FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Optimalizace spotřeby el. energie v budově na základě spotových cen
Student:	Jiří Matoušek
Vedoucí:	Ing. Jindřich Hegr
Studijní program:	Informatika
Obor / specializace:	Manažerská informatika 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem práce je vytvoření prototypu aplikace, která umožní na základě spotových cen elektrické energie regulovat příkon vybraných systémů budovy, zejména vytápění, ochlazování a vzduchotechniky.
Dílčí cíle jsou následující:

1. Seznamte se s projektem firmy v oblasti instalovaných technologií a s jejich možnostmi ovládání.
2. Specifikujte požadavky na funkcionalitu aplikace.
3. Analyzujte programovací prostředky pro vývoj aplikace, zvolte vhodnou vývojovou platformu.
4. Navrhněte a vytvořte prototyp řídicí aplikace.
5. Modelujte možné ekonomické úspory navrhovaného řešení.
6. Navrhněte případná rozšíření aplikace vzhledem k potřebám firmy.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 22. ledna 2024 v Praze.

Bakalářská práce

OPTIMALIZACE SPOTŘEBY EL. ENERGIE V BUDOVĚ NA ZÁKLADĚ SPOTOVÝCH CEN

Jiří Matoušek

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jindřich Hegr
16. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2024 Jiří Matoušek. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Matoušek Jiří. *Optimalizace spotřeby el. energie v budově na základě spotových cen*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval Ing. Jindřichu Hegrovi za jeho ochotu, čas a přínosné rady poskytované v průběhu vedení této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

Zvyšování cen energií motivuje spotřebitele k úvahám i reálným opatřením vedoucím jednak ke snižování spotřeby energií a také k možnostem redukce finančních nákladů za odebrané energie. Možností pro snížení finančních nákladů může být přechod z fixního účtování na účtování průběhové (spotové). Průběhové měření umožňuje lépe plánovat spotřebu energií v čase. To znamená, že spotřebitelé mohou snížit nebo odložit spotřebu v době vysoké ceny a naopak více spotřebovávat nebo akumulovat energii v době, kdy jsou ceny nižší.

Tato bakalářská práce poskytuje čtenářům základní výčet technologií používaných v moderních budovách, zmiňuje právní aspekt a terminologii v oblasti energetického trhu, identifikuje spotřebiče energií v budovách včetně možností jejich vzdáleného ovládání. Dále se práce zabývá analýzou požadavků a návrhem prototypu řídicí aplikace a s tím souvisejícím výběrem vývojové platformy. V další fázi jsou základní funkcionality implementovány v jazyku C++ (společně s jednoduchou webovou aplikací), jejichž zdrojový kód je součástí práce. Prototyp komunikuje s rozhraním portálu OTE, kde získává data o cenách z denního trhu. Výstupem je REST rozhraní se zpracovanými daty.

Práce vnikla na základě podnětu softwarové firmy VERA, spol. s r.o., která staví novou budovu pro své potřeby. Právě prototyp aplikace a zmiňované možnosti komunikace s jednotlivými technologiemi budovy jsou přínosné pro další rozhodování v oblasti ovládání, měření a regulace.

Klíčová slova denní trh, dynamické určování cen, energetický trh, inteligentní budovy, OTE, optimalizace spotřeby, průběhové měření, řídicí aplikace, spotové ceny, vyhláška o měření elektřiny

Abstract

Rising energy prices motivate consumers to consider and take real measures to reduce energy consumption and to reduce the financial costs of energy consumption. Switching from fixed billing to pay-as-you-go (spot) billing may be an option for reducing financial costs. Continuous metering allows better planning of energy consumption over time. This means that consumers can reduce or postpone consumption when prices are high and, conversely, consume more or store energy when prices are lower.

This bachelor thesis provides the reader with a basic list of technologies used in modern buildings, mentions the legal aspect and terminology in the energy market, and identifies energy-consuming appliances in buildings, including their remote control capabilities. Furthermore, the thesis deals with the requirements analysis and design of a prototype control application and the related selection of the technology stack. In the next phase, the basic functionalities are implemented in C++ (together with a simple web application), the source code of which is included in the thesis. The prototype interacts with the energy market operator's public interface to obtain price data from the daily market. Output of the apliaction is communicated over a REST interface for other systems to use.

The thesis was initiated based on the suggestion of the software company VERA, spol. s r.o., which is constructing a new building for its needs. The prototype application and the mentioned possibilities for communication with the various building technologies are beneficial for the project owner's further decision-making in the areas of HVAC and building automation systems.

Keywords control applications, day-ahead market, dynamic pricing, energy market, interval metering, OTE, smart buildings, spot prices

Seznam zkratek

API	Application (Programming) Interface
BACS	Building Automation and Control System
CBOR	Concise Binary Object Representation
CLI	Command Line Interface
CoAP	Constrained Application Protocol
CORE	Constrained RESTful Environments
DALI	Digital Addressable Lighting Interface
DOM	Document Object Model
EC motor	Elektronicky komutovaný motor
ERU	Energetický regulační úřad
FPGA	Field Programmable Gate Array
GHz	Gigahertz
GPL	GNU General Public License
HP	Hewlett-Packard
HTTP	Hypertext Transfer Protocol
HVAC	Heating, ventilation, and air conditioning
IDE	Integrated Development Environment
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
MaC	Management Client
MaR	Měření a regulace
MIME	Multipurpose Internet Mail Extensions
MWh	Megawatt hodina
OSCORE	Object Security for Constrained RESTful Environments
OTE	OTE a.s. – operátor energetického trhu ČR
OZE	Obnovitelné zdroje energie
PLC	Programmable Logic Controller
REST	Representational state transfer
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TČ	Tepelné čerpadlo
TLS	Transport Layer Security
UX	User Experience
VZT	Vzduchotechnika
WS	Web Services
WSDL	Web Services Description Language
XML	Extensible Markup Language
Zadavatel	VERA, spol. s r.o.

