VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE PRO ENERGETICKÝ MANAGEMENT BUDOV

WEB APPLICATION FOR BUILDING ENERGY MANAGEMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Ondřej Rygl

AUTHOR

VEDOUCÍ PRÁCE — Ing. Lukáš Jablončík

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Ondřej Rygl ***ID:*** 223327

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Webová aplikace pro energetický management budov

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je vytvoření webové aplikace pro energetický management budov. V rámci teorie student provede analýzu dostupných řešení, seznámí se s požadavky na aplikace pro řízení chytrých budov a aktuálním řešením backendové aplikace. V praktické části bude navržena a naprogramována webová aplikace s plnou podporou autentizace a autorizace uživatelů na několika úrovních práv. Bude navrženo rozhraní pro komunikaci s prvky chytré budovy. Aplikace umožní import a export dat. Finální aplikace bude nasazena na testovacím serveru a otestuje se v reálném provozu budovy, kde bude ověřena její funkčnost, rychlost, bezpečnost a stabilita.

DOPORUČENÁ LITERATURA:

[1] Lathkar, M. (2021). Building web apps with python and flask learn to develop and deploy responsive restful web applications using flask framework. BPB Publications.

[2] Martin, R. C. (2009). Clean code: A Handbook of Agile Software Craftsmanship. Prentice Hall. ISBN: 9780132350884.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Lukáš Jablončík

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá implementací webové aplikace sloužící jako nástroj pro zobrazování dat z chytrých prvků domácnosti. Cílem práce bylo seznámit se s již podobnými řešeními, které se zabývají podobnou problematikou, existujícím backendem, který má sloužit jako zdroj dat aplikace a v neposlední řadě vytvořením webové aplikace. Ta má uživatelům umožnit lehké zobrazení dat vyčtených z prvků chytré domácnosti. Webová aplikace byla vytvořena pomocí frameworku pythonu Django a javascriptové knihovny React.js.

KLÍČOVÁ SLOVA

Webová aplikace, Python, Django, JavaScript, React.Js, API, Material UI

ABSTRACT

Bachelor thesis describes implementation of web aplication used as a tool to visualize data from smart home devices. The goal was to study alternative sollutions, get to know the current backend implementation that will serve as data provider and lastly implementation of web aplication. The aim is to easily display data from the smart home gadgets. Application was created using python framework Django and javascript library React.js

KEYWORDS

Web application, Python, Django, JavaScript, React.Js, API, Material UI

RYGL, Ondřej. *Webová aplikace pro energetický management budov*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 51 s. Bakalářská práce. Vedoucí práce: Ing. Lukáš Jablončík

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Ondřej Rygl
VUT ID autora:	223327
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Webová aplikace pro energetický management budov

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu bakalářské práce panu Ing. Lukáši Jablončíku za odborné vedení, konzultace, podnětné návrhy k práci a zpřístupnění dat, bez kterých by práce nemohla být vytvořena.

Obsah

Seznam obrázků

Úvod

S rostoucími cenami energii spojenými s aktuální energetickou krizí přichází lidé se snahou hlídat si spotřebu elektrické energie ve své domácnosti a zaměstnaní. Díky moderním technologiím již lze jednoduše vyčítat data o spotřebě energie ze zařízení jako elektroměr nebo nabíječka na elektromobily. Mezi hlavní výhody monitorování spotřeby elektrické energie může patřit například možnost rozvrhnutí spotřeby elektrické energie do doby, kdy je cena energie na spotovém trhu nižší, čímž může dojít k úsporám v řádu desítek procent.
Cílem bakalářské práce je vytvořit webovou aplikaci, která bude schopna vyčítat a zobrazovat data z prvků chytré domácnosti do grafu. Aplikace bude poskytovat možnost autentizace uživatele na základě jeho přihlašovacích údajů a autorizace na základě jeho úrovni oprávnění. Data ze všech zařízení půjde exportovat a stahovat.
Práce je rozdělena do několika celků. První část se zabývá teoretickým vysvětlením pojmů spojených s vývojem webové aplikace a představením zabezpečeni webové aplikace. Druhá kapitola obsahuje analýzu již existujících řešení, které se zabývají podobnou problematikou. Ve třetí kapitole je popsán aktuální stav backendové aplikace, která zajišťuje výčet a zpracování dat ze zařízení. Poslední kapitola se věnuje implementaci řešení a představení výsledné podoby vytvořené aplikace.

1 Teoretická část studentské práce

V této kapitole budou teoreticky vysvětleny základy webových aplikací, základní principy API a programovací jazyky python a javascript, jejichž frameworky a knihovny byly využity při tvorbě aplikace. V další části kapitoly budou popsány principy autentizace a ověření uživatele při přístupu k datům skrz API.

1.1 Webová aplikace

Webovou aplikaci lze definovat jako aplikaci typu klient-server, to znamená, že aplikace běží na serveru a klient - uživatel, k ní přistupuje vzdáleně pomocí webového prohlížeče skrz internet. Webová aplikace se často skládá ze dvou částí – `backendu` - části aplikace, která běží v rámci serveru a `frontendu` – grafického rozhraní, které vidí koncový uživatel a pomocí kterého aplikaci ovládá.

1.1.1 Backend webové aplikace

Část aplikace běžící na serveru. Nezabývá se vzhledem, ale operacemi s daty a komunikací s připojenou databází, popřípadě dalšími službami, na které je aplikace napojená a odkud čerpá data. Backend webové aplikace vytváří a poskytuje pro frontend API, pomocí kterého jsou posílány dotazy a požadavky na data.

1.1.2 API

Otevřené rozhraní označené jako API pochází z anglického `Application Programming Interface`. Jedná se o způsob, jakým si 2 nebo více aplikací předávají data a vzájemně komunikují. Nejpoužívanějším typem otevřeného rozhraní je tzv. REST API z anglického Representational State Transfer. Jedná se o styl, který jasně definuje jak přistupovat k datům z otevřeného rozhraní a jak toto rozhraní tvořit. Tento styl jako první ve své disertační práci z roku 2000 představil Roy Fielding. [1] Základní jednotkou REST API je tzv. `Resource`, v překladu zdroj. Zdroje se mohou shlukovat do kolekcí, nebo vystupovat jako jedinečné prvky. Ke zdrojům se pak přistupuje pomocí tzv. `URI - (Uniform Resource Identifier)` - v překladu Jednotný identifikátor zdrojů. Jedná se o adresu, která definuje cestu ke konkrétní kolekci zdrojů, či zdroj jako prvek.[2]

Základní metody REST API

1. **GET** – Získává data daného zdroje
2. **POST** – Vytváří nové zdroje na serveru
3. **PUT** – Změní, respektive aktualizuje konkrétní zdroje na serveru

4. **DELETE** – Odstraní zdroje ze serveru

Posílání dat prostřednictvím API

V současné době se nejčastěji používají 3 druhy formátování dat posílaných pomocí API.

1. **XML** – Z anglického `Extensible Markup Language`, lze název přeložit jako rozšiřitelný značkovací jazyk. Jak už z názvu vyplývá, strukturu dat formátu XML definují tzv. značky – podobně jak je tomu v jazyce HTML. Příklad použití XML zápisu je ukázán v příloze A.1.
2. **JSON** – Nejpoužívanější formát dat při komunikaci s API, je jasně definován a rozdělen pomocí různých druhů závorek, které vytváří jeho tělo. Jazyk je pro člověka jednoduchý na čtení a psaní a pro stroj je lehce generovatelný a zpracovatelný.[3] Ukázka zápisu JSON formátu je znázorněna v příloze A.2
3. **YAML** – Je zkratkou pro `YAML Ain´t Markup Language`, v překladu YAML není značkovací jazyk. Dle autorů se jedná o lidsky přívětivý způsob serializace dat, který je podporován všemi programovacími jazyky.[4] Podobně jako programovací jazyk Python, ani YAML nepoužívá závorky a pro oddělení dat jsou použity mezery. Při zápisu seznamu jsou použity odrážky pro definici jednotlivých prvků. Představení zápisu jazyku YAML je představeno v příloze A.3.

1.1.3 Frontend webové aplikace

Část aplikace, se kterou interaguje cílový uživatel. Získává vstup uživatele, který poté posílá ke zpracování do backendu aplikace. Zobrazuje data, která od backendu obdrží. Cílem frontendu je být co nejvíce přehledný a pro uživatele přívětivý, aby jej bylo možné nejlehčím způsobem ovládat. K vývoji frontendu webových aplikací se používají jazyky HTML, CSS a javascript, který aplikaci dodává dynamičnost.

1.2 Programovací jazyky

1.2.1 Python

Python je objektově orientovaný programovací jazyk, jehož vysokoúrovňové datové struktury spojené s dynamickým typováním proměnných z něj dělá velmi oblíbený jazyk například pro rychlé vytváření aplikací, skriptování nebo nástroj pro jednoduché spojení již existujících aplikací dohromady. Zároveň také podporuje balíčky a další zásuvné moduly, pomocí kterých jej lze lehce rozšířit. [5]

1.2.2 Framework Django

Django je open source framework rozšiřující programovací jazyk Python. Podobně jako další webové frameworky se vyznačuje architekturou MVC, která je popsaná v následující sekci. Nově vytvořený projekt ve frameworku Django v sobě již zahrnuje administraci webu, správu databáze a další funkce, které uživateli ulehčí vývoj webové aplikace.[6]

Django MVT model

Framework Django patří mezi webové frameworky využívající architekrutu `MVC - Model-View-Controller`. Tato architektura jasně rozděluje logickou část webu od části se kterou pracuje uživatel. V případě Djanga však `View` nedefinuje jak se mají data zobrazit, ale která data se mají uživateli zobrazit. `View` tedy můžeme chápat jako `Controller` v obecné definici `MVC` architektury. O to, jak se data zobrazují se v Djangu stará tzv. `Template`, který tedy z klasického `MVC` modelu nahrazuje `View`. V podání Djanga by se tedy tato architektura dala nazvat `MVT - Model-View-Template`.[7]

1.2.3 HTML

HyperText Markup Language je nejzákladnější stavební prvek webové aplikace. Definuje strukturu obsahu zobrazovaného na webové stránce, se kterou poté další programovací jazyky pracují. Jak již z anglického názvu vyplývá, HTML pracuje s tzv. značkami, které mají předdefinované vlastnosti, pomocí kterých se určuje formátování jejich obsahu.[8]

1.2.4 DOM - Document Object Model

Document Object Model je reprezentací objektů, které tvoří strukturu webu. DOM tak umožňuje dalším komponentům webu s touto strukturou a jejím obsahem pracovat nebo obsah upravovat. Obsah je uchován ve stromové struktuře, s niž lze rozlišovat nadřazené a podřazené prvky webových stránek. Po úpravě dat se však musí všechny podřazené prvky znovu vykreslit, aby se aktualizoval vzhled stránky.[9]

1.2.5 JavaScript

JavaScript je objektově orientovaný programovací jazyk, který se primárně využívá k vytváření dynamických webových aplikací. Jeho kód je kompilovaný `just-in-time`, čímž je především zvýšen výkon aplikace. Pro tvorbu frontendových aplikací se často

využívají frameworky rozšiřující javascript o další funkcionality. Mezi ty nejpoužívanější patří React[1], Vue.js[2], nebo Angular[3].

1.2.6 Knihovna React.js

Knihovna React.js vznikla jako interní nástroj firmy Meta pro vytváření jejich aplikací. Nyní se jedná o jednu z nejrozšířenějších knihoven určených pro tvorbu webových stránek. Základním principem knihovny React je tvorba takzvaných jednostránkových aplikací. Jejich princip spočívá v načtení obsahu stránky pouze jednou a následně všechny změny proběhnou pouze na straně klienta. Veškerý obsah zobrazující se uživateli je tvořen z komponentů, které spolu navzájem pracují a vytváří tak plnohodnotnou stránku.

JSX

React JSX lze nazvat kombinací klasického HTML (Hypertext Markup Language) kódu s Javascriptovým kódem, kterému tak dodává možnost definovat, jak má daný komponent vypadat. Prvky JSX jsou vzhledově podobné HTML kódu, část prvků je identická, je tudíž snazší pro uživatele s tímto kódem pracovat. [10]

React komponenty

Komponenty jsou základním stavebním prvkem reactu. Rozdělí uživatelské rozhraní do na sobě nezávislých, znovu použitelných částí, které jsou od sebe izolované. Daly by se přiblížit k funkcím v javascriptu, jelikož přijímají atributy, se kterými pracují a vrací prvky, které definují, co se zobrazí na obrazovce.
React komponenty můžeme dělit na funkční a třídní. Komponenty definované funkcí jsou nejjednodušší způsob, jak vytvořit komponent v reactu. Funkce přijímá `props`, kterými získává data. Od verze reactu 16.8 lze pomocí tzv. `Hooks` nahradit konstruktor, čímž lze funkčními komponenty kompletně nahradit ty třídní[11]. Funkční komponenty pak vrací JSX kód. Naopak třídní komponenty jsou javascriptové `ES6 třídy`[12], rozšířené o `React.Component`. Musí obsahovat funkci `render()`, která vrací react prvky v podobě JSX kódu. Počáteční data musí být vytvořena pomocí konstruktoru.

