Aplikace pro správu spotřeby energií

Stanislav Greschner

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Stanislav Greschner**
Osobní číslo:	**A21327**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Aplikace pro správu spotřeby energií**
Téma práce anglicky:	**Application for Energy Management**

Zásady pro vypracování

1. Prostudujte způsoby účtování energií a proveďte rešerši aplikací zabývajících se spotřebou energií.
2. Popište vhodné technologie pro tvorbu webových aplikací.
3. Definujte požadavky na aplikaci pro správu energií.
4. Navrhněte, realizujte a otestuje aplikaci s využitím vhodných technologií.
5. Věnujte se zabezpečení aplikace.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PILGRIM, Mark. Ponořme se do HTML5. 1. vydání. Brno: Jan Melvil Publishing, 2015. ISBN 978-80-905802-6-8.
2. KING, Todd; THRONTVEIT, Dean. Power Monitoring in the Data Center. Boca Raton: CRC Press, 2012. ISBN 978--1439879092.
3. ALLEN, Jason L. a ROBBINS, Scott. Building Cross-Platform Apps using Titanium, Alloy, and Appcelerator Cloud Services. Indianapolis: Wiley, 2014. ISBN 978-1118673256.
4. FREEMAN, Adam. Pro ASP.NET Core MVC 2. 7. vydání. New York: Apress, 2017. ISBN 978-1484231494.
5. KLEIN, John M. Energy Management and Conservation Handbook. 2. vydání. Boca Raton: CRC Press, 2016. ISBN 978--1498733053.

Vedoucí bakalářské práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.
- že při tvorbě této práce jsem použila nástroj generativního modelu AI [ChatGPT; www.chat.openai.com] za účelem zkvalitnění textu a pomoci při stylování aplikace. Po použití tohoto nástroje jsem provedla kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne 10. 5. 2024 Stanislav Greschner v. r.

ABSTRAKT

Cílem této bakalářské práce je vyvinout aplikaci pro správu spotřeby elektrické energie, která je nezávislá na konkrétním dodavateli a umožňuje uživatelům vkládat vlastní tarifní parametry a monitorovat spotřebu energie. Aplikace poskytuje analýzu spotřeby a předpovídá budoucí spotřebu. Přístup k aplikaci je možný prostřednictvím webového rozhraní a mobilních zařízení. Práce rovněž zahrnuje vytvoření uživatelského účtu pro zabezpečený přístup.

Klíčová slova: Správa spotřeby energie, elektrická energie, webové rozhraní, mobilní aplikace, analýza spotřeby

ABSTRACT

The objective of this bachelor's thesis is to develop an application for managing electricity consumption that is independent of any particular supplier and enables users to input their own tariff parameters and monitor energy consumption. The application provides consumption analysis and forecasts future consumption. Access to the application is available through a web interface and mobile devices. The thesis also includes the creation of a user account for secure access.

Keywords: Energy management, electrical energy, web interface, mobile application, consumption analysis.

Poděkování

Děkuji panu Jiřímu Vojtěškovi za jeho neocenitelnou pomoc a výjimečně ochotný přístup při práci na mé bakalářské práci. Jeho podpora a odborné vedení byly klíčové pro dosažení výsledků, kterých si velmi vážím.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **8**
I TEORETICKÁ ČÁST **9**
1 SPOTŘEBA ELEKTRICKÉ ENERGIE **10**
1.1 Základy měření spotřeby energie 10
1.2 Čtení a Interpretace Dat z Elektroměrů 12
1.3 Rozdíl mezi dodavatelem a distributorem elektřiny 13
1.4 Distributoři elektřiny 14
1.5 Největší dodavatelé elektřiny V ČR 14
1.6 Tarifní systémy a jejich struktura 15
1.7 Vyúčtování elektřiny 17
1.7.1 Přeplatek 17
1.7.2 Nedoplatek 17
1.7.3 Obchodní část vyúčtování 17
1.7.4 Distribuční část vyúčtování 17
1.7.5 Stálá platba a účtování spotřeby 18
1.7.6 Ukázka vyúčtování elektřiny 18
1.8 Aplikace pro výpočet spotřeby energie 19
1.8.1 Domácí odečty jednoduše 19
1.8.2 Odečty elektrické energie 20
1.8.3 Kalkulátor nákladů na elektřinu 20
1.8.4 PROUD ČEZ 20
1.8.5 E.ON 21
1.8.6 Přínos těchto aplikací 21
2 WEBOVÉ A MOBILNÍ APLIKACE **22**
2.1 Principy webových aplikací 22
2.2 Principy mobilních aplikací 22
2.3 Progressive Web Apps (PWA) 23
2.4 Bezpečnostní aspekty webových a mobilních aplikací 23
2.5 Přehled použitých programovacích jazyků 23
2.5.1 Typescript 23
2.5.2 HTML 24
2.5.3 SCSS 25
2.6 Frameworky a knihovny pro vývoj 26
2.6.1 Angular 26
2.6.2 Vue.js 26
2.6.3 React 28
2.7 Backend technologie a databáze 29
2.7.1 Node.js 29
2.7.2 SQL/NoSQL 30
2.8 Cloudové služby a API 31
2.9 Firebase jako platforma pro serverovou stranu a autentifikaci 31
II PRAKTICKÁ ČÁST **33**

3 ÚVOD DO APLIKACE 34
3.1 POŽADAVKY NA APLIKACI 34
3.1.1 Funkční požadavky 34
3.1.2 Nefunkční požadavky 35
3.2 IMPLEMENTACE APLIKACE 35
3.2.1 Bezpečnost aplikace 35
3.3 IMPLEMENTACE 36
3.3.1 Přihlašovací obrazovka 36
3.3.2 Registrační obrazovka 38
3.3.3 Obrazovka zapomenutého hesla 39
3.3.4 Vizualizace obnovy hesla 40
3.3.5 Obrazovka kalkulátoru 41
3.3.6 Obrazovka měsíční přehled 47
3.3.7 Obrazovka porovnání tarifů 50
3.3.8 Obrazovka nastavení 53
3.4 STRUKTURA DATABÁZE 58
3.4.1 Tariff: 58
3.4.2 Reading: 58
3.4.3 InitialState a StateData: 58
3.4.4 ComparisonResult: 58
3.4.5 Calculations: 59
3.5 VÝVOJ A TESTOVÁNÍ 59
4 FIREBASE 60
4.1 INICIALIZACE 60
4.2 PŘÍKLADNÁ UKÁZKA ZÁPISU DO DATABÁZE 62
4.3 PŘÍKLADNÉ NAČÍTANÍ DAT Z DATABÁZE 63
4.4 ZABEZPEČENÍ APLIKACE 64
ZÁVĚR 65
SEZNAM POUŽITÉ LITERATURY 66
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 68
SEZNAM OBRÁZKŮ 69
SEZNAM TABULEK 70
SEZNAM ZDROJOVÝCH KÓDŮ 71
SEZNAM PŘÍLOH 72

ÚVOD

Tato bakalářské práce se zaměřuje na vývoj aplikace pro správu a monitoring spotřeby elektřiny, což představuje aktuální a významné téma v kontextu rostoucího důrazu na energetickou efektivitu a udržitelnost. Spotřebitelé čelí obtížím v efektivním monitorování a správě své spotřeby energie, což může vést k nadměrným výdajům a zbytečnému plýtvání zdroji. Práce si klade za cíl vyvinout aplikaci, která umožní uživatelům nejen sledovat a analyzovat svou spotřebu, ale také optimalizovat její využití pomocí personalizovaných tarifních nastavení a předpovědí budoucí spotřeby.

Výzkum kombinuje teoretický přístup s praktickou implementací. Iniciační fáze zahrnuje detailní rešerši existujících technologií a aplikací v oblasti monitorování spotřeby energie. Následuje fáze návrhu, kde jsou specifikovány požadavky na aplikaci a vypracován její koncept. Implementační fáze práce se soustředí na vývoj aplikace s využitím vybraných technologií a její testování ve spolupráci s cílovou skupinou uživatelů.

Celková struktura práce je systematicky rozvržena do několika kapitol, které postupně pokrývají od teoretických základů, přes metodologii výzkumu, až po praktické aspekty vývoje a evaluace aplikace. V závěrečné části jsou prezentovány výsledky výzkumu a diskutovány možnosti dalšího rozvoje aplikace. Tímto způsobem práce přispívá k teoretickému poznání v oblasti energetického managementu a nabízí praktické řešení pro zlepšení energetické efektivity na úrovni koncových uživatelů.

I. TEORETICKÁ ČÁST

1 SPOTŘEBA ELEKTRICKÉ ENERGIE

V úvodní části kapitoly "Spotřeba elektrické energie" bude prezentována rešerše zaměřená na způsoby účtování energií a na existující aplikace monitorující spotřebu elektrické energie. Způsoby účtování energie jsou klíčovým faktorem pro pochopení, jak jsou náklady na energii rozdělovány mezi koncové uživatele. Bude zkoumáno, jaké metodiky jsou aplikovány v různých regionech a jak se liší v závislosti na poskytovateli energie.

Dále bude provedena rešerše dostupných aplikací, které umožňují uživatelům monitorovat a řídit jejich spotřebu elektrické energie. Tyto aplikace hrají zásadní roli v zvyšování energetické efektivity a v edukaci uživatelů o jejich spotřebním chování. V rámci rešerše budou identifikovány hlavní funkce, uživatelské rozhraní a technologie použité v těchto aplikacích. Kritické hodnocení těchto aplikací odhalí jejich přínosy a potenciální omezení ve snaze o optimalizaci spotřeby elektrické energie.

1.1 Základy měření spotřeby energie

Měření spotřeby energie je klíčovým prvkem pro správné vyúčtování elektřiny a efektivní správu energetické spotřeby v domácnostech i v průmyslu. Měřící zařízení, které se běžně nazývají elektroměry, jsou instalovány u každého odběrného místa a zaznamenávají množství spotřebované elektrické energie v kilowatthodinách (kWh).

Měření spotřeby je důležité nejen pro účtování spotřebované elektřiny, ale také pro monitorování a řízení energetické efektivity. Přesné měření umožňuje odběratelům lépe pochopit svou spotřebu a identifikovat příležitosti pro úspory energie.

Distribuční sazby elektřiny jsou klíčovým faktorem v účtování za elektrickou energii, které ovlivňuje finanční náklady spojené se spotřebou energie. Tyto sazby se mohou lišit v závislosti na distribučním území, typu odběrného místa (domácnost, firma), a čase spotřeby (vysoký a nízký tarif).

Název sazby	Pro koho je sazba vhodná?	Co musíte splňovat?	Máte nárok na nízký tarif?
Malá spotřeba D01D	Chaty, garáže a domácnosti s nízkou spotřebou elektřiny.	Běžné spotřebiče jako lednice, osvětlení, televize... V domácnosti nesmí být připojená výrobna a jistič nad 3 x 63 Ampér	Ne, D01D je jednotarifní sazba, která má jednotnou cenu elektřiny po celý den.
Běžná spotřeba D02D	Domácnosti s běžnými spotřebiči.	Běžné spotřebiče jako TV, lednička, sporák, pračka, myčka...	Ne, D02D je jednotarifní sazba, která má jednotnou cenu elektřiny po celý den.
Bojler D25D	Domácnosti, které používají elektřinu pro ohřev vody.	Elektrický bojler	Ano, D25D je dvoutarifní sazba, kde je 8 hodin denně cena eletřiny v levnějším tarifu.
Akumulační topení D26D	Domácnosti, které k vytápění používají akumulační spotřebiče.	Akumulační topení	Ano, D26D je dvoutarifní sazba, kde je 8 hodin denně cena eletřiny v levnějším tarifu.
Elektromobil D27D	Majitele elektromobilů.	Elektromobil	Ano, D27D je dvoutarifní sazba, kde je 8 hodin denně cena eletřiny v levnějším tarifu.
Kombinované vytápění D35D	Domácnost, která využívá smíšené elektrické spotřebiče pro vytápění objektu.	Sazbu bylo možné přiznat pouze do 1. 4. 2017. Nově již není možné tuto sazbu získat	Ano. D35D je dvoutarifní sazba, kde je 16 hodin denně cena eletřiny v levnějším tarifu.
Elektrické topení D45D	Domácnost, která využívá přímotopné elektrické spotřebiče pro vytápění objektu.	Sazbu bylo možné přiznat pouze do 1. 4. 2017. Nově již není možné tuto sazbu získat.	Ano. D45D je dvoutarifní sazba, kde je 20 hodin denně cena eletřiny v levnějším tarifu.
Tepelné čerpadlo D56D	Domácnost, která využívá systém tepelného čerpadla pro vytápění objektu.	Sazbu bylo možné přiznat pouze do 1. 4. 2017. Nově již není možné tuto sazbu získat.	Ano. D56D je dvoutarifní sazba, kde je 22 hodin denně cena eletřiny v levnějším tarifu.
Elektrické topení D57D	Domácnosti, které používají elektřinu k vytápení.	Elektrické topení (tepelné čerpadlo, elektrokotel, přímotopy, podlahové vytápění)	Ano. D57D je dvoutarifní sazba, kde je 20 hodin denně cena eletřiny v levnějším tarifu.
Víkend D61D	Místo, které využíváte převážně o víkendu.	Bez podmínek	Ano, D61D je dvoutarifní sazby, kde je levnější tarif od pátku 12:00 do neděle 22:00

Obr. 1: Přehled distribučních sazeb elektřiny (z webu cez.cz)[1]

Na Obr.1 je poskytnut přehled distribučních sazeb, který ilustruje, jaké podmínky musí být splněny pro získání různých tarifů a jak dlouho má odběratel nárok na nízký tarif, což je často označováno jako levnější nebo noční proud. Tento přehled pomáhá odběratelům lépe se orientovat ve svých možnostech a volbách týkajících se tarifů elektřiny.

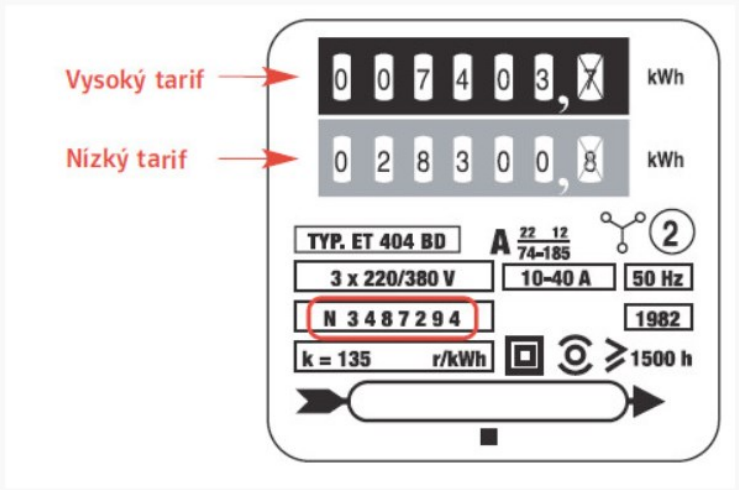
Získání přístupu k nízkým tarifům může být podmíněno splněním určitých kritérií, jako je instalace speciálního měřícího zařízení, které umožňuje dálkové ovládání a přepínání tarifů. Tato zařízení jsou často spojena s tzv. hromadným dálkovým ovládáním (HDO), což je systém umožňující dodavatelům energie automaticky přepínat tarify v závislosti na aktuální zátěži elektrické sítě a spotřebě [2].

Více informací o specifických distribučních sazbách a podmínkách pro jejich získání může odběratel nalézt na webových stránkách svého distribučního provozovatele, například na stránce ČEZ, jak je uvedeno ve zdroji obrázku 1.

1.2 Čtení a Interpretace Dat z Elektroměrů

Elektroměry, které měří množství odebrané elektrické energie, jsou instalovány v elektroměrových rozvaděčích na odběrném místě. Tyto přístroje jsou využívány k hodnocení spotřebované elektřiny a k následnému vyúčtování.

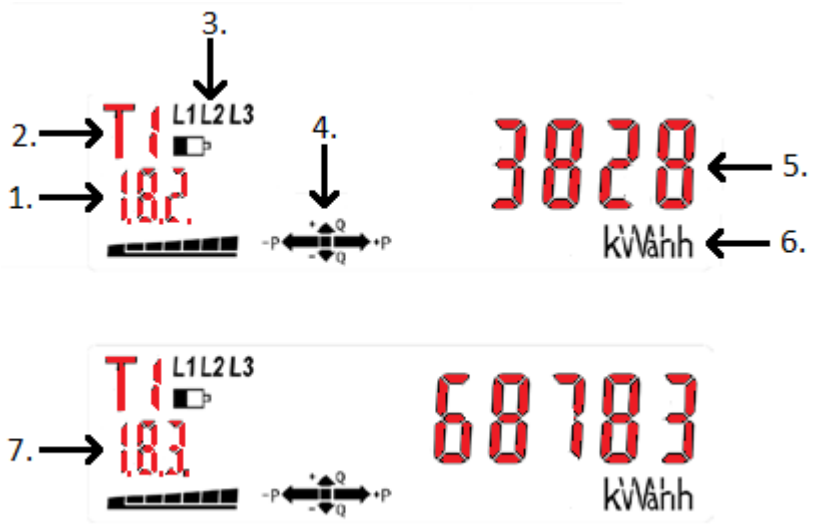
V případě jednotarifních elektroměrů je na přístroji přítomen pouze jeden číselník pro vysoký tarif (VT). U dvoutarifních elektroměrů se mohou číselníky nacházet buď nad sebou, kde nahoře je číselník pro vysoký tarif a dole pro nízký tarif, nebo vedle sebe, kde vlevo je vysoký tarif označený I. a vpravo nízký tarif označený II.



Obr. 2: Mechanický elektroměr [3]

Naměřené hodnoty jsou zaznamenávány z číslic před desetinnou čárkou (viz Obr. 2). Pro nahlášení naměřených hodnot je nutné znát výrobní číslo elektroměru, které je nejčastěji umístěno vedle písmen N nebo Nr.

Pokud elektroměr zobrazuje digitální displej bez kódů OBIS, u jedno tarifního přístroje se neustále zobrazuje hodnota vysokého tarifu – VT. U dvou tarifního přístroje se hodnoty VT (T1 na displeji) a NT (T2 na displeji) střídají v osmisekundových intervalech. Na hlavním displeji může být zobrazen stav vysokého tarifu (VT) T1 (1.8.2) nebo stav nízkého tarifu (NT) T2 (1.8.3) (viz Obr. 3).