Úvod

Jak touha po technologickém pokroku, zisku a všeobecném blahobytu, tak i stanoviska Evropské komise jsou hnacím motorem změn nejen na energetickém trhu. Vyhláška o měření elektřiny z roku 2020 zapracovává část evropské směrnice o společných pravidlech pro vnitřní trh s elektřinou. Tato vyhláška uzákoňuje povinnosti zejména provozovatelům částí elektrizační soustavy v oblasti metod měření elektrických toků. V praxi se hovoří o tzv. chytrých či inteligentních elektroměrech umožňujících průběhové měření a vzdálené odečty. Mimo jiné z vyhlášky vyplývá povinnost provozovatelům soustavy nainstalovat tyto průběhové elektroměry u odběratelů s odběrem nad 6 MWh silové elektřiny za rok. Tito odběratelé představují nejčetnější skupinu, u které bude ze zákona povinné průběhové měření od července roku 2027. Digitalizace, všeobecně vnímána jako pokrok, se takto dotýká i energetických sítí, jakkoli se uplatňuje i jinde. Tento stav věcí bude informatik nejspíš hodnotit kladně, už jen z toho důvodu, že vyvolává poptávku po jeho schopnostech.

Očekávaným vývojem je, že vícero domácností i firem bude vstupovat na energetický trh i v roli výrobce. Pročež je nutné, jak technologicky tak právně, aby měly odpovídající měřič. Tyto prvky pokročilé měřicí infrastruktury typicky komunikují s distributory a ti s operátorem energetického trhu, kterým je v současné době OTE, a.s. V praxi to znamená, že domácnosti a firmy s inteligentními měřiči, typicky elektroměry, mají možnost energie kupovat a prodávat za tzv. spotové ceny. Tedy za ceny, které jsou v současné době stanovené po hodinách, od 1. 7. 2024 po čtvrthodinách a jsou určovány energetickým trhem.

Tento spotový trh, nazýván jako *denní trh*, i když ve skutečnosti udává ceny na příští den (*ang. Day-Ahead Market*), umožňuje účastníkům vyšší flexibilitu než fixní sazby. Spotřebitel má volbu rozplánovat si spotřebu energie na následující den a optimálně odebírat největší množství, když vyhodnotí spotovou cenu jako nízkou. Naopak menší výrobce, motivován ziskem z vyšších cen, se bude snažit přizpůsobit výrobu tak, aby prodával co nejvíce v době s vysokou cenou, pokud mu to budou technologické možnosti umožňovat. Lze očekávat, že pokud tento trh bude nadále tak *liberalizovaný*, jak se o něm dnes hovoří, cenové výkyvy, a tak i příležitosti individuálního zisku či optimalizace,

se budou postupně zmenšovat, pokud tedy bude dovoleno snadného vstupu na tento trh a další okolnosti zůstanou nezměněny.

Realisticky je však nutné se změnou budoucích okolností počítat. Co zůstává nepopiratelným faktem je, že tento trend otevírá ziskové okénko pro řadu podnikatelů z technických oborů, zejména energetiky a producenty informačních technologií. Nad tím, zda koncový odběratel bude mít v dlouhodobém horizontu elektřinu levnější, můžeme zatím spekulovat.

Role technika v této doméně je nezastupitelná. Informatik může přispět svým dílem díky svým dovednostem analyzovat, navrhovat a programovat řídicí systémy a hardware, navrhovat komunikační protokoly, propojovat existující systémy, tvořit přívětivá rozhraní pro uživatele ad. Bude potřebovat podnikatele, který jeho dovednosti bude schopen využít k naplnění potřeb spotřebitelů.

K motivaci pro tuto práci přispívá projekt novostavby společnosti zadavatele, který sestává z řady oblastí vhodných pro uplatnění softwarového přístupu. Ultimátním cílem je vytvoření komplexního řídicího a monitorovacího systému, propojujícího senzorická data budovy s vnějšími faktory, agregující široké spektrum systémů řízení, měření, regulace a uchovávání energie.

Práce může sloužit jako podklad pro další firmy a domácnosti mající zájem softwarově řídit své chytré budovy.

Cíle

Cílem práce je vytvoření prototypu aplikace, která umožní na základě spotových cen na denním trhu s energiemi poskytovat data o výhodnosti ceny elektrické energie, zejména systémům vytápění, chlazení a vzduchotechniky. Dílčí cíle jsou následující:

- poskytnutí přehledu v oblasti instalovaných technologií (na základě projektu novostavby společnosti zadavatele) a možností jejich ovládání
- specifikace (sběr a analýza) požadavků na funkcionalitu aplikace
- analýza programovacích prostředků pro vývoj aplikace, volba vhodné vývojové platformy
- návrh a tvorba prototypu řídicí aplikace
- návrh případných rozšíření aplikace vzhledem k potřebám zadavatele

Kapitola 1

Technologie, terminologie a teorie domény

1.1 Klasifikace atributů software

Na začátku projektu je nutné specifikovat, co od konečného řešení očekáváme. Průběhem času se v oboru ustálily nápomocné modely, které klasifikují kategorie atributů software.