[1] https://react.dev/
[2] https://vuejs.org/
[3] https://angular.io/

Virtual DOM

Virtual DOM je čistě virtuální reprezentace DOMu, která vznikne po každé změně dat. Tento virtuální DOM je poté porovnán oproti reálnému DOM, načež se spočítá rozdíl a minimální počet operací nutných pro aktualizaci reálného DOMu. Změny se poté propíší do reálného DOMu. V knihovně React má každý komponent svůj `state`, který se po každé upravě změní, čímž zavolá změnu virtual DOM. [9]

1.3 Autentizace

Autentizace je proces ověřování faktu, že daný prvek je ten, za který se vydává. V kontextu webových aplikací se běžně ověřuje uživatel pomocí přihlašovacího jména nebo ID a svého privátního klíče, který by měl znát pouze daný uživatel. Nejčastěji je klíč v podobě hesla nebo kódu. [13]

1.3.1 Autentizace vůči API

API může sloužit jako nástroj pro získávání dat. Ty jsou ve většině případů soukromá a neměl by se k nim dostat neautorizovaný uživatel. Proto je nezbytné zajistit způsob, jak povolit dotazy na API pro ověřené uživatele aplikace.

HTTP header

Basic Authentication je nejjednodušším způsobem autentizace vůči API je do hlavičky HTTP požadavku přidat `Authorization: Basic token`, přičemž token je ve tvaru:

$$Base64(username : password) \tag{1.1}$$

Tento způsob však pro svůj jednoduchý způsob dešifrování není doporučován bez dalších bezpečnostních mechanizmů jako je například HTTPS/SSL.
JSON Web Token, zkráceně JWT je jedním ze způsobů ověřování uživatele vůči API umožňující uživateli posílat požadavky na API bez nutnosti neustálého prokazování své identity pomocí jména a hesla. JWT je spočítán ověřovacím serverem na základě přihlašovacích údajů zaslaných uživatelem. Token se skládá ze tří částí:

1. HEADER - Hlavička dat, často se skládá ze 2 částí – označení typu tokenu a názvu algoritmu použitého pro podepsání dat.
2. PAYLOAD - Data, jenž obsahují informaci o uživateli (např. login) a další doplňková data, například čas expirace tokenu.
3. SIGNATURE - podpis, který obsahuje hash skládající se z hlavičky a dat již zašifrovaných pomocí funkce Base64.

Celkový token se poté skládá ze spojení všech tří části zašifrovaných pomocí funkce Base64, jak je znázorněno ve následujícím vzorci:[14]

$$[Base64(HEADER)].[Base64(PAYLOAD)].[Base64(SIGNATURE)] \quad (1.2)$$

API klíče

API klíče jsou tokeny, které klient posílá společně s HTTP požadavkem. Tento token lze poslat jako string v adrese, jako hlavičku požadavku nebo jako cookie. Jelikož se jedná o privátní klíč známý pouze klientu a API serveru, je společně s tímto druhem autentizace nutné implementovat i zabezpečené spojení například pomocí HTTPS/SSL.

OAuth 2.0

Jedná se autorizační protokol, který dá API dočasný přístup k údajům uživatele na jiném web serveru. Uživatel bude vyzván, aby se přihlásil vůči jinému verifikačnímu serveru, který poté předá API přístupový token, pomocí kterého se uživatel autentizuje bez nutnosti posílat API jakékoliv heslo. Mezi největší provozovatele OAuth2 serverů patří například Google[4] nebo Facebook.[5] [15]

1.4 Zabezpečení webových aplikací

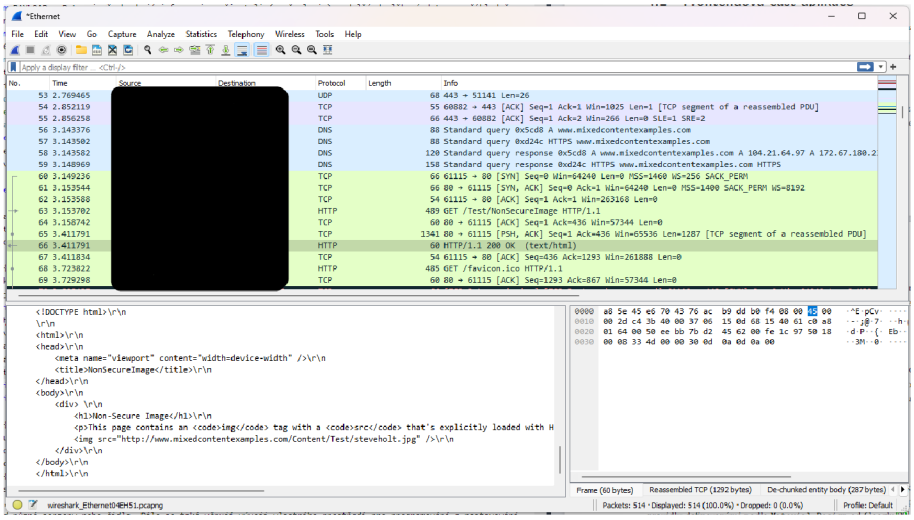
V době, kdy se celý svět přenáší do digitálního prostoru, je třeba digitální komunikaci zabezpečit tak, aby nebylo možné její obsah odposlechnout a zneužít.

1.4.1 HTTPS

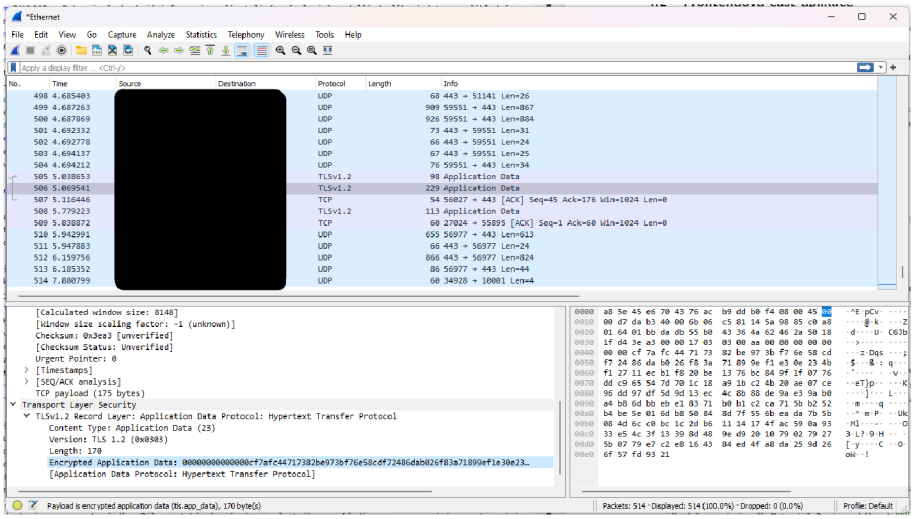
Zkratka HTTPS pochází z anglického `Hypertext trasfer protokol secure` a dodává tak klasickému protokolu HTTP přívlastek o zabezpečení. Na rozdíl od něj je totiž veškerá komunikace, která je pomocí protokolu HTTPS přenesena, šifrovaná pomocí protokolu SSL/TLS. Komunikace je zašifrována pomocí veřejného klíče aplikace a její dešifrování a následné přečtení je možné jen pomocí soukromého klíče, který by měl zůstat utajený a znát by jej měl jen vlastník dané webové aplikace, se kterou uživatel komunikuje. Na obrázku 1.1 je vidět nezašifrovaná komunikace odeslaná pomocí protokolu HTTP. Na obrázku 1.2 je znázorněna ta stejná komunikace, přenesená zašifrovaně pomocí protokolu HTTPS.

[4] https://developers.google.com/identity/protocols/oauth2
[5] https://developers.facebook.com/docs/facebook-login/

Komunikace pomocí HTTPS probíhá na portu 443. Ve všech moderních internetových prohlížečích je znázorněna pomocí zámečku, který se nachází vedle vyhledávací lišty. Po kliknutí na něj lze nahlédnout na certifikát, který obsahuje informace o veřejném klíči, pomocí kterého je komunikace zabezpečena.



Obr. 1.1: Odchycení paketů odeslaných pomocí HTTP v aplikaci Wireshark



Obr. 1.2: Odchycení paketů odeslaných pomocí HTTPS v aplikaci Wireshark

1.4.2 SSL/TLS

SSL neboli `Secure Socket Layer` je síťový protokol, který pomocí šifrovaní zajišťuje autenticitu dat odeslaných při komunikaci přes internet. S první verzí tohoto protokolu přišla v roce 1995 firma Netscape, která si dala za cíl zajistit soukromí, ověření a integritu dat na internetu.[16]
V roce 1999 byl nahrazen protokol SSL protokolem TLS - `Transport layer security`. Protokol TLS funguje na principu asymetrické kryptografie. V momentě kdy se uživatel připojí na server, zahájí se tzv. `TSL handshake`. Během něj klient specifikuje své dostupné možnosti šifrování komunikace (dostupné šifry, protokoly a jejich verze). Server poté vybere nejvhodnější z nich, odešle svůj certifikát a spolu s ním způsob pomocí kterého bude komunikace šifrována společně s parametry na základě kterých bude vygenerován klíč pro šifrování zpráv. Následně je vygenerován a ověřen tzv. `session key`, pomocí kterého je celá komunikace šifrována, což zajišťuje, že pouze klient a server budou schopni tuto komunikaci dešifrovat a číst. Od nejnovější verze protokolu TLS 1.3 je tento handshake zjednodušen, původních 37 možných šifrovacích sad bylo zredukováno pouze na 5, díky čemuž je možné odeslat klíče pro všechny varianty v jedné zprávě a není tedy nutné čekat než server zvolí konkrétní šifrování. Tímto z celé komunikaci odpadá jeden krok a zpráva, kterou vrací server může být pomocí klíče zvolené metody zašifrována. Celý proces vytváření `session keys` je tedy urychlen.[17][18]

1.4.3 OWASP Top ten

OWASP neboli Open Worldwide Application Security Project je nezisková organizace zaměřující se na zlepšení zabezpečení aplikací v internetu. Funguje na principu otevřené komunity, což znamená, že každý může přispět do projektů, které jsou pořádány. Organizace každoročně po celém světě pořádá školení a konference, které představují zranitelnosti a potencionální bezpečnostní hrozby. Pravidelně je také vydáván přehled s názvem OWASP Top 10, který představuje 10 nejkritičtějších zranitelností webových aplikací zaznamenaných za posledních pár let. Nejaktuálnější seznam byl vydán v roce 2021 a vypadá následovně:

1. Broken Access Control - Útočník by nikdy neměl být schopen přistoupit jako neautorizovaný uživatel tam, kam by měl mít přístup pouze uživatel, který autorizovaný je. Úpravou parametrů URL adresy nebo například paměti aplikace uložené v prohlížeči může být získán přístup do nezabezpečené aplikace, popřípadě odeslat API požadavek, který může vést i ke smazání dat.

2. Cryptographic Failure - Veškerá komunikace klienta se serverem by měla být šifrovaná pomocí dostatečně silné šifry, například symetrické šifry AES s 256 bitovým klíčem. Nezabezpečená komunikace může vést ke ztrátě přihlašovacích údajů, popřípadě ke ztrátě osobních dat uživatelů, kteří danou aplikaci používají.
3. Injection - Každý neošetřený vstup, kam uživatel zadává data může být potencionální zranitelností webové aplikace. Zabránit injektování škodlivých příkazů lze třeba validací dat, které odeslal uživatel, nebo ošetřením vstupních polí, jenž by měla být chráněna například limitací typů symbolů, které do něj uživatel může zadat. Není nutné, aby do pole, kam má být uživatelem zadáno číslo šel vložit jiný znak než číslice. Zvláště nebezpečné poté mohou být speciální symboly, které se používají například u databázových jazyků. Mezi nejčastější injekce patří třeba SQL injekce, díky které může útočník získat přístup k databázi.
4. Insecure Design - Bezpečný design webové aplikace je její navržení v takové podobě, aby vždy byly vyhodnocovány nové potencionální hrozby a kód by byl navržen a otestován na jeho odolnost vůči známým útokům. Modelování potencionálních hrozeb by mělo být začleněno do procesu zdokonalení aplikace. V momentě kdy tomu tak není, může se stát celý návrh nebezpečným a potencionálně zranitelným.
5. Security Misconfiguration - Chybná konfigurace webové aplikace a serveru na kterém běží může útočníkovi ukázat nebo dokonce otevřít vstup, pomocí kterého se dokáže dostat k datům, nebo získat přístup k danému server. Často se může jednat o zbytečně otevřený port, který není k běhu aplikace třeba, nebo použití originálních nezměněných přihlašovacích údajů. Špatně nastavené loggování může ukázat část zdrojového kódu, ke kterému by se uživatel neměl dostat. Server, který hostuje danou webovou aplikaci by měl být co nejvíce minimalistický. Neměl by obsahovat další služby, které by jeho fungování mohly ohrozit a především by všechny aplikace, které nejsou potřeba pro jeho běh měly být smazány nebo odděleny.
6. Vulnerable and Outdated Components - Zastaralé a neaktualizované komponenty aplikace jsou častým cílem útočníků. Postupem času se objevují zranitelnosti, na které jsou od tvůrců aplikací vydávány bezpečností opravy a záplaty. Často se tyto aktualizace nenainstalují samy od sebe a je tedy nutné, aby byl stav systému pravidelně monitorován a aktualizován. Existuje řada nástrojů, které dokáží monitorovat stav zranitelností systému. Jedním z nich třeba nástroj Shodan[6].

[6]https://www.shodan.io/

7. Identification and Authentication Failures - Nedostatečné ověření uživatele a nízké nároky na kvalitu jeho přihlašovacích údajů jsou dalším častým způsobem jak útočník může získat neoprávněný přístup do aplikace. Systém by neměl dovolit uživateli použít triviální hesla, která lze metodou bruteforce prolomit. Pokud webová aplikace podporuje autentizaci uživatele pomocí Single sign-on, musí správně pracovat s jejich tzv. `session tokeny`, které je nutné po odhlášení, nebo určitém časovém intervalu neaktivity mazat.
8. Software and Data integrity Failures - Integrita dat aplikace může být porušena v momentě, kdy se aplikace spoléhající se na externí pluginy a knihovny z nedůvěryhodných zdrojů. Zapnutá funkce automatických aktualizací může také potenciálně narušit integritu a kompatibilitu jednotlivých komponentů aplikace, čímž se může webová aplikace stát zranitelnou.
9. Security Logging and Monitoring Failures - Webová aplikace musí pro správce systému generovat kvalitní logy a záznamy o chybách, které jasně identifikují problémy aplikace, které by mohly vést k neoprávněným vstupům do aplikace.
10. Server-Side Request Forgery - Chyba tohoto typu může nastat v momentě, kdy webová aplikace čerpá data z uživateli dodaných externích zdrojů, může dojít k žádostem na neověřené zdroje dat[19]

2 Analýza již existujících řešení

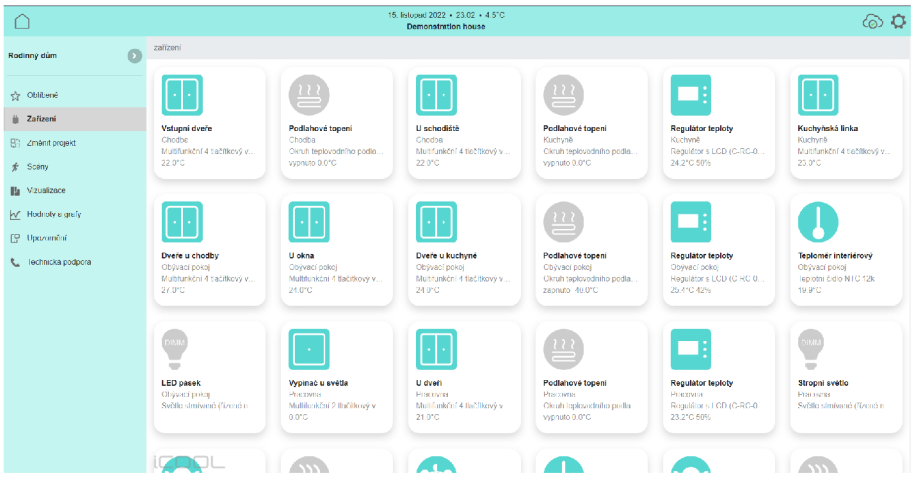
V následující kapitole budou představena již existující řešení, která lze využít pro vytvoření, resp. ovládání chytré domácnosti. Skupina řešení se skládá jak z open-source řešení, které jsou dostupné všem zdarma, tak z komerčních řešení, jenž jsou placené, nebo fungují pouze s konkrétním zařízením sloužícím jako brána.

2.1 Unipi technology

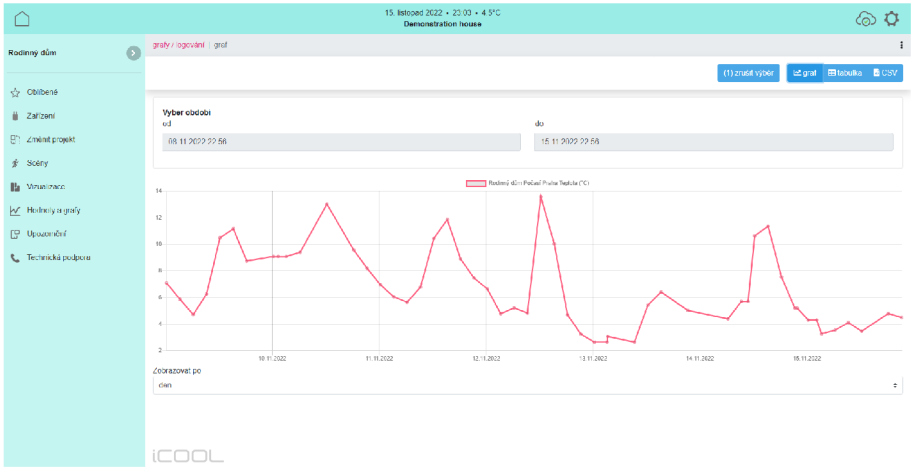
Společnost Unipi technologies vytváří vlastní řídící jednotky, které lze v budově zapojit nejen jako výpočetní jednotku, ale také jako vstupní bránu, tzv. `Gateway`, čímž umožňuje začlenit další zařízení jako jsou například různé senzory nebo čidla. Dále se také věnují vývoji vlastního prostředí pro programování a nastavování řídících jednotek, nazvaného Mervis. Kromě programování samotného PLC je také součástí balíčku Mevris také aplikace Mevris SCADA, které slouží jako dispečerský systém, pomocí kterého lze vzdáleně monitorovat popřípadě řídit jednotlivé prvky k tomuto systému připojeny. Unipi technologies celý balíček Mevris dává k jednotlivým řídícím jednotkám zdarma. Součástí jednotlivých řídících jednotek je i API, na které lze napojit vlastní aplikaci.
Kromě již existujících řešení lze jednotlivé prvky použít pro zakázkový vývoj, díky kterému lze ovládací jednotky rozšířit například o bezdrátové technologie jako je `4G` nebo `Zigbee`.[20]

2.2 iCool4

Řešení společnosti iCool4 nabízí řešení ovládání chytré budovy pomocí svého softwaru napojeného na řídící a regulační systém TECOMAT Foxtrot. [21] Aplikace vytvořená jak pro web, tak pro mobilní zařízení s operačními systémy iOS a Android nabízí velké množství funkcí, jako ovládání prvků domácnosti, možnosti vizualizace dat vyčtených z jednotlivých zařízení, schopnosti vizualizace bytové jednotky včetně jednotlivých prvků pro zjednodušení přehlednosti, možnosti plánovat a optimalizovat spouštění jednotlivých prvků pro optimální využití elektrické energie. Frontend webové aplikace je vytvořen pomocí programovacího jazyku javascript a jeho frameworku Vue.js. Na obrázku 2.1 je zobrazenu ukázka přehledu zařízení v aplikaci. Obrázek 2.2 poté vyobrazuje konkrétní zobrazení dat z teploměru.



Obr. 2.1: Přehled dostupných zařízení aplikace iCOOL4

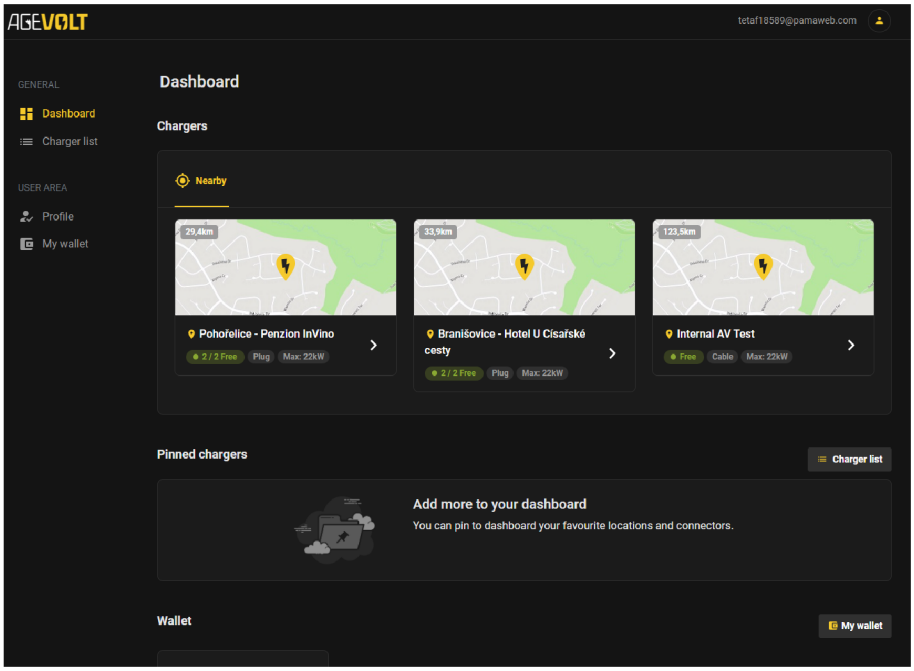


Obr. 2.2: Zobrazení dat z teploměru v aplikaci iCOOL4

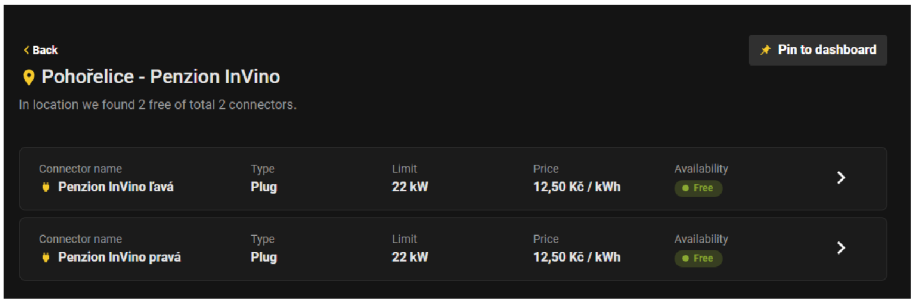
2.3 AgeVolt

Slovenská firma AgeVolt nabízí řešení pro ovládání nabíjecích stanic pro elektromobily. Podobně jako výše zmíněná firma iCool4, AgeVolt využívá hardwarové řešení TECOMAT Foxtrot navázané na vlastní nabíjecí stanice. Celkové řízení je poté umožněno díky spojení s vlastní implementací webové aplikace, která se zaměřuje na tři základní pilíře EMS systému - Měření, Monitorování a řízení systému.[22] Webová aplikace napsaná v javascriptovém frameworku React.js je ve své bezplatné verzi dostupná všem registrovaným uživatelům. Nabízí přehled a následné informace o dobíjecích stanicích pro elektrická vozidla dostupných v rámci systému AgeVolt. Skrz systém je možné si požadovanou stanici zapnout a začít nabíjet. Platba za dobití automobilu je uhrazena v kreditech, které je třeba si předem nahrát na svůj účet. Na obrázcích 2.3 je vidět Dashboard zobrazující nejbližší dobíjecí místa. Na ob-

rázku 2.4 je poté vidět detail konkrétního dobíjecího místa s přehledem jednotlivých dobíjecích stanic.



Obr. 2.3: Dashboard aplikace ageVolt

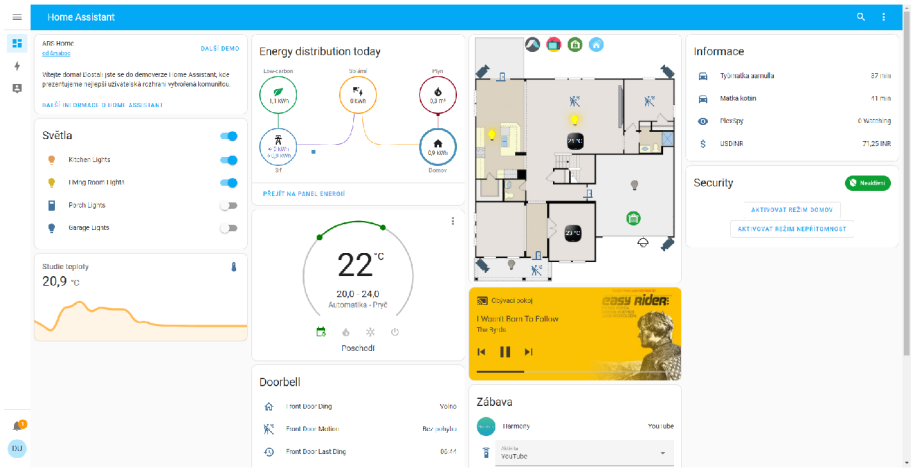


Obr. 2.4: Detail dobíjecích stanic

2.4 Home Assistant

Nástroj Home Assistant je Open-source projekt, který slouží jako ovládací brána chytré domácnosti. Tvůrci řešení kladou velký důraz na bezpečnost a především soukromí uživatele. Celé řešení je totiž vytvořeno pro provozování na lokální síti bez přístupu k internetu a provozování v cloudu, čímž je může být značně zajištěno soukromí a obecně celé zabezpečení chytré domácnosti.[23]

Provozování Home Assistant v domácí síti je vývojáři doporučováno na dedikovaném systému například pomocí zařízení RaspberryPi, pro které nabízí vlastní operační systém. Aplikace je však také dostupná na další platformy jako Windows, Linux, macOS a to v podobě aplikace nebo jako image pro virtuální stroje. Home Assistant aktuálně podporuje přes 2300 integrací chytrých zařízení, implementací a protokolů.[24]
Program je napsán v programovacím jazyce Python. Frontendové řešení webové aplikace je pak napsáno v jazyce javascript. Pro jeho lehkou možnost úprav se vyskytuje velké množství již vytvořených komunitních vzhledů aplikace. Na obrázku 2.5 je zobrazen jeden z nich.



Obr. 2.5: Detail dobíjecích stanic

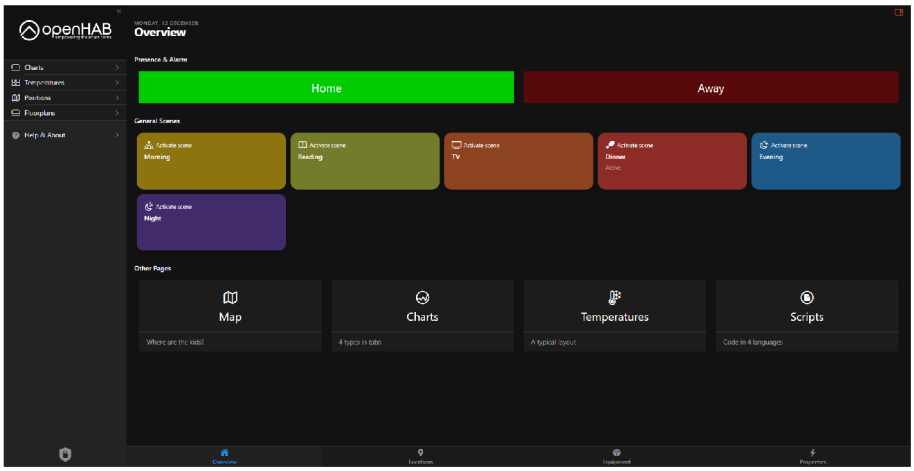
2.5 openHAB

Název openHAB je zkratkou anglického názvu open Home Automation Bus, co lze přeložit jako otevřená domácí automatizační sběrnice. Jedná se o open-source program, který stejně jako Home Assistant slouží jako řídící jednotka chytré domácnosti. Mezi silné stránky svého řešení řadí tvůrci:

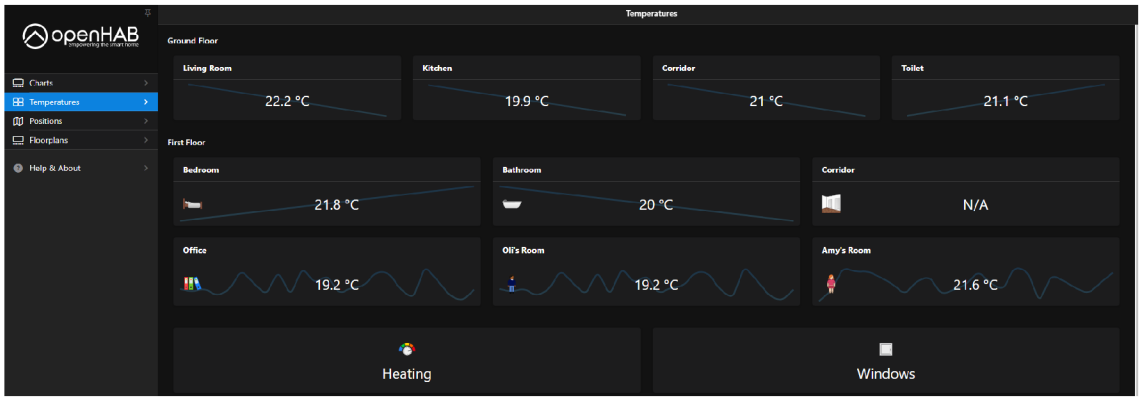
1. Možnost integrovat vetší počet různých zařízení využívající rozdílné systémy.
2. Schopnost vytvoření jednotného rozhraní se společným přístupem k řízení automatizací v celým systému, nehledě na počet výrobců.
3. Flexibilitu nástroje, který lze upravit a přizpůsobit všem požadavkům uživatele.[25]

Systém openHAB je kompletně napsán v programovacím jazyce Java, která je dostupná na většinu operačních systémů jako je Windows, Linux nebo macOS. Pro Raspberry Pi existuje instalační nástroj `openHABian`, které obsahuje předkonfigurovaný image operačního systému s již nainstalovaným řešením openHAB. Frontend

webové aplikace je naprogramován v jazyce javascript pomocí frameworku Vue.js. Na ukázce 2.6 je vidět dashboard a základní menu aplikace. Obrázek 2.7 ukazuje detail zobrazení dat z teploměru.



Obr. 2.6: Dashboard aplikace OpenHAB



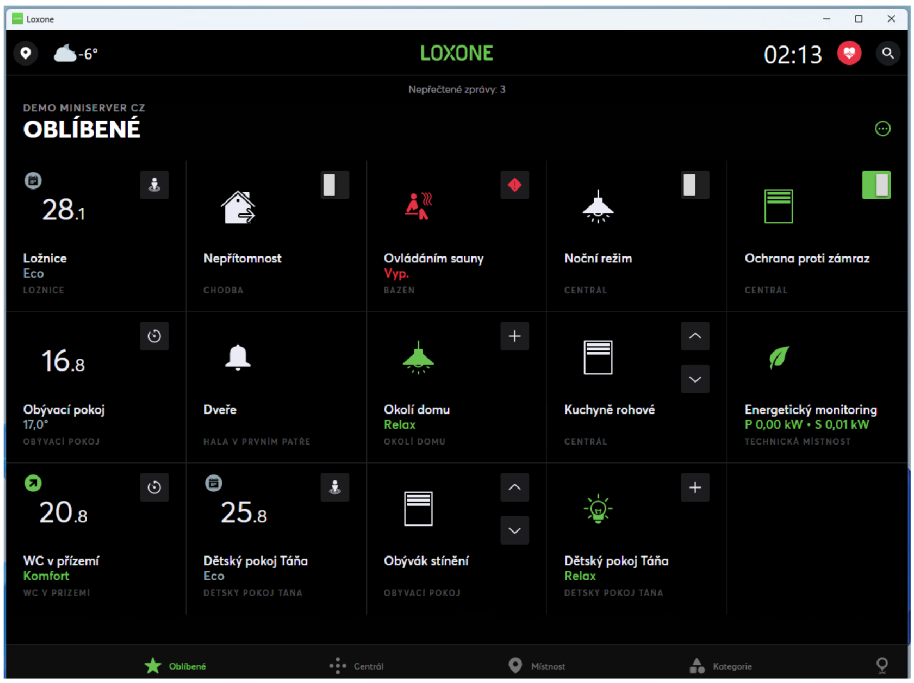
Obr. 2.7: Detail zobrazení dat teploty

2.6 Loxone

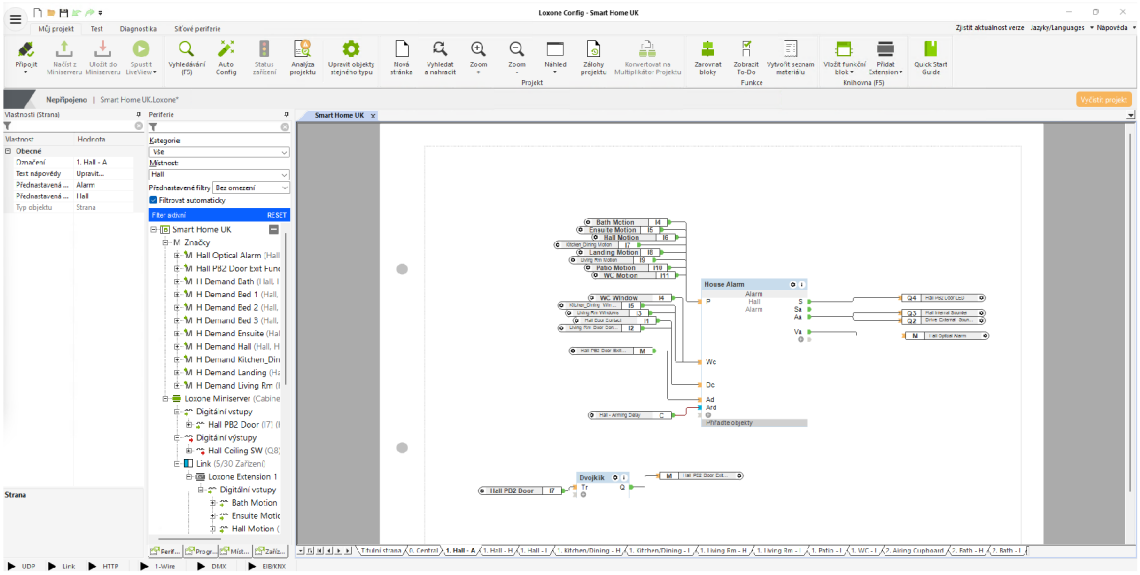
Systém chytrého řízení budovy od společnosti Loxone je kompletně založen na vlastním řešení. Počínaje řídící jednotkou `Miniserver` umístěnou v rozvaděči, přes možnost na míru vytvořeného ovládacího panelu, až po aplikaci, ze které lze celý systém ovládat. Všechny prvky ekosystému Loxone lze instalovat ve dvou variantách. Řešení `Tree`, která ovládá jednotlivé prvky po kabeláži, nebo variantě `Air`, která pro komunikaci využívá radiové vlny o délce 868 MHz definované podle směrnice ETSI EN 300 220-1.[26]

Nastavení a ovládání chytrého řízení budov mají na starost dvě aplikace.

1. Loxone Config – Nástroj sloužící k naplánování a konfiguraci všech prvků systému Loxone. Díky funkcím jako je `Auto config` a `Simulace & Live View` umožňuje snadnou základní konfiguraci a následné vyzkoušení celého systému v simulaci. Další velkou výhodou je také schopnost naprogramovat systém energetického managementu budov, čímž lze optimalizovat spotřebu energie ze sítě a naplno využít energii vytvořenou například pomocí solárních panelů nebo dalších obnovitelných zdrojů. Ukázka aplikace je znázorněna na obrázku 2.9. Aplikace Loxone Config je dostupná ke stažení na počítače s operačním systémem Windows 7 a novější.[27]
2. Loxone App – Aplikace sloužící pro jednoduché ovládání chytré budovy. Plně kompatibilní se všemi prvky ekosystému Loxone. Aplikace nabízí možnost připojení k více vstupním zařízením, čímž umožňuje ovládat více objektů z jedné aplikace. Aplikace je dostupná v desktopové variantě pro počítače s operačním systémem Windows, Linux s architekturou arm64 nebo armv7l, macOS, v mobilní verzi pro telefony s operačním systémem Android, iOS, a také pro chytré hodinky s operačním systémem Wear OS. Na obrázku 2.8 je ukázáno základní menu aplikace. [28]



Obr. 2.8: Dashboard aplikace Loxone



Obr. 2.9: Ukázka aplikace Loxone Config

3 Aktuální backendové řešení

Kapitola popisující stávající backendové řešení aplikace, které je zajišťováno nástrojem Thingsboard. V kapitole je Thingsboard společně jeho součástmi představen.

3.1 Thingsboard

Nástroj Thingsboard je open-source program sloužící pro sběr dat z IoT zařízení, jejich vizualizaci a správu připojených zařízení.
Aplikace je vytvořena tak, aby byla schopna provozu na většině hardwaru a operačních systémech od lokálního mini počítače Raspberry Pi po cloudovém řešení jako `Amazon Web Services`[1], `Microsoft Azure`[2], `DigitalOcean`[3] nebo `Google Cloud Platform`[4].[29]
Řešení je v základní verzi zdarma distribuované jako open-source projekt volně dostupný ke stažení. Pro komerční použití je nutné zakoupit licenci v podobě `Professional Edition`, která je pak také rozšířená o spoustu funkcí, které mohou být klíčové pro implementaci řešení ve firmách a větších projektech.

3.2 Součásti aplikace Thingsboard

Celé řešení Thingsboard se skládá z několika na sobě závislých celků, které budou v následující sekci popsány.

3.2.1 Thingsboard Node

Thingsboard Node je řídící jednotka napsaná v programovacím jazyce Java, která je odpovědná za následující operace: [30]

1. Zpracování REST API dotazů, které mohou využít například aplikace třetích stran.
2. WebSocket odběrů, které jsou využívány primárně webovým rozhraním Thingsboard.
3. Zpracování přijatých zpráv pomocí Rule Engine.
4. Monitorování stavu připojených zařízení.

[1] https://aws.amazon.com/
[2] https://azure.microsoft.com/
[3] https://www.digitalocean.com/
[4] https://cloud.google.com/

3.2.2 ThingsBoard IoT Gateway

ThingsBoard IoT Gateway je open-source software, který slouží jako vstupní brána pro zařízení třetích stran do systému ThingsBoard. Aplikace běží jako služba na IoT zařízení s operačním systémem Linux. Je napsaná v jazyce Python a odesílá MQTT zprávy, pomocí kterých komunikuje se zbytkem systému ThingsBoard.[31]

3.2.3 Rule Engine

Rule engine je rozšíření aplikace Thingsboard umožňující vytváření a spouštění automatizací na základě událostí, které se stanou v průběhu běhu aplikace. Pro uživatele se tak vyskytuje možnost vytváření automatizací, které mohou značně zjednodušit práci s celým systém a autonomně systém řídit.

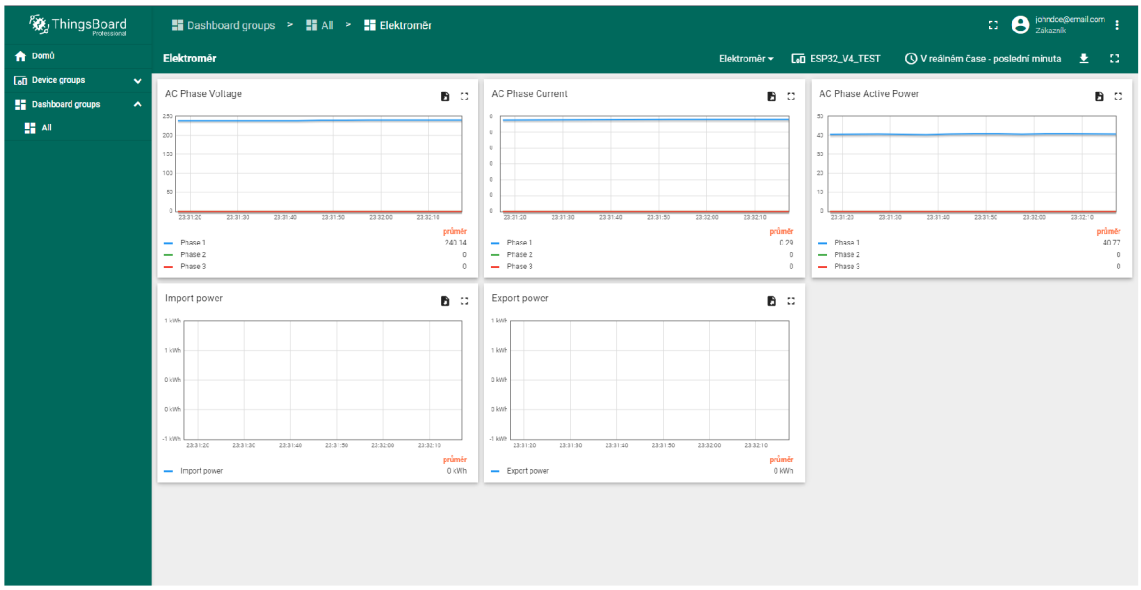
3.2.4 Databáze

Pro ukládání dat v databázi lze využít SQL, NoSQL nebo jejich kombinaci. Tvůrci nástroje ThingsBoard doporučují použití PostgreSQL pro SQL řešení a Apache Cassandra jako NoSQL databázi.[32]
V aktuálním řešení je použito hybridní řešení a propojení SQL i NoSQL databází, přičemž PostgreSQL zastupující SQL databázi uchovává data o připojených zařízeních a Cassandra jako zástupce NoSQL řešení ukládá data vyčtená z připojených zařízení.

3.2.5 ThingsBoard Web UI

Frontendový nástroj sloužící k ovládání systému a zobrazení uložených dat z připojených zařízení. Aplikace je poměrně komplexní a pro koncového uživatele může být velmi složitá na ovládání. Nástroj je napsán pomocí javascriptového frameworku Express.js[5]. Ukázka aplikace vzhledu aplikace je zobrazena na obrázku 3.1. Na levé straně se nachází ovládací panel, pomocí kterého se lze celou aplikací navigovat. Uprostřed je vždy zobrazená celá konkrétní stránka a v horním panelu možnosti a nastavení zobrazení dat. V konkrétním dashboardu tak lze specifikovat období a časový interval dat, jakým se mají data vyčítat z databáze a zobrazovat.

[5]https://expressjs.com/



Obr. 3.1: Dashboard aplikace ThingsBoard

3.3 Nedostatky řešení ThingsBoard

Řešení ThingsBoard je velmi komplexní aplikace skládající se z mnoha ovládacích prvků, nastavení a možností, které může uživatel využít. Z pohledu správce se aplikace stává mocným nástrojem pro monitoring a automatizaci chytré domácnosti. Pro koncového uživatele, kterého zajímají pouze výsledné hodnoty zobrazené v grafech však právě komplexnost a rozsáhlost aplikace může být největší překážkou v jejím každodenním používání.

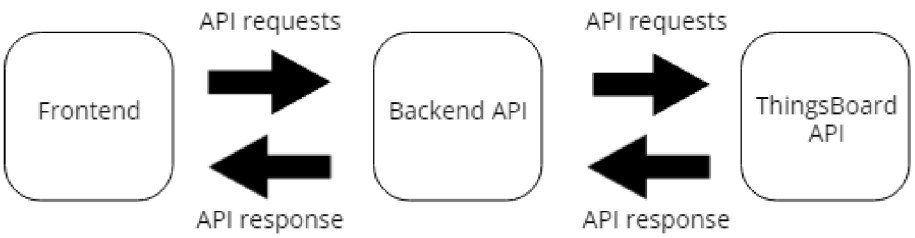
4 Programové řešení práce

Kapitola zabývající se implementací řešení bakalářské práce, v rámci které byla vytvořena webová aplikaci, která po úspěšné autentizaci uživatele umožňuje zobrazit data vyčtená z různých prvků chytré domácnosti.

Webová aplikace byla naprogramována pomocí kombinace javascriptového frameworku React.js určeného pro tvorbu frontendu a python frameworku Django, který vytváří serverovou část praktické části bakalářské práce.

4.1 Struktura aplikace

Django projekt tvořící praktickou část bakalářské práce obsahuje dvě Django aplikace. První z nich je aplikace frontend, obsahující soubor `index.html` odkud je veškerá funkcionalita přebrána frameworkem React.js. Druhá aplikace nazvaná `api` vytváří backend aplikace. Dále aplikace využívá také externí API, pomocí kterého se připojuje do systému Thingsboard a stahuje z něj data. Frontend posílá požadavky na REST API, které je součástí backendu webové aplikace, to pak požadavek zpracuje a přepošle jej dále na externí API. Vrácena data poté backend zpracuje a pošle jako odpověď na originální požadavek zpět frontendu. Komunikace a tok dat probíhající v rámci běhu aplikace jsou zobrazeny na obrázku 4.1.



Obr. 4.1: Zobrazení komunikace mezi prvky aplikace

Aplikace pracuje se dvěma úrovněmi práv `CUSTOMER_USER` a `TENANT_ADMIN`. Tyto role jsou přebrány ze systému Thingsboard. Při přihlášení do aplikace je na API odeslán dotaz, který si vyžádá aktuální informace o přihlášeném uživateli. Součástí odpovědi vrácené externím API je i informace o aktuálním oprávnění uživatele. Tato informace je uložena do React `useContext`, což je část paměti frontendové části aplikace, která umožňuje předávat data mezi komponenty, které jsou tomuto contextu podřazeny. Ukázka kódu 4.1 ukazuje, jak vytvořený `<AuthContext>` předává v parametru informaci o oprávnění uživatele. Pokud má oprávnění administrátora, bude schopný přistoupit i do modulů pro správu uživatelů a zařízení, kam se naopak normální uživatel aplikace nedostane.

Výpis 4.1: Ukázka routeru aplikace

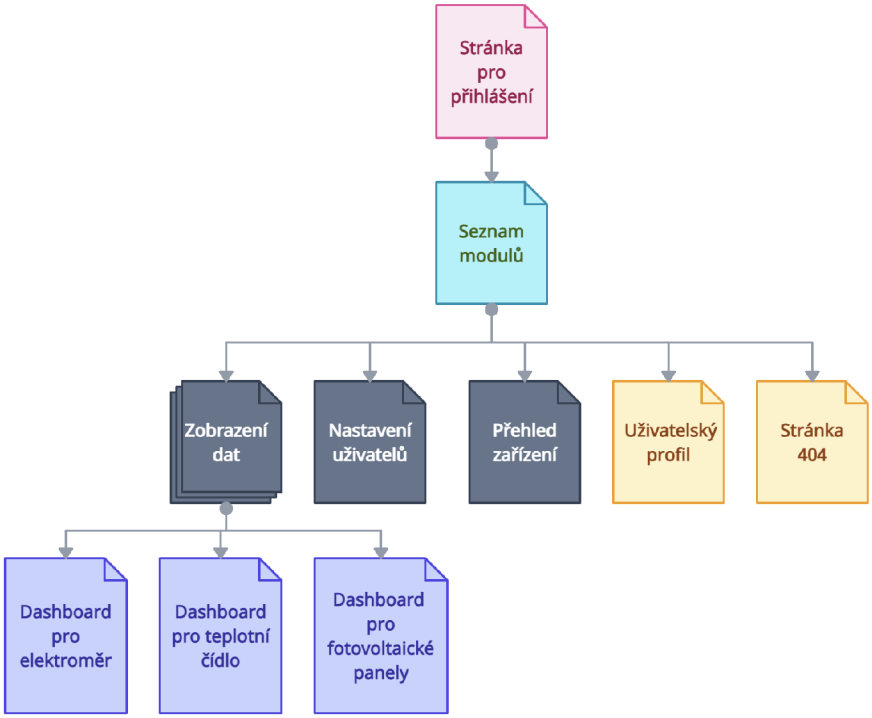
```
<BrowserRouter basename="/">
    <AuthContext.Consumer>
        {({isAdmin}) => <Routes>
            {Auth.isAuthenticated() && (
            <>
                isAdmin && (<>
                    <Route
                        path='/ListDevices'
                        element={<ListDevices />}/>
                    <Route
                        path='/Users'
                        element={<ListUsers />}/>
                    </>)}
            <Route
                path='/'
                element={<ModuleList />}/>
            <Route
                path='/Devices'
                element={<DeviceDashboard />}/>
            <Route
                path='/Device/:devicetype/:id'
                element={<Device_General />}/>
            <Route
                path='/Profile'
                element={<CurrentUserProfile/>}/>
            <Route
                path="*"
                element={<NotFoundPage/>}/>
            </>)}
            <Route
                path='/login'
                element={<LoginPage />} />
            <Route
                path="*"
                element={<Navigate to="/login"/>}/>
        </Routes>
        }
     </AuthContext.Consumer>
</BrowserRouter>
```

4.2 Frontendová část aplikace

Část aplikace, kterou vidí uživatel, může z ní číst a skrze ní ovládat zbytek systému. Posílá požadavky na API backendu a přijímá zpět data, které jsou dále zpracovány a zobrazeny. Grafická část aplikace je vytvořená pomocí React JSX, které spolu kombinuje prvky HTML doplněné o javascript kód.

4.2.1 Návrh uživatelského rozhraní

Po spuštění aplikace je po uživateli vyžadováno přihlášení, bez kterého se nedostane dál. Aplikace je dále rozdělena do modulů, které jsou načteny z JSON souboru uchovaného ve zdrojové složce kódu. Každý záznam tohoto souboru uchovává informaci o názvu modulu, URL odkaz pro `react-router`, úroveň práv potřebnou pro jeho zobrazení a odkaz na ikonu vystihující konkrétní modul. Struktura uživatelského rozhraní aplikace je představena na obrázku 4.2.



Obr. 4.2: Struktura uživatelského rozhraní aplikace

4.2.2 Využité frontendové knihovny

Při implementaci řešení byly použity knihovny zjednodušující práci s vytvářením grafických prvků a zobrazováním dat v podobě grafů. Knihovny byly instalovány a přidány do projektu pomocí balíčkovacího programu `npm`[1]. Ukázka instalace balíčku pomocí nástroje `npm`:

```
$ npm install recharts
```

Material UI

Knihovna[2] obsahující předdefinované React.js komponenty, které jsou uživatelsky přívětivé a zároveň lehce využitelné vývojáři při vytváření aplikací. Design se řídí pravidly definovanými podle `Material Design`[3] od Googlu[33]. Obsahuje naprostou většinu základních komponentů, které by mohl vývojář potřebovat pro tvorbu jednoduchých webových aplikací. Veškerý kód je dostupný jako open-source projekt vydaný pod MIT licencí.[34]

Recharts

Knihovna[4] umožňující uživateli vytvářet grafy v podobě komponentů pro React.js a dále je využívat pro zobrazení dat v aplikacích. Knihovna obsahuje různé typy grafů v podobě komponentů, které lze dále upravovat a personalizovat tak, aby výsledný graf vypadal podle potřeb aplikace. Veškerý kód je dostupný jako open-source projekt vydaný pod MIT licencí.[35]

json-to-csv-export

Rozšíření[5] do jazyka Javascript, které umožňuje jednoduchý převod dat do formátu CSV - Comma separated value a jejich následné stažení. Krom dat samotných přijímá vstupní příkaz také volitelné parametry, které umožňují vytvořit vlastní název výstupního souboru, přidat vlastní hlavičku dat a možnost změnit `Delimetr` - rozdělovač dat.

[1] https://www.npmjs.com/
[2] https://mui.com/
[3] https://m3.material.io/
[4] https://recharts.org/
[5] https://www.npmjs.com/package/json-to-csv-export

4.3 Backendová část aplikace

Backendová část aplikace slouží jako brána mezi frontendovou částí a externím API. Obsahuje vlastní API, na které jsou posílány ze strany frontendu požadavky, které jsou zpracovány, upraveny a odeslány dále na externí API. Jakmile se vrátí odpověď od API, bude zabalena do odpovědi originálního požadavku, který přišel z uživatelské části aplikace.

4.3.1 Použité backendové knihovny

Instalace a správa knihoven je v jazyce python zajištěna pomocí balíčkovacího nástroje `pip`[6] následovně:

```
pip install djangorestframework
```

Django rest framework

Knihovna[7] rozšiřující možnosti a schopnosti frameworku django. Ulehčuje práci s vytvářením Web API, sloužící ke komunikaci s frontendovou částí aplikace. Knihovna poskytuje také nástroje pro práci s daty a jejich serializací do formátu JSON nebo XML.

requests

Knihovna `requests`[8] je nástroj pro vytváření HTTP požadavků v jazyce Python, umožňující snadnou komunikaci s API. Novému požadavku lze lehce specifikovat v parametrech funkce věci jako hlavičky dotazu, parametry URL, data jako tělo požadavku, nebo informace o autorizaci uživatele. Odpověď je potom vrácena ve formátu JSON, což umožňuje jejich další snadné zpracování.

[6] https://pypi.org/project/pip/
[7] https://www.django-rest-framework.org/
[8] https://requests.readthedocs.io/en/latest/

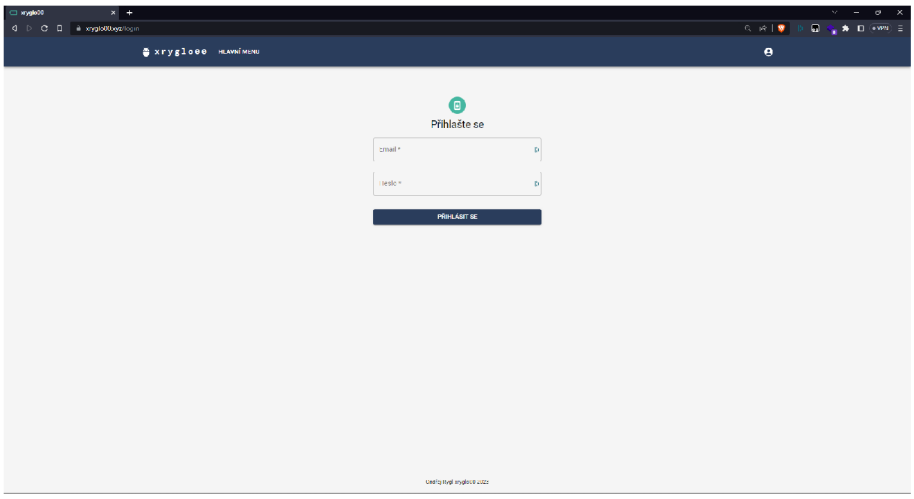
4.3.2 Struktura aplikace backendu

Souborová struktura aplikace vytvářející backend využívá kromě předdefinovaných souborů aplikace Django především následující tři soubory:

1. `urls.py` - Soubor obsahující URI s endpointy, na které frontend posílá požadavky. Je zde obsažen seznam s `urlpatterns`, který definuje názvy jednotlivých endpointů API spolu s odkazem na konkrétní `APIView`.
2. `views.py` - V tomto souboru jsou definovány třídy typu `APIView`, ve kterých jsou definovány metody vyplývající z definice REST API. Jsou zde tedy konkrétní metody `.get()` nebo `.post()`, které řeší odpovídající požadavky zaslané na daný endpoint.
3. `util.py` - Zde jsou obsaženy pomocné funkce, které primárně slouží k odesílání zpracovaných požadavků na externí API a jejich následné zpracování do podoby, ve které jsou odeslány zpět frontendu.

4.4 Vzhled a práce s aplikací

Po přístupu na stránku bude uživatel vyzván k přihlášení pomocí přihlašovacích údajů. Pokud zadá správné přihlašovací údaje, bude přesměrován na stránku s přehledem všech jemu dostupných modulů. Pokud jsou zadány špatné přihlašovací údaje, na stránce se objeví chybová hláška a uživatel nebude moci pokračovat, dokud nezadá přihlašovací údaje správně. Na obrázku 4.10 je zobrazen přihlašovací dialog.



Obr. 4.3: Přihlašovací dialog aplikace

Po úspěšném přihlášení uživatele se jeho JWT token uloží do tzv. `session storage` prohlížeče, kde zůstane dokud nebude okno, ve kterém aplikace běží, zavřeno. Z důvodu potřeby práce s přihlašovacím jménem uživatele je i to ukládáno stejným způsobem do `session storage` prohlížeče. Krom JWT a přihlašovacího jména je uloženo také ID zákazníka, kterému patří tento uživatel.

Jak je uvedeného kódu 4.2 jasné, ke smazání JWT tokenu dojde také při odhlášení uživatele z aplikace, který následně bude odkázán na přihlašovací stránku.

Po úspěšném přihlášení bude uživatel přesměrován na stránku, která dynamicky zobrazí všechny jemu dostupné moduly aplikace. Pro uživatele s oprávněním `CUSTOMER_USER` se zobrazí pouze modul pro zobrazování dat. Pro uživatele s oprávněním `TENANT_ADMIN` budou zobrazeny i moduly pro správu uživatelů a modul pro správu zařízení.

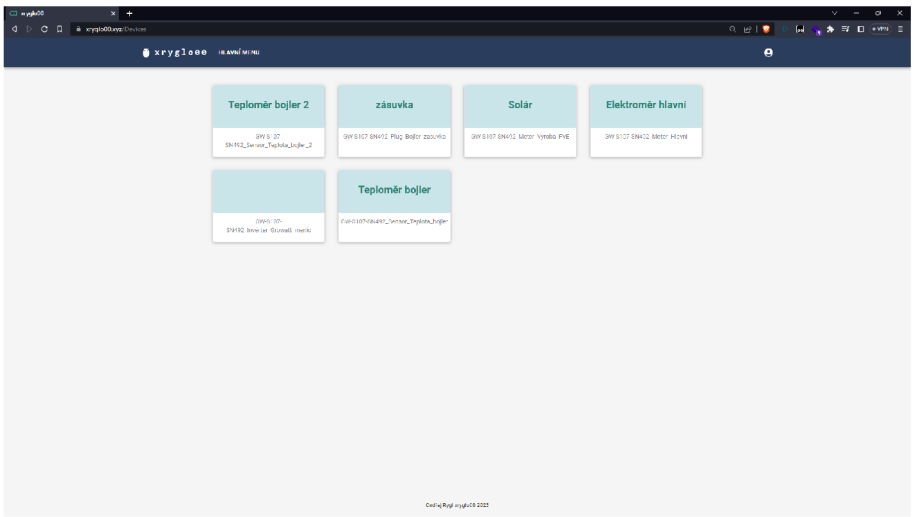
Výpis 4.2: Způsob ukládání JWT tokenu v do paměti prohlížeče

```
const AUTH_TOKEN_NAME = "JWT"

class Auth {
    static logout = async () => {
      sessionStorage.removeItem(AUTH_TOKEN_NAME)
      window.location.href = "/login";
    }
    static setJwt = async (jwt) => {
        sessionStorage.setItem(AUTH_TOKEN_NAME, jwt)
    }
    static getJwt = () => {
      return sessionStorage.getItem(AUTH_TOKEN_NAME)
    }
    static isAuthenticated = () => Boolean(Auth.getJwt())
  }
```

Modul Zobrazení dat

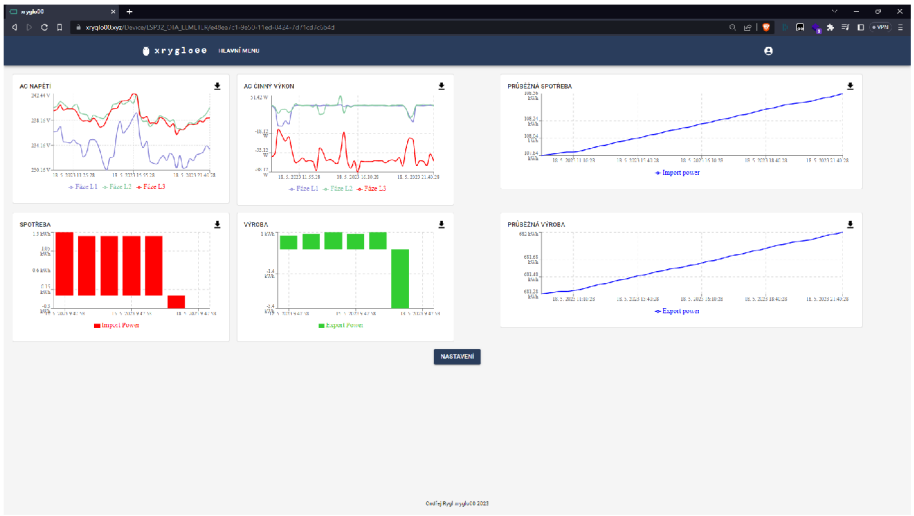
Otevření modulu je doprovázeno dotazem na API, který si vyžádá seznam všech zařízení, které jsou dostupná pro uživatele. Odpověď se zpracuje a z jejího obsahu se dynamicky vytvoří karta pro každé zařízení, které je uživateli dostupné.



Obr. 4.4: Dynamické zobrazení dostupných zařízení.

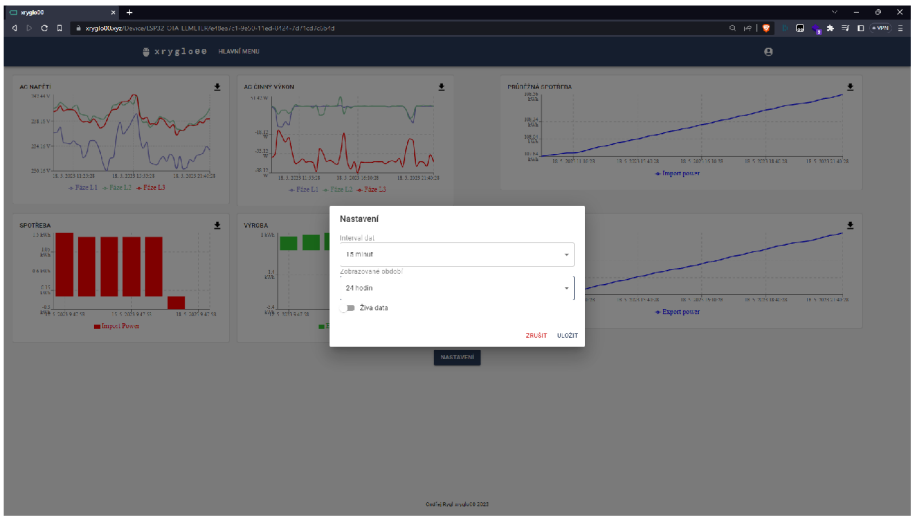
Každá karta je nadepsána uživatelským označením zařízení a názvem, pod kterým ho eviduje systém. Karta také obsahuje URL s informacemi o typu a ID zařízení. Po kliknutí na zvolenou kartu, komponent `<Device_General>` na základě parametru URL adresy vyhodnotí typ zařízení a podle něj zobrazí komponent obsahující dashboard s grafy pro daný typ zařízení. Z druhého parametru v odkazu potom vyčte ID zařízení, jenž při načítání dat bude odesláno jako parametr dotazu na odpovídající endpoint API, který vrátí konkrétní data. Při otevření stránky zařízení bude podle počtu předdefinovaných grafů zařízení odeslán odpovídající počet dotazů na data, které je vyplní.

Data z každého grafu lze exportovat ve formátu CSV, tedy Comma separated value. Po kliknutí na tlačítko stažení, které se nachází v pravém horním rohu každého z grafů se objeví prohlížeč souborů, umožňující uživateli zvolit umístnění, kam se soubor stáhne a možnost definovat jeho jméno. Data jsou vždy rozdělena pomocí symbolu středníku a obsahují názvy hlaviček sloupců podle názvů os v grafu.



Obr. 4.5: Zobrazení dat z elektroměru v grafech.

Každý typ zařízení má předdefinované grafy, které zobrazují různá data. Na obrázku 4.5 je vyobrazen komponent `<Device_Gateway>` vykreslující dashboard s grafy pro elektroměr. Pod těmito grafy se vždy nachází tlačítko `Nastavení`, které po stisknutí otevře vyskakovací okno ukázané na obrázku 4.6. Zle lze nastavit zobrazované období a interval po kterém budou data rozdělena. Nastavení také obsahuje možnost zobrazit tzv. živá data. Při zapnutí této funkce, bude v pozadí aplikace zapnut časovač, který každých pět vteřin spustí funkci odesílající nový dotaz na API s žádostí o nová data, která budou okamžitě zobrazena.



Obr. 4.6: Nastavení zobrazení dat v grafech.

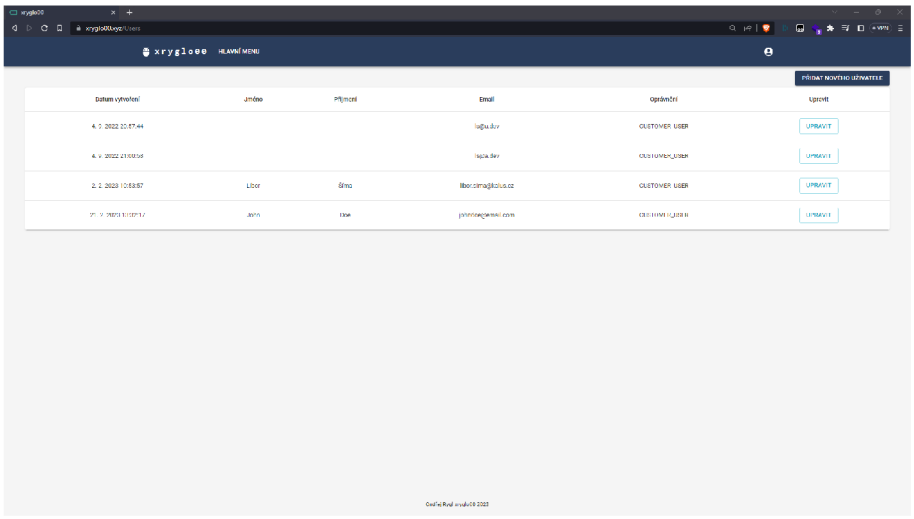
V momentě kliknutí na zvolenou kartu zařízení se odešle dotaz na API, který zkontroluje, zda je zařízení aktuálně připojeno do systému a je z něj možné vyčítat data. Jestliže připojeno je, systém pokračuje dál a zobrazí grafy. Pokud však připojeno není, uživateli se zobrazující komponent `<DeviceDisconected>`, který ho informuje o odpojeném stavu zařízení a odkáže jej zpět na hlavní stránku.

Jelikož je aplikace dynamická a není vázaná jen na jednoho zákazníka a jemu přidělené zařízení, může se stát, že uživatel bude mít přiřazené zařízení, které není systémem podporováno. V momentě, kdy se uživatel pokusí otevřít kartu takového zařízení, bude mu vyobrazen komponent `<Device_not_implemented>`, který jej o tomto faktu informuje a nabídne mu možnost vrátit se zpět na hlavní stranu, odkud může pokračovat dále do aplikace.

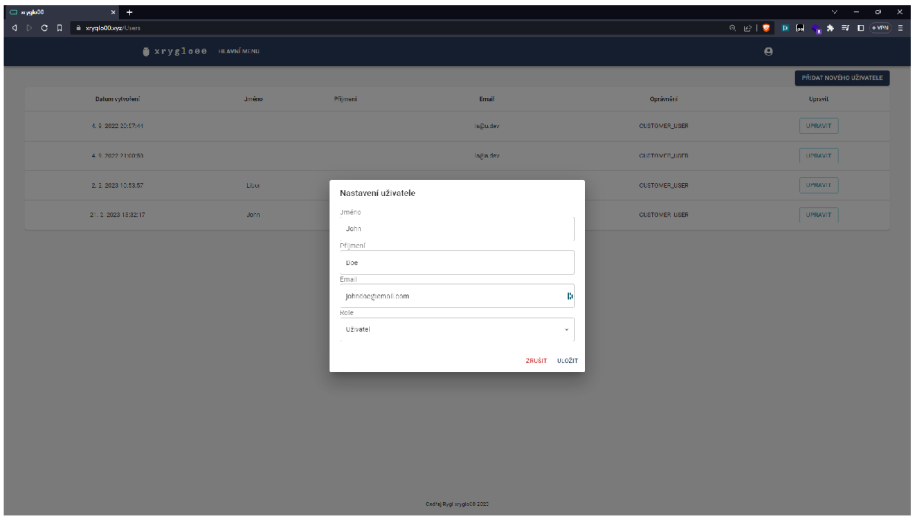
Modul Nastavení uživatelů

Při spuštění modulu je odeslán dotaz na externí API, který na základě JWT tokenu a ID zákazníka, které jsou od přihlášení do aplikace uložené v `session storage` prohlížeče vyžádá seznam všech uživatelů, které systém eviduje pod odeslaném ID zákazníka. Odpověď v podobě seznamu ve formátu JSON je vyobrazena jako tabulka, obsahující informace jako jméno, příjmení, email uživatele a jeho oprávněni. Každý záznam také obsahuje tlačítko umožňující upravit všechny výše uvedené informace. Po jeho stisknutí se otevře vyskakovací okno, které obsahuje formulář s předvyplněnými údaji daného uživatele. Tyto údaje lze změnit. Po kliknutí na tlačítko uložit bude odeslán POST dotaz obsahující aktualizované údaje uživatele.

Uživatel s oprávněním `TENANT` bude moci upravit informace o konkrétním uživateli. Je také možné změnit jeho oprávnění.



Obr. 4.7: Modul nastavení uživatelů

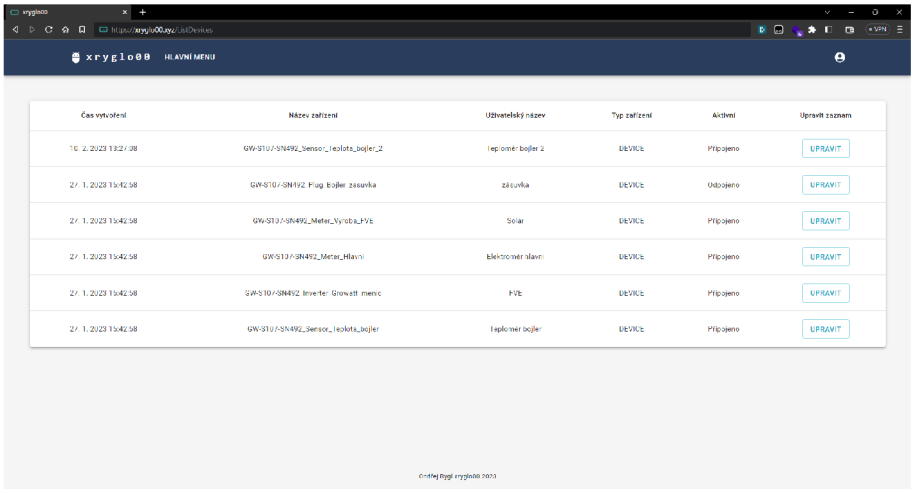


Obr. 4.8: Úprava uživatele

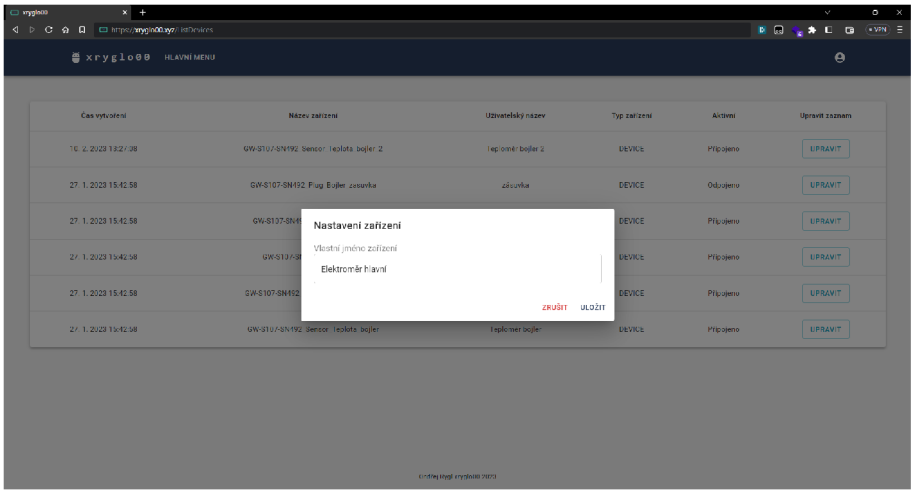
Modul Přehled zařízení

Modul s přehledem zařízení slouží pro administrátora konkrétního zákazníka - uživatele s oprávněním `TENANT`. Načtení dat je vyvoláno hned při otevření modulu, kdy se zavolá GET dotaz na API, který ve svém parametru obsahuje JWT token jako autorizaci dotazu a ID zákazníka. Obě informace vyčte ze `session storage` prohlížeče. API vrátí jako odpověď všechny zařízení, které jsou registrovány pod konkrétním zákazníkem. Data jsou zobrazena jako tabulka obsahující informace o datu přidání konkrétního zařízení do systému, názvu zařízení, označení zařízení, typu zařízení a informaci, zda je zařízení aktuálně připojeno nebo odpojeno. V neposlední řadě je zde také přidána možnost upravit označení tohoto zařízení. To slouží primárně jako

možnost pro administrátora definovat vlastní název zařízení. Při kliknutí na tlačítko upravit se otevře nové okno, které vyžaduje po uživateli zadat jeho označení konkrétního zařízení. Po kliknutí na uložit se okno zavře a na API se odešle POST dotaz, který obsahuje aktualizované zařízení doplněné o nový uživatelský název.



Obr. 4.9: Modul přehled zařízení



Obr. 4.10: Úprava názvu zařízení

Uživatelský profil

Ikonka uživatele na hlavním panelu umožňuje kromě možnosti odhlášení také přesměrování na profil aktuálně přihlášeného uživatele. Po přesměrování na tuto stránku se uživateli zobrazí komponent `<CurrentUserProfile>` obsahující informace o jeho jméně a přihlašovacím emailu. Dále bude mít možnost změnit si své přihlašovací heslo. Tato funkce vyžaduje zadání jeho aktuálního hesla, zadání nové hesla a jeho ověření, aby měl uživatel jistotu, že ho zadal opravdu správně. Pokud bude ověření rozdílného od obsahu políčka s novým heslem, políčko zčervená, čímž bude uživatel upozorněn na tuto nesrovnalost. Hesla je možné zobrazit pomocí tlačítka na konci zadávacího pole. Jakmile jsou pole pro nové heslo a jeho ověření vyplněny správně, odemče se tlačítko, jehož stisknutím bude odeslán příkaz na API. Pokud je změna v pořádku, heslo bude změněno. Přihlášení včetně JWT tokenu zůstane platné a nové heslo bude po uživateli vyžadováno při jeho dalším přihlášení. Pokud je původní heslo při odeslání chybné, bude uživatel pomocí vyskakovací hlášky upozorněn o selhání při pokusu o změnu hesla.

Stránka 404

HTTP chyba s označením `404 - Not found` je chybou na straně klienta. Nastává z pravidla v momentě, kdy se uživatel snaží přistoupit na stránku, které neexistuje. V řešení práce na ni server odkáže pokaždé, když se uživatel pokusí vstoupit na stránku, která není definovaná v nastavení `react-router` ukázaném na výpisu 4.1. Komponent `<NotFoundPage>` zobrazený uživateli při této chybě obsahuje informaci o tom, že takováto stránka nebyla nalezena a odkaz zpět na hlavní menu se seznamem dostupných modulů.

4.5 Testování aplikace

Praktická část bakalářské práce v podobě webové aplikace byla pro testovací účely spuštěna na zařízení `Raspberry Pi 4b` pomocí HTTP serveru `Apache 2`[9], odkud byla s využitím služby `Zero trust tunnell` od firmy Cloudflare otevřena do internetu. Pro účely testování funkčnosti webové aplikace byla zakoupena doména `xryglo00.xyz`.

4.5.1 Cloudflare

Americká firma Cloudflare[10] celosvětová firma nabízející služby umožňující rychlé, zabezpečené a spolehlivé připojení aplikací svých zákazníků do internetu. Firemním cílem je tvořit lepší internet. Mezi nejpopulárnější produkty nabízené firmou Cloudflare patří například webový firewall, který dokáže monitorovat a blokovat pokusy o `DDOS útoky`, `SQL injekce`, `Crossite scripting` a dalšími útoky vycházející například z `OWASP Top 10`. Jeho umělá inteligence dokáže chránit webové aplikace také před nově objevenými `zero days` útoky.

Zero Trust Tunnel

Nástroj `Zero trust` je bezpečnostní řešení, které nahrazuje potřebu klasické VPN. Tato služba umožňuje nastavení přístupu do vnitřní sítě například pomocí `Zero trust tunnel`. Ten umožňuje vytvořit zabezpečenou cestou, kterou lze připojit služby z vlastní sítě na servery Cloudflare, odkud jsou dostupné z celého internetu a to bez potřeby vlastnit veřejnou IP adresu. [36]

V pozadí hostitelského serveru běží služba s názvem `cloudflared`, která vytváří odchozí spojení do sítě firmy Cloudflare. Touto cestou lze hostovat například webové servery nebo třeba nastavit možnost připojení do sítě přes protokoly jako je SSH, RDP nebo VNC a další.[36] Cloudflare nabízí tuto službu pro jednotlivce zdarma, Tento způsob hostování například webové aplikace šetří náklady na pronájem komerčního VPS serveru, nebo na náklady spojené s koupí či pronájmem veřejné IP adresy.

K účelům testování aplikace vytvořené v rámci bakalářské práce byl pomocí zero trust vytvořen tunel, který generuje šifrované spojení mezi vnitřní domácí sítí autora a servery firmy Cloudflare, což umožňuje hostovat server na kterém běží webová aplikace v domácí síti a zároveň jej bezpečně pouštět mimo tuto síť do internetu.

[9] https://httpd.apache.org/
[10] https://www.cloudflare.com/

Závěr

Tato bakalářská práce se zabývala vytvořením webové aplikace čerpající data z externího API, která zvládne zpracovat a dynamicky zobrazit.

V teoretické části se jsou představeny jednotlivé programovací jazyky a jejich součásti, pomocí kterých byla webová práce vyhotovena. Dále se zde nachází podkapitola vysvětlující pojem autentizace společně s ukázkou různých druhů autentizace pro čerpání dat z externích zdrojů. Teoretickou část práce ukončuje podkapitola popisující zabezpečení webové aplikace a představuje nejčastější bezpečnostní zranitelnosti webových aplikací. Druhá kapitola byla věnována představení šesti již existujících řešení, které se zabývají podobnou problematikou. Třetí kapitola popisuje aktuální backendové řešení aplikace, které zajišťuje vyčítání dat z prvků chytré domácností, jejich uchovávání v databázích a následné zpřístupnění pomocí API. Ve čtvrté kapitole byl popsán návrh a vzhled vytvořené webové aplikace. Ta po přihlášení zobrazí uživateli jemu dostupné moduly aplikace. Aplikace dynamicky zobrazí uživatelům jejich dostupná zařízení, jejichž data je možné prohlédnout. každý uživatel má také možnost exportu jeho aktuálně zobrazených dat v grafu. Uživatel přihlášený pomocí účtu s oprávněním administrátora má navíc možnost spravovat jemu svěřené uživatele a lehce upravovat zařízení přidané pro jeho skupinu. V této kapitole jsou také popsány knihovny a další nástroje, pomocí kterých byla aplikace vytvořena a otestována. Testování jako takovému je věnován konec této části práce.

V rámci dalšího rozšíření aplikace by bylo vhodné přidat více funkcí administrátora, například pro přidávání nových zařízení do systému, nebo pro vytváření uživatelských skupin. dalším vylepšením by mohl být modul, ze kterého by se daly zařízení jako taková ovládat, jelikož aktuálně proudí data pouze jedním směrem a to ze zařízení do aplikace. Dále je třeba vytvořit dashboardy pro zobrazení dat z dalších zařízení, jelikož aktuálně jsou podporovány pouze elektroměry, teplotní čidla a fotovoltaické panely.

Literatura

[1] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Irvine, 2000. Dostupné také z: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.DisertaÄDnÃŋprÃące.UniversityofCalifornia.>

[2] BERNERS-LEE, T, R FIELDING a L MISINTER. *Uniform Resource Identifier (URI): Generic Syntax* [online]. 2005 [cit. 2022-12-06]. Dostupné z: <https://www.rfc-editor.org/rfc/pdfrfc/rfc3986.txt.pdf>

[3] *Introducing JSON* [online]. 2022 [cit. 2022-12-12]. Dostupné z: <https://www.json.org/>

[4] *YAML: YAML Ain't Markup Language* [online]. 2022 [cit. 2022-12-12]. Dostupné z: <https://yaml.org/>

[5] *What is Python? Executive Summary* [online]. 2022 [cit. 2022-12-06]. Dostupné z: <https://www.python.org/doc/essays/blurb>

[6] *Documentation: Django at a glance* [online]. 2022 [cit. 2022-12-12]. Dostupné z: <https://docs.djangoproject.com/en/4.1/intro/overview/#django-at-a-glance>

[7] *FAQ: General: Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don-t use the standard names?* [online]. 2022 [cit. 2022-12-06]. Dostupné z: <https://docs.djangoproject.com/en/4.1/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>

[8] *HTML: HyperText Markup Language* [online]. MDN contributors, 2022 [cit. 2022-12-07]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTML>

[9] RAVICHANDRAN, Adhiti. *React Virtual DOM Explained in Simple English* [online]. In: . 2018 [cit. 2022-12-07]. Dostupné z: <https://programmingwithmosh.com/react/react-virtual-dom-explained/>

[10] *Introducing JSX* [online]. California, USA: Meta Platforms, 2022 [cit. 2022-12-07]. Dostupné z: <https://reactjs.org/docs/introducing-jsx.html>

[11] *CHANGELOG: 16.8.0 (February 6, 2019)* [online]. California, USA: Meta Platforms, 2019 [cit. 2022-12-07]. Dostupné z: <https://github.com/facebook/react/blob/main/CHANGELOG.md#1680-february-6-2019>

[12] *Classes* [online]. MDN contributors, 2022 [cit. 2022-12-07]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes>

[13] *OWASP Cheat Sheet Series: Authentication Cheat Sheet* [online]. 2021 [cit. 2022-12-07]. Dostupné z: <https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html>

[14] *JWT: Introduction to JSON Web Tokens* [online]. 2022 [cit. 2022-12-07]. Dostupné z: <https://jwt.io/introduction>

[15] *Swagger: OAuth 2.0* [online]. SmartBear Software, 2022 [cit. 2022-12-07]. Dostupné z: <https://swagger.io/docs/specification/authentication/oauth2/>

[16] What is SSL? | SSL definition. *Cloudflare* [online]. Cloudflare, 2023 [cit. 2023-05-23]. Dostupné z: <https://www.cloudflare.com/learning/ssl/what-is-ssl/>

[17] What is TLS (Transport Layer Security)?. *Cloudflare* [online]. Cloudflare, 2023 [cit. 2023-05-23]. Dostupné z: <https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/>

[18] CARSON. TLS 1.2 and TLS 1.3 Handshake Walkthrough. *Medium* [online]. 18 Mar 2021 [cit. 2023-05-23]. Dostupné z: <https://cabulous.medium.com/tls-1-2-andtls-1-3-handshake-walkthrough-4cfd0a798164>

[19] OWASP Top 10:2021: Welcome to the OWASP Top 10 - 2021. *OWASP* [online]. [cit. 2023-05-23]. Dostupné z: <https://owasp.org/Top10/>

[20] *Customization: Complete solution* [online]. Unipi technology, 2022 [cit. 2022-12-11]. Dostupné z: <https://www.unipi.technology/customization/complete-solution-415>

[21] *ICool4: Komponenty* [online]. ICT EXPERT, 2022 [cit. 2022-12-11]. Dostupné z: <https://www.icool4.cz/produkty/>

[22] *Charging solutions* [online]. AgeVolt Slovakia, 2022 [cit. 2022-12-11]. Dostupné z: <https://www.agevolt.com/charging-solutions>

[23] DAVYDOV, Michael. HomeAssistant for newcomers:: What it is, what is hassio, hassos, hassbian, 101 and cookies. In: *Home Assistant* [online]. 22 June 2019 [cit. 2022-12-12]. Dostupné z: <https://community.home-assistant.io/t/homeassistant-for-newcomers-what-it-is-what-is-hassio-hassos-hassbian-101-and-cookies/123004>

[24] *Home Assistant: Integrations* [online]. Home Assistant, 2022 [cit. 2022-12-11]. Dostupné z: <https://www.home-assistant.io/integrations/#all>

[25] *OpenHAB: empowering the smart home* [online]. openHAB Community, 2022 [cit. 2022-12-11]. Dostupné z: <https://www.openhab.org/docs/>

[26] *ETSI EN 300 220-1 V3.1.1 (2017-02): Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz;* [online]. Francie, Feb 2022 [cit. 2022-12-12]. Dostupné z: <https://www.etsi.org/deliver/etsi_en/300220_300299/30022001/03.01.01_60/en_30022001v030101p.pdf>

[27] *Loxone: Loxone Config* [online]. Loxone Electronics, 2022 [cit. 2022-12-12]. Dostupné z: <https://www.loxone.com/cscz/produkty/loxone-config/>

[28] *Loxone: Loxone App* [online]. Loxone Electronics, 2022 [cit. 2022-12-12]. Dostupné z: <https://www.loxone.com/cscz/produkty/aplikace/>

[29] *ThingsBoard: ThingsBoard Community Edition* [online]. The ThingsBoard Authors, 2022 [cit. 2022-12-12]. Dostupné z: <https://thingsboard.io/docs/>

[30] *ThingsBoard Node* [online]. The ThingsBoard Authors, 2022 [cit. 2022-12-11]. Dostupné z: <https://thingsboard.io/docs/reference/msa/#thingsboard-node>

[31] *IOT Gateway: What is ThingsBoard IoT Gateway?* [online]. The ThingsBoard Authors, 2022 [cit. 2022-12-11]. Dostupné z: <https://thingsboard.io/docs/iot-gateway/what-is-iot-gateway/>

[32] *ThingsBoard architecture: SQL vs NoSQL vs Hybrid database approach* [online]. The ThingsBoard Authors, 2022 [cit. 2022-12-11]. Dostupné z: <https://thingsboard.io/docs/reference/#sql-vs-nosql-vs-hybrid-database-approach>

[33] *MATERIAL DESIGN* [online]. Google, 2022 [cit. 2022-12-11]. Dostupné z: <https://m2.material.io/>

[34] *Material UI - Overview* [online]. Material UI, 2022 [cit. 2022-12-11]. Dostupné z: <https://mui.com/material-ui/getting-started/overview/>

[35] Recharts. <Recharts /> [online]. Recharts Group, 2021 [cit. 2022-12-11]. Dostupné z: <https://recharts.org/>

[36] Cloudflare Tunnel. *Cloudflare Docs* [online]. 2023 [cit. 2023-05-16]. Dostupné z: https://developers.cloudflare.com/cloudflare-one/connections/connect-apps/

A Ukázka XML, JSON a YAML formátu

Výpis A.1: Příklad XML zápisu

```xml
<?xml version="1.0"?>
<SeznamOsob>
    <Osoba id=1>
        <Jmeno>Petr</Jmeno>
        <Prijmeni>Mráz</Prijmeni>
        <Vek>35</Vek>
    </Osoba>
    <Osoba id=2>
        <Jmeno>Marie</Jmeno>
        <Prijmeni>Ružičková</Prijmeni>
        <Vek>47</Vek>
    </Osoba>
</SeznamOsob>
```

Výpis A.2: Příklad JSON zápisu

```json
{
"SeznamOsob": {
    "Osoba": [
        {
        "id": "1",
        "Jmeno": "Petr"
        "Prijmeni": "Mráz"
        "Vek": "35"
        }
        {
        "id": "2",
        "Jmeno": "Marie"
        "Prijmeni": "Ružičková"
        "Vek": "47"
        }
    ]
}
}
```

Výpis A.3: Příklad YAML zápisu

```
# Seznam Osob
- Osoba 1:
    Jmeno: Petr
    Prijmeni: Mráz
    Vek: 35
- Osoba 2:
    Jmeno: Marie
    Prijmeni: Růžičková
    Vek: 47
```