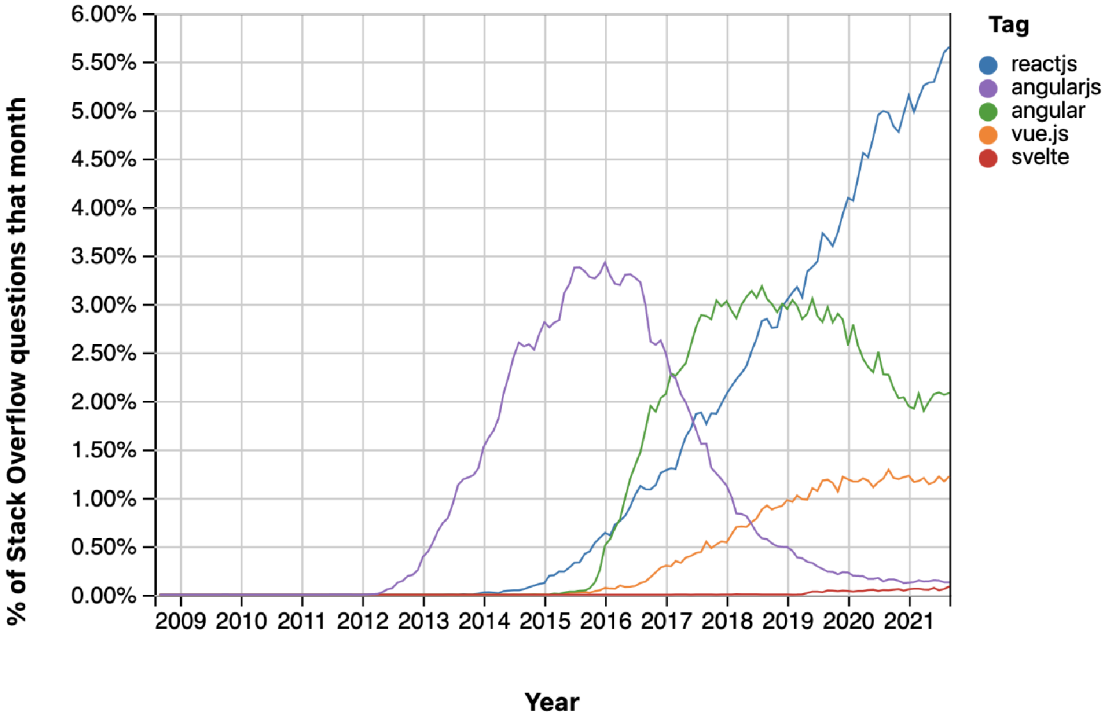
Obrázek 2.1: Příklad Softwaru pro správu bytů [4]

2.1 Architektura

Všechny vyhledané projekty jsou webové aplikace. Většina těchto open-source projektů je založena na architektuře klient-server. Výjimkou je projekt *Housing Management* [5], který je založen na blockchainu za účelem decentralizace. Cílem tohoto projektu je vytvořit transparentnější transakce a snížit tím riziko podvodů a chyb v oblasti správy vlastnictví.

2.2 Klient

Převládajícím programovacím jazykem pro vytvoření klientské části je Javascript a dalšími jsou Typescript a PHP. Protože se jedné o webové stránky, všechny projekty obsahovaly část projektu napsanou v HTML a CSS. Obvykle byl z důvodu zjednodušení vývoje využit nějaký framework. Framework je soubor programů, API a šablon, který pomáhá a zjednodušit vývoj aplikace. Mezi nejpoužívanější frontend frameworky patří React, Angular a Vue. 2.2 Všechny tři jsou pro programovací jazyk Javascript. Angular vyvinul Google, React je od Facebooku a Vue je open-source projekt. Další, méně používané frameworky jsou pro PHP. Nejpoužívanější z nich je Laravel.



Obrázek 2.2: Popularita frontend frameworků [6]

2.3 Server

Pro vytváření backendu pro aplikaci s architekturou klient-server existuje mnoho frameworků pro mnoho programovacích jazyků. Mezi nejpoužívanější patří Django a Flask, které používají jazyk Python, ExpressJS a NestJS pro Javascript, Laravel a CakePHP pro PHP a .NET Core pro C#.

2.4 Generování pdf

Generovat soubory PDF na klientské straně lze několika způsoby. Lze jej vytvářet v Javascript kódu vytvořením dokumentu a vkládáním jednotlivých elementů do tohoto dokumentu. Tento způsob používá knihovna *jsPDF*. Další možností je generování PDF přímo z HTML a CSS souborů. V tomto případě je třeba, aby součástí použitého nástroje pro generování PDF byly všechny použité vlastnosti z CSS a HTML. V opačném případě bude mít vygenerovaná stránky jiný vzhled než originální. Tento způsob využívá knihovna *html2canvas*, která generuje obrázek z původní stránky a pro generování souboru PDF lze použít knihovnu *html2pdf*, která využívá knihovny *html2canvas* a *jsPDF*. Poslední možností je generování PDF pomocí prohlížeče. Uživatel může dát webovou stránku k tisku, kde lze dále uložit do souboru PDF. V případě potřeby využití generování PDF pomocí prohlížeče v kódu, lze využít knihovnu *Puppeteer*, která poskytuje API pro ovládání různých funkcí chromu a chromia, včetně vygenerování stránky v PDF.

3 Použté techhnologie

3.1 Architektura Klient-Server

Tato architektura je použita, když se očekává využití služby několika uživateli zároveň. Jeden server může zpracovávat požadavky od několika klientů najednou. Za server považujeme jeden nebo více počítačů zaměřených na jistý účel a klientem je pracovní stanice, obvykle s GUI, která umožňuje vytváření požadavků pro server a zpracovává přijaté výsledky. Nejčastějším příkladem klienta je webový prohlížeč. Server obvykle zpracovává přístup k databázi, aktualizuje, odstraňuje, přidává a vyhledává data. Může také data dopočítávat a zpracovávat je před odeslání klientovi nebo před uložením do databáze. Obvykle je zde zařízena autorizace a autentizace. [7]

3.2 ReactJS

React je Javascriptová knihovna pro tvorbu uživatelského rozhraní. Jedná se o open-source knihovnu od Facebooku, která zjednodušuje vývoj a údržbu webových aplikací.

3.2.1 Komponenty

Webová aplikace vyvinutá za pomoci knihovny React se skládá z jednotlivých komponent. Celá aplikace je reprezentovaná stromem komponentů 3.1. Tyto komponenty představují Javascriptové třídy nebo funkce, kde vstupem jsou vlastnosti vložené z nadřazené komponenty a výstupem je JSX kód, který umožňuje vytvářet HTML elementy v Javascriptu nebo kód, který generuje HTML. Komponenty vytvořené Javascriptovou třídou mohou mít uložený stav proměnných. Funkční komponenty tuto funkcionalitu nahrazují takzvanými hooky.