Mezi tyto modely (v ang. se jedná o *software quality models*) se řadí například McCallův model, Boehmův model, FURPS(+) model, ISO/IEC 9126 a nástupce ISO/IEC 25010.[1,2] Dále stojí za zmínku i primitivní, ale přínosný model *MoSCoW**.[3]

Tyto modely nám mohou napomoct k tomu, abychom neopomněli některou zásadní část z počátku vývoje. Vyjádřením se k atributům, které tyto modely shrnují, tak můžeme zamezit případným potížím v budoucnu.†

1.1.1 Model FURPS+

FURPS+ je model používaný pro specifikaci a hodnocení softwaru. Přínos FURPS modelu je připisován Robertu Gradymu, který ho, společně s Deborah Caswellovou, uvedl v publikaci *Software metrics: establishing a company-wide program*.[4] Dále ho Grady uvádí v *Practical software metrics for project management and process improvement*, kde již hovoří o FURPS+.[5]

Model pomáhá definovat různé aspekty softwarových požadavků a je užitečný pro zajištění jejich uceleného sběru. Napomáhá systematickému přístupu

*model si popíšeme dále

†Ne, že by nešlo vyvíjet bez použití standardizovaných modelů, někdy naopak může být žádoucí pustit se do projektu bez přílišné počáteční byrokracie a reagovat na podněty v průběhu. Jaká metoda bude úspěšnější už stojí na individuálním posouzení. Tyto nástroje mohou mít větší váhu, pokud objednáváme dílo na zakázku. Pak změnová řízení mohou být velmi nákladná.

k analýze požadavků projektu. Možným využitím jsou implementace informačních systémů. Nicméně kategorie jsou poměrně obecné a mají průnik s většinou softwarových řešení. Jelikož FURPS je akronym, model je snadno zapamatovatelný.

Grady uvádí, že FURPS+ dostal příznak plus aby se dosáhlo uspokojení v rámci společnosti HP, kde jednotlivé divize kladli důraz na jednotlivé atributy. Například FLURPS pro důraz na lokalizovatelnost. Pro sjednocení používání variant FURPS modelu mezi divizemi, spol. HP zavedla akronym FURPS+ jako kompromis.[5]

Model FURPS+ se skládá z pěti hlavních kategorií. Uvedeny jsou anglické kategorie dle *Figure 4-1: Components of FURPS+* Gradyho publikace. Jsou doplněny o české překlady z prezentace *Dimenze kvality FURPS+* Dušana Vaňka a o další kategorie:

Funkčnost (Functionality)

- Náplň programu, základní rysy, množina funkcionalit, případy užití (Feature set)
- Rozsah pokrytí, schopnosti (Capabilities)
- Obecnost, aplikovatelnost na širší škálu, použitelnost dodaných funkcí (Generality)
- Celková bezpečnost systému, šifrování, databáze, osobní údaje (Security)

Použitelnost (Usability)

- Lidské faktory, interakce s uživatelem, školení (Human Factors)
- Estetika, uživatelská rozhraní, vizuál (Aesthetics)
- Rozhraní, front end, potřebné webové služby
- Virtualizace, baremetal, cloud, SaaS, hosting, docker
- Operační systémy, programovací jazyky, vývojová prostředí
- Spolupráce přes rozhraní, API, REST atp.
- Konzistence, jednotnost, konzistence GUI (Consistency)
- Dokumentace, uživatelská, výuková, příklady (Documentation)

Spolehlivost (Reliability)

- Četnost a závažnost selhání (Frequency/ Severity of Failure)
- Obnovitelnost ze selhání, výpadku (Recoverability)
- Předvídatelnost (Predictability)
- Přesnost (Accuracy)
- Průměrný čas mezi selháními (Mean Time to Failure)

Výkonnost (Performance)

- Rychlost (Speed)
- Efektivita, hospodárnost se zdroji (Efficiency)
- Spotřeba zdrojů, paměťová náročnost, disková úložiště (Resource Consumption)
- Průchodnost (Thruput)
- Čas odezvy (Response Time)
- Čas zotavení (Recovery Time)

Podporovatelnost (Supportability)

- Ověřitelnost, testovatelnost (Testability)
- Rozšiřitelnost, modularita, rozhraní (Extensibility)
- Přizpůsobivost (Adaptability)
- Udržovatelnost, vendor lock-in, udržitelnost podpory (Maintainability)
- Kompatibilita, slučitelnost, interoperabilita (Compatibility)
- Konfigurovatelnost (Configurability)
- Podpora, obslužnost, servisní služby a opravy (Serviceability)
- Škála instalace, průvodci, kontejnery, virtuální stroje, cloud (Installability)
- Lokalizovatelnost, podpora jazyků a přístupnost (Localizability)[5,6]

1.1.1.1 Alternativy

Výhodou FURPS+ modelu je jeho snadná zapamatovatelnost, avšak jeho základy jsou relativně staré.[‡] Je možné se obrátit na jiné modely, které vesměs reflektují podobné spektrum jako FURPS, například zmiňovaný ISO/EIC 25010, a mohou jít více do detailu.

ISO/EIC 25010 (Software Product Quality) v sobě zahrnuje devět kategorií: *Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, Flexibility a Safety.*[7]

1.1.2 MoSCoW

Jednoduchý model s akronymem pro *Must have, Should have, Could have a Won't have.*

[‡]Opět, ne že by bylo třeba měnit, co funguje. FURPS+ model je navrhnut jistě přínosně, koneckonců je také ozkoušen v rámci velké a historicky úspěšné firmy HP. Nicméně vývoj v oboru může posouvat terminologii a také relevanci některých kategorií.

Must have Vymezení požadavků a atributů, které jsou naprosto nutné k aktuálnímu vydání.

Should have Požadavky s vysokou prioritou, které by měly být zahrnuty ve vydání.

Could have Požadavky, které by se mohly dostat do vydání a je možné je přidat bez přílišných dodatečných nákladů.