Obr. 3: Digitální displej s kódy OBIS [3]

Symboly na elektroměru ukazují stav zařízení: svítí-li, elektroměr je zapojený správně; bliká-li, je na elektroměru nesprávný sled fází; a pokud nesvítí vůbec, některá fáze chybí. Aktivní tarif, směr toku činné a jalové energie jsou indikovány svítícími šipkami [3].

1.3 Rozdíl mezi dodavatelem a distributorem elektřiny

Hlavní rozdíl mezi dodavatelem a distributorem elektřiny spočívá v jejich základních funkcích a možnostech výběru ze strany spotřebitele.

Dodavatel elektřiny je společnost, od které si spotřebitelé kupují elektřinu. Role dodavatele zahrnuje určení ceny elektřiny, fakturaci a obecně obchodní vztahy se zákazníkem. Spotřebitelé mají možnost vybrat si svého dodavatele podle vlastních preferencí a mohou ho kdykoli změnit, pokud najdou výhodnější nabídku nebo lepší služby.

Distributor elektřiny na druhé straně je společnost zodpovědná za fyzickou distribuci elektřiny k zákazníkům. Tato role zahrnuje údržbu a správu elektrického vedení, instalace a servis měřicích zařízení, řešení technických problémů a zajištění nepřetržitého přívodu elektřiny. Na rozdíl od dodavatele si distributora nemůže spotřebitel vybrat. Distributor je přidělen na základě geografického umístění spotřebitele a je v dané oblasti obvykle monopolním poskytovatelem.

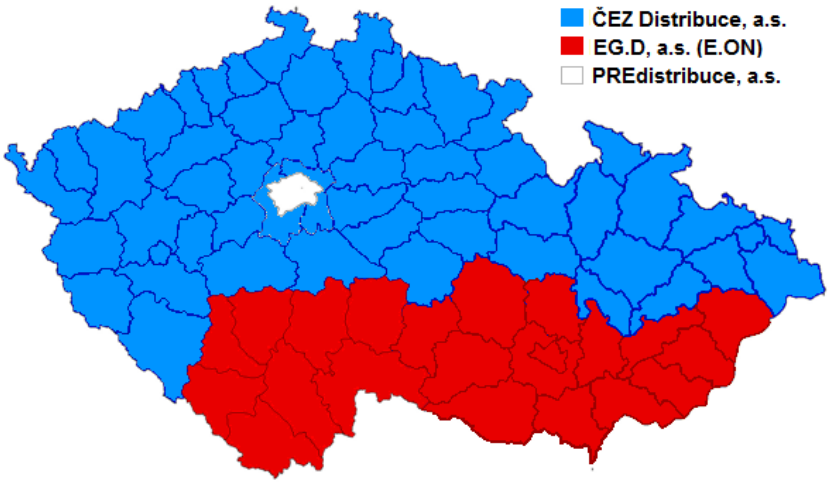
Tedy, zatímco dodavatelé jsou voleni spotřebiteli a jsou zodpovědní za obchodní aspekty dodávky elektřiny, distributoři jsou určeni regionálně a odpovídají za technické a infrastrukturní aspekty distribuce elektřiny [4].

1.4 Distributoři elektřiny

V České republice je distribuční soustava elektřiny strukturována do tří hlavních distribučních území. Každé z těchto území je spravováno jednou z tří velkých distribučních společností: ČEZ Distribuce, a.s., PREdistribuce, a.s., a E.ON Distribuce, a.s. Toto rozdělení území mezi jednotlivé distributory je klíčové pro organizaci a efektivní řízení dodávek elektřiny napříč zemí.

Každá z těchto společností má na starost nejen distribuci elektrické energie do domácností a podniků ve svém přiděleném regionu, ale také údržbu a rozvoj distribuční sítě. To zahrnuje vše od výstavby nových vedení, přes rekonstrukce stávajících zařízení, až po rychlé řešení výpadků a poruch [5].

Geografické rozdělení území mezi jednotlivé distributory a jejich specifické role v elektroenergetickém systému ČR jsou vizuálně znázorněny na Obr. 4.



Obr. 4: Rozdělení území ČR mezi distributory elektřiny [5]

1.5 Největší dodavatelé elektřiny V ČR

ČEZ, akciová společnost, je významným lídrem v produkci elektřiny v České republice a zároveň mateřskou společností Skupiny ČEZ, založené v roce 2003. Skupina, která sdružuje více než 120 společností, hraje klíčovou roli nejen v domácím, ale i ve středoevropském a východoevropském energetickém sektoru. V České republice ČEZ dominuje nejen jako největší výrobce elektřiny a tepla, ale také jako provozovatel distribuční sítě a přední hráč na

velkoobchodním i maloobchodním trhu s elektřinou. Společnost se také věnuje vědeckým a inovačním aktivitám a je předním subjektem v rozvoji elektrických sítí a elektromobility [6].

Innogy, jedna z pěti největších energetických společností v Evropě, nabízí široké spektrum profesionálních služeb. Zabývá se výrobou, obchodem, transportem a dodávkami elektřiny a plynu, přičemž klade důraz na kvalitní zákaznický servis, včetně nonstop zákaznické linky a poradenských center po celé republice [6].

E.ON Energie, který na český trh vstoupil v 90. letech, se postupně etabloval jako důležitý akční hráč v energetickém sektoru. Dnes dodává elektřinu 1,1 milionu zákazníků a zaujímá 20% podíl na trhu [6].

Pražská energetika (PRE), založená v roce 1994, je mezi třemi nejvýznamnějšími dodavateli elektřiny v České republice s více než 600 tisíci zákazníky. Společnost garantuje vysokou spolehlivost a kvalitu služeb, a nabízí také 'PREekoproud' – elektřinu z obnovitelných zdrojů [6].

Pražská plynárenská, založená v roce 1847, se od roku 2012 věnuje také dodávkám elektřiny. S více než 400 tisíci zákazníky nabízí řadu cenově výhodných produktů, včetně kombinovaného balíčku plynu a elektřiny a slevy pro online sjednání smluv [6].

Centropol Energy, založený v roce 2002, poskytuje elektřinu i plyn přes 300 tisícům zákazníků. Kromě standardní nabídky energií podporuje výkup elektřiny z obnovitelných zdrojů a nabízí slevy u partnerů na fotovoltaické panely a další technologie pro moderní domácnosti [6].

1.6 Tarifní systémy a jejich struktura

Očekávání spojené s úsporami při používání elektrospotřebičů v nočních hodinách v důsledku aplikace nočního tarifu již nejsou považována za relevantní. Termín "noční proud" je sice stále běžně užíván v každodenní konverzaci, ale současné tarifní systémy již nejsou vázány výhradně na noční čas. Možnost volby mezi nízkým a vysokým tarifem stále existuje, avšak důležité je být obeznámen s příslušnými postupy a důvody pro jejich využití. Není pravidlem, že využití vysokého tarifu automaticky vede k nadměrným platbám za elektřinu.

Historicky byly noční tarify zavedeny primárně za účelem snížení zátěže na elektrickou síť během nejvytíženějších hodin, což mělo motivovat spotřebitele k přesunutí energeticky náročnějších aktivit do pozdních hodin. V současnosti se tarify upravují nejen v nočních, ale i v jiných časech.

Pokud jde o topení, ohřev vody a elektromobily, většina domácností v České republice využívá elektřinu primárně pro osvětlení a běžné domácí spotřebiče a platí ve vysokém tarifu. Tato situace se mění v případě, že elektřinu spotřebujete pro specifické účely, jako je vytápění nebo ohřev vody, či vlastníte elektromobil. V těchto případech je doporučováno, aby byl kontaktován příslušný distributor a požádáno o změnu na dvoutarifní sazbu, což je moderní ekvivalent dřívějšího nočního proudu.

Distributoři nabízejí různé tarifní sazby, které jsou diferencovány podle specifických potřeb zákazníků:

Jednotarifní sazby jsou určeny pro domácnosti s nižší (D01d) nebo střední až vysokou spotřebou (D02d).

Dvoutarifní sazby zahrnují tarify speciálně určené pro ohřev vody (D25d, D26d), nabíjení elektromobilů (D27d) a topení přímotopem (D45d, D56d, D57d).

Získání nízkého tarifu elektřiny vyžaduje správné nastavení spotřebičů a jejich jmenovité spotřeby, aby distributor mohl rozhodnout o přizpůsobení tarifu. Proces sjednání tarifu začíná konzultací s dodavatelem elektřiny, po které následuje kontaktování distributora s žádostí o změnu tarifu a případnou instalaci potřebných zařízení.

Pro domácnosti s nízkou spotřebou elektřiny nemusí být výhodné usilovat o dvoutarifní sazbu vzhledem k vyšším distribučním poplatkům. Důležité je vybrat dodavatele, který nabízí férové ceny a transparentní tarifní strukturu bez ohledu na čas spotřeby [7].

1.7 Vyúčtování elektřiny

Vyúčtování elektřiny se týká všech domácností, bez ohledu na to, od kterého dodavatele energie jsou odebírány. Standardně je vyúčtování energií strukturováno podle jednotného vzoru.

Na úvodní straně vyúčtování jsou poskytnuty souhrnné informace, které zahrnují období, na které se vyúčtování vztahuje, distribuční sazbu a produkt odebíraný od dodavatele. Dále je zde uveden konečný stav vyúčtování, tj. zda byl zaznamenán přeplatek nebo nedoplatek [8].

1.7.1 Přeplatek

Přeplatek, který znamená, že výše uhrazených záloh převýšila skutečnou spotřebu, je někdy automaticky vracen na účet zákazníkem, zatímco jiné společnosti ho používají k úhradě záloh v následujícím období. Postup vypořádání přeplatku je specifikován na faktuře [8].

1.7.2 Nedoplatek

Nedoplatek vzniká, pokud zálohy nebyly uhrazeny ve výši odpovídající skutečné spotřebě, což znamená, že zákazník dluží dodavateli za spotřebovanou elektřinu. Informace o výši nedoplatku a způsobu jeho úhrady do stanoveného termínu jsou uvedeny ve faktuře. Často se platí stejným způsobem jako zálohy, obvykle inkasem, přičemž je třeba zajistit, aby byl na účtu dostatečný limit pro stažení platby.

Dodavatelé ve vyúčtování uvádějí, jakým způsobem došlo k vzniku přeplatku nebo nedoplatku, a to porovnáním zaplacených záloh s reálnými náklady na spotřebu elektřiny. Vyúčtování dále obsahuje graf, který ilustruje rozdělení plateb za spotřebovanou elektřinu, její distribuci a další poplatky [8].

1.7.3 Obchodní část vyúčtování

Obchodní část vyúčtování je odvozena od produktu, který zákazník u dodavatele sjednal, a tedy závisí na volbě dodavatele [8].

1.7.4 Distribuční část vyúčtování

Distribuční část je určena podle distribučního území, ve kterém zákazník sídlí, a regulovaných poplatků stanovených Energetickým regulačním úřadem, které jsou jednotné pro všechny domácnosti. Česká republika je rozdělena do tří distribučních území: EG.D

Distribuce pro jižní Čechy a jižní Moravu, PRE Distribuce pro Prahu a ČEZ Distribuce pro zbytek republiky [8].

1.7.5 Stálá platba a účtování spotřeby

V obchodní části vyúčtování je uvedeno, kolik zákazníka stála spotřeba elektřiny podle spotřebovaných megawatthodin. V případě, že zákazník využívá dvoutarifní sazbu, je položka rozdělena na spotřebu v nízkém a vysokém tarifu. Dvoutarifní sazba je typická pro domácnosti, které využívají elektřinu pro vytápění, ohřev vody nebo nabíjení elektromobilu. Stálá platba, která je fixní měsíční poplatek dodavateli, a daň z elektřiny, vypočítaná podle spotřeby, jsou rovněž specifikovány ve vyúčtování [8].

1.7.6 Ukázka vyúčtování elektřiny

Tato sekce poskytuje podrobný výpočet ročních nákladů na elektřinu a demonstruje, jak jsou využívány údaje o spotřebě a tarify pro určení celkové částky vyúčtované zákazníkovi. Výpočet zahrnuje všechny složky tarifu, včetně poplatků za distribuci, obchod, systémové služby a další.

Je předpoklad, že roční spotřeba ve vysokém tarifu (VT) je 2000 kWh a v nízkém tarifu (NT) je 3000 kWh. Měsíční poplatek za jistič je 100 Kč, což za celý rok představuje náklady ve výši 1200 Kč. Tento poplatek je účtován měsíčně.

Poplatky za distribuci se počítají na základě spotřeby v kWh:

Ve vysokém tarifu se za distribuci platí 200 Kč za každý MWh, takže při spotřebě 2 MWh činí tento poplatek 400 Kč.

V nízkém tarifu se platí 250 Kč za každý MWh, což při spotřebě 3 MWh vede k nákladům 750 Kč.

Systémové služby jsou účtovány za 212 Kč za každý MWh, což při celkové spotřebě 5 MWh (součet VT a NT) dělá 1060 Kč. Tento poplatek se počítá na základě spotřeby v kWh.

Poplatek za POZE je 600 Kč za MWh, takže celkové náklady za 5 MWh jsou 3000 Kč, rovněž počítané na základě spotřeby v kWh.

Měsíční poplatek za OTE je 4 Kč, což za rok činí 48 Kč a je účtován měsíčně. Stálý měsíční poplatek je 120 Kč, což za celý rok dělá 1440 Kč, účtované měsíčně.

Daň z elektřiny je stanovena na 30 Kč za MWh, takže při celkové roční spotřebě 5 MWh zaplatí zákazník 150 Kč. Tento poplatek je vypočítán na základě spotřeby v kWh.

Obchodní poplatky za energii:

Ve vysokém tarifu je cena 3400 Kč za MWh, což v případě spotřeby 2 MWh činí 6800 Kč.

V nízkém tarifu je cena také 3400 Kč za MWh, při spotřebě 3 MWh to je 10200 Kč.

Oba tyto poplatky jsou vypočítány na základě spotřeby v kWh.

Celkové roční náklady na elektřinu tak činí 25,048 Kč. Tento výpočet poskytuje transparentní pohled na to, jak jsou jednotlivé složky tarifu aplikovány na spotřebu zákazníka a jak tyto složky přispívají k celkovým nákladům na energie.

1.8 Aplikace pro výpočet spotřeby energie

Ceny za kubický metr zemního plynu a cena za kilowatthodinu elektřiny byly výrazně zvýšeny. Z provedeného srovnání vyplývá, že současné ceny elektřiny jsou mnohonásobně vyšší než v minulém roce. Aby konečné roční vyúčtování a faktura za elektřinu nepřekvapily, je doporučeno sledovat vlastní spotřebu energií. V tom mohou být využity moderní technologie; není nutné, aby byly údaje o spotřebě zapisovány ručně na papír. Pro bezstarostné sledování může být pořízen senzor připojený k internetu věcí, který pravidelně měří spotřebu elektřiny nebo plynu, odesílá naměřené hodnoty do cloudu, a ty pak mohou být zobrazovány v připojené aplikaci. V budoucnu by měly být všechny elektroměry vybaveny podobnou „chytrou“ technologií a odečty energií by měly probíhat distančně [9].

Aktuální stav na trhu s mobilními aplikacemi, které napomáhají k výpočtu spotřeby energie, bude nyní ukázán.

1.8.1 Domácí odečty jednoduše

Aplikaci „Domácí odečty jednoduše“ je možné stáhnout z Google Play a její používání je zcela zdarma. Umožňuje ukládat záznamy o spotřebě elektřiny, plynu, teplé a studené vody, přičemž stav se zaznamenává k určitému datu a můžete k němu připojit i fotografii měřidla. V aplikaci si lze také zanést ceny za jednotlivé služby a měsíční zálohy. Aplikace automaticky vytváří grafy měsíční spotřeby a nabízí jejich porovnání.

Tato aplikace dlouhodobě drží vysoké hodnocení, blížící se 5 hvězdám, což odráží její uživatelsky příjemné rozhraní dostupné i v češtině, které umožňuje snadné přepínání mezi

funkcemi. Základní kategorie zahrnují plyn, teplou a studenou vodu a elektřinu, s možností přidání dalších kategorií podle potřeby. Hodnoty měřidel jsou zaznamenávány na konci každého měsíce, s automatickým vyhodnocením spotřeby a grafickým porovnáním s ostatními měsíci. Data jsou ukládána přímo do telefonu, ale lze je také zálohovat na Google Drive nebo externí disk [10].

1.8.2 Odečty elektrické energie

V této aplikaci se na začátku přidá odběrné místo, to se pojmenuje, popíše (např. zda se jedná o domov, nebo chatu), zadá se informace o jističi, číslu elektroměru a číslu odběrného místa. Všechny tyto údaje lze nalézt na faktuře za elektřinu. Následně je možné do aplikace přidat ceník produktu dodavatele, do části měsíční odpočty pak ukládáte měsíční stavy elektřiny. V aplikaci je možné zároveň propojit ceník s odečtem, přidat údaj o měsíční záloze a aplikace bude informovat o tom, jak se spotřeba vyvíjí, případně zda bude mít odběratel doplatek, nebo nedoplatek [9].

1.8.3 Kalkulátor nákladů na elektřinu

Pokud není problém angličtina, je možné si stáhnout program Electricity Bill Calculator, která pohlídá náklady na klimatizaci, chladničku, topení, pračku, nebo žehlení. U každého spotřebiče se zadává jeho příkon, počet spotřebičů, počet minut, po které spotřebič používáte, cena za kWh elektřiny. Díky aplikaci je snadno viditelné, kolik používání těchto spotřebičů stojí [9].

1.8.4 PROUD ČEZ

Nová mobilní aplikace ČEZ Distribuce s názvem PROUD umožňuje klientům této firmy samoodečtem elektřiny kontrolu spotřeby v jednotlivých odběrných místech. Má pomoci zákazníkům v době současné energetické krize a častých výkyvů cen.

Aplikace PROUD, která má ve své první generaci sedm funkcí, je zdarma ke stažení v Google Play a AppStore. Kromě samoodečtů poskytuje přehled spotřeby spolu s dalšími informacemi o odběrném místě a informuje o odstávkách a poruchách.

Přímá komunikace z aplikace umožňuje nahlašování poruch, zadávání požadavků včetně fotografií a GPS a sledování stavu řešených záležitostí [11].

Bohužel aplikace není dostupná pro uživatele, kteří nejsou odběrateli energie u společnosti ČEZ.

1.8.5 E.ON

Společnost E.ON nedávno aktualizovala svou mobilní aplikaci Energie24, která nyní nabízí kromě standardních informací o zálohách, smlouvě a stavu spotřeby energií i unikátní funkci pro získání aktuálního odhadu vyúčtování. Tato funkce umožňuje uživatelům nafotit stav elektroměru a na základě toho získat odhad spotřeby a výši očekávané platby za elektřinu v aktuálním fakturačním období. Nové funkce aplikace poskytují uživatelům přesnější informace o jejich spotřebě a nákladech na energie, což jim umožňuje lépe nastavit výši záloh a předejít tak zbytečným přeplatkům či nedoplatkům.

Aplikace Energie24 je kompatibilní s jednotarifními i dvoutarifními distribučními sazbami a nabízí tuto službu bezplatně. Dále umožňuje uživatelům ručně upravit velikost jističe a poskytuje snadný přístup k zákaznické a poruchové lince společnosti E.ON. Uživatelé mohou také upravovat výši záloh a získat přehled o vyúčtování včetně možnosti stažení faktur za elektřinu ve formátu PDF. Po uvedení nových funkcí se počet uživatelů aplikace více než zdvojnásobil a byl zaznamenán téměř třicetiprocentní nárůst v počtu stažení. Aplikace je dostupná pro systémy Android a iOS a lze ji zdarma stáhnout v obchodech Google Play a Apple Store [12].

Bohužel aplikace také není dostupná pro uživatele, kteří nejsou odběrateli energie u společnosti E.ON

1.8.6 Přínos těchto aplikací

Pokud by odběratelé rádi na energiích něco ušetřili, chytré aplikace jsou ideálním způsobem, jak začít. Nejenže data zaznamenají, ale také vyhodnotí a jednotlivá období porovnají. Navíc je díky uloženým hodnotám možné zkontrolovat, že skutečná faktura za energie odpovídá spotřebě energií.

2 WEBOVÉ A MOBILNÍ APLIKACE

Webové aplikace jsou hostovány na serverech a přístup k nim je možný prostřednictvím webových prohlížečů bez ohledu na operační systém uživatelova zařízení. Na druhou stranu, mobilní aplikace jsou vyvíjeny specificky pro konkrétní mobilní operační systémy, jako jsou iOS nebo Android, a vyžadují instalaci přímo na zařízení.

Pokud jde o vývoj a údržbu, u webových aplikací je udržována jednotná kódová báze, což zjednodušuje provádění aktualizací a údržbu, které jsou realizovány na serverové straně. Mobilní aplikace však mohou vyžadovat vytvoření a udržování několika verzí kódu pro různé platformy, což zvyšuje složitost údržby a aktualizací. Použití multiplatformních frameworků může tento problém částečně řešit.

Další důležitý rozdíl je v přístupu k hardwarovým funkcím. Webovým aplikacím je umožněn omezenější přístup k hardwarovým funkcím zařízení ve srovnání s mobilními aplikacemi, které mohou plně využívat například kamery, GPS senzory, akcelerometry a další. Tento rozdíl může výrazně ovlivnit typy funkcí, které aplikace mohou uživatelům nabídnout.

2.1 Principy webových aplikací

Webové aplikace jsou software, který se spouští na webovém serveru a je přístupný prostřednictvím webového prohlížeče pomocí internetového připojení. Tyto aplikace jsou vyvíjeny v různých programovacích jazycích a frameworcích, jako je HTML, CSS, JavaScript, PHP, Ruby on Rails a další. Webové aplikace umožňují dynamickou interakci s uživatelem a často využívají databáze pro ukládání a správu dat.

2.2 Principy mobilních aplikací

Mobilní aplikace jsou aplikace navržené k běhu na mobilních zařízeních, jako jsou smartphony a tablety. Existují tři hlavní typy mobilních aplikací: nativní aplikace, webové aplikace a hybridní aplikace. Nativní aplikace jsou vyvíjeny specificky pro jednu platformu (iOS nebo Android) a nabízejí nejlepší výkon a přístup k funkcím zařízení. Webové mobilní aplikace jsou vlastně webové stránky optimalizované pro mobilní zařízení. Hybridní aplikace spojují prvky obou metodik a jsou spouštěny ve webovém prohlížeči, který je integrován do nativní aplikace.

2.3 Progressive Web Apps (PWA)

Progressive Web Apps (PWA) jsou typem aplikace, který se snaží kombinovat nejlepší z webových a mobilních aplikací. Tyto aplikace jsou přístupné přes webový prohlížeč, ale nabízejí funkce typické pro mobilní aplikace, jako jsou push notifikace, offline práce a přístup k hardwarovým funkcím zařízení. Výhody PWA zahrnují snadnou instalaci bez nutnosti stahování z obchodu s aplikacemi, aktualizace na straně serveru a lepší dosah uživatelů. Omezení zahrnují omezenější přístup k funkcím zařízení ve srovnání s nativními aplikacemi a závislost na podpoře prohlížeče.

2.4 Bezpečnostní aspekty webových a mobilních aplikací

Bezpečnost je kritický aspekt vývoje jak webových, tak mobilních aplikací. Mezi běžné hrozby patří SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), nezabezpečené API a problémy s ukládáním dat na straně klienta. Zabezpečení aplikace vyžaduje komplexní přístup, včetně šifrování dat, bezpečného programování, pravidelného testování zranitelností a aktualizace softwaru. Je důležité dodržovat nejlepší postupy a bezpečnostní standardy, jako jsou OWASP Top 10 pro webové aplikace a mobilní aplikace, a implementovat robustní autentizační a autorizační mechanismy.

2.5 Přehled použitých programovacích jazyků

V rámci projektu bylo zásadní zvolit správné technologie, které umožní efektivně realizovat požadované funkcionality. Proto byly využity následující programovací jazyky: TypeScript, SCSS a HTML. Každý z těchto jazyků přináší specifické výhody, které jsou klíčové pro různé aspekty vývoje webové aplikace.

2.5.1 Typescript

TypeScript je otevřený programovací jazyk vyvinutý společností Microsoft, který byl poprvé uvedený na trh v roce 2012. Jako rozšíření JavaScriptu přidává TypeScript statické typování a objektově orientované funkce, jako jsou třídy a rozhraní. Tato vylepšení činí TypeScript zvláště vhodným pro vývoj rozsáhlejších aplikací, neboť umožňuje efektivněji identifikovat a opravovat potenciální chyby již během vývoje a zjednodušuje procesy refaktorizace a správy kódu.

Umožňuje vývojářům explicitně specifikovat typy pro proměnné, funkce a objekty, což přispívá k větší bezpečnosti kódu a jeho srozumitelnosti. Dále je plně kompatibilní s

jakýmkoliv platným JavaScriptovým kódem, což zjednodušuje integraci a přechod pro existující projekty. TypeScript je navíc pravidelně aktualizován o podporu nejnovějších funkcí ECMAScript a zahrnuje vlastní rozšíření, jako jsou typové anotace. To umožňuje vývojářům využívat nejnovější technologie bez obav z kompatibility s prohlížeči.

TypeScript je také integrován s rozšířenou podporou pro nástroje, jako jsou linters, editory a integrované vývojové prostředí (IDE), které zlepšují celkový vývojový proces a pomáhají vývojářům psát čistější a efektivnější kód. Tyto vlastnosti činí TypeScript široce přijímaným ve vývojářské komunitě, zvláště v projektech, kde je potřeba vyšší míra robustnosti a škálovatelnosti, jako jsou velké webové aplikace a enterprise řešení [13].

2.5.2 HTML

HTML, což je zkratka pro HyperText Markup Language, je standardním značkovacím jazykem používaným pro tvorbu webových stránek. Umožňuje strukturování obsahu, jako jsou sekce, odstavce a odkazy, prostřednictvím prvků HTML, které zahrnují značky a atributy. I když HTML není považováno za programovací jazyk, protože nemůže vytvářet dynamickou funkcionalitu, je nyní oficiálně uznáváno jako webový standard. Specifikace HTML jsou udržovány a vyvíjeny Světovým konsorciem pro web (W3C), které poskytuje pravidelné aktualizace.

Využívá se především ve webovém vývoji, kde vývojáři pomocí kódu HTML navrhují způsob zobrazení webových prvků v prohlížeči, jako jsou texty, hypertextové odkazy a mediální soubory. Dále se HTML používá pro internetovou navigaci a organizaci webových dokumentů podobně jako v Microsoft Word.

HTML pracuje na principu souborů, které mají příponu .html nebo .htm a jsou čteny webovými prohlížeči, které zobrazují jejich obsah uživatelům internetu. Každá HTML stránka obsahuje řadu HTML prvků, které se skládají ze sady značek a atributů. Tyto prvky jsou základními stavebními bloky webové stránky.

Dokument HTML musí začínat deklarací *<!DOCTYPE>*, která informuje webový prohlížeč o typu dokumentu. S příchodem HTML5, moderní deklarace veřejného typu dokumentu vypadá jako *<!DOCTYPE html>*.

V současné době je k dispozici 142 HTML značek, které umožňují vytváření různých prvků. HTML5 přineslo podporu nových typů formulářových prvků a zavedlo několik sémantických značek, které jasně popisují obsah, například *<article>, <header>* a *<footer>*.

HTML má své výhody i nevýhody. Mezi výhody patří, že je přívětivé pro začátečníky, má širokou podporu a je otevřené a zcela zdarma. Na druhou stranu, HTML se primárně používá pro statické webové stránky a pro dynamické funkcionality je potřeba použít JavaScript nebo backendový jazyk, jako je PHP.

HTML je úzce spojeno s CSS (Cascading Style Sheets) a JavaScriptem. Zatímco HTML se používá k přidání textových prvků a vytváření struktury obsahu, CSS zajišťuje stylování, jako jsou pozadí, barvy, rozvržení, mezery a animace, a JavaScript přidává dynamickou funkcionalitu, jako jsou posuvníky, vyskakovací okna a fotogalerie. Tyto tři jazyky tvoří základ front-endového vývoje [14].

2.5.3 SCSS

SCSS (Sassy Cascading Style Sheets) je jednou ze dvou syntaxí dostupných pro populární CSS preprocesor Sass (Syntactically Awesome Stylesheets). Jako pravý superset CSS, všechny platné CSS kódy jsou zároveň platné SCSS kódy. SCSS umožňuje stylovat vizuální prvky webové stránky, včetně tlačítek, posuvníků, obrázků, barevných schémat, písem, témat a rozložení stránky.

SCSS funguje tak, že se kompiluje do nativního CSS kódu, který je kompatibilní s jakýmkoliv prohlížečem. Tento proces kompilace přináší úsporu času pro vývojáře, kteří mohou psát stručný SCSS kód, který se následně kompiluje do rozsáhlejšího CSS kódu. Tímto způsobem SCSS umožňuje vývojářům dělat více s kratším kódem, aniž by byla ohrožena kompatibilita s webem.

Mezi výhody používání SCSS patří zvýšená produktivita, lepší organizace kódu, zlepšená udržitelnost a čitelnost, a větší síla oproti samotnému CSS. SCSS přidává do CSS kódu funkce, které nejsou dostupné v standardním CSS. Toto zahrnuje proměnné, vnořené syntaxe a mixiny, které snižují potřebu opakovaného používání kódu. Tyto funkce umožňují deklarovat určité styly jednou a volat je v celém stylsheetu bez nutnosti opakování pro každý element na stránce.

Například vnořená syntaxe umožňuje seskupit kód uvnitř jiného kódu, což vede k čistším a efektivnějším stylesheetům. SCSS také podporuje proměnné, které lze použít pro uchovávání hodnot, které budou použity napříč stylesheetem. To je užitečné při práci s barvami, velikostmi a dalšími hodnotami, které mohou být často měněny. Mixiny v SCSS umožňují vytvářet skupiny CSS deklarací, které lze používat opakovaně v celém stylesheetu, což je

užitečné při práci s dodavatelskými prefixy, složitými animacemi a dalšími kódy, které mohou být použity na více místech.

Celkově SCSS poskytuje významné vylepšení workflow CSS, což umožňuje webdesignérům a vývojářům efektivnější a organizovanější práci na stylech webových stránek a aplikací [15].

2.6 Frameworky a knihovny pro vývoj

2.6.1 Angular

Angular je open-source JavaScriptový framework vyvinutý společností Google, který je známý svým využitím TypeScriptu a standardního DOM pro zlepšení čitelnosti a udržitelnosti kódu. Tento framework je populární díky svému modulárnímu designu, kompatibilitě napříč platformami a integrovaným testovacím možnostem.

Jednou z klíčových funkcí Angularu je dvousměrné vázání dat, které umožňuje automatickou synchronizaci mezi modelem (daty) a zobrazením, což usnadňuje manipulaci s prvky webových stránek. Angular také implementuje architekturu Model-View-Controller (MVC), což usnadňuje správu složitých projektů tím, že rozděluje aplikace do tří propojených komponent.

V porovnání s jinými populárními frameworky, jako jsou ReactJS a VueJS, Angular poskytuje rozsáhlejší sadu nástrojů pro vývoj komplexních aplikací. Na rozdíl od Vue, který má jednodušší syntaxi, a Reactu, který se zaměřuje na výkon pomocí virtuálního DOM, Angular nabízí rozsáhlý ekosystém a pokročilé funkce, jako jsou animace a přístupnost, které zvyšují uživatelskou zkušenost a podporují opakované použití komponent.

Angular je obzvláště účinný při vývoji jednostránkových aplikací (SPA), které poskytují hladký a desktopově podobný uživatelský zážitek, a je také široce využíván v e-commerce a obsahově řízených webových stránkách, jako ukazuje příklad T-Mobile a weather.com, které využívají jeho schopnosti pro real-time aktualizace a správu složitých funkcionalit [16].

2.6.2 Vue.js

Vue je JavaScriptový framework, který usnadňuje vývoj uživatelského rozhraní (UI) webových stránek a jednostránkových aplikací. Byl navržen tak, aby byl postupně přijímatelný a vyhýbal se mnoha nedostatkům stávajících, monolitických frameworků. Jádro Vue je snadno

integrovatelné s dalšími knihovnami a existujícími projekty, což usnadňuje jeho osvojení díky zaměření pouze na vrstvu zobrazení knihovny. Vue je také schopný pohánět složité jednostránkové aplikace s využitím moderního nástrojí a podpůrných knihoven pro rozšířenou funkcionalitu.

Vue navíc využívá metody dvousměrného vázání dat, aby zajistil, že více částí aplikace používající stejná data vždy disponuje nejnovější verzí těchto dat.

Učení Vue JS je díky progresivní povaze frameworku jednoduché, což umožňuje jeho implementaci do existujících projektů. To znamená, že Vue je snadné pro každého vývojáře s předchozími znalostmi JavaScriptu. Na rozdíl od jiných frameworků Vue nevyžaduje rozsáhlé znalosti knihoven ani JSX. Vue je také přizpůsobitelný pro vývojáře všech úrovní díky své rozsáhlé dokumentaci. V průzkumu z roku 2019 uvedlo 76 % dotázaných vývojářů, že výjimečná dokumentace je největší výhodou Vue.

Pokud jde o srovnání s Reactem, Vue a React jsou oba JavaScriptové frameworky, ale každý nabízí síly v různých oblastech a jeden může být preferován v závislosti na potřebách uživatele. Vue je primárně navržen pro úkoly související s uživatelským rozhraním a může být integrován do jakéhokoli JavaScriptového projektu pro zvládání designu a vývoje rozhraní, zatímco jiný framework zvládá složitější úkoly. Naproti tomu React nabízí bohatý ekosystém knihoven pro design/rozvoj UI a umožňuje uživatelům psát vlastní komponenty v HTML prostřednictvím rozšíření .JSX. React také podporuje mobilní vývoj prostřednictvím React Native, rychlé vývojářské nástroje React, které jsou dostupné jako rozšíření pro Firefox nebo Chrome, a je přívětivý pro SEO. React je v současnosti nejpopulárnějším frameworkem a používá se na dvojnásobném počtu webových stránek než Vue. Nabízí velkou flexibilitu, ale je významně těžší z hlediska kódu, což znamená, že je také pomalejší v provozu.

Celkově mají Vue i React své silné stránky a slabiny. React je výrazně více všestranný a adaptabilní, zatímco Vue je více zjednodušený, snadnější na učení a disponuje více vestavěnými funkcemi. React je ve svém základním stavu minimalistický, ale nabízí výrazně více knihoven a podporu. Každý z nich je vysoce schopný framework schopný složité funkcionality JavaScriptu [17].

2.6.3 React

React je populární knihovna JavaScriptu, zaměřená na výstavbu uživatelských rozhraní, která byla původně vyvinuta společností Facebook. Jako open-source nástroj si získala oblibu pro svou schopnost zjednodušit a zefektivnit vývoj rozhraní aplikací. React je známý svým použitím deklarativních zobrazení, která umožňují vývojářům vytvářet interaktivní aplikace, které se okamžitě aktualizují při příchodu nových informací.

Jednou z klíčových vlastností Reactu je modulární přístup k UI, který podporuje rozdělení uživatelského rozhraní na opakovaně použitelné komponenty. Tyto komponenty jsou samostatné jednotky UI, které lze kombinovat pro sestavení složitých rozhraní. Tento přístup nejenže zvyšuje udržitelnost a škálovatelnost aplikací, ale také zlepšuje jejich výkonnost díky konceptu virtuálního DOM. Virtuální DOM umožňuje Reactu efektivně porovnat aktuální stav komponent s novým a aktualizovat pouze změněné části skutečného DOM.

React je primárně frontendová technologie, avšak díky nástrojům jako Next.js, který umožňuje server-side rendering, se jeho použití rozšiřuje i na backendové operace. Tento přístup rozostřuje hranici mezi frontendem a backendem. V porovnání s čistým JavaScriptem, známým jako Vanilla JS, poskytuje React strukturovanější a efektivnější způsob stavby dynamických uživatelských rozhraní, zejména pro větší a složitější aplikace.

React, často mylně označovaný za framework, je ve skutečnosti knihovna zaměřená na vrstvu zobrazení aplikací, což vývojářům umožňuje vytvářet opakovaně použitelné UI komponenty a efektivně spravovat stav. React může být rozšířen o další knihovny a nástroje pro poskytování funkcionality podobné frameworkům, avšak jeho základní filozofie zůstává soustředěná na stavbu komponent UI.

Pro efektivní využití Reactu je zásadní mít pevné základy v JavaScriptu, včetně znalostí proměnných, datových typů, funkcí, objektů, polí a asynchronního JavaScriptu. React aplikace často interagují s API a zpracovávají asynchronní operace, jako je načítání dat, což vyžaduje dobrou znalost Promise a async/await.

React se osvědčil jako účinný nástroj pro tvorbu dynamických a interaktivních uživatelských rozhraní, což ho činí ideálním pro široké spektrum aplikací, od jednoduchých webových stránek po složité podnikové aplikace [18].

2.7 Backend technologie a databáze

Backend technologie a databáze jsou základními pilíři pro vývoj moderních webových a mobilních aplikací. Mezi populární backendové technologie patří Node.js, které je založeno na JavaScriptovém enginu V8 od Google a umožňuje vývoj serverové části aplikací v JavaScriptu. Node.js je ceněné pro svou schopnost zpracovávat asynchronní požadavky, což výrazně zlepšuje výkon aplikací při práci s více uživateli současně.

Co se týče databází, rozlišujeme mezi relačními (SQL) a nerelečními (NoSQL) databázemi. SQL databáze, jako jsou MySQL, PostgreSQL, a Microsoft SQL Server, používají strukturovaný dotazovací jazyk pro manipulaci a dotazování dat, která jsou uspořádána do tabulek. Díky pevně definované schématu jsou vhodné pro aplikace vyžadující vysokou úroveň konzistence dat.

Na druhé straně, NoSQL databáze jako MongoDB, Redis, nebo Cassandra, nabízejí flexibilnější schémata pro ukládání dat, což je výhodné pro aplikace s velkým množstvím nestrukturovaných dat nebo pro situace, kdy rychlost a škálovatelnost jsou prioritní před striktní konzistencí. NoSQL technologie jsou často využívány v systémech, kde se počítá s rychlým růstem dat a požadavků na zpracování.

Výběr mezi SQL a NoSQL databázemi by měl být vždy založen na specifických potřebách projektu a charakteristikách dat, která aplikace bude zpracovávat.

2.7.1 Node.js

Node.js je populární běhové prostředí open-source, které umožňuje spouštění JavaScriptu mimo prohlížeč. Bylo původně vyvinuto ve Facebooku a nyní ho spravují vývojáři z Facebooku i Instagramu. Node.js funguje na JavaScriptovém enginu V8 a využívá architekturu založenou na událostech a neblokujících vstupně-výstupních operacích, což jej činí vhodným pro aplikace v reálném čase.

Hlavní předností Node.js je jeho schopnost zpracovávat více klientů současně pomocí jednoho vlákna díky "Single Threaded Event Loop" architektuře. Tato vlastnost redukuje potřebu zdrojů a paměti, což zrychluje vykonávání úloh.

Node.js není programovací jazyk ani framework, ale prostředí, ve kterém můžete běžet JavaScriptové aplikace. Primárně se využívá pro backend, ale díky své flexibilitě a škálovatelnosti nachází uplatnění i na frontendu.

Mezi klíčové vlastnosti Node.js patří:

Jednoduchost: Node.js je snadno přístupné začátečníkům s mnoha tutoriály a rozsáhlou komunitou.

Škálovatelnost: Díky své jednovláknové povaze zvládá Node.js obrovské množství simultánních spojení.

Rychlost: Ne-blokující provádění vláken činí Node.js rychlejším a efektivnějším.

Balíčky: Existuje rozsáhlý ekosystém open-source balíčků v NPM (Node Package Manager), který zjednodušuje práci vývojářům.

Node.js se běžně používá pro real-time komunikaci, aplikace Internetu věcí (IoT), streamování dat a komplexní jednostránkové aplikace (SPA), kde jeho event-loop efektivně zpracovává požadavky v neblokujícím režimu.

Node.js je tedy ideální volbou pro vývoj aplikací, které vyžadují vysokou úroveň interakce uživatele v reálném čase a zpracování velkého množství dat s minimálním zpožděním [19].

2.7.2 SQL/NoSQL

SQL (Structured Query Language) je tradiční přístup pro práci s relačními databázemi, který se používá od 70. let 20. století. SQL databáze jsou strukturované do tabulek s řádky a sloupci, což umožňuje efektivní ukládání a získávání dat pomocí komplexních dotazů. Tyto databáze jsou vhodné pro aplikace, které vyžadují pevně definovaná schémata a jsou závislé na pevných vztazích mezi daty.

Na druhé straně, NoSQL databáze poskytují flexibilní přístup k ukládání dat, který není omezen pevnými schématy. NoSQL, což znamená "not only SQL", zahrnuje databáze, které mohou podporovat různé datové struktury, jako jsou dokumenty, grafy, klíč-hodnota páry nebo široké sloupce. Tyto systémy jsou ideální pro práci s nestrukturovanými daty a jsou schopné efektivního horizontálního škálování, což je užitečné pro aplikace vyžadující rychlé zpracování velkých objemů dat.

NoSQL databáze jsou vhodné pro real-time webové aplikace, velké datové sklady a projekty vyžadující rychlou iteraci vývoje, protože umožňují vývojářům přidávat nové atributy bez nutnosti reorganizovat celé databázové schéma. Na rozdíl od SQL databází, které vynikají v zajištění transakční integrity a komplexních dotazů, NoSQL databáze nabízejí vysokou dostupnost a flexibilitu při práci s velkými datovými objemy rozdělenými přes mnoho serverů.

Ve výběru mezi SQL a NoSQL by měl být zohledněn typ aplikace, požadavky na výkon, a očekávané škálování. Zatímco SQL databáze mohou být ideální pro aplikace vyžadující pečlivě strukturované dotazy a datovou integritu, NoSQL je lepší volbou pro projekty, které potřebují rychle zpracovat velké množství různorodých dat [20].

2.8 Cloudové služby a API

Poskytovatelé cloudových služeb, jako jsou Amazon Web Services (AWS), Google Cloud Platform (GCP) a Microsoft Azure, nabízejí širokou škálu služeb pro vývoj, hostování a škálování aplikací. Tyto služby zahrnují výpočetní kapacity, databázové služby, ukládání dat, AI a machine learning, a mnoho dalších.

API umožňují aplikacím komunikovat mezi sebou a sdílet data a funkcionality. RESTful a GraphQL API jsou běžně používané pro vývoj webových a mobilních aplikací, poskytují standardizovaný způsob interakce s backendovými systémy.

2.9 Firebase jako platforma pro serverovou stranu a autentifikaci

Výběr Firebase jako platformy pro serverovou stranu a autentifikaci aplikace byl motivován několika klíčovými výhodami, které tato služba nabízí, a s ohledem na konkrétní potřeby a strukturu projektu. Zde jsou hlavní důvody, pro které bylo rozhodnuto využít Firebase, a některé potenciální nevýhody, které může tato volba přinést.

Firebase poskytuje kompletní backendové řešení bez nutnosti správy serverů, což umožňuje soustředit se na vývoj klienta bez starostí o údržbu infrastruktury. Pro práci s databází bylo rozhodnuto využít Firestore z ekosystému Firebase, což umožňuje strukturované a efektivní ukládání dat s možností real-time synchronizace. Tento přístup podporuje rychlý vývoj a snadnou správu datových modelů, jakými jsou tarify, odečty a uživatelské údaje.

Databázové služby Firebase, jako jsou Realtime Database a Firestore, umožňují efektivní a real-time manipulaci s daty. Vzhledem k tomu, že má aplikace zahrnuje funkce jako chat, je schopnost Firebase synchronizovat data v reálném čase napříč všemi klienty bez složitých konfigurací velkou výhodou.

Firebase Authentication poskytuje jednoduché a bezpečné řešení pro správu uživatelských účtů a integraci různých metod přihlašování, což šetří čas, který by jinak musel být věnován implementaci těchto funkcí. Tento systém umožňuje rychlou a bezpečnou autentizaci

uživatelů, podporuje vícefaktorovou autentifikaci a integraci s hlavními poskytovateli identity, jako jsou Google, Facebook a Twitter.

Jako platforma integrovaná s Google Cloud, Firebase usnadňuje využívání dalších služeb Google, jako jsou analytika, cloudové úložiště nebo strojové učení, což rozšiřuje možnosti aplikace. Díky bohatým SDK a jednoduchému API mohu rychle vyvíjet aplikaci, což je zásadní pro agilní vývoj a iteraci.

Nicméně, výběr Firebase přináší i určité nevýhody. Ačkoliv Firebase nabízí zdarma úroveň, náklady mohou rychle narůstat s rostoucím počtem uživatelů a zvýšeným využíváním dat. Je důležité pečlivě monitorovat a spravovat využití zdrojů, aby se předešlo nečekaně vysokým nákladům. Firebase klade určité limity na operace databáze a počet současně připojených uživatelů ve free tieru. Pro velké nebo rychle rostoucí aplikace může být nutné přejít na placený plán. Použití Firebase znamená značnou závislost na službách Google. Jakékoli změny v politice, cenách, nebo ukončení podpory služby by mohly vyžadovat náročné změny na straně aplikace. Ačkoliv Firestore poskytuje flexibilnější možnosti dotazování než Realtime Database, stále existují omezení ve srovnání s tradičními relačními databázemi, což může být omezující pro složité datové modely. Přesun dat z Firebase do jiné platformy může být komplikovaný, pokud rozhodnu v budoucnu změnit backendovou technologii.

Zvažování těchto výhod a nevýhod bylo klíčové pro rozhodovací proces a nakonec výběr Firebase jako řešení, které nejlépe vyhovuje potřebám aplikace, zatímco poskytuje flexibilitu a efektivitu potřebnou pro rychlý vývoj [21] [22].

II. PRAKTICKÁ ČÁST

3 ÚVOD DO APLIKACE

Aplikace je určena pro analýzu elektrické spotřeby a funguje nezávisle na dodavateli energie. Nabízí uživatelům možnost nastavit si vlastní tarify a upravit ceny podle svých potřeb. Uživatelé mohou v aplikaci ukládat svá data, provádět výpočty s různými tarify a také si prohlédnout odhady své spotřeby na konci měsíce podle aktuálního tarifu od svého dodavatele energie. Tato flexibilita umožňuje uživatelům lepší kontrolu nad svými náklady na energie a pomáhá jim optimalizovat jejich spotřebu.

3.1 Požadavky na aplikaci

Před zahájením programování aplikace je nezbytné stanovit jak funkční, tak nefunkční požadavky. Funkční požadavky zajišťují správnou funkčnost aplikace a umožňují její bezproblémové používání. Nefunkční požadavky na druhé straně zabezpečují aplikaci z hlediska bezpečnosti a zaručují kompatibilitu s relevantními verzemi operačních systémů na mobilních zařízeních a ve webových prohlížečích.

3.1.1 Funkční požadavky

ID požadavku	Popis požadavku
FUN1	Systém umožní uživatelům registrovat se pomocí e-mailu, hesla.
FUN2	Uživatelé se mohou přihlásit pomocí svého e-mailu a hesla.
FUN3	Systém umožní uživatelům obnovit zapomenuté heslo prostřednictvím e-mailové adresy.
FUN4	Uživatelé mohou přidávat, mazat tarify pro kalkulaci energetických nákladů.
FUN5	Systém vypočítá a zobrazí měsíční spotřebu elektřiny a náklady na základě zadaných odečtů a tarifů.
FUN6	Uživatelé mohou zobrazit historii své spotřeby elektřiny v časové ose.
FUN7	Systém poskytne možnost porovnat dva různé tarify a ukázat, který je výhodnější.

Tabulka 1: Funkční požadavky

3.1.2 Nefunkční požadavky

ID požadavku	Popis požadavku	
NFUN1	Systém musí být vždy dostupný	
NFUN2	Uživatelská data musí být chráněna	
NFUN3	Webové rozhraní musí být responzivní a funkční na všech zařízeních (desktop, tablet, smartphone).	
NFUN4	Stránky by se měly načítat do 3 sekund a odpovědi na dotazy by neměly trvat déle než 2 sekundy.	
NFUN5	Uživatelské rozhraní by mělo být intuitivní a snadno navigovatelné pro všechny typy uživatelů.	
NFUN6	Systém by měl být navržen tak, aby bylo možné snadno přidávat nové funkce nebo integrace.	
NFUN7	Aplikace by měla být kompatibilní s hlavními prohlížeči, jako jsou Chrome, Firefox, Safari a Edge	

Tabulka 2: Nefunkční požadavky

3.2 Implementace aplikace

Pomocí frameworku Angular s jazykem TypeScript pro všechny tři platformy bylo možné vytvořit jednotnou aplikaci, která funguje na Androidu, iOS a také jako webová aplikace. Volba Ionicu, Angularu a TypeScriptu zajistila efektivní správu stavu a reaktivní programování, což vede k lepší udržitelnosti kódu a snazší údržbě aplikace.

3.2.1 Bezpečnost aplikace

Data jsou bezpečně ukládána a spravována pomocí cloudových technologií Firebase, což zahrnuje autentizaci, autorizaci a ukládání dat v databázi. Všechny operace, jako je mazání a aktualizace údajů, jsou prováděny s důrazem na ochranu uživatelských dat a soukromí.

3.3 Implementace

3.3.1 Přihlašovací obrazovka

Pro úspěšném přihlášení na webovou stránku je implementováno uživatelské rozhraní pomocí Angularových reaktivních formulářů, kde *FormGroup* a *FormControl* slouží k zajištění validace (viz Zdroj. kód 1) a správy stavu formulářů. Změny ve formuláři jsou sledovány pomocí *valueChanges observables*, které umožňují reagovat na změny v reálném čase a aktualizovat stav aplikace podle uživatelského vstupu.

```
1 <form [formGroup]="ionicForm"  novalidate>
2 <div style="padding-top: -26px;padding-bottom: 16px;" *ngIf="this.ionicForm.con-
trols?.['email'].touched && this.ionicForm.controls?.['email'].invalid">
3         <ion-text *ngIf="errorControl.email.errors?.required" class="errorInput" color="dan-
ger">Email je vyžadován.</ion-text>
4         <ion-text *ngIf="errorControl.email.errors?.pattern" class="errorInput" color="dan-
ger">Prosím zadejte validní email.</ion-text>
5       </div>
6 </form>
```

Zdroj. kód 1 :Zdrojový kód FormGroup

V případě, že uživatel vyplní formulář a údaje projdou validací, je volána funkce *login* (viz Zdroj. kód 2), která pomocí služby *Firebase* a její metody *signInWithEmailAndPassword* (viz Zdroj. kód 3) zkusí uživatele autentizovat. Jakékoli výsledky tohoto pokusu jsou zpracovány pomocí RxJS subscribe, což odpovídá metodě then v Promises, zatímco chyby jsou zachytávány a zpracovávány *v catchError*. Pokud je přihlášení úspěšné, uživatel je přesměrován na požadovanou stránku, což je reakce na úspěšnou autentizaci.

```
 1 async login() {
 2     const loading = await this.loadingController.create();
 3     await loading.present();
 4     if (this.ionicForm.valid) {
 5       try {
 6         const userCredential = await this.authService.loginUser(this.ionicForm.value.email,
this.ionicForm.value.password);
 7         if (userCredential) {
 8           console.log('uživatel je přihlášen');
 9           await loading.dismiss();
10           this.router.navigate(['/tabs/calculator']);
11         } else {
12           throw new Error('přihlášení selhalo');
13         }
14       } catch (error) {
15         console.error(error);
16         await this.presentToast('Příhlášení selhalo prosím zkontrolujte že jste zadali
správné data');
17         await loading.dismiss();
18       }
19     } else {
20       console.log('Prosím vyplňte všechny hodnoty');
21       loading.dismiss();
22       await this.presentToast('Prosím vyplntě všechny hodnoty');
23     }
```

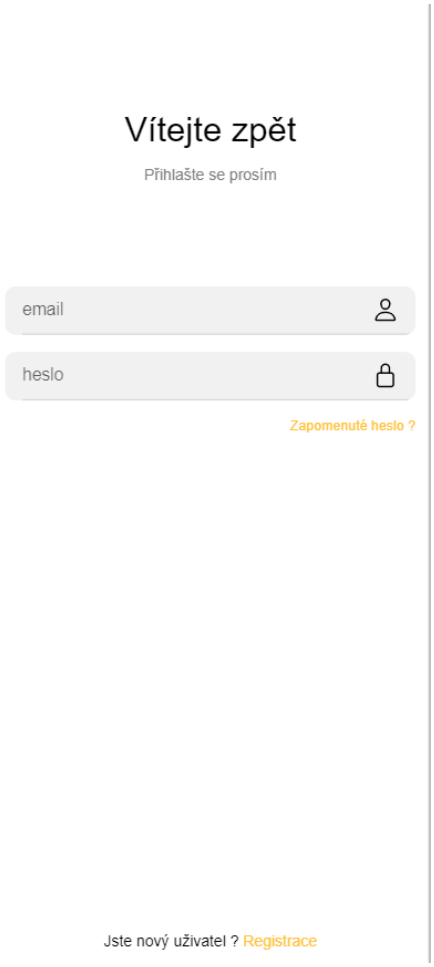
```
24    }
```

Zdroj. kód 2: Přihlašovaní uživatele

```
1 loginUser(email: string, password: string): Promise<any> {
2     return this.afAuth.signInWithEmailAndPassword(email, password);
3   }
```

Zdroj. kód 3: Funkce loginUser

Na přihlašovací stránce je uživatelům k dispozici formulář pro přihlášení, který vyžaduje uživatelské jméno ve formě emailové adresy a heslo (viz Obr. 5). V případě, že uživatel zapomene své heslo, je zde implementována funkce pro jeho obnovení. Po zadání přihlašovacích údajů se v pravém dolním rohu obrazovky zobrazí tlačítko pro odeslání formuláře. Pro osoby, které nejsou zaregistrovány, je na spodní části stránky umožněn přístup na další stránku, kde se mohou zaregistrovat.



Obr. 5: Přihlašovací obrazovka

3.3.2 Registrační obrazovka

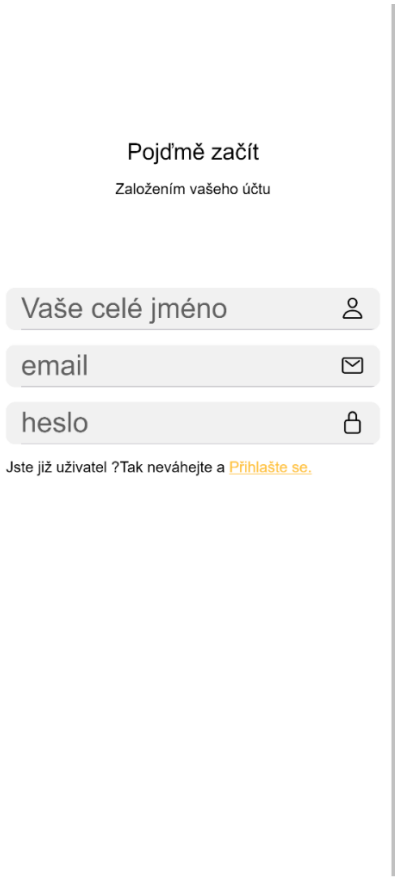
Na registrační obrazovce webové aplikace byl implementoval formulář, který vyžaduje, aby uživatelé zadali následující povinné údaje: celé jméno, e-mail a heslo (viz Obr. 6). Zde jsou specifické podmínky, které byly nastaveny pro každé pole:

Systém zahrnuje zobrazení chybových hlášení v případě, že uživatel zadá neplatné údaje, což pomáhá okamžitě identifikovat, co je potřeba opravit. Po úspěšném vyplnění a odeslání formuláře je zpracována registrace prostřednictvím Firebase (viz Zdroj. kód 4).

```
1   async registerUser(email:string,password:string ,fullname:string) {
2     return await  this.ngFireAuth.createUserWithEmailAndPassword(email,password)
3   }
```

Zdroj. kód 4: Funkce registerUser

V případě jakýchkoliv chyb při registraci, jako je například "uživatel již je zaregistrovaný s daným e-mailem" nebo "e-mail je ve špatném formátu", se zobrazí příslušné chybové hlášky. Tyto chyby jsou uživateli prezentovány červeným a tučným písmem, aby byly snadno viditelné.



Obr. 6: Registrační obrazovka

Pro úspěšnou registraci musí uživatel vyplnit tato pole:

Celé jméno

E-mail: Tento údaj musí být ve správném formátu, obsahující zavináč a správnou doménu, a je nezbytný pro registraci.

Heslo: Pro zabezpečení účtu musí heslo obsahovat kombinaci velkých a malých písmen, čísel a speciálních znaků s minimální délkou osmi znaků.

3.3.3 Obrazovka zapomenutého hesla

Na obrazovce pro obnovu zapomenutého hesla byl implementován jednoduchý formulář, který vyžaduje od uživatele zadání jeho e-mailové adresy. Proces obnovy hesla je řízen pomocí autentizačního servisu, který je součástí aplikace a poskytuje funkci *resetPassword*.

Zde je stručný popis přístupu a implementace:

Uživatelské rozhraní: Uživatel je vyzván, aby zadal svůj e-mail do textového pole. Toto je jediné pole na této obrazovce, a jeho účel je jasný a přímočarý.

Funkce resetování hesla: Po zadání e-mailu uživatelem je volána funkce *resetPassword* (viz Zdroj. kód 5) autentizačního servisu. Tato funkce přijímá e-mail jako parametr a odesílá odkaz pro resetování hesla na tento e-mail.

```
async resetPassword(email:string){
    return await this.ngFireAuth.sendPasswordResetEmail(email)
  }
```

Zdroj. kód 5: Funkce resetovaní hesla

Potvrzení odeslání: Jakmile je odkaz na reset hesla úspěšně odeslán, zobrazím uživateli toast notifikaci s informací, že odkaz byl odeslán (viz Zdroj. kód 6). Notifikace se zobrazuje na spodní části obrazovky a má nastavenou dobu trvání 2000 milisekund.

Přesměrování po akci: Po zavření toastu je uživatel automaticky přesměrován zpět na přihlašovací obrazovku. Toto přesměrování je implementováno jako reakce na událost *onDidDismiss* toastu.

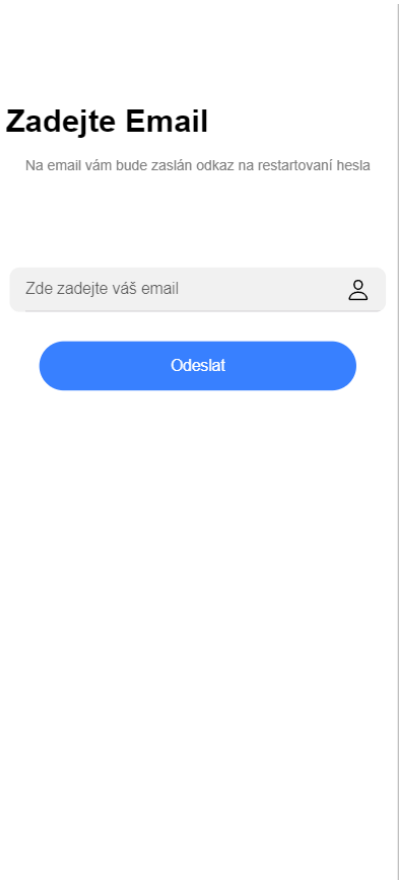
```
async presentToast() {
    const toast = await this.toastController.create({
      message: 'Odkaz na obnovení hesla byl odeslán na váš e-mail',
      duration: 2000, // Duration in milliseconds
      position: 'bottom' // Position of the toast (top, bottom, middle)
    });

    toast.present();
    toast.onDidDismiss().then(()=>{
      this.router.navigate(['/login']);
```

```
11      })
12    }
```

Zdroj. kód 6: Funkce presentToast()

Tímto způsobem bylo zajištěno, že uživatelé, kteří zapomněli své heslo, mohou snadno zahájit proces jeho obnovy. Uživatelské rozhraní je navrženo tak, aby bylo co nejpřístupnější a nejsnazší pro použití, což minimalizuje frustraci uživatelů při zvládání potenciálně stresující situace (viz. Obr. 7).

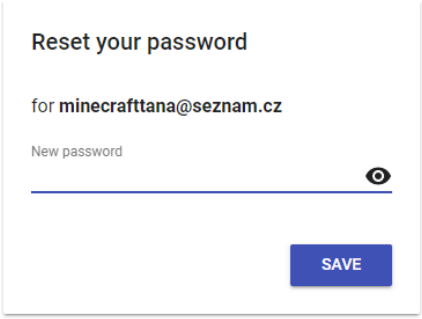


Obr. 7: Obrazovka zapomenutého hesla

3.3.4 Vizualizace obnovy hesla

Email s odkazem pro resetování hesla (viz Obr. 8) je uživateli zaslán po ověření jeho e-mailové adresy. Po kliknutí na tento odkaz je uživatel přesměrován na stránku, kde si může nastavit nové heslo (viz Obr. 9), kterým se ihned může přihlásit. Tímto způsobem je zajištěn hladký a bezpečný proces obnovy hesla.

Obr. 8: Ukázka emailu zapomenutého hesla



Obr. 9: Okno pro změnu zapomenutého hesla

3.3.5 Obrazovka kalkulátoru

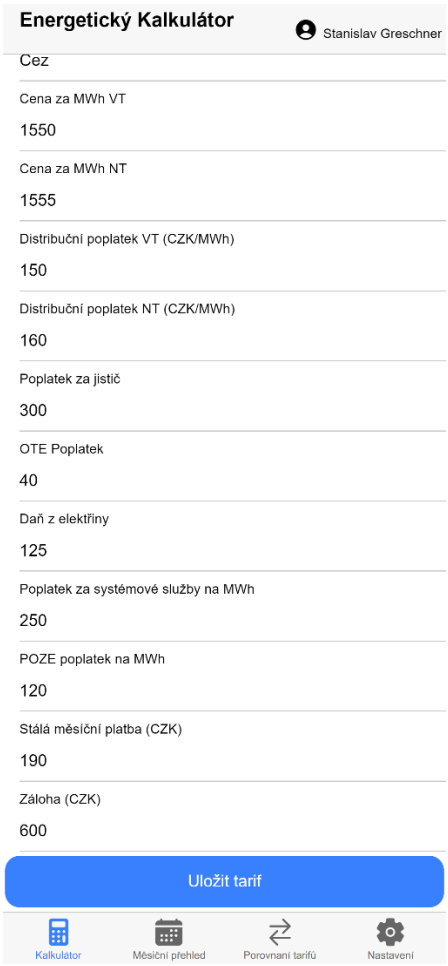
V rámci obrazovky kalkulátoru ve své aplikaci umožňuji uživatelům vypočítat a spravovat spotřebu elektřiny a s ní spojené náklady podle jejich osobních tarifů a odečtů. Uživatelé mohou přidávat nové tarify (viz Obr. 10) pomocí funkce *saveTariff* (viz Zdroj. kód 7) a zaznamenávat počáteční a aktuální stavy spotřeby, což umožňuje detailní sledování a výpočet předpokládaných měsíčních nákladů.

```
async saveTariff(tariff: Tariff, globalUID: string): Promise<void> {
    if (!tariff.name) {
      // Vytvoření a zobrazení alertu místo vypsání chyby do konzole
      await this.presentAlert('Chyba', 'Název tarifu chybí, nelze uložit tarif');
      return Promise.reject('Tariff name is missing');
    }

    try {
      // Zajistěte, že cesta k dokumentu zahrnuje subkolekci
      await this.firestore.doc(`tariffs/${globalUID}/userTariffs/${tariff.name}`).set(tariff, { merge: true });
      // Zobrazení potvrzení o úspěšném uložení
      await this.presentAlert('Úspěch', 'Tarif byl úspěšně uložen');
    } catch (error) {
      // Zobrazení chybové zprávy, pokud nastane problém při ukládání
      await this.presentAlert('Chyba při ukládání', 'Nastal problém při ukládání tarifu. Prosím, zkuste to znovu.');
      console.error('Error saving tariff:', error);
      return Promise.reject(error);
    }
  }
```

Zdroj. kód 7: Funkce SaveTariff

Hlavní funkcionalita kalkulátoru zahrnuje přidání tarifů, kde uživatelé mohou prostřednictvím formuláře specifikovat název tarifu, ceny za MWh pro vrcholové a mimo vrcholové období, různé poplatky a daně. Tato data jsou uložena a jsou přístupná pro každého uživatele individuálně. Dále mohou uživatelé zadávat počáteční a aktuální stavy své spotřeby elektřiny, které slouží jako základ pro výpočet celkové spotřeby a nákladů.

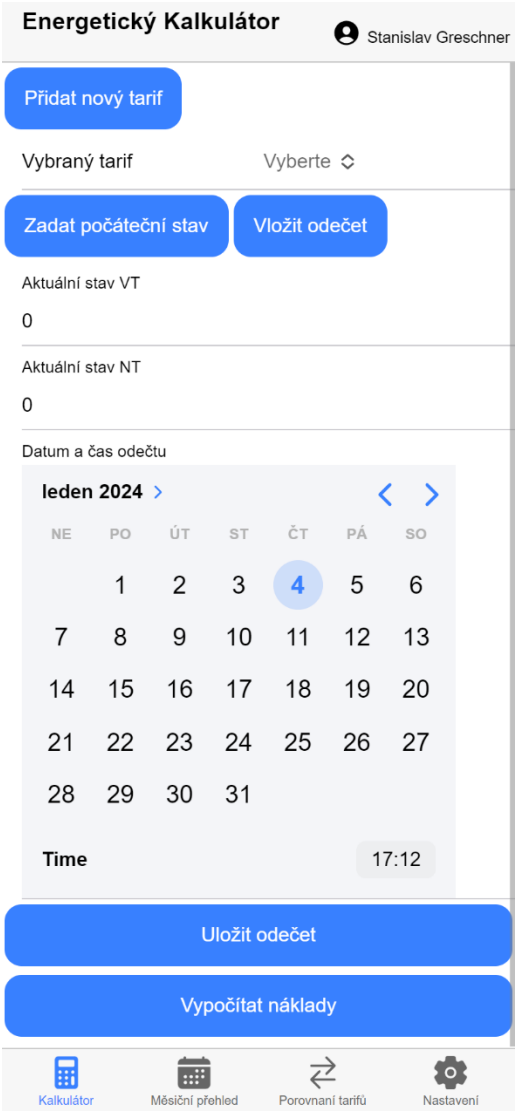


Obr. 10: Ukázka ukládání tarifu

Pro výpočet spotřeby a nákladů je k dispozici funkce, která po zpracování vstupních údajů určí průměrnou denní spotřebu, celkovou spotřebu za měsíc a odhadované měsíční náklady. Výsledky jsou zobrazeny uživatelům v přehledné a srozumitelné formě (viz Obr. 11), včetně porovnání s předem zadanou zálohou.

Obr. 11: Ukázka výsledků výpočtu

Interaktivní prvky (viz Obr. 12) uživatelského rozhraní, jako jsou rozbalovací formuláře a výběrové seznamy, zvyšují uživatelský komfort tím, že umožňují snadnou navigaci a rychlé přepínání mezi různými úkony. Data související s tarify, spotřebou a náklady jsou bezpečně ukládána a spravována s využitím cloudových služeb Firebase, což zahrnuje funkcionalitu pro autentizaci uživatelů a databázové operace.



Obr. 12: Ukázka zadávání odečtu

Další důležitou funkcionalitou kalkulátoru je automatické zpracování situací, kdy uživatel zapomene zadat počáteční stav spotřeby elektřiny. V takovém případě je počáteční stav vypočítán na základě průměru z předchozích odečtů, což zajišťuje nepřetržitou přesnost výpočtů a umožňuje uživatelům pokračovat v monitorování a správě svých nákladů i bez kompletních dat. Tento mechanismus je implementován prostřednictvím funkce *checkAndUpdateInitialState()*, která se automaticky spouští při každém zavolání funkce *saveReading()* (viz Zdroj. kód 8) která ukládá zadané odečty.

```
saveReading() {
        const timestamp = this.convertDateToMilliseconds(this.readingDate);
        console.log(`Reading timestamp to be saved: ${timestamp}`);
        const readingInstance = new ReadingClass(this.currentVT, this.currentNT, time-
stamp);
        this.dataService.saveReading(readingInstance,this.globalUID).then(() => {
            console.log('Odečet byl úspěšně uložen.');
            this.checkAndUpdateInitialState();
        }).catch(error => {
            console.error('Došlo k chybě při ukládání odečtu:', error);
        });
    }
```

Zdroj. kód 8: Funkce saveReading

Funkce *checkAndUpdateInitialState()* (viz Zdroj. kód 9) v aplikaci slouží jako facilitátor pro aktualizaci počátečního stavu spotřeby, pokud nebyl pro aktuální měsíc zadán. Tato funkce sama o sobě neprovádí žádné kontroly; namísto toho přímo volá metodu *calculateAndSetInitialReadingIfNewMonth()* z datového servisu. Tímto způsobem se zjednodušuje logika a zvyšuje se efektivita správy kódu tím, že se odpovědnost za výpočet a aktualizaci deleguje přímo na specializovanou metodu v rámci datového servisu.

```
checkAndUpdateInitialState() {
    this.dataService.calculateAndSetInitialReadingIfNewMonth(this.globalUID);
  }
```

Zdroj. kód 9: Funkce checkAndUpdateInitialState

Metoda *calculateAndSetInitialReadingIfNewMonth()* (viz Zdroj. kód 10) je jádrem systému pro zajištění plynulého přechodu mezi měsíci ve spotřebě elektřiny. Tato metoda nejprve načte poslední dva dostupné odečty pro daného uživatele. Pokud jsou k dispozici, porovná časová razítka těchto odečtů a určí, zda došlo k přechodu mezi měsíci. Pokud ano, vypočítá průměrnou denní spotřebu za vrcholové a mimo vrcholové období a následně odhadne počáteční stav pro nový měsíc na základě těchto průměrů a dnů uplynulých od posledního záznamu do prvního dne nového měsíce. Výsledný počáteční stav je pak uložen do systému.

```
 1 calculateAndSetInitialReadingIfNewMonth(globalUID: string): void {
 2         this.getLastTwoReadings(globalUID).subscribe(readings => {
 3             if (readings.length < 2) {
 4                 console.log('Not enough readings to calculate new initial state.');
 5                 return;
 6             }
 7             const [latestReading, previousReading] = readings;
 8             const latestDate = new Date(latestReading.timestamp);
 9             const previousDate = new Date(previousReading.timestamp);
10             if (latestDate.getMonth() !== previousDate.getMonth() || latestDate.get-
FullYear() !== previousDate.getFullYear()) {
11                 const daysDifference = (latestDate.getTime() - previousDate.getTime()) /
(1000 * 3600 * 24);
12                 const avgDailyVT = (latestReading.VT - previousReading.VT) / daysDiffe-
rence;
13                 const avgDailyNT = (latestReading.NT - previousReading.NT) / daysDiffe-
rence;
14                 const firstDayNextMonth = new Date(previousDate.getFullYear(), previ-
ousDate.getMonth() + 1, 1);
15                 const daysUntilMonthStart = (firstDayNextMonth.getTime() - previousDate.ge-
tTime()) / (1000 * 3600 * 24);
16                 const estimatedVT = previousReading.VT + (avgDailyVT * daysUntil-
MonthStart);
17                 const estimatedNT = previousReading.NT + (avgDailyNT * daysUntil-
MonthStart);
18                 const initialState = {
19                     vtConsumption: estimatedVT,
20                     ntConsumption: estimatedNT,
21                     timestamp: firstDayNextMonth.getTime(),
22                     userId: globalUID
23                 };
24                 this.saveInitialState(initialState, globalUID);
25             } else {
26                 console.log('No month transition detected, no need to update initial
state.');
27             }
28         }, err => {
29             console.log('Error getting documents', err);
30         });
31     }
```

Zdroj. kód 10: Funkce calculateAndSetInitialReadingIfNewMonth

Tímto způsobem aplikace zajistí, že uživatelé nejsou penalizováni za chybějící data a mohou bez problémů pokračovat v monitorování a správě své spotřeby. Automatická aktualizace počátečních stavů tedy přináší významné zvýšení uživatelského komfortu a zajišťuje přesnost dat pro další výpočty spotřeby a finančních nákladů spojených s energetickou spotřebou.

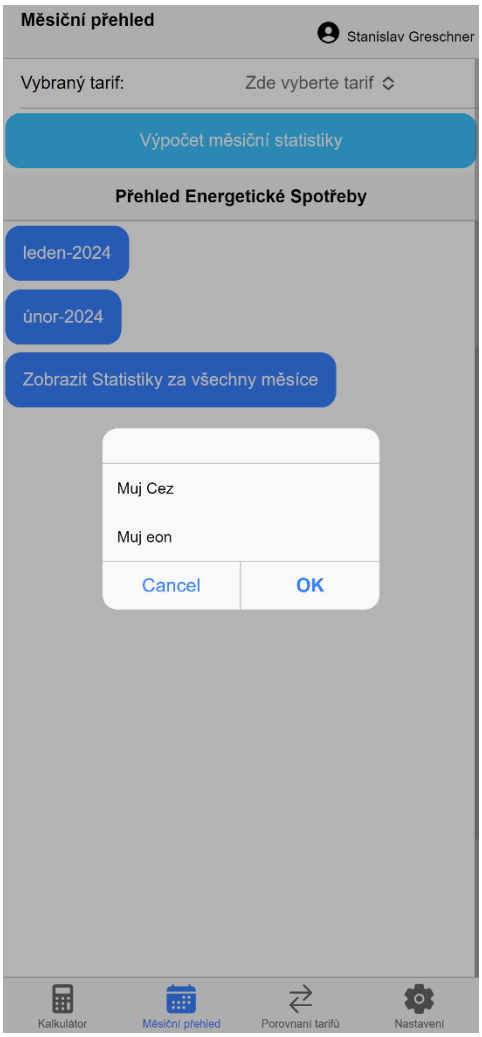
Kalkulátor tedy představuje klíčový nástroj v aplikaci, který uživatelům poskytuje efektivní prostředky pro správu a optimalizaci jejich energetické spotřeby a finančních výdajů, což je v dnešní době vysoce relevantní a žádaná funkcionalita.

3.3.6 Obrazovka měsíční přehled

Na stránce měsíčního přehledu ve aplikaci je uživatelům poskytován komplexní přehled o jejich energetické spotřebě a finančních nákladech spojených s tarify za elektřinu. Uživatelé mohou vybrat specifický měsíc a zobrazit detailní informace o spotřebě, nákladech a úsporách jak na denní, tak na celkové měsíční bázi. Funkce *loadMonthlyCalculations(globalUID: string)* (viz Zdroj. kód 11) hraje klíčovou roli v načítání těchto dat z Firebase Firestore, aktualizujíc statistiky energetické spotřeby v reálném čase.

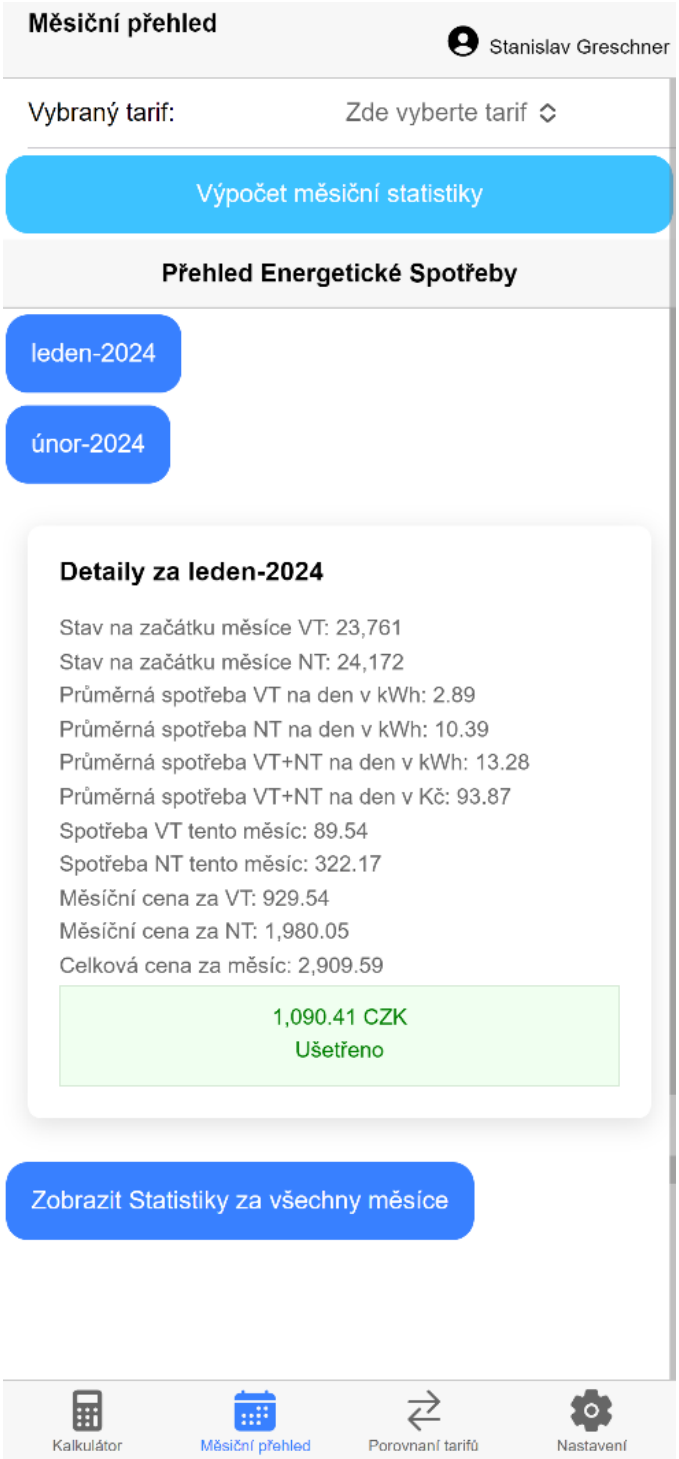
```
1 loadMonthlyCalculations(globalUID: string): void {
2         this.dataService.getMonthlyCalculations(globalUID).subscribe(data => {
3             this.monthlyCalculations = data;
4             console.log(data);
5             this.calculateTotalSavings();  // Calculate total savings after data is loaded
6         }, error => {
7             console.error('Failed to load monthly calculations:', error);
8         });
9 }
```

Zdroj. kód 11: Funkce loadMonthlyCalculations



Obr. 13: Ukázka výběru tarifu

Další hlavní funkcionalita je propojena s interaktivním tlačítkem na uživatelském rozhraní, které, když je stisknuto, aktivuje funkci *calculateMonthlyConsumption()* (viz Zdroj. kód 12). Tato funkce zajišťuje výpočet a zobrazení údajů o spotřebě a finančních nákladech za vybraný měsíc. Provádí detailní analýzu spotřebních dat a vypočítává celkové náklady na energii na základě aktuálně zvoleného tarifu (viz Obr. 13) a spotřebního vzoru.



Obr. 14: Ukázka detailů za měsíc

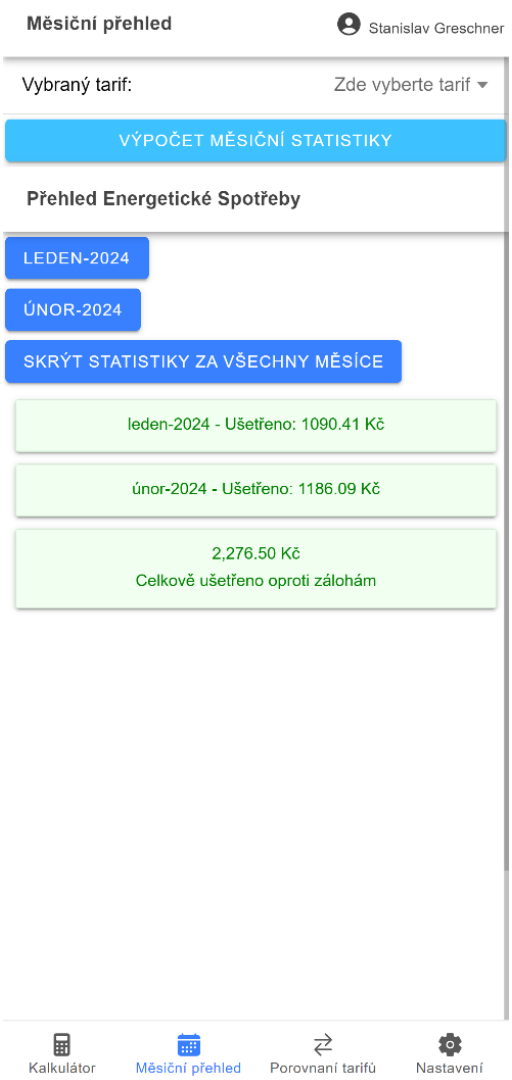
```
calculateMonthlyConsumption() {
    this.dataService.getLastTwoInitialStates(this.globalUID).subscribe(states => {
        if (states.length < 2) {
            console.error('Not enough initial states for calculation.');
            return;
        }
        const [latestState, previousState] = states;
        const days = (latestState.timestamp - previousState.timestamp) / (1000 * 3600 *
24);
        if (days <= 0) {
            console.error('Invalid time difference. Cannot calculate consumption.');
            return;
        }
        // Calculate total and average consumption
        const totalVTConsumption = latestState.vtConsumption - previousState.vtConsumption;
        const totalNTConsumption = latestState.ntConsumption - previousState.ntConsumption;
        const avgDailyVT = totalVTConsumption / days;
        const avgDailyNT = totalNTConsumption / days;
        const avgDailyTotal = avgDailyVT + avgDailyNT;
        // Calculate costs
        const costVT = this.calculateCost(totalVTConsumption, 'VT');
        const costNT = this.calculateCost(totalNTConsumption, 'NT');
        const totalCost = costVT + costNT;

        const savings = this.selectedTariff.deposit - totalCost;
        // Save all calculated data to Firebase
        this.saveCalculationData({
            initialVT: previousState.vtConsumption,
            totalMonthlyVT: totalVTConsumption,
            avgDailyVT: avgDailyVT,
            initialNT: previousState.ntConsumption,
            totalMonthlyNT: totalNTConsumption,
            avgDailyNT: avgDailyNT,
            monthlyCostNT: costNT,
            monthlyCostVT: costVT,
            avgDailyTotal: avgDailyTotal,
            dailyCostTotal: totalCost / days,
            totalCost: totalCost,
            timestamp: previousState.timestamp,
            Savings: savings
        });
    });
}
```

Zdroj. kód 12: Funkce calculateMonthlyConsumption

Statistiky o celkové spotřebě ve vrcholových a mimo vrcholových hodinách, průměrné denní spotřebě a nákladech, a celkových měsíčních nákladech jsou poskytovány, aby uživatelům pomohly lépe pochopit, jak jejich spotřeba ovlivňuje měsíční účty za energie (viz Obr. 14). Výpočet úspor, který ukazuje buď úsporu, pokud skutečná spotřeba nedosahuje předpokládané zálohy, nebo doplatek, pokud je spotřeba vyšší, je rovněž zobrazen.

Možnost zobrazení statistik za všechny měsíce, která uživatelům umožňuje rychle získat přehled o úsporách nebo doplatcích za každý měsíc, je také klíčovou funkcí. Tato možnost poskytuje uživatelům přehled o tom, jak se jejich spotřební návyky mění a jak efektivně spravují svou energetickou spotřebu v průběhu roku.



Obr. 15: Ukázka zobrazení statistiky za všechny měsíce

Interaktivní prvky, jako jsou výběrové seznamy a tlačítka pro zobrazení detailů (viz Obr. 15), usnadňují navigaci a poskytují přesné informace dle potřeby. Tato kombinace přehlednosti a funkcí, jako je *loadMonthlyCalculations* a *calculateMonthlyConsumption*, činí tuto stránku měsíčního přehledu efektivním nástrojem pro správu a analýzu energetické spotřeby a finančních závazků spojených s tarify za elektřinu.

3.3.7 Obrazovka porovnání tarifů

Na obrazovce porovnání tarifů v aplikaci mají uživatelé k dispozici nástroj umožňující efektivní porovnání dvou různých energetických tarifů na základě jejich osobní spotřeby. Tento proces porovnání začíná výběrem dvou tarifů z rozbalovacího seznamu, který je součástí uživatelského rozhraní implementovaného pomocí komponenty ion-select (viz Zdroj. kód

13). Každý tarif je vybírán ze seznamu dostupných tarifů, který je dynamicky načten a uživatelům nabízí možnost vybrat tarif pro každé porovnání zvlášť.

```
        <ion-item>
          <ion-label>Tarif 1:</ion-label>
          <ion-select [(ngModel)]="selectedTariff1" (ionChange)="loadAvailableMonths()">
            <ion-select-option *ngFor="let tariff of tariffs" [value]="tariff">
              {{ tariff.name }}
            </ion-select-option>
          </ion-select>
        </ion-item>
```

Zdroj. kód 13: Ukázka kódu výběru tarifů ze seznamu

Po výběru těchto tarifů uživatelé kliknou na tlačítko "Porovnat tarify". Toto tlačítko aktivuje funkci *compareAndSaveTariffs()* (viz Zdroj. kód 14), která nejprve ověří, zda byly oba tarify správně vybrány a zda je k dispozici globální uživatelské ID. Pokud některá z těchto informací chybí, uživatelé jsou upozorněni prostřednictvím vyskakovacího okna s chybou, které je vyzývá, aby se ujistili, že mají oba tarify vybrané.

```
compareAndSaveTariffs(): void {
    if (!this.selectedTariff1 || !this.selectedTariff2 || !this.globalUID) {
      this.presentAlert('Chyba', 'Ujistěte se, že máte oba dva tarify vybrané.');
      return;
    }

    this.dataService.getAllInitialStates(this.globalUID).subscribe(states => {
      if (states.length < 2) {
        this.presentAlert('Chyba', 'Nemáte žádný měsíc úspěšně uzavřený.');
        return;
      }

      states.sort((a, b) => a.timestamp - b.timestamp);

      for (let i = 0; i < states.length - 1; i++) {
        const startState = states[i];
        const endState = states[i + 1];
        this.processTariffCalculations(startState, endState);
      }
    }, error => {
      this.presentAlert('Chyba', 'Něco se pokazilo: ' + error);
    });
  }
```

Zdroj. kód 14: Funkce comparAndSaveTariffs

Pokud jsou všechny potřebné údaje k dispozici, *compareAndSaveTariffs()* načte všechny počáteční stavy spojené s uživatelovým globálním ID. Funkce kontroluje, zda existují alespoň dva uzavřené měsíce pro analýzu. Jestliže data chybí, uživatel je opět upozorněn prostřednictvím vyskakovacího okna s chybou o nedostatku uzavřených měsíců. V opačném případě se načtená data seřadí podle časového razítka a funkce přechází k vlastnímu porovnání tarifů.

Pro každý uzavřený měsíc funkce *processTariffCalculations(startState, endState)* (viz Zdroj. kód 15) formátuje datum z časového razítka do podoby měsíce a roku a následně vypočítá měsíční náklady pro vrcholové i mimo vrcholové období obou tarifů pomocí funkce *calculateCosts(startState, endState, tariff)*. Tato funkce vypočítá náklady a spotřebu a připraví data pro zobrazení.

```
 1 processTariffCalculations(startState: StateData, endState: StateData): void {
 2     const month = this.formatMonthYear(startState.timestamp); // Month label from the end
state's timestamp
 3     const result1 = this.calculateCosts(startState, endState, this.selectedTariff1);
 4     const result2 = this.calculateCosts(startState, endState, this.selectedTariff2);
 5
 6     const tariffName = `${this.selectedTariff1.name} vs ${this.selectedTariff2.name}`;
 7     if (result1 && result2) {
 8       const combinedResult = {
 9         month,
10         totalVT: result1.totalMonthlyVT + result2.totalMonthlyVT,
11         totalNT: result1.totalMonthlyNT + result2.totalMonthlyNT,
12         monthlyCostNT1: result1.monthlyCostNT,
13         monthlyCostVT1: result1.monthlyCostVT,
14         monthlyCostNT2: result2.monthlyCostNT,
15         monthlyCostVT2: result2.monthlyCostVT,
16         dailyCostTotal1: result1.dailyCostTotal,
17         dailyCostTotal2: result2.dailyCostTotal,
18         totalCost1: result1.totalCost,
19         totalCost2: result2.totalCost
20       };
21       this.saveResultsToFirestore(month, combinedResult);
22     }
23   }
```

Zdroj. kód 15: Funkce processTariffCalculations

Výsledky jsou pak prezentovány na obrazovce (viz Obr. 16), kde funkce *getCostComparison(tariffResult)* (viz Zdroj. kód 16) poskytuje srovnání obou tarifů a vyhodnocuje, který z nich je finančně výhodnější. Tato funkce nabízí uživatelům jasný a srozumitelný přehled o rozdílu v nákladech mezi oběma tarify a umožňuje jim informovaně rozhodnout o nejvhodnějším tarifu pro jejich potřeby.

```
 1   getCostComparison(tariffResult: any): string {
 2     const totalCost1 = tariffResult.totalCost1;
 3     const totalCost2 = tariffResult.totalCost2;
 4     const difference = Math.abs(totalCost1 - totalCost2);
 5
 6     if (totalCost1 < totalCost2) {
 7       return `Tariff č1 je pro vás výhodnější o ${difference.toLocaleString()} Kč.`;
 8     } else if (totalCost2 < totalCost1) {
 9       return `Tariff č2 je pro vás výhodnější o ${difference.toLocaleString()} Kč.`;
10     } else {
11       return "Oba tarify jsou cenově ekvivalentní.";
12     }
13   }
```

Zdroj. kód 16: Funkce getCostComparison

Tato funkcionalita zvyšuje transparentnost a poskytuje uživatelům lepší kontrolu nad jejich energetickými výdaji, což přispívá nejen k efektivnějšímu hospodaření s energií, ale také k

udržitelnějšímu využívání energetických zdrojů. Tento komplexní a propracovaný systém je klíčem k optimalizaci nákladů a energetické efektivitě v domácnostech uživatelů.

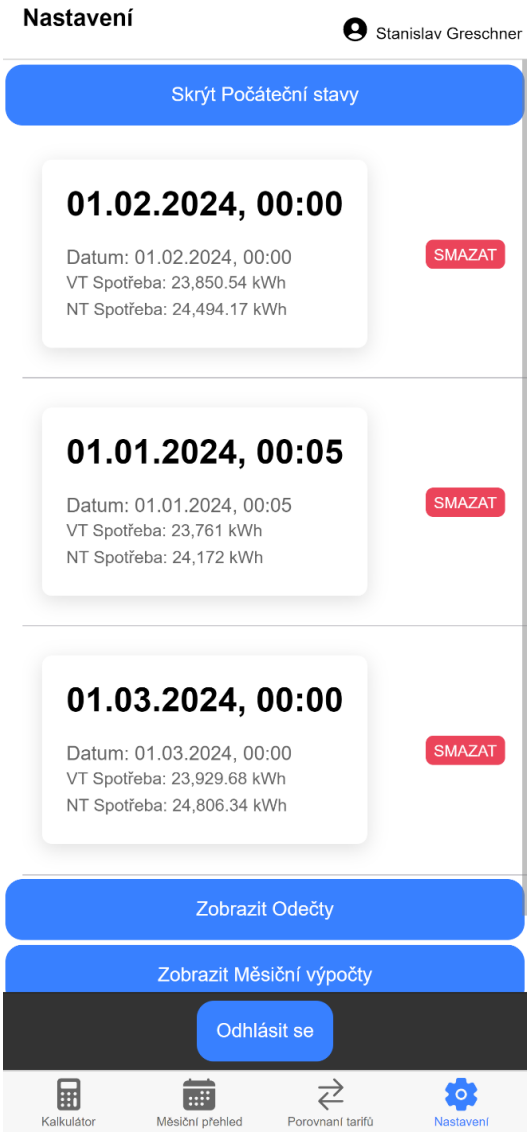
Obr. 16: Ukázka porovnaní tarifů

3.3.8 Obrazovka nastavení

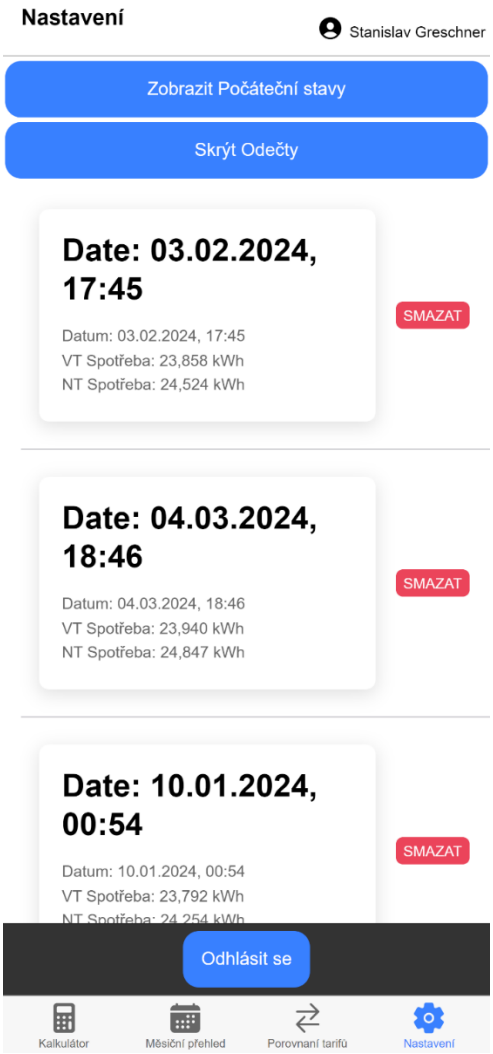
Obrazovka nastavení v mé aplikaci slouží jako centrální místo pro správu uživatelských dat a konfiguraci aplikace. Umožňuje uživatelům přehledně spravovat počáteční stavy, odečty, tarify a měsíční výpočty. Tato stránka je klíčová pro udržení kontroly nad osobními a finančními informacemi spojenými s energetickou spotřebou.

Správa počátečních stavů a odečtů

Uživatelé mohou prohlížet a odstraňovat záznamy o počátečních stavech každého měsíce (viz Obr. 17) a odečtech (viz Obr. 18). Každý záznam obsahuje podrobnosti o datu a množství spotřebované elektřiny, což uživatelům umožňuje sledovat jejich historickou spotřebu.



Obr. 17: Obrázek správy počátečních stavů



Obr. 18: Obrázek správy odečtů

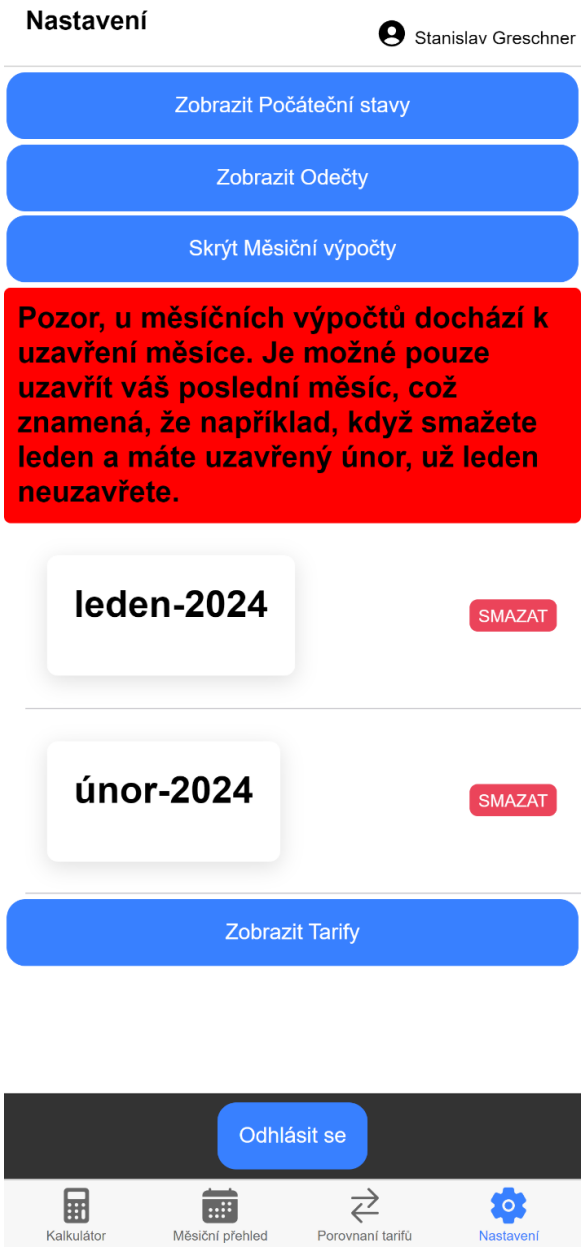
Přehled a správa tarifů

Nabízí možnost prohlížet detaily jednotlivých tarifů (viz Obr. 19) a případně je odstranit. Detaily zahrnují informace jako cena za jednotku, měsíční poplatky a další související náklady, které pomáhají uživatelům lépe porozumět struktuře jejich tarifů

Obr. 19: Obrázek tarifů

Zobrazení a správa měsíčních výpočtů

Uživatelé mají možnost nejen zobrazit, ale i odstranit záznamy uzavřených měsíců(viz Obr. 20), které ilustrují jejich měsíční finanční závazky na základě spotřeby. Tato funkce je klíčová pro uživatele, kteří chtějí mít možnost vymazat všechna data z aplikace, například pro účely obnovení nastavení aplikace do výchozího stavu.



Obr. 20: Obrázek měsíční výpočty

Funkce, která ručí za načtení dat, je implementována prostřednictvím metody *loadCollectionData(path: string, property: string, isRecursive: boolean = false)* (viz Zdroj. kód 17). Tato metoda načítá data z určené kolekce nebo skupiny kolekcí v závislosti na tom, zda je požadováno rekurzivní prohledávání. Data jsou poté předána a zobrazena v aplikaci, což umožňuje uživatelům správu a přehled nad všemi relevantními informacemi.

```
1 loadCollectionData(path: string, property: string, isRecursive: boolean = false) {
2     let collection: AngularFirestoreCollection<any> | AngularFirestoreCollectionGroup<any>;
3     if (isRecursive) {
4       // If recursive, you're looking for a collection group
5       collection = this.firestore.collectionGroup(path.split('/').pop()!);
6     } else {
7       // Regular collection
```

```
 8       collection = this.firestore.collection(path);
 9     }
10     collection.snapshotChanges().subscribe(data => {
11       this[property] = data.map(e => {
12         return {
13           id: e.payload.doc.id,
14           ...e.payload.doc.data() as any
15         };
16       });
17     }, error => {
18       console.error(`Failed to load data from ${path}:`, error);
19     });
20   }
```

Zdroj. kód 17: Funkce loadColletionData

Možnost odhlášení: Pro zvýšení bezpečnosti a soukromí mohou uživatelé snadno a bezpečně odhlásit z aplikace, což zamezí neoprávněnému přístupu k jejich údajům.

3.4 Struktura databáze

Struktura databáze ve Firestore je optimalizována tak, aby reflektovala specifické potřeby aplikace pro správu energetických tarifů a spotřeby uživatelů. Zde jsou příklady klíčových entit a jejich úložiště ve Firestore:

3.4.1 Tariff:

Kolekce: /tariffs/{userID}/userTariffs

Každý dokument obsahuje detaily o jednom energetickém tarifu, včetně názvu, cen za vrcholové a mimo vrcholové období, daní, systémových služeb a předplacených záloh.

3.4.2 Reading:

Kolekce: /readings/{userID}/userreadings

Dokumenty zaznamenávají odečty elektrické energie pro vrcholové (VT) a mimo vrcholové (NT) období, doplněné o časové razítko, které ukazuje, kdy byl odečet proveden.

3.4.3 InitialState a StateData:

Kolekce: /initialStates/{userID}/states

Ukládají informace o počátečních a aktuálních stavech spotřeby energie, které jsou klíčové pro výpočet měsíčních nákladů a spotřeby.

3.4.4 ComparisonResult:

Kolekce: /TariffComparison/{userID}/Months/{month}/Comparisons

Dokumenty obsahují výsledky porovnání tarifů, například "Muj Cez vs Muj eon", které poskytují uživatelům podrobné srovnání nákladů a spotřeby.

3.4.5 Calculations:

Kolekce: /calculations/{userID}/monthlyCalculations/{month}

Uchovává výsledky měsíčních výpočtů spotřeby a nákladů pro každý měsíc, jako je "leden-2024". Tyto informace jsou základem pro finanční plánování a účetnictví spotřeby energie.

Firestore databáze umožňuje dynamické a flexibilní spravování dat díky své struktuře kolekcí a dokumentů, což je ideální pro aplikace vyžadující rychlé a škálovatelné řešení pro správu velkého množství uživatelských dat. Každý uživatel interaguje pouze se svými daty, díky unikátnímu User ID, což zajišťuje bezpečnost a izolaci dat. Integrace s Firebase Auth poskytuje robustní autentizační mechanismy a usnadňuje správu uživatelských sesí, což zvyšuje bezpečnost celé aplikace.

3.5 Vývoj a testování

Aplikace byla vyvíjena s použitím frameworků a nástrojů, které podporují cross-platform kompatibilitu. Díky tomu je možné zajistit konzistentní uživatelské rozhraní a funkčnost napříč všemi platformami. Během vývoje byla aplikace podrobena manuálnímu testování, aby se ověřila její funkčnost a stabilita na různých zařízeních. Tento proces zahrnoval důkladnou kontrolu všech funkcí a uživatelského rozhraní, aby se zajistilo, že odpovídají stanoveným požadavkům a že jsou konzistentně realizovány napříč různými platformami. Manuální testování bylo provedeno na emulátorech. Tento přístup zajišťuje, že aplikace poskytuje stabilní a spolehlivý výkon.

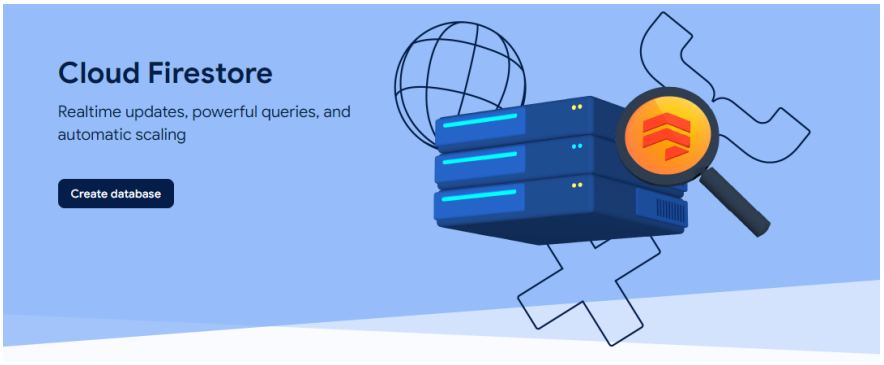
4 FIREBASE

Pro implementaci serverové strany aplikace existuje řada technologií a přístupů. Pro tuto aplikaci byla zvolena technologie Firebase, což je platforma pro vývoj mobilních a webových aplikací, která poskytuje bohaté a flexibilní back-end služby. Firebase umožňuje efektivní správu dat a autentizaci uživatelů, integraci s cloudovými funkcemi a mnoho dalších služeb, které usnadňují rychlý vývoj aplikací a minimalizují potřebu vlastního serverového řešení.

Jako klíčový prvek serverové architektury byl zvolen Firestore, konkrétně Cloud Firestore, což je NoSQL databáze, která umožňuje real-time synchronizaci dat mezi klienty a serverem. Tato databáze je ideální pro aplikace vyžadující rychlou odezvu a schopnost práce v reálném čase. Díky své škálovatelnosti a flexibilitě se Firestore dobře hodí pro aplikace, které mohou růst v počtu uživatelů i v náročnosti na data. O technologii serverové strany je možno použít více přístupů. Jako optimální řešení bylo rozhodnuto použít Firebase.

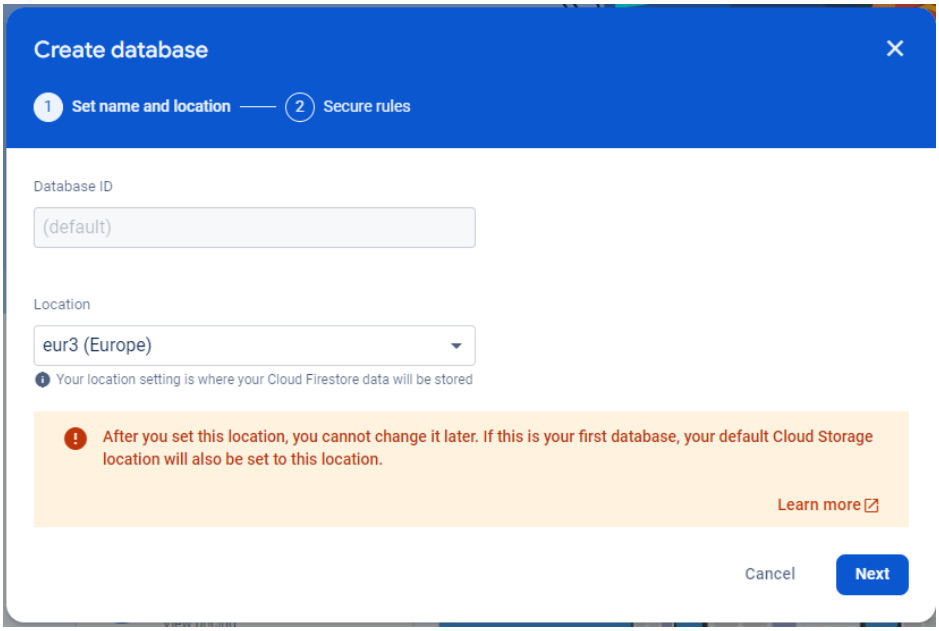
4.1 Inicializace

Pro inicializaci databáze v Firestore byla nejdříve v uživatelském rozhraní Cloud Firestore zvolena možnost vytvoření nové databáze. V prvním kroku je potřeba kliknout na tlačítko "Create database" (viz Obr. 21), které je zobrazeno na úvodní stránce Firebase konzole.



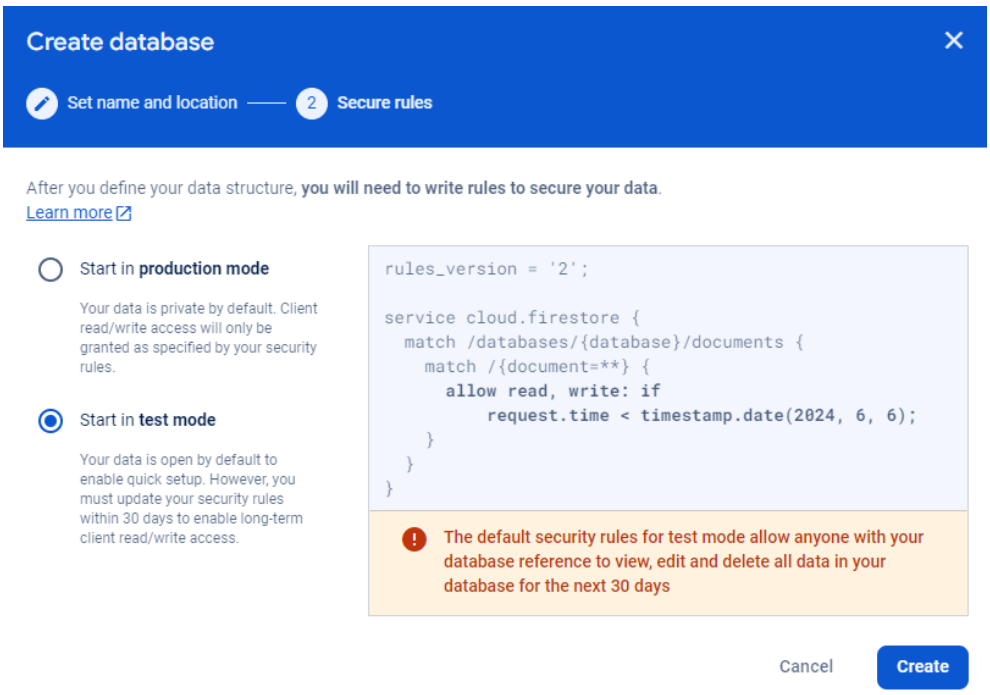
Obr. 21: Část uvodní stránky firebase konzole

Poté se otevře nová stránka, kde je možnost zvolit lokaci databáze (viz Obr. 22). Pro umístění databáze byl vybrán region "eur3 (Europe)", což znamená, že všechna data budou uložena ve fyzickém umístění v Evropě. Volba lokality je důležitá z hlediska latence a právních požadavků na ukládání dat.



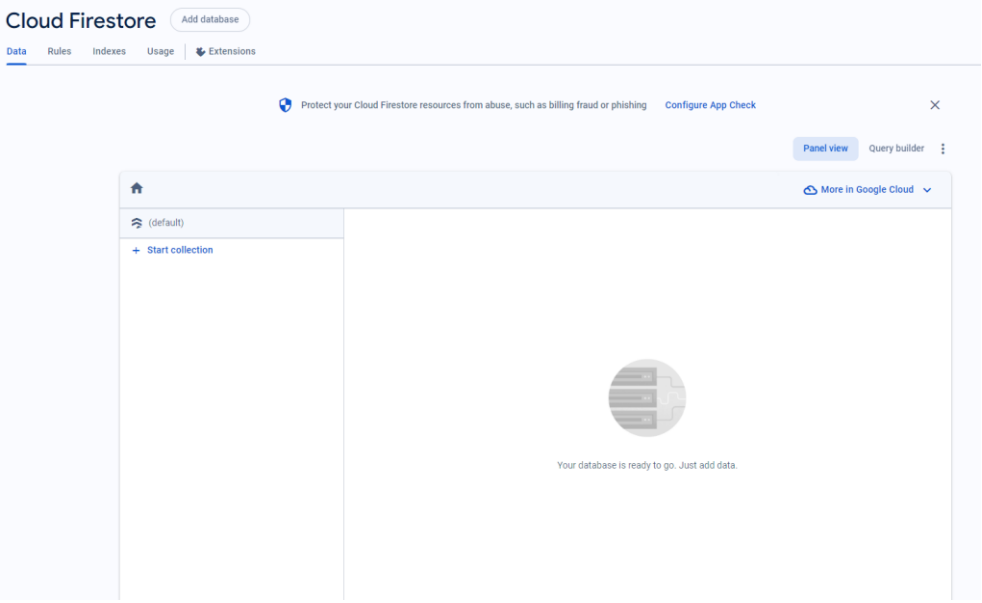
Obr. 22: Ukázka výběru regionu databáze

Následně je nutné nastavit bezpečnostní pravidla pro databázi (viz Obr. 23). Zvolí se "Start in test mode", což znamená, že databáze bude mít otevřený přístup, umožňující čtení a zápis dat bez autentizace. Tento režim je vhodný pro počáteční vývoj a testování aplikace, ale vyžaduje pozdější aktualizaci pravidel, aby se zajistilo bezpečné užívání databáze.



Obr. 23: Ukázka stránky bezpečnostních pravidel

Po nastavení bezpečnostních pravidel je potřeba kliknout na tlačítko "Create" (viz Obr. 23) pro dokončení procesu inicializace databáze (viz Obr. 24).



Obr. 24: Ukázka nově založené databáze

4.2 Příkladná ukázka zápisu do databáze

Metoda *saveTariff* (viz Zdroj. kód 18) je navržena k ukládání tarifů do cloudové databáze *Firestore* pod konkrétním uživatelským ID. Funkce nejprve zkontroluje, zda objekt tarifu obsahuje název, což je klíčový atribut potřebný pro jeho identifikaci a správné uložení do databáze.

Pokud název tarifu chybí, funkce okamžitě informuje uživatele prostřednictvím dialogového okna s chybovou zprávou *"Název tarifu chybí, nelze uložit tarif"* a proces ukládání je přerušen s chybou. Toto je zajištěno voláním *presentAlert* metody, která zobrazuje upozornění v uživatelském rozhraní.

Pokud je název tarifu přítomen, funkce pokračuje v zápisu do databáze *Firestore*. Tarif je uložen pod specifickou cestou, která zahrnuje uživatelské ID a název tarifu, umožňující jeho jednoduché dohledání a správu v budoucnu. Po úspěšném uložení tarifu je uživateli zobrazeno potvrzovací dialogové okno se zprávou *"Tarif byl úspěšně uložen"*, což signalizuje korektní dokončení operace.

```
 1 async saveTariff(tariff: Tariff, globalUID: string): Promise<void> {
 2     if (!tariff.name) {
 3       // Vytvoření a zobrazení alertu místo vypsání chyby do konzole
 4       await this.presentAlert('Chyba', 'Název tarifu chybí, nelze uložit tarif');
 5       return Promise.reject('Tariff name is missing');
 6     }
 7
 8     try {
 9       // Zajistěte, že cesta k dokumentu zahrnuje subkolekci
10       await this.firestore.doc(`tariffs/${globalUID}/userTariffs/${tariff.name}`).set(ta-
riff, { merge: true });
11       // Zobrazení potvrzení o úspěšném uložení
12       await this.presentAlert('Úspěch', 'Tarif byl úspěšně uložen');
13     } catch (error) {
14       // Zobrazení chybové zprávy, pokud nastane problém při ukládání
15       await this.presentAlert('Chyba při ukládání', 'Nastal problém při ukládání tarifu.
Prosím, zkuste to znovu.');
16       console.error('Error saving tariff:', error);
17       return Promise.reject(error);
18     }
19   }
```

Zdroj. kód 18: Ukázka zápisu do databáze pomocí funkce saveTariff

4.3 Příkladné načítaní dat z databáze

Metoda *getTariffs* (viz Zdroj. kód 19) slouží k načítání tarifů pro konkrétního uživatele z cloudové databáze *Firestore*. Tato funkce je definována tak, aby přijímala unikátní uživatelské ID *(globalUID)* jako parametr, které identifikuje, pod kterým uživatelem mají být tarify vyhledány.

Při volání této funkce se dotazuje databáze *Firestore* na kolekci `tariffs`, konkrétně na subkolekci `userTariffs` pod cestou, která je specifikována pomocí uživatelského ID. Tímto způsobem je zajištěno, že načítaná data jsou přímo spojena s daným uživatelem.

Výsledkem dotazu je *Observable*, což je typ objektu z knihovny *RxJS*, který umožňuje asynchronní operace a streamování dat. *Observable* vrací pole tarifů *(Tariff[])*, kde každý tarif je automaticky doplněn o pole *id* díky volbě *valueChanges({ idField: 'id' })*. Tato volba zajistí, že každý objekt tarifu bude obsahovat identifikátor přímo z databáze, což usnadňuje další manipulaci s těmito objekty, jako je například mazání.

Tato metoda je zásadní pro aplikace, které potřebují efektivně a bezpečně načítat uživatelská data a zobrazovat je v uživatelském rozhraní, přičemž umožňuje snadnou integraci s frontendovými frameworky, které podporují reaktivní programování, jako je *Angular* s jeho architekturou založenou na *RxJS Observables*.

```
1 getTariffs(globalUID: string): Observable<Tariff[]> {
2         return this.firestore.collection<Tariff>(`tariffs/${globalUID}/userTariffs`).value-
Changes({ idField: 'id' });
3     }
```

Zdroj. kód 19: Ukázka načítaní dat z databáze pomocí funkce getTariffs

4.4 Zabezpečení aplikace

Pro zabezpečení aplikace je zásadní, aby každý uživatel měl přístup pouze k těm datům, která jsou spojena s jeho unikátním identifikátorem *GlobalUID*. Tento přístup zaručuje, že osobní a citlivá data jsou chráněna proti neoprávněnému přístupu. O autentizaci uživatelů se stará služba Firebase Authentication, která na straně poskytovatele kompletně zajišťuje bezpečnost a ochranu uživatelských účtů. Díky této službě jsou účty chráněny proti běžným hrozbám, jako je neoprávněný přístup nebo únik dat.

Jako primární databázová platforma pro ukládání a správu dat byl zvolen Firestore, což je NoSQL databáze od Firebase. Firestore nabízí výhody v podobě snadné škálovatelnosti, real-time synchronizace dat a integrace s ostatními službami Firebase. Zabezpečení dat při přenosu mezi klientem a serverem je zajištěno pomocí protokolu HTTPS, který zaručuje, že všechna data jsou šifrována během přenosu, což minimalizuje riziko odposlechu dat třetími stranami.

Firestore rovněž poskytuje šifrování dat uložených na disku, známé jako šifrování "at rest". Toto šifrování je realizováno automaticky bez potřeby jakékoliv další konfigurace ze strany vývojáře, což výrazně zjednodušuje proces implementace bezpečnostních opatření. Všechna data uložená v Firestore jsou šifrována standardními šifrovacími algoritmy, které jsou spravovány a aktualizovány Google Cloud Platformou, zajišťující vysoký standard ochrany dat.

Díky těmto bezpečnostním opatřením mohou vývojáři aplikací na platformě Firestore zůstat klidní, vědomi si, že data jejich aplikací jsou chráněna nejen během přenosu, ale i při uložení na fyzických úložištích. Integrované bezpečnostní řešení umožňuje vývojářům soustředit se více na funkcionalitu a uživatelskou zkušenost aplikace, než na zabezpečení dat, které je robustně a efektivně spravováno platformou Firebase.

ZÁVĚR

V závěrečné části této bakalářské práce je shrnuto, že primární cíl, kterým bylo vyvinutí nezávislé aplikace pro správu spotřeby elektrické energie, byl úspěšně dosažen. V rámci teoretické části byl realizován důkladný průzkum existujících aplikací a metodik v oblasti energetického managementu, což umožnilo získat hluboké porozumění zkoumané problematice. Praktická část práce demonstruje, jak navržená aplikace přináší inovativní řešení pro efektivní správu a optimalizaci energetické spotřeby, což uživatelům umožňuje signifikantně snížit jejich energetické náklady a zvýšit kontrolu nad jejich vlastní spotřebou. Klíčovým aspektem je, že aplikace byla vyvinuta s důrazem na kompatibilitu napříč různými platformami a využívá cloudovou databázi, což zajišťuje konzistentnost uživatelského rozhraní a funkčnost napříč všemi zařízeními.

Ačkoli současná verze aplikace neumožňuje automatizovaný sběr dat prostřednictvím integrace s digitálními měřiči, je v rámci budoucího rozvoje aplikace doporučena implementace této funkcionalit. Toto rozšíření by významně přispělo k zlepšení uživatelské přívětivosti a efektivity sběru dat. Rovněž je doporučeno, aby byl v budoucích verzích aplikace zahrnut rozšířený rozsah energetických zdrojů a integrovány nové technologie, které by podpořily udržitelné řízení energetických zdrojů. Takové inovace by nejenom usnadnily širší adopci aplikace mezi koncovými uživateli, ale také by mohly přispět k celkovému zlepšení energetické efektivnosti na trhu.

Celkově tato bakalářská práce nejen že představuje přínos pro autora, který si výrazně rozšířil znalosti v relevantním oboru, ale nabízí také potenciální přínos pro širší komunitu zainteresovaných subjektů, včetně domácností a malých podniků, hledajících efektivnější metody hospodaření s energiemi a snižování jejich výdajů.

SEZNAM POUŽITÉ LITERATURY

[1] *[Přehled distribučních sazeb elektřiny]*. Online. In: Cez. ©2024. Dostupné z: https://www.cez.cz/cs/podpora/prehled-distribucnich-sazeb-elektriny. [cit. 2024-05-10].

[2] *Co je to nízký tarif/noční proud*. Online. Cez. ©2024. Dostupné z: https://www.cez.cz/cs/podpora/co-je-to-nizky-tarif-nocni-proud. [cit. 2024-05-10].

[3] *Jak číst naměřená data?* Online. Egd. ©2024. Dostupné z: https://www.egd.cz/jak-cist-namerena-data. [cit. 2024-05-10].

[4] *Jaký je rozdíl mezi distributorem a dodavatelem elektrické energie*. Online. Eon. Dostupné z: https://www.eon.cz/radce/energie/ceny-energie/jaky-je-rozdil-mezi-distributorem-a-dodavatelem-elektriny/. [cit. 2024-05-10].

[5] *Distributoři elektřiny – území distribuce, kontakty*. Online. Kurzy. ©2000-2024. Dostupné z: https://www.kurzy.cz/elektrina/distributori. [cit. 2024-05-10].

[6] *Dodavatelé elektřiny: Jak vybrat toho pravého?* Online. Usetreno. ©2010–2024. Dostupné z: https://www.usetreno.cz/energie-elektrina/dodavatele-elektriny/. [cit. 2024-05-10].

[7] *Noční proud aneb Jak získat vysoký a nízký tarif*. Online. Spp. 2024. Dostupné z: https://www.spp.cz/magazin/nocni-proud-jak-ziskat-vysoky-a-nizky-tarif. [cit. 2024-05-10].

[8] *Jak se vyznat ve vyúčtování za energie?* Online. Srovnejto. 2022. Dostupné z: https://www.srovnejto.cz/blog/jak-se-vyznat-ve-vyuctovani-za-energie/. [cit. 2024-05-10].

[9] *Chytré aplikace na hlídání spotřeby*. Online. Mobilizujeme. 2022. Dostupné z: https://mobilizujeme.cz/clanky/chytre-aplikace-na-hlidani-spotreby. [cit. 2024-05-10].

[10] *TOP 3 uživatelsky příjemných aplikací, které vám pomohou šetřit za energie*. Online. Epet. 2020. Dostupné z: https://www.epet.cz/top-3-uzivatelsky-prijemnych-aplikaci-ktere-vam-pomohou-setrit-za-energie/. [cit. 2024-05-10].

[11] *ČEZ spouští mobilní aplikaci. V energetické krizi umožní kontrolu spotřeby*. Online. Forbes. 2022. Dostupné z: https://forbes.cz/cez-spousti-mobilni-aplikaci-v-energeticke-krizi-umozni-kontrolu-spotreby/. [cit. 2024-05-10].

[12] *Mobilní aplikace společnosti E.ON vyfotí elektroměr a zobrazí odhad vyúčtování*. Online. Tzb-info. 2019. Dostupné z: https://www.tzb-info.cz/ceny-paliv-a-energii/124517-mobilni-aplikace-spolecnosti-e-on-vyfoti-elektromer-a-zobrazi-od-had-vyuctovani. [cit. 2024-05-10].

[13] *What is TypeScript?* Online. Typescriptlang. ©2012-2024. Dostupné z: https://www.typescriptlang.org/. [cit. 2024-05-10].

[14] *What Is HTML? Hypertext Markup Language Basics Explained*. Online. Hostinger. 2023. Dostupné z: https://www.hostinger.com/tutorials/what-is-html. [cit. 2024-05-10].

[15] *What Is SCSS? Learn How To Use SCSS To Style HTML*. Online. Upwork. 2022. Dostupné z: https://www.upwork.com/resources/what-is-scss. [cit. 2024-05-10].

[16] *Angular overview*. Online. Sanity. 2024. Dostupné z: https://www.sanity.io/glossary/angular. [cit. 2024-05-10].

[17] *What Is Vue JS?* Online. Builtin. 2022. Dostupné z: https://builtin.com/software-engineering-perspectives/vue-js. [cit. 2024-05-10].

[18] *What is React and why use it for your app?* Online. Uxpin. 2024. Dostupné z: https://www.uxpin.com/studio/blog/what-is-react/. [cit. 2024-05-10].

[19] *What Is Node.js and Why You Should Use It*. Online. Kinsta. 2023. Dostupné z: https://kinsta.com/knowledgebase/what-is-node-js/. [cit. 2024-05-10].

[20] *SQL vs. NoSQL Databases: What's the Difference?* Online. Ibm. 2022. Dostupné z: https://www.ibm.com/blog/sql-vs-nosql/. [cit. 2024-05-10].

[21] *Google Firebase*. Online. Techtarget. 2023. Dostupné z: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase. [cit. 2024-05-10].

[22] *Firebase Firestore Database*. Online. Medium. 2024. Dostupné z: https://medium.com/@abubakarsaddqiuekhan/firebase-firestore-database-78dae380d5cf. [cit. 2024-05-10].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

kWh	Kilowatthodina
VT	Vysoký tarif
NT	Nízký tarif
MWh	Megawatthodina
IoT	Internet of Things
SPA	Single Page Application
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SQL	Structured Query Language
NoSQL	Not Only SQL
PWA	Progressive Web Apps
SCSS	Sassy Cascading Style Sheets
AWS	Amazon Web Services
GCP	Google Cloud Platform
OBIS	Object Identification Systém
HDO	Hlavní Dálkové Ovládání
ČR	Česká Republika
PHP	Personal Home Page
HTTPS	Hypertext Transfer Protocol Secure

SEZNAM OBRÁZKŮ

Obr. 1: Přehled distribučních sazeb elektřiny (z webu cez.cz)[1] 11
Obr. 2: Mechanický elektroměr [3] 12
Obr. 3: Digitální displej s kódy OBIS [3] 13
Obr. 4: Rozdělení území ČR mezi distributory elektřiny [5] 14
Obr. 5: Přihlašovací obrazovka 37
Obr. 6: Registrační obrazovka 38
Obr. 7: Obrazovka zapomenutého hesla 40
Obr. 8: Ukázka emailu zapomenutého hesla 41
Obr. 9: Okno pro změnu zapomenutého hesla 41
Obr. 10: Ukázka ukládaní tarifu 42
Obr. 11: Ukázka výsledků výpočtu 43
Obr. 12: Ukázka zadávání odečtu 44
Obr. 13: Ukázka výběru tarifu 47
Obr. 14: Ukázka detailů za měsíc 48
Obr. 15: Ukázka zobrazení statistiky za všechny měsíce 50
Obr. 16: Ukázka porovnaní tarifů 53
Obr. 17: Obrázek správy počátečních stavů 54
Obr. 18: Obrázek správy odečtů 55
Obr. 19: Obrázek tarifů 56
Obr. 20: Obrázek měsíční výpočty 57
Obr. 21: Část uvodní stránky firebase konzole 60
Obr. 22: Ukázka výběru regionu databáze 61
Obr. 23: Ukázka stránky bezpečnostních pravidel 61
Obr. 24: Ukázka nově založené databáze 62

SEZNAM TABULEK

Tabulka 1: Funkční požadavky 34
Tabulka 2: Nefunkční požadavky 35

SEZNAM ZDROJOVÝCH KÓDŮ

Zdroj. kód 1 :Zdrojový kód FormGroup....36
Zdroj. kód 2: Přihlašovaní uživatele....37
Zdroj. kód 3: Funkce loginUser....37
Zdroj. kód 4: Funkce registerUser....38
Zdroj. kód 5: Funkce resetovaní hesla....39
Zdroj. kód 6: Funkce presentToast()....40
Zdroj. kód 7: Funkce SaveTariff....41
Zdroj. kód 8: Funkce saveReading....45
Zdroj. kód 9: Funkce checkAndUpdateInitialState....45
Zdroj. kód 10: Funkce calculateAndSetInitialReadingIfNewMonth....46
Zdroj. kód 11: Funkce loadMonthlyCalculations....47
Zdroj. kód 12: Funkce calculateMonthlyConsumption....49
Zdroj. kód 13: Ukázka kódu výběru tarifů ze seznamu....51
Zdroj. kód 14: Funkce comparAndSaveTariffs....51
Zdroj. kód 15: Funkce processTariffCalculations....52
Zdroj. kód 16: Funkce getCostComparison....52
Zdroj. kód 17: Funkce loadColletionData....58
Zdroj. kód 18: Ukázka zápisu do databáze pomocí funkce saveTariff....63
Zdroj. kód 19: Ukázka načítaní dat z databáze pomocí funkce getTariffs....64

SEZNAM PŘÍLOH

prilohy.zip:

fullwork	fullwork.zip
app-release	app-realease.apk
fulltext	fulltext.pdf