Obrázek 3.1: Příklad struktury reaktové aplikace

3.2.2 Hooky

Hooky jsou funkce, které umožňují využívat některé vlastnosti implementované v knihovně React. Mezi nejpoužívanější patří useState, který ukládá vnitřní stav pro funkční komponenty a useEffekt, který spouští funkce při vytvoření komponenty a při změně jejich proměnných. Hook useEffekt má jako argument funkci, která je spuštěna při vytvoření a aktualizování komponenty. Pro zamezení příliš častého spuštění lze jako druhý argument vložit list hodnot a funkce bude spuštěna jen při změně nějaké hodnoty z listu. Jestliže funkce vložená do useEffektu má jako návratovou hodnotu další funkci, pak bude zavolána při odstranění komponenty 3.1.

Funkce hooky mohou být použity jen v komponentách vytvořené jako funkce. Komponenty vytvořené ze tříd mají metody, které nahrazují funkce hooků, např. useEffect je nahrazen metodami componentDidMount, componentDidUpdate a componentWillUnmount. Vzhledem k tomu, že funkční komponenty a hooky jsou novější a předpokládá se pro ně větší podpora než pro třídové komponenty, jsou v této práci použity výhradně funkční komponenty.

```
1 useEffect(()=>{
2     // called when updating a component or dependency
3     return function cleanUp(){
4         // called when the component is deleted
5     }
6 }, [
7     // dependencies
8 ])
```

```
9 export default useGet
```

Zdrojový kód 3.1: Ukázka hooku useEffect

3.2.3 Kontext

Obvyklý způsob předávání dat v reaktové aplikaci je z vyšších komponent do nižších přes argumenty (props). V případě potřeby předávání informací, které se týkají více komponentů ve velké části projektu, jako například UI styl nebo lokalizace, lze využít reaktový kontext. Kontext se vytvoří funkcí React.createContext. Pro použití kontextu se vloží do projektu komponenta Context.Provider jako předek části projektu, kde mají být dostupná data z kontextu. Přístup k datům je možné provést několika způsoby. Využitím hooku useContext, komponentou Context.Provider nebo v případě tříd, definování statického kontextového typu.

3.3 Typescript

Jedná se o nadstavbu Javascriptu přidanou o syntaxi pro určení datových typů. Javascript je skriptovací jazyk vytvořen pro umožnění interaktivity webových stránek. Javascript je interpretovaný jazyk a potřebuje ke spuštění Javascript engine, který je součástí webových prohlížečů, ale lze jej mít i mimo webový prohlížeč.[8] Typescript je kompilován do Javascriptu. Výhodou Typescriptu je, že kontroluje správné datové typy proměnných. Díky tomu lze předcházet mnoha chybám při psaní skriptu ještě před spuštěním programu.

3.3.1 Datové typy

Proměnné mohou být deklarovány pomocí tří klíčových slov - const, var a let. Pomocí const se vytvářejí konstantní hodnoty, které nemohou být změněny, klíčovým slovem var a let se vytvářejí hodnoty, které mohou být změněny. Var je použito k vytváření globálních proměnných a let k vytvoření lokálních proměnných.

Základní datové typy v Typescriptu jsou textový řetězec (string), boolean, který může nabývat hodnot pravda a nepravda a číslo (number). Zde jsou vyjmenovány další datové typy, které mohou být v Typescriptu použity.

- základní datové typy
  - textový řetězec - string
  - boolean
  - číslo - number

- funkce - jsou definované datovými typy vstupů a výstupem funkce
- objekty - množina dvojic klíčů (textový řetězec) a hodnot (jakýkoli datový typ)
- množinový typ - union
- pole - array
- uspořádané n-tice - tuples
- Dynamický typ - any
- prázdné typy
  - prázdné - null
  - nedefinované - undefined
  - nic - void
  - hodnota, která nenastane - never

3.3.2 Generické datové typy

V případě entit nebo funkcí, které jsou totožné až na použitý datový typ, lze tyto entity sloučit pomocí takzvaných generických datových typů. Tyto datová typy umožňují vytvořit generické funkce, tzn. funkce které jsou vytvořeny pomocí generickým typem, obvykle značeným T. Tento datový typ je při volání funkce nahrazen datovým typem dle potřeby. generické datové typy mohou být omezeny pomocí klíčového slova extends. Tímto způsobem se zaručí, že datový typ musí osahovat určité parametry.

3.4 Komunikace klient - server

3.4.1 REST API

Architektura REST API umožňuje komunikaci mezi serverem a klientem pomocí čtyř základních metod – CRUD (Create, Retrieve, Update, Delete). Jedná se o vytvoření dat metodou PUT, získání dat pomocí metody GET, k odstranění dat se používá metoda DELETE a metodou POST se aktualizují již vytvořená data.

3.4.2 Generování entit

Mezi klientem a serverem se posílají data o určitém schématu (datovém typu). Ke komunikaci je vhodné, aby obě strany dopředu věděly, jaký datový typ je třeba poslat a jaký datový typ očekávat jako odpověď. V případě využití programovacích jazyků, které mají pevně definované datové typy je třeba tyto typy dopředu vytvořit a je nutné, aby byly shodné na klientské i serverové straně. Pro zjednodušení této synchronizace je možné automaticky generovat entity a přenášet je. K tomuto účelu slouží OpenAPI specifikace, která standardizuje JSON objekt obsahující schémata entit. Na základě této specifikace mohou být generovány JSON nebo YAML soubory, které jsou využity pro přenesení definice datových typů.

3.4.3 Autorizace - Autentizace

V případě poskytování dat ze serveru, které nejsou veřejné je potřeba ověřit, zda uživatel má oprávnění tato data zobrazit. První krok v tomto procesu je identifikace a ověření uživatele. Obvykle se k tomuto používá uživatelské jméno a heslo. Tomuto procesu se říká autentizace. Autorizace je proces ověření, zda daný uživatel má přístup k datům.

JSON Web Token

Jeden z možných řešení autentizace je pomocí JSON Web Tokenu (JWT). Tento token se skládá z hlavičky, dat a podpisu. Hlavička a data jsou JSON objekty. Hlavička obsahuje informaci o typu tokenu a použitém algoritmu k zakódování. Data obsahují samotné předávané informace, obvykle o uživateli. Podpis je vytvořen zakódováním hlavičky, dat a soukromého klíče. Tento podpis slouží k ověření, že s tokenem nebylo manipulováno a že byl vytvořen někým, kdo zná soukromý klíč. Token je při komunikaci přes HTTP vkládán do autorizační hlavičky v podobě „Bearer {token}“ [9].

3.5 Internactionalizace a lokalizace

V případě vytváření softwaru, který má být použit lidmi z více zemí, případně kultur nebo regionů, je třeba program podle tohoto kritéria upravit. Tento proces se nazývá internacionalizace (i18n). Cílem i18n je vytvořit software, který není pro jednu specifickou zemi, ale je možné jej jednoduše upravit pro lidi z různých regionů. Proces, vytváření z i18n softwaru specifický pro konkrétní region se nazývá lokalizace (l10n). Nejobvyklejší aspekt pro i18n a l10n je změna jazyku, může se ale také jednat o změnu zápisu formátu data nebo změnu vzhledu a umístění UI elementů.

3.6 Bootstrap

Bootstrap je framework, který zjednodušuje stylizování webových stránek. Je vytvořen se zaměřením na responzivní web a podporou mobilních zařízení. Základní položkou při vytváření stránek pomocí Bootstrap je kontejner. Jeho hlavní využití je při používání mřížkového systému. Kontejner může obsahovat řádky a sloupce, které určují pozici elementů na stránce. Šířka kontejneru je 12 jednotek, které jsou rozděleny mezi jednotlivé sloupce. Pozice sloupců se může dynamicky měnit na základě velikost obrazovky. Díky tomu je možné mít stejný kód pro zobrazení na mobilních zařízeních, tabletech i počítačích. Bootstrap má předdefinované třídy, které se vkládají do HTML elementů a tím určují jejich vzhled a pozici.

3.7 Yarn

Yarn je software, který se stará o správu balíčků. Balíčky jsou spravovány pomocí souboru *package.json*, který obsahuje základní informace o projektu, jako je název a verze, balíčky, které jsou obsaženy v projektu jejich verze, skripty pro vytvoření, testování a vývoj projektu a další informace o konfiguraci projektu. Yarn umožňuje především jednoduché kopírování projektů a snadnou manipulaci s balíčky. [10]

4 Vývoj aplikace

Cílem této práce je vytvořit webovou aplikaci, která umožňuje zobrazení, správu, vytváření a odebírání dat týkajících se společenství vlastníků bytových jednotek. Mezi tyto data patří naměřené hodnoty z měřidel, informace o platbách, členech společenství a bytových jednotkách. Aplikace by měla přehledně zobrazovat všechna tato data, generovat rozúčtování a umožnit vytvářet tyto informace.

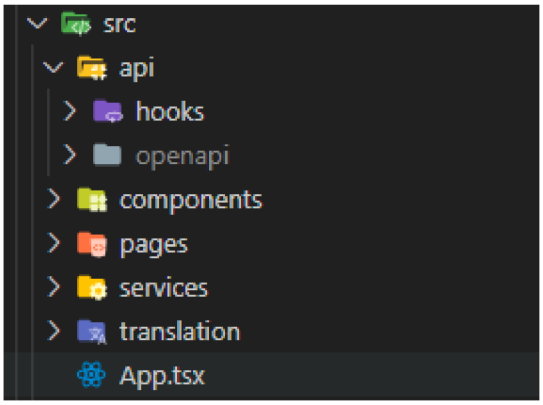
4.1 Klient

Základ klientského kódy byl vytvořen přes React s šablonou pro Typescript. Tím se vytvořil základní projekt s ukázkovými stránkami.

4.1.1 Struktura

Základní projekt je vytvořen v jedné složce se všemi potřebnými souboru. Součástí vytvořené struktury je složka src, ve které je uložen zdrojový kód. Zde jsou vytvořeny tyto podsložky pro přehlednost projektu: api pro automaticky generovaný kód, který je použit ke komunikaci se serverem a hooky použití v aplikaci. Components, kde jsou uloženy React komponenty, které nejsou specifické jen pro jednu stránku, ale mohou být použity na více místech v projektu. Ve složce pages jsou uloženy komponenty konkrétních stránek a komponenty, které jsou použity jen v jedné určité stránce. Složka services je použita pro pomocné funkce a ve složce translation jsou uloženy JSON soubory s textem použitým na stránkách a jejich překladem 4.1.

Struktura samotných komponentů, z kterých je webová stránka tvořena začíná základním komponentem App, který obsahuje navigační lištu a všechny stránky, na které je možno se dostat z navigační stránky, také je zde kontrola, zda je uživatel přihlášen a případně ho přesměruje na přihlašovací stránku. Další struktura je určena obsahem dané stránky.



Obrázek 4.1: Struktura projektu

4.1.2 Generování entit

Pro zjednodušení komunikace se serverem byla použita knihovna OpenAPI Typescript Codegen[11], která generuje z JSON souboru, poskytnutého serverem, datové typy objektů, které se posílají nebo přijímají a funkce pro samotnou komunikaci. Tato knihovna má možnost generovat tyto funkce za pomoci několika HTTP klientů. V tomto projektu byl použit Axios, protože umožňuje zachycovat jednotlivé žádosti a odpovědi. Pro implementaci komunikace v React komponentech jsou vytvořeny hooky, které používají generované funkce a starají se o chyby, zrušení žádostí a uložení odesílaných nebo přijímaných dat. Pro většinu typů žádostí byl vytvořen generický hook, kde vstupem je funkce vygenerovaná funkce z knihovny OpenAPI Typescript Codegen 4.1. Pro některé metody, které potřebují odlišnou strukturu byl vytvořen samostatný hook.

Generování samotných entit probíhá po zavolání příkazu openapi s určením z jakého souboru se mají entity generovat, do jaké složky se mají vygenerovat a případné s dalšími parametry pro upřesnění generování. Pro zjednodušení procesu byly vytvořeny dva yarn skripty, *generate*, který generuje entity na základě testovacího serveru a *generate-local*, který generuje z lokálního serveru.

```
1  import { useEffect, useState } from "react";
2  import { CancelablePromise } from "../openapi";
3  import { ApiState } from "./APITypes";
4  import useApi from "./useApi";
5
6  function useGet<T>(apiCall: ()=>CancelablePromise<T>):[T,
       ApiState] {
7      const [variable, setVariable] = useState<T>({} as T);
8      const [apiState, handleRequest] = useApi()
9      useEffect(()=>{
10         const promise = handleRequest(apiCall())
```

```
11          promise.then((response)=>{
12              setVariable(response ? response : {} as T)
13          })
14          return function cleanup(){
15              promise.cancel()
16          }
17          // eslint-disable-next-line react-hooks/exhaustive-
                deps
18      }, [handleRequest])
19      return [variable, apiState];
20  }
21
22  export default useGet
```

Zdrojový kód 4.1: Hook pro GET dotaz

4.1.3 Import a export komponent

V projektu má obvykle každá komponenta vlastní soubor. Pro vložení jednotlivé komponenty je třeba exportovat tuto komponentu z původního souboru a importovat do dalšího, kde má být využita. Při importování je třeba uvést relativní cestu k souboru, ze kterého je komponenta importována. Aby nedošlo k vytvářením složitých cest a zbytečně velkému počtu příkazů import na začátků souborů, jsou vytvořeny ve většině složek soubory *index.ts*. Tyto soubory zjednodušují tento proces tak, že jsou do nich vloženy všechny komponenty, které mohou být použity mimo danou složku a jsou v tomto souboru exportovány. V případě potřeby použití těchto komponent je pak potřeba určit jen cestu do dané složky a nikoli přímo k souboru. Příkladem je vkládání jednotlivých stránek do komponenty App. Bez použití souborů index by bylo potřeba každou stránku, která je obsažena v projektu importovat zvlášť, ale s využitím indexů je vše importováno ze složky pages 4.2.

```
1  // import a export v souboru index.ts
2  import CreateUnitVariable from "./CreateUnitVariable";
3  import HomePage from "./HomePage";
4  ...
5  import YearTotal from "./YearTotal";
6
7  export {CreateUnitVariable, HomePage, ... YearTotal}
8
9  // import stránek v souboru App.tsx
10 import { CreateUnitVariable, HomePage, ... YearTotal} from
       './pages'
```

Zdrojový kód 4.2: Ukázky využití souboru index.ts

4.1.4 Autentizace

Na klientské straně je třeba pro autentizaci získat JWT token pomocí metody GET Token s platnými přihlašovacími údaji. Tento token je uložen a při každém požadavku je přidán do HTTP hlavičky headers-auth. Token musí zůstat uložen i po obnovení stránky, a proto je uložen do místího úložiště. Token má omezený čas platnosti. Proto je třeba kontrolovat, zda uložený token je pořád platný a jestliže není, přesměrovat uživatele na přihlašovací stránku. Toho je docíleno pomocí HTTP klientu Axios, který dokáže zachytit každou odezvu a zkontrolovat kód stavu odpovědi. Když je kód 401 - Unauthorized, pak dojde k přesměrovaní na stránku pro přihlášení.

O vkládání tokenu do záhlaví požadavků se stará knihovna OpenAPI Typescript Codegen. Ukládání tokenu a kontrola jeho platnosti je implementována v hooku useToken, který je použit v základní komponentě App.

4.1.5 Internacionalizace

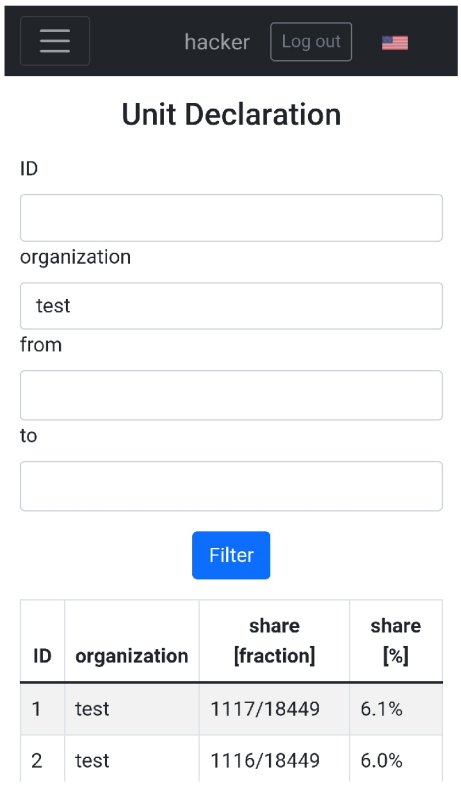
O i18 se stará knihovna *react-intl*, která poskytuje reaktový kontext umožňující v každé části projektu zjistit, který jazyk je aktuálně vybrán a na základě toho vybrat správný překlad. Vložení textu v různých jazycích je umožněno komponentou FormattedMessage. Do této komponenty je vložen id textu, který má být zobrazen a komponenta najde odpovídající text podle id a aktuálně zvoleného jazyku. Jazyky jsou uloženy v JSON souborech ve složce translation, kde každý jazyk má svůj soubor, který obsahuje dvojice id:překlad. Jazykový kontext i komponenty umožňující vložení textu byly upraveny pro Typescript tak, aby datové typy id a jazyku nebyly jen string, ale byly výčet možných hodnot. Tyto data jsou vzata přímo z JSON souborů s překlady. Díky tomu, když se vloží id, které není obsaženo v souborech s překladem, tak kompilátor zobrazí chybu. V případě, když je potřeba vložit text do elementu jako parametr string a není možné vložit celou komponentu, je vytvořen hook useTranslatedMessage, který využívá hooku poskytnutého knihovnou *react-intl* a je upraven pro odpovídající datové typy.

Jazyk se na webové stránce vybírá tlačítkem, který se nachází v pravé části navigační lišty. Jelikož knihovna Bootstrap nemá možnost umístění tohoto tlačítka tak, aby zůstalo na stejném místě i při zobrazení na mobilním zařízení, je pro tuto komponentu vytvořen CSS soubor, který pevně definuje umístění komponenty na obrazovce a její vzhled. Pro jasné zobrazení výběru jazyků je k názvu přidána vlajka. Obrázky vlajek jsou načítány z webové stránky https://flagpedia.net/.

4.1.6 Boostrap

Pro úpravu vzhledu byla použita primárně knihovna react-boostrap, která obsahuje komponenty využívající frameworku Bootstrap. Díky tomu je přehlednější kód, protože není třeba vypisovat HTML elementy a vkládat do nich CSS třídy z knihovny bootstrap, ale stačí importovat příslušné komponenty a pracovat s nimi. V případě, kdy knihovna react-boostrap neobsahuje komponenty, které by dosáhly požadovaného vzhledu nebo funkcionality, byla využita knihovna Bootstrap. V několika případech nestačí ani knihovna Bootstrap a potom jsou vytvořeny soubory CSS, které jsou pojmenovány stejně jako komponenty, v kterých jsou využity a jsou umístěny ve stejné složce.

Výhodou, která přináší využití knihovny Bootstrap, je jednoduché řešení zobrazení pro různé velikosti displeje. Jednotlivé elementy na stránkách jsou součástí kontejneru, který obsahuje řádky a sloupce, ve kterých je samotný obsah stránky. Pomocí tohoto uspořádání lze jednoduše naprogramovat posunutí určitých elementů na další řádek, když se již nevejdou vedle sebe. Navigační lišta byla také vytvořena pro možnost zobrazení na mobilních zařízeních a odkazy na jednotlivé stránky jsou zobrazeny jen po zmáčknutí k tomu určeného tlačítka. 4.2



Obrázek 4.2: Zobrazení webové stránky na mobilním zařízení

4.1.7 Navigace

Pro navigaci mezi jednotlivými stránkami je využita knihovna react-router-dom, která umožňuje přesměrování na danou komponentu podle URL cesty. Určení cest a komponent, které k sobě patří je naprogramováno v souboru App.tsx, kde je také vytvořena podmínka pro přesměrování na přihlašovací stránku v případě, kdy není uložen token pro autentizaci. Navigace mezi stránkami probíhá přes navigační lištu umístěnou v horní části stránek. Tato navigační lišta je stejná pro všechny stránky, kromě přihlašovací. Navigační lišta obsahuje odkazy na stránky, které jsou umístěny vlevo a vpravo je jméno přihlášeného uživatele, tlačítko pro odhlášení a tlačítko pro změnu jazyka. Odkazy na přesměrování jsou vytvořeny komponentou Nav.Link z knihovny react-bootstrap pro správný vzhled, a pro správnou funkcionalitu je jako vlastní element vložena komponenta NavLink z knihovny react-router-dom 4.3.

```
1  import { Nav } from "react-bootstrap"
2  import { NavLink } from "react-router-dom"
3  import { TranslatedMessage, TranslatedMessageKey } from "
       ../l18n"
4
5  function NavItem(props: {text: TranslatedMessageKey, path:
       string}){
6      return (
7          <Nav.Link as={NavLink} exact to={props.path} href={
               props.path} >
8              <TranslatedMessage id={props.text}/>
9          </Nav.Link>
10     )
11 }
12 export default NavItem
```

Zdrojový kód 4.3: Odkaz pro navigaci

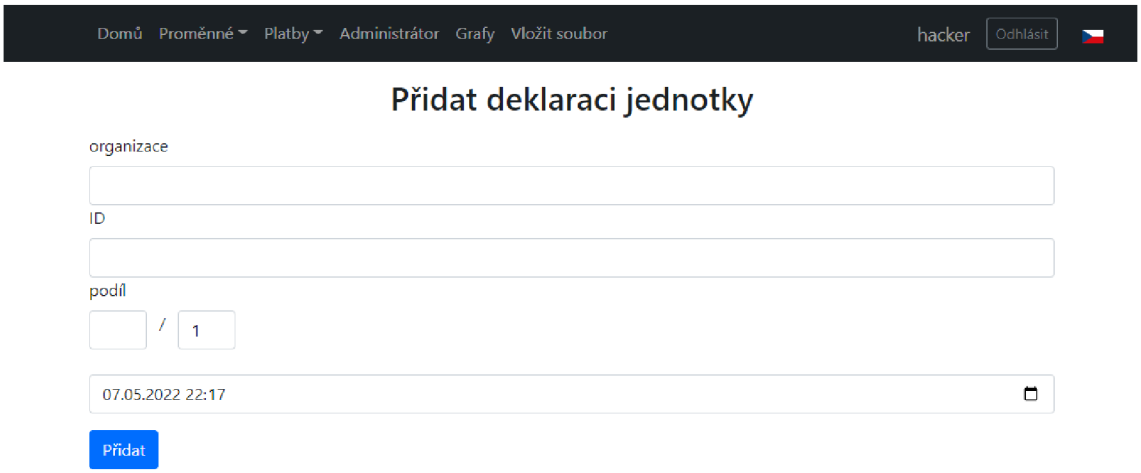
4.1.8 Typy stránek

Vytváření dat

Pro přidání dat byla vytvořena komponenta, která se stará o zobrazení stavu komunikace se serverem a základní vzhled stránky. Při odeslání zobrazuje spinner a po přijetí odpovědi ukáže, zda byla úspěšná a případně zobrazí, jaká chyba nastala. Potomek této komponenty je formulář vytvořený dle potřeby daných datových typů. Formulář se obvykle skládá z komponent ve složce Input. Zde jsou připravené elementy pro vstup textového řetězce, čísla, data, roku, zlomku a výčtového typu 4.3.

Další stránka, pro vkládání dat je pro nahrání souborů. Soubory jsou posílány v datovém typu FormData, který obsahuje pole objektů Blob. Blob je datový typ

obsahující neměnná a nezpracovaná data. Samotné soubory nejsou přímo datového typu Blob, ale File, který je založen na Blobu, ale doplňuje ho o funkce specifické pro soubory. Aby bylo umožněna takzvaná „drag and drop“ funkcionalita, je použita knihovna react-dropzone, která poskytuje hook z kterého máme především atributy pro elementy div a input, z kterých se komponenta Drag and Drop skládá.



Obrázek 4.3: Stránka pro přidání jednotky

Zobrazování dat

Zobrazení dat probíhá převážně v tabulkách. Pro jednoduché zobrazení byla vytvořena generická tabulka 4.4, ale lze ji použít jen pro jednodušší datové typy. Pro komplikovanější datové typy jsou vytvořeny konkrétní tabulky. Stránky se zobrazením dat se obvykle skládají z formuláře pro filtraci zobrazených dat, tabulkou s daty, tlačítek, které umožňují přesunutí na další stránku a tlačítko, které přesměruje na stránku pro vytvoření dané entity.

Pro filtrování byla vytvořena komponenta specifická pro datový typ UnitVariableFilter, která umožňuje vybrat, jaké parametry mají být filtrovány. Některé data jsou filtrovány jinými filtry, ale jestliže všechny jejich parametry jsou obsaženy v datovém typu UnitVariableFilter, pak pro vytvoření filtru lze využít stejnou komponentu.

Tabulka se skládá z hlavičky a jednotlivých řádků. Řádek je vlastní komponenta rozdílná pro každou tabulku. Výjimkou jsou řádky UnitRow, které jsou vytvořeny tak, aby zobrazily hodnoty z vloženého objektu dle vložených klíčů. Řádky jsou

do tabulky vkládány pomocí funkce map, které vezme hodnoty z listu a po jedné je přiřazuje ke komponentě řádku. Ke správnému fungování funkce map v Reaktu je třeba do dané komponenty, která se vytváří, vložit parametr klíč, který je pro každou hodnotu v listu rozdílný. Jako hodnota klíče byl použit index pozice dat v listu. Součástí tabulky je také upozornění, které zobrazuje načítání dat, chybové hlášení a oznámení při nenalezení dat s daným filtrováním.

Další stránkou, která zobrazuje data je výpis rozúčtování. Filtr této stránky určuje, jaká rozúčtování budou zobrazena, ale data obsažená v samotných rozúčtováních jsou pevně daná. Data pro rozúčtování jsou vložena do připravené šablony. Tato stránka je upravena tak, aby při tisku stránky z webového prohlížeče bylo zobrazeno pouze rozúčtování 4.4. Filtr, navigační lišta a tlačítka jsou při tisku schovány. Toho bylo docíleno pomocí CSS třídy *d-print-none*.

```
function UnitTable<Type, Key extends keyof Type>(props: {
    apiState: ApiState,
    variables: Type[], textColumns: TranslatedMessageKey[],
        keyColumns: Key[]}){
    return (
        <Table striped bordered responsive>
            <TableHead
                text={props.textColumns}
            />
            <tbody>
                <TableWarning
                    apiState={props.apiState}
                    isEmpty={props.variables.length===0}
                    mainErrorMessage="error.loadUV"
                />
                {props.variables.map((unit, index) => {
                    return (<UnitRow key={index} unit={unit
                        } columns={props.keyColumns}/>)
                })}
            </tbody>
        </Table>
    )
}
```

Zdrojový kód 4.4: Tabulka jednotkových proměnných

Zobrazení dat s možností úpravy

Některé stránky jsou vytvořeny tak, aby kombinovaly zobrazování a editaci dat, případně vytváření nových dat. Obecně se jedná o tabulky, u kterých je možné přidávat a odebírat jednotlivé řádky a je možné editovat některé hodnoty řádků.

Předpis plateb jednotky č.1 platný od 01.02.2021 **Vlastnický podíl 1117/18449**

Vlastník jednotky	Doručovací adresa z KN	E-mail
Žena Příjmení	Javorová 1324/9, 46014 Liberec, Liberec XIV - Ruprechtice	jmeno.primjmeni@email.cz
Můž Příjmení	Javorová 1324/9, 46014 Liberec, Liberec XIV - Ruprechtice	jmeno.primjmeni@email.cz

Předepsaná měsíční úhrada činní **2900** Kč

na účet č. pod variabilním symbolem

Složka	Koeficient	Základ	Úhrada
Fond oprav	117,7 m2	108	1006
Vodné a stočné	59,433 m3/1675,82 m3	159008	441

Předepsanou platbu je nutno hradit do 15.dne v měsíci ve stanovené výši! Prosím překontrolujte aktuálnost vašeho E-mailu a doručovací adresy! Případné dotazy prosím napiště na e-mail javorova@outlook.cz.

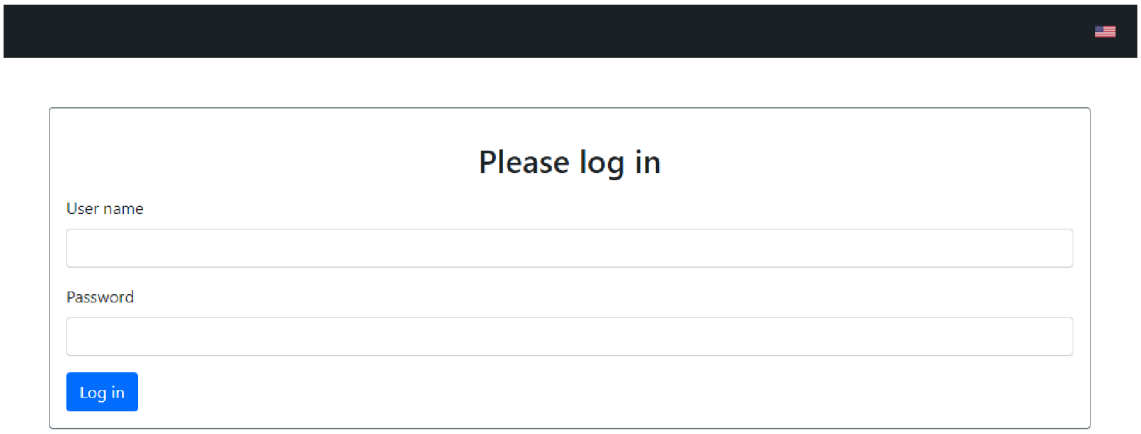
07.05.2022

Obrázek 4.4: Vygenerované rozúčtování

Možnost editace je pomocí input elementu, který je vložen do buněk tabulky, které umožňují editaci. Na stránkách, kde se mají zobrazovat i editovat data, je třeba mít hooky pro POST a GET metody.

Přihlašování

Stránka pro přihlašování má vlastní navigační lištu, která obsahuje jen tlačítko pro změnu jazyka. Obsah stránky se skládá z formuláře se dvěma vstupy pro uživatelské jméno a heslo. Komponenta pro tuto stránku se stará o uložení přihlašovacích údajů, jejich odeslání na server a zpracování odpovědi. Při odeslání přihlašovacích údajů je zobrazen spinner a v případě odpovědi s chybou je tato chyba zobrazena uživateli. 4.5



Obrázek 4.5: Stránka pro Přihlašování

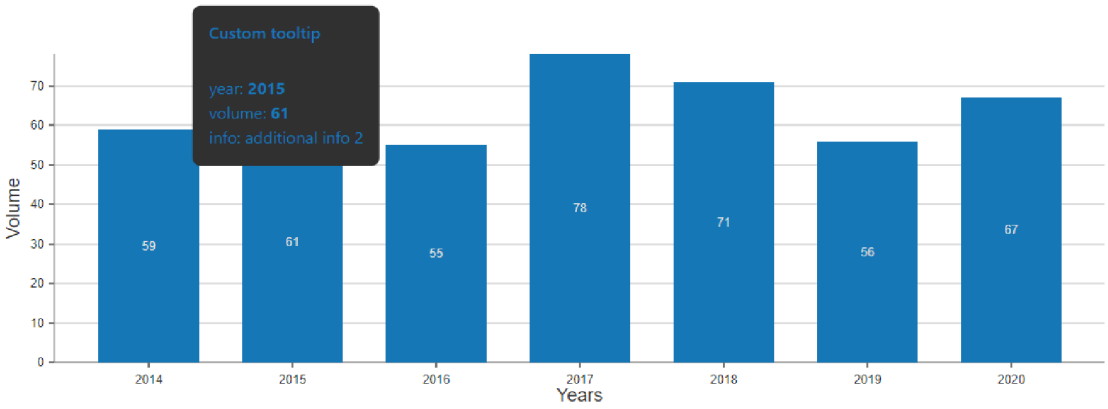
Administrátorské akce

Tato stránka byla vytvořena pro funkce, ke kterým by měl být přístup pouze administrátor daného společenství. Zatím nebyla naprogramována autorizace pro různé druhy přihlášených uživatelů, takže přístup k této stránce má každý přihlášený uživatel. Tato stránka obsahuje jen možnost odstranění všech dat v databázi. Počítá se, že s rozšiřováním aplikace bude třeba zde vložit více funkcí. Akce jsou upořádány v tabulce, která obsahuje popis akce, tlačítko pro vykonání akce a zobrazení, zda byla akce úspěšně provedena.

4.1.9 Grafy

Pro vytvoření grafů byla použita knihovna *Nivo*. Tato knihovna obsahuje velké množství různých typů grafů a její výhodou je, že má možnost nainstalování a použití jen určitých částí knihovny. V projektu jsou vytvořeny tři druhy grafů. Dva jsou spojnicové a jeden z nich je upraven pro zobrazení času na ose x. Třetí typ grafu je sloupcový. Spojnicové grafy jsou udělány se skokovou změnou, pro zobrazení počtu lidí v jednotce a sloupcové grafy jsou pro zobrazení spotřeby v daném časovém období 4.6. Komponenty pro tyto grafy byly vytvořeny pro jednoduché použití s výchozími hodnotami vzhledu, které mohou být upraveny bez změny samotné komponenty vložením stylu vzhledu jako vstupní parametr do komponenty.

V grafech je umožněno použít takzvaný tooltip. To je popis, který se zobrazí při umístění kurzoru nad určitou hodnotu v grafu. Tento popisek může být pro každou hodnotu v grafu jiný a může obsahovat doplňující informace k dané hodnotě.



Obrázek 4.6: Sloupcový graf

4.2 Server

Serverový kód byl vytvořen pomocí platformy .NET framework v jazyce C#. Slouží pro zpracování dat a jejich ukládání, vyhledávání, odstraňování a editaci v databázi. Také se stará o autorizaci a vytváření JWT tokenů. Tato práce je zaměřená především na klientskou část aplikace, proto se zaměříme na API kontrolery pro komunikaci s klientem.

4.2.1 Kontrolery

Kontrolery jsou C# třídy vycházející v tomto případě z třídy ControllerBase. Každý kontroler má svoji cestu, na které může mít vytvořeno několik funkcí pro dotazy. Tyto dotazy jsou součástí REST API a musí být definovány jako PUT, POST, GET nebo DELETE. V těchto funkcích je naprogramována samotná logika, tzn. práce s databází, úprava dat a dopočítávání dat. Práce s daty v backendu aplikace není součástí této bakalářské práce a většina této logiky byla vytvořena konzultantem, panem Ing. Janem Pěničkou. 4.5

```
1  [ApiController]
2  [Route("[controller]")]
3  [Authorize(AuthorizePolicyNames.Api)]
4  [Produces(MediaTypeNames.Application.Json)]
5  public class AdminController : ControllerBase
6  {
7      private readonly ILogger _logger;
8      private readonly StorageService _storageService;
9
10     public AdminController(ILogger<UnitVariableController>
           logger, StorageService storageService)
```

```
11      {
12          _logger = logger;
13          _storageService = storageService;
14      }
15
16      [HttpGet("StorageDropAll")]
17      public async Task StorageDropAll(CancellationToken
            cancellationToken)
18      {
19          _logger.LogWarning("Storage drop all [enter]!");
20          //await _storageService.LoadRepositoryAsync(
                cancellationToken);
21          await _storageService.DropAll(cancellationToken);
22          await _storageService.SaveChangesAsync(
                cancellationToken);
23          _logger.LogWarning("Storage drop all [exit]!");
24      }
25  }
```

Zdrojový kód 4.5: Kontroler pro administrátorské akce

4.2.2 Generování entit

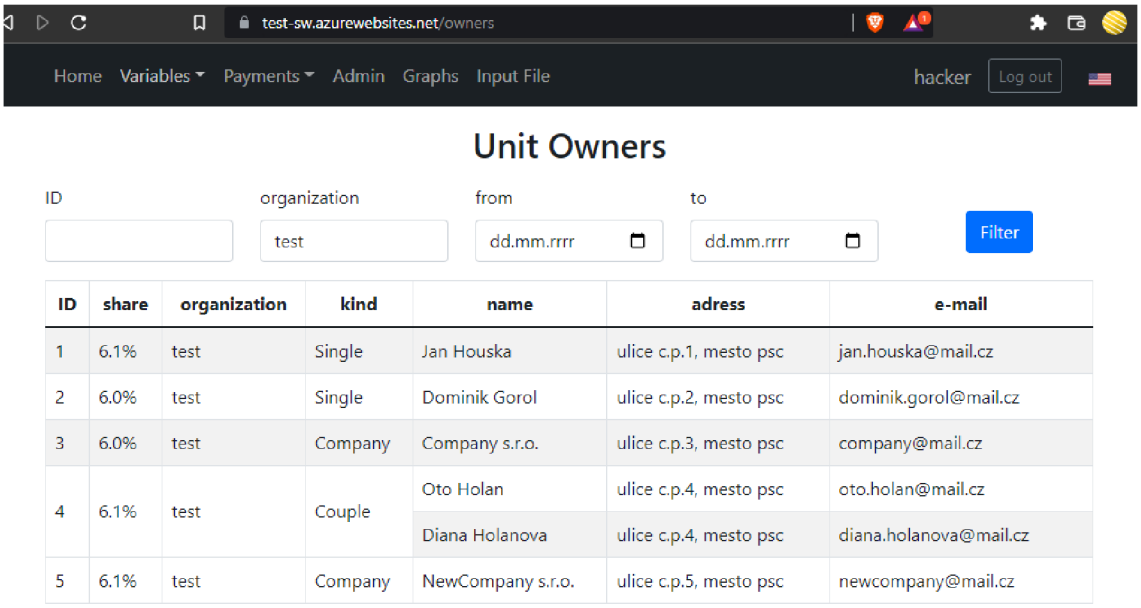
Na serverové části je potřeba generovat JSON soubor dle openAPI specifikace, který je použit pro generování datových typů na klientské straně. Pro samotnou generaci byly použity dvě knihovny NSwag a Swashbuckle, kde obě dvě knihovny generují podobný JSON soubor.

4.3 Testování

Při vývoji klientské části aplikace je využíván vývojářský server, který je spuštěn lokálně. Tento server umožňuje při vývoji zobrazit vytvářené elementy a ladit program. Pro zobrazení samotných dat je třeba mít lokální server pro serverovou část aplikace nebo se připojit k testovacímu serveru. Pro rozlišení rozdílných připojení byly vytvořeny 2 skripty *yarn start-dev* a *yarn start-prod*. Rozdíl mezi těmito příkazy je, že jsou spuštěny v různém prostředí a obsahují jiné hodnoty globálních proměnných. Globální proměnné jednotlivých prostředí jsou uloženy v souboru *.env-cmdrc.json*.

Testovací server pro klientskou stranu aplikace je vytvořen pomocí Microsoft Azure. Zde je vytvořen základní účet a založen server pro webovou aplikaci, který má jako operační systém Debian 10 s využitím NODE 16. Webová stránka je vytvořena na adrese https://test-sw.azurewebsites.net 4.7. K nasazení aplikace na server byl

použit editor *Visual Studio code* s použitím rozšíření *Azure app service* a do složky public byl přiložen soubor *web.config*, který obsahuje nastavení pro webový server.



ID	share	organization	kind	name	adress	e-mail
1	6.1%	test	Single	Jan Houska	ulice c.p.1, mesto psc	jan.houska@mail.cz
2	6.0%	test	Single	Dominik Gorol	ulice c.p.2, mesto psc	dominik.gorol@mail.cz
3	6.0%	test	Company	Company s.r.o.	ulice c.p.3, mesto psc	company@mail.cz
4	6.1%	test	Couple	Oto Holan	ulice c.p.4, mesto psc	oto.holan@mail.cz
				Diana Holanova	ulice c.p.4, mesto psc	diana.holanova@mail.cz
5	6.1%	test	Company	NewCompany s.r.o.	ulice c.p.5, mesto psc	newcompany@mail.cz

Obrázek 4.7: Ukázka webové stránky na testovacím serveru

5 Závěr

Cílem této práce bylo vytvořit webovou aplikaci umožňující správu bytových jednotek. K dosažení bylo třeba seznámit se s možnými architekturami a technologiemi k vytvoření aplikace.

V první části práce jsou shrnuty technologie v oblasti délkového odečtu měřičů, přenos dat a způsoby zobrazení získaných informací. Důraz je kladen především na architekturu softwaru pro zobrazení dat a frameworky použité pro vytváření jednotlivých částí aplikace. V další části jsou popsány informace o použitém frameworku, programovacím jazyce, knihovnách a koncepty potřebné pro vytvoření aplikace.

Na základě těchto poznatků byla vytvořena šablona aplikace pomocí React pro Typescript. Dále byla vytvořena základní struktura pro ukládání souborů s jednotlivými komponenty a struktura aplikace pro možnost vložení nových stránek. Pro vytvoření samotných stránek zobrazující data, bylo potřeba vytvořit kód pro generování datových typů a funkcí pro komunikaci se serverem. Pro implementaci komunikace v komponentách byly vytvořeny hooky, které se také starají o odesílání, přijímání dat a status požadavku. Poté byla vytvořena stránka pro zobrazení přijatých dat v tabulce. Tato stránka obsahuje formulář pro filtraci dat, tabulku s daty a tlačítka umožňující zobrazit další stránky v případě, kdy server odeslal jen část určených dat. Tabulka také zobrazuje upozornění v případě chybné odezvy ze serveru.

Stránka, která byla vytvořena dále je určena pro posílání dat na server. Tato stránka se skládá z formuláře pro vložení informací k danému datovému typu. Stránky pro zobrazení tabulky a vložení informací byly vytvořeny nejprve pro datový typ UnitVariable, což je schéma společné pro všechny ukládané hodnoty do databáze. Dále byly vytvořeny obdobné stránky pro vložení a zobrazení dat pro naměřené hodnoty z vodoměru, vlastníky jednotek a bytové jednotky.

Také byla vytvořena stránka pro zobrazení výpisu rozúčtování jednotlivých bytových jednotek. Tato stránka obsahuje informace o dané jednotce, vlastnících a jednotlivých nákladů, které jsou třeba za jednotku uhradit. Rozúčtování mělo být generováno jako html pro mail, ale místo toho je tato stránka upravena pro tisk tak, ať při tisku z prohlížeče se vygeneruje rozúčtování, které je dále možné uložit do formátu PDF. Dále aplikace obsahuje stránku umožňující vložení položek nákladů, které se zobrazují na vyúčtování jednotek a tabulku s ročními výdaji, u které je

možné editovat částku jednotlivých položek a datum, kdy byly zaplaceny.

Byly vytvořeny komponenty pro zobrazení grafů, které nejsou zatím využity pro žádná konkrétní data. Pro jejich ukázku je na stránce grafy vytvořen sloupcový a časový graf s pevnými daty. Dále aplikace obsahuje stránku s možností nahrát a odeslat soubor na server a stránku s administrátorskými akcemi, která umožňuje zatím jen odstranění všech dat v databázi. Jako součást aplikace byla vytvořena základní autorizace, kde uživatel potřebuje nejprve zadat uživatelské jméno a heslo pro přístup k zobrazení dat.

Po vytvoření, byla aplikace umístěna na veřejný server pomocí Microsoft Azure na adresu https://test-sw.azurewebsites.net. Pro vložení aplikace na server byl použit editor *Visual Studio code* s rozšířením *Azure app service*, který umožňuje jednoduché vytvoření serveru a nasazení aplikace.

Pro praktické využití této webové aplikace pro správu společenství bytových jednotek je třeba vytvořit další zobrazení, které by dané společenství mohlo potřebovat. Další očekávaná vlastnost od takové aplikace je možnost vytvoření různých druhů účtů pro uživatele, tak aby měl například vlastník bytové jednotky jiné možnosti zobrazení než správce společenství.

Použitá literatura

[1] CHREN, Stanislav; ROSSI, Bruno; PITNER, Tomáš. Smart grids deployments within EU projects: The role of smart meters. In: *2016 Smart Cities Symposium Prague (SCSP)*. 2016, s. 1–5. Dostupné z DOI: 10.1109/SCSP.2016.7501033.

[2] ALVISI, Stefano; CASELLATO, Francesco; FRANCHINI, Marco; GOVONI, Marco; LUCIANI, Chiara; POLTRONIERI, Filippo; RIBERTO, Giulio; STEFANELLI, Cesare; TORTONESI, Mauro. Wireless Middleware Solutions for Smart Water Metering. *Sensors*. 2019, roč. 19, č. 8. ISSN 1424-8220. Dostupné z DOI: 10.3390/s19081853.

[3] *M-Bus* [online]. M-Bus, 2020 [cit. 2022-05-10]. Dostupné z: https://m-bus.com/.

[4] KHATWA, Aditya. *Managers* [online]. Avion Housing Management, 2020 [cit. 2022-05-14]. Dostupné z: https://arcane-tundra-59376.herokuapp.com/admin/managers/manager/.

[5] ABUAESH, Noha. *Abuaesh / HousingManagement* [online]. San Francisco: Github Inc., 2020 [cit. 2022-04-24]. Dostupné z: https://github.com/abuaesh/HousingManagement.

[6] KROTOFF, Tanguy. *Frontend Frameworks Popularity* [online]. San Francisco: Github Inc., 2022 [cit. 2022-04-24]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.

[7] POKORNÝ, Jaroslav; VALENTA, Michal. *Databázové systémy*. 1. vyd. Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

[8] *Javascript* [online]. Moscow: Ilya Kantor, 2021 [cit. 2022-04-11]. Dostupné z: https://javascript.info/intro.

[9] *JWT* [online]. Bellevue: Auth0, 2013 [cit. 2022-05-11]. Dostupné z: https://jwt.io/.

[10] *Yarn* [online]. Yarn, 2016 [cit. 2022-04-29]. Dostupné z: https://yarnpkg.com/.

[11] *Openapi-typescript-codegen* [online]. San Francisco: Github Inc., 2020 [cit. 2022-04-18]. Dostupné z: https://github.com/ferdikoomen/openapi-typescript-codegen.

Příloha I - zdrojové kódy

- Klientská část aplikace - sv-client.zip