Won't have Chtěné zbylé funkcionality do budoucna, které nebudou v aktuálním vydání.[3]

Jedná se o koncept, který se používá naprosto běžně. Slouží k rychlé klasifikaci požadavků dle důležitosti implementace. Lze spojit s FURPS+ modelem (prakticky i s jakýmkoli jiným) do matice (viz tabulka 1.1), která může posloužit jako podklad pro rozvržení času a práci se zdroji.[3]

	FURPS	**M**	**S**	**C**	**W**
F	functionality	*must*	*should*	*could*	*won't*
	...				
	...				
U	usability	*must*	*should*	*could*	*won't*
	...				
R	reliability	*must*	*should*	*could*	*won't*
	...				
P	performance	*must*	*should*	*could*	*won't*
	...				
S	supportability	*must*	*should*	*could*	*won't*
	...				

Tabulka 1.1 FURPS–MoSCoW matice, zdroj: Jonathan Dyson

1.2 Strategická analýza

Jedná se o *„proces shromáždění a vyhodnocení relevantních fakt a informací, potřebných pro formulaci strategií."* Celkově sestává z několika analýz (např. SLEPT, Porterova, firemní kultury, konkurence, trhů, dodavatelů atp.). Mezi ně se řadí i SWOT analýza.[8]

1.2.1 SWOT

Další metoda obsahující akronym v názvu. Jedná se o *Strengths, Weaknesses, Opportunities a Threats*, tedy Silné stránky, Slabé stránky, Příležitosti a Hrozby.

Analýza slouží jako pomůcka k hodnocení faktorů ovlivňujících úspěšnost konkrétního záměru. Časté využití nachází v rámci strategického řízení a marketingu. Lze ji uplatnit téměř na jakýkoli projekt, organizaci nebo záměr díky její jednoduché a obecné podstatě. Hrozby, identifikované SWOT analýzou, představují klíčové zdroje rizik, ke kterým můžeme dále specifikovat protiopatření.[9]

1.2.1.1 Silné a slabé stránky

Jak se uvádí v publikaci *Swot Analysis: A Guide to Swot for Business Studies Students*, u silných stránek a slabých stránek se zaměřuje na vnitřní faktory (uvnitř organizace, uvnitř týmu). V čem je tým dobrý, v čem špatný, jaké má technologické výhody, know-how, jakou má pověst. Dále finanční stránky, rozpočet, cash flow, rating atp. Schopnost poskytovat služby, odbornost a proškolenost zúčastněných lidí ad. Silné stránky podporují příležitosti a napomáhají ve zdolávání hrozeb.[10]

1.2.1.2 Příležitosti a hrozby

Příležitosti a hrozby jsou zaměřeny naopak na vnější faktory. Jako například: konkurence přicházející na trh nebo z něho vystupující, zvýšená nebo pokleslá poptávka, ztráta nebo zisk reputace, politický vývoj, legislativa, ekologie atd.[9,10]

Silné stránky	**Slabé stránky**
Příležitosti	**Hrozby**

Obrázek 1.1 Příklad struktury SWOT analýzy, zdroj: Management mania

1.2.1.3 Původ a SOFT metoda

V publikaci *The origins of SWOT analysis* se o SWOT hovoří jako o jednom z nejstarších a celosvětově nejrozšířenějších strategických nástrojů. Kvůli častému užívání a někdy nadužívání se místy analýza zasloužila o kritiku a o odkazy na použití sofistikovanějších modelů.[11]

Mimo jiné se ve výše uvedené podrobné publikaci autoři zmiňují o zjištěném *předchůdci* SWOT analýzy, tzv. SOFT metodě. Ta spočívá v položení si čtyřech (SOFT) otázek:

*„1. What must be done to **s**afeguard the **s**atisfactory in present operations? 2. What must be done to **o**pen the door to **o**pportunities in future operations? 3. What must be done to **f**ix the cause of **f**aults of present operations? 4. What must be done to **t**hwart, ameliorate or avert the **t**hreats to future operations? (Stewart et al., 1965a, p. 18 […]).“*[11]

V *origins of SWOT* se dále uvádí jméno Roberta Franklina Stewarta, který se dle jejich výzkumu významně zasloužil o rozšíření jak SOFT, tak dále SWOT metody.[‡11] Dalšími nástroji strategické analýzy se práce nebude zabývat.

1.3 Knihovny a frameworky

Tato kapitola se věnuje popisu technologií a prostředků, které budou dále použity v rámci rozhodování o návrhu prototypu. Uvádí specifické knihovny, vývojová prostředí a dále systémy pro sestavování software.

1.3.1 Poskytující rozhraní

1.3.1.1 libknxnet

Jedná se o malou (cca 580 řádek*), opensource knihovnu dostupnou z GitHubu. Knihovna implementuje KNXnet/IP rozhraní pro Linux. Poskytuje minimum funkcí pro konverze dat a dále specifikuje výčtové typy (enumy) pro základní KNX datové typy a operace. Jedná se o C++ knihovnu, knxnet funkcionalita je zapouzdřena do namespace. Nenajdeme jinak příliš typických funkcionalit pro C++ oproti C, také z důvodu použitého soketového rozhraní `<sys/socket.h>`, které si se šablonami, iterátory nebo smart pointery nerozumí. Není využit enum class atp.[12]

Knihovna definuje uživatelské typy například pro KNX adresu (`address_t`), přičemž pro 16bitovou adresu používá union (viz kód 1.1) pro přístup přes 16bitovou hodnotu, group adresu (`ga`), fyzickou adresu (`pa`) a bytearray.**

Komunikace mezi linuxovým strojem a nějakou KNX bránou je, jak jsme zmínili, založená na soketech. Komunikuje se přes multicast adresu (typicky 224.0.23.12).[12]

[‡]R. Stewart se znal s Albertem S. Humphreym, který je běžně uváděn jako autor SWOT analýzy. Další informace lze najít ve zmiňované publikaci *The origins of SWOT analysis*, která představuje zajímavé souvislosti.

*metodou `cat * | wc -l`

**Někdo by mohl spekulovat, zda toto chování je v C++ *undefined behaviour*. Jedná se o praktiku nedoporučovanou C++ Core Guidelines (tzv. *union punning*)[13] a například Pieter P. v článku *Don't use unions or pointer casts for type punning* argumentuje tím, že je to nedefinované chování, pokud se union naplní přes jeden prvek a přistoupí přes jiný[14] a pak je rozhodující implementace v kompilátoru. Ovšem tím, že všechny prvky mají stejnou délku a kompilátor naimplementuje přístup přes jedno místo v paměti, nehrozí riziko přetečení a přístup by měl být konzistentně správný s benefitem snadného přístupu k datům různou sémantikou. Určitě by se dal použít `std::variant`[15] jak doporučuje standard, nicméně otázka zůstává, zda v tomto případě práce se sokety to má opravdu význam.

```
typedef union __address
  {
    uint16_t value;
    struct
    {
      uint8_t high;
      uint8_t low;
    } bytes;
    struct __attribute__((packed))
    {
      uint8_t line:3;
      uint8_t area:5;
      uint8_t member;
    } ga;
    struct __attribute__((packed))
    {
      uint8_t line:4;
      uint8_t area:4;
      uint8_t member;
    } pa;
    uint8_t array[2];
  } address_t;
```

Výpis kódu 1.1 knxnet, typ `address_t`, zdroj: libknxnet

1.3.1.2 Alternativy

Například Qt KNX (v rámci Qt, součást Qt for Automation) poskytuje ještě jinou metodu: TCP tunelování přes KNX/IP tunel. To je patrně náročnější na implementaci, ale při TCP komunikaci se drží spojení s KNX tunelem, které může být šifrované[†]. U multicast metody v podstatě bez vlastního protokolu nic nevíme o KNX bráně (role, kterou může zastávat např. KNX tunel), jestli správně přeposílá data do KNX sítě, či jestli je vůbec v provozu.[16,17]

Nevýhodou a zároveň výhodou může být vysoká provázanost s Qt SDK. S tím související úskalí použití modulu Qt KNX mimo Qt framework. Lze pod GPLv3 sestavit ze zdrojových kódů. Přiložen je postup autora pro sestavení v příloze B.[*]

1.3.1.3 gSOAP

GSOAP je rozsáhlá sada programů a knihoven (toolkit) umožňující implementovat efektivní a bezpečná rozhraní k webovým službám (SOAP, XML, JSON, REST). Umožňuje datovou vazbou mapovat XML do programů v C a C++. Stará se o serializaci a deserializaci XML dat. Umí transformovat datové struktury (C, některé C++) na XML a zpět. Toolkit je vybaven nástroji pro práci s WSDL soubory a nadstavbami, jako jsou např. WS-Security a WS-ReliableMessaging.[18]

Dokumentace k gSOAP poskytuje podrobné návody na vytváření webových služeb, včetně příkladů kódu a pokynů pro implementaci funkcí, jako jsou autentizace a šifrování zpráv. Uživatelé mohou využívat různé pluginy, které rozšiřují funkcionality gSOAP, včetně podpory pro cURL nebo integraci s Apache webovými servery.[18]

GSOAP je schopné z WSDL vygenerovat potřebné hlavičkové soubory pro C++ program. Postup může vypadat takto (viz kód 1.2). Ve výpisu 1.3 je ukázán příklad jednoduchého SOAP klienta v C++ (SOAP API vrátí součet dvou čísel). Pozn. gSOAP si umí poradit i se strukturovanými typy, kdyby result byl nějaký seznam, dokáže vygenerovat strukturu s použitím například `std::vector`.

Nástroj wsdl2h konzumuje WSDL a vytvoří z něj hlavičkový soubor. Soapcpp2 pak generuje implementační kód s datovými vazbami. Příznak `-j` značí, že se mají vygenerovat i C++ proxy třídy (zabalí funckonalitu do objektů) a `-C` říká, že se má vygenerovat pouze část klienta. GSOAP umí naopak generovat WSDL i z kódu.[18]

[†]Data na multicastu mohou být samozřejmě šifrovaná taky. TCP spojení pro zaznamenávání dění na KNX síti je bezpečnější, při nechtěném zapojení KNX brány do nepřipravené sítě dojde multicast přenos (dle nastavení směrovačů ad.) ke všem zařízením. V rámci udržitelnosti dlouhodobého provozu je jakékoli opatření, které mitiguje budoucí rizika, žádoucí.

[*]Existuje ještě spousta dalších nástrojů pro práci s KNX, kterými se práce nezabývá. Existuje např. Falcon SDK pro dotnet, weinzierl.de KNX SDK (C), thelsing/knx, Calimero-core (Java) ad.

```
curl http://some/public/wsdl/source > out.wsdl
wsdl2h out.wsdl
soapcpp2 -j -C out.h
```

Výpis kódu 1.2 WSDL do hlavičkového souboru, zdroj: gSOAP, upraveno autorem

```
// File: calclient.cpp
#include "soapcalcProxy.h"
#include "calc.nsmap"
int main()
{
  calcProxy service;
  double result;
  if (service.add(1.0, 2.0, result) == SOAP_OK)
    std::cout << "The sum of 1.0 and 2.0 is "
              << result << std::endl;
  else
    service.soap_stream_fault(std::cerr);
  service.destroy(); // delete data and release memory
}
```

Výpis kódu 1.3 Příklad použití proxy class v C++ v rámci jednoduchého SOAP klienta, zdroj: gSOAP

1.3.1.4 Pistache

Pistache je moderní C++ framework pro vývoj REST API, který se zaměřuje na výkon s cílem poskytovat elegantní a asynchronní API. Pistache je napsaný v čistém C++17 a nabízí nízkoúrovňovou abstrakci. Umožňuje snadnou a intuitivní práci s HTTP serverem, nebo (asynchronním) HTTP klientem. Přes tzv. router můžeme směrovat (bind) požadavky z REST API na C++ funkce.[19]

Framework obsahuje type-safe implementaci a podporuje MIME typy. Mezi další funkce patří streamování odpovědí, které umožňuje odesílat data v sériích (chunked encoding). Podporuje dynamické cesty přes wildcards a pojmenované parametry.

Dokumentace Pistache je bohužel (zatím) poměrně krátká. Popisuje pouze minimum, pár příkladů jak začít, jak konfigurovat a spouštět HTTP server, jak přijímat a odesílat HTTP požadavky a odpovědi a jak nastavit router. Lákavé na něm je právě podpora C++17, což může usnadňovat integraci do nových projektů.[19] Pro příklad použití viz výpis 1.4 (`listenAndServe` zahájí blokující čekání a pokud nenastane chyba, program bude v cyklu obsluhovat příchozí požadavky. Spíše než dědit z `Http::Handler` je výhodnější použít `Router`, viz 1.5).

Nejrozšířenější alternativou pro Pistache je C++ REST SDK od Microsoftu.[20*]

```
using namespace Pistache;
class HelloHandler : public Http::Handler{
public:
    HTTP_PROTOTYPE(HelloHandler)
    void onRequest(const Http::Request& request,
           Http::ResponseWriter response) {
           response.send(Http::Code::Ok,
                        "Hello, World\n");
    }
};
// [...]
int main() {
    Http::listenAndServe<HelloHandler>("*:9080");
}
```

■ **Výpis kódu 1.4** Příklad použití Pistache knihovny, zdroj: Pistache Docs, Getting started

*Typicky se dále bude chtít komunikovat přes JSON, na to může posloužit knihovna `nlohmann::json`, pro větší efektivitu rapidjson nebo simdjson (což je pouze parser).

```
using namespace Pistache;
// [...]
void UsersApi::getUserId(const Rest::Request& request,
                         Http::ResponseWriter response){
    auto id = request.param(":id").as<int>();
    // ...
}
void UsersApi::linkUsers(const Rest::Request& request,
                         Http::ResponseWriter response){
    auto u1 = request.splatAt(0).as<std::string>();
    auto u2 = request.splatAt(1).as<std::string>();
    // ...
}
void UsersApi::startServer(){
    Http::Router router;
    Routes::Get(router, "/users/all",
      Routes::bind(&UsersApi::getAllUsers, this));
    Routes::Post(router, "/users/:id",
      Routes::bind(&UsersApi::getUserId, this));
    Routes::Get(router, "/link/*/to/*",
      Routes::bind(&UsersApi::linkUsers, this));

    Address addr(Ipv4::any(), Port(9080));
    auto opts = Http::Endpoint::options().threads(1);
    Http::Endpoint server(addr);
    server.init(opts);
    server.setHandler(router.handler());
    server.serve();
}
```

■ **Výpis kódu 1.5** Příklad použití routeru a bind, zdroj: Pistache Docs, Routing, upraveno autorem

1.3.2 Build systémy

1.3.2.1 CMake

Jedná se o open source, multiplatformní nástroj pro organizaci, sestavování a testování software. V rámci sestavování (spouštění kompilace, linkování atp.) je navrhnut tak, aby využíval nativní build systémy (automatická kompilace, hlídání změn v souborech a částečná kompilace atp.), jako je GNU make nebo ninja. Strukturu a konfiguraci projektu reprezentuje `CMakeLists.txt` soubor, ze kterého během procesu sestavování vygeneruje `Makefile` resp. `build.ninja`. Můžeme přidávat různé cíle (targets) sestavení (příkaz `add_executable`). Definovat podadresáře (`add_subdirectory`), které budou obsahovat podružné `CMakeLists.txt`. Linkovat knihovny (`target_link_libraries`), specifikovat mody vydání (release, debug, test) ad. Výhodou je, že se de facto jedná o standard a podporují jej většína C++ a C IDE. Samozřejmě, že lze používat také z příkazové řádky.[21]

1.3.2.2 npm/pnpm

Npm je registr software, který umožňuje sdílení a stahování balíčků (packages). Poskytuje CLI (`npm install`, `npm run dev` atp.). Registr sestává z veřejné databáze JavaScriptových balíčků.[22] Npm také řeší závislosti projektu (specifikované v `package.json` souboru), které v případě jejich absence na lokálním PC stahuje z registru na internetu. Jedná se o standardního správce balíčků pro Node.js, což je JS runtime.[23].

Jelikož tzv. `node_modules` (stažené balíčky z registru pro Node.js projekt) někdy zabírají hodně místa na disku a npm je stahuje znovu pro každý projekt, tato skutečnost dala vznik několika projektům ve snaze npm zefektivnit. Populárním je pnpm, který mj. závislosti stahuje na jedno místo na disku a do projektových adresářů vytváří hard linky. Používá také vlastní postup řešení závislostí. Tím se docílí zrychlení oproti npm. Základní příkazy jsou s npm identické.[24]

1.3.3 Pro webové aplikace a frontendy

1.3.3.1 Svelte

Svelte je framework pro vývoj webových aplikací. Odlišuje se od přístupů používaných ve většině jiných populárních frameworků, jako jsou React, Angular nebo Vue.js. Svelte využívá kompilace komponent na optimalizovaný JavaScript během fáze sestavení, namísto interpretace komponent za běhu na straně klienta. To vede k výraznému zlepšení výkonu a menší velikosti finálních sestavení.*[26,27]

*Kompilátor je napsán v JavaScriptu[25], tedy o nějaké obrovské efektivitě nemůžeme hovořit. To má zase své výhody, jako rychlost vývoje a velká přenositelnost. Také se nemusí

Způsoby, jakými se pracuje se Svelte se vyznačují svým *odporem* k nadbytečnému *boilerplate*[**], tedy zbytečně popisnému a dlouhému kódu, který se musí opakovat (typicky kopírovat nebo generovat – snippets atp.) na mnoha místech a nepřináší příliš užitné zátěže. Boilerplate zase na druhou stranu může být snadněji strojově zpracováván s jednoduššími parsery.

V populárním porovnání *krausest/js-framework-benchmark* JS frameworků se Svelte umisťuje na předních pozicích.[28] V porovnaní s React, Angular, Vue.js a Preact (viz tabulky A.1–A.9, podrob. informace v příloze) vychází vždy nejlépe.[†]

V bakalářské práci *Comparison of Modern JavaScript Web Frameworks* autorka (S. Jameel) dochází k závěrům, že (Svelte a React) oba frameworky mají své silné stránky. Svelte nepotřebuje tolik kódu (méně boilerplate), což ale uvaluje určitá omezení na styl psaní, zatímco React disponuje vyšší flexibilitou (za cenu zvýšené kvantity kódu). Svelte exceluje v nízké paměťové náročnosti, časech načítání a zpracovávání. React exceluje v čase vykreslení. Dle autorky Svelte nabízí více intuitivní průběh učení a vyžaduje méně kódování.[29]

Svelte umožňuje jednoduchou deklarativní syntaxi pro reaktivní aktualizace. Kdykoliv se změní stav aplikace, Svelte automaticky aktualizuje DOM, aby odrážel tyto změny (zjednodušení správy stavu). A to bez nutnosti použití virtuálního DOM. Při změně stavu dochází k přímé aktualizaci DOM, což přispívá k rychlejší reakci uživatelského rozhraní na interakce uživatele. Svelte má integrovanou podporou pro animace, efekty ad. Svelte lze poměrně snadno začlenit do stávajících aplikací.[27]

1.3.3.2 Alternativy

Alternativ k Svelte je mnoho. Nadneseně můžeme zmínit ve vývojářských kruzích známe fráze jako *0 dní od vydání dalšího frontend frameworku* nebo *co*

míchat několik naprosto odlišných technologií do sebe.

[**]Příkladem boilerplatu je například XML, kde je užitečná zátěž obalena štítky – což je žádoucí pro snadné strojové zpracovávání – ale tvořit taková delší XML ručně není ani zamýšlené, ani žádoucí, ani pohodlné. Takový boilerplate také trpí malou výpovědní hodnotou, resp. podílem užitné zátěže na celkové zátěži – to je zase kompenzováno dokonalejšími komprimačními nástroji. Java (8 aj.) je častým příkladem boilerplate programovacího jazyka – se svým vlajkovým `public static void main(String[] args)` v čele. U formátů typu XML, HTML atp. je boilerplate v podstatě esenciální, zatímco u programovacích jazyků se již názory více rozchází. Opakovat pořád stejné vzorce může být pro programátora zbytečně unavující. Nicméně také je důležité podotknout, že existují vesměs důležitější atributy a snesitelný boilerplate může být vyvážen precizním kompilátorem nebo bezchybnou správou paměti.

[†]React, Angular i Vue jsou nyní používanější než Svelte a stojí za nimi podstatně větší organizace. Můžeme předpokládat, že se budou také snažit jejich technologie zrychlit, tedy výsledky z porovnaní se musí brát s rezervou. Nicméně Svelte funguje taky nějakou dobu (cca od konce 2016) a pořád si vede velmi úspěšně, vycházejí nové verze a popularita spíše stoupá. Ve volbách webových technologií dnes člověk může stěží odhadovat, jaké technologie budou populární za deset let. Nicméně ze spousty lákavých alternativ Svelte připadá v úvahu momentálně nejvíce.

šedesát sekund, to nový JS framework atp. Zajisté nepokryjeme všechny alternativy. Nicméně uvedeme alespoň ty běžně známé, mezi které se řadí React, Angular a Vue.js.[30] Ještě můžeme zmínit Preact, což je jak se sám popisuje: *„Fast 3kB React alternative with the same modern API.“*[31]

1.3.3.3 Svelte Kit

Součástí ekosystému Svelte je také *SvelteKit* (používá build systém Vite), který poskytuje podporu pro směrování (založené na souborovém systému), SSR (server side rendering), prerendering, optimalizace obrázků ad. Jedná se o podobnou technologii jako je Next pro React nebo Nuxt pro Vue. Podporuje též integraci TypeScriptu. Svelte se řadí do kategorie tzv. *metaframeworks*.[32]

1.3.3.3.1 Form actions a Progresive enhancement SvelteKit nabízí tzv. form actions, což umožňuje odeslání dat na server prostřednictvím HTML elementu `<form>` bez nutnosti použití JavaScriptu na straně klienta. Formulář může mít výchozí akci nebo pojmenované akce, které lze vyvolat pomocí parametru dotazu. Tyto akce umožňují správu akcí HTML formuláře jako jsou: přihlášení uživatele, registrace a další. Data se dále zpracují na serveru. Výsledky akcí mohou být následně zobrazeny na stránce bez nutnosti celostránkového načítání. Pro příklad implementace viz výpis 1.6.[33]

Užitečnou funkcí Svelte Kitu je tzv. Progresive enhancement. Poskytuje to stránce odolnost vůči nepřítomnosti JavaScriptu na straně prohlížeče (ať už z důvodu pomalého připojení, zákazu nebo chybějící podpory). Tedy pokud JavaScript bude dostupný, mohou se provádět různá UX (nebo jiná) zlepšení, ale pokud ne, stránka bude fungovat nadále, pouze bez těchto *vylepšovacích* funkcí.

Funkčnosti, které se mají takto progresivně zavádět se dávají do atributu `use:enhance`. Typické využití je u formulářů (viz výpis 1.6).[34]

```
<form method="POST"
      action="?/create"
      use:enhance={() =>{
      // ... enhance code here ...
      }}>
  <input/> // ... form elements ...
</form>
```

Výpis kódu 1.6 Enhance a actions u formulářů, zdroj kit.svelte.dev, upravil autor

1.4 Vývojová prostředí

Dle průzkumu Stack Overflow z roku 2023 zobrazeného tabulkou 1.2 je **Visual Studio Code** nejpoužívanější vývojové prostředí vůbec (někdo by VS Code mohl nazvat editorem a až Visual Studio vývojovým prostředím).[35] Samozřejmě průzkum bude podléhat různým zkreslením. Můžeme spekulovat, zda na Stack Overflow budou zastoupeni všechny typy programátorů, nejspíš ne. Nicméně popularitu má velkou, také z důvodu, že se jedná o bezplatný opensource software, který je bohatý na funkcionality a stojí za ním společnost Microsoft. Opensource komunity také distribuují vlastní sestavy opensource kódu (např. VSCodium), jelikož sestava od Microsoftu obsahuje jimi nechtěnou telemetrii.[36,37]

IDE	zastoupení v %
Visual Studio Code	73.71
Visual Studio	28.43
IntelliJ IDEA	26.82
Notepad++	24.54
Vim	22.29
Android Studio	16.82
PyCharm	14.63
Jupyter Notebook/JupyterLab	12.74
Sublime Text	12.61
Neovim	11.88

■ **Tabulka 1.2** Průzkum používání IDE z roku 2023, zdroj Stack Overflow, vybráno horních 10, otázka: *Which development environments did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.* 86544 odpovědí

VSCode je dostupné na všechny běžné platformy, zejména z důvodu, že je napsáno za pomocí technologie Electron (mj.). Nevýhodou toho (zatím) je nutnost přibalit k sestavě téměř celý Chromium a Node.js stack, což mimo jiné plýtvá místem* a zvyšuje distribuční náklady.[38]

VS Code má bohatou nabídku tzv. *extensions*, rozšíření, které dodávají pokročilejší funkce a podpory pro různé technologie. Typicky pokud nějaká knihovna, framework atp. jsou alespoň trochu používané, budou mít rozšíření do VS Code (např. jazyk jako COBOL přes rozšíření IBM Z Open Editor[39]).

*V podstatě proti celé filozofii sdílených (.so nebo .dll) knihoven v operačních systémech a jakési efektivitě využití hardware, typické pro jazyky jako C, C++, Rust, Go ad. Jak neefektivní umí moderní technologie být, je někdy udivující. Nicméně zápory vyvažuje, jak říkají Američané, *more streamline process*, který spočívá v přenositelnosti a také v zpřístupňování domény programování a vývoje širšímu spektru lidí. Také JavaScript (příp. TypeScript) v kombinaci s HTML, CSS a nad tím vystavěnými frameworky dělá vývoj daleko interaktivnější a pro některé zábavnější, než terminálové seance s překlepy v `#define`.

Nejlepší zkušenost autora pro vývoj C nebo C++ je u IDE **Qt Creator**, který podporuje qmake a CMake build systém. Prostředí je vytvořeno zejména pro Qt, nicméně jakékoli jiné projekty v CMake jsou naprosto podporovány. CLion je alternativa s obsáhlejší funkcionalitou, nicméně je místy málo responzivní a zdrojově náročné (zkušenost autora). Obě dvě prostředí pro jazyky C/C++ podporují širokou škálu nástrojů statické analýzy kódu a QoL funkcionalit (generování konstruktorů, refactoring, přeskakovaní mezi hlavičkovými soubory atp.). Pro JavaScript lze využít **Webstorm** (JetBrains IDE, stejný styl jako CLion) nebo výše zmíněné VS Code.

Neovim je lákavou volbou pro spektrum programátorů používajících editory vi/Vim. Umožňuje modování v programovacícm jazyku Lua. Navazuje na tradici projektu Vim a využívá jeho stále aktuálního vývoje. Díky integraci jazyku Lua dokáže vykonávat funkce jako statická analýza kódu, *fuzzy search* atp. Může se chovat prakticky stejně jako vim, s tím, že další funkce jsou mnohem jednodušší na přidání. Každý si Neovim může přizpůsobit k nepoznání. To některé programátory přitahuje, některé zvyklejší na kompletnější balíčky zase odrazuje. Z principu věci je daleko tvárnější než typická vývojová prostředí a může se hodit (podobně jako vim, nano atp.) při potřebě rychle upravit nějaký soubor, kdy je nežádoucí čekat na načtení, opět v anglickém znění, *full-fledged IDE* (plnohodnotného, *se vším všudy*).[40]

Dobré zmínit, že více grafická a *klikací* vývojová prostředí mají často volbu nějaké vi/Vim emulace buď zabudovanou nebo v podobě pluginů, rozšíření atp. Například vim rozšíření pro JetBrains IDE, fakevim pro Qt Creator, pro Kate, VS Code a další. Takže lze mít do jisté míry výhody z obou přístupů, pokud je programátor ochoten podstoupit takový kompromis (ne vždy může být integrace chování vimu stoprocentní, u JetBrains prostředí a Qt Creatoru funguje velmi obstojně, dokonce je možné definovat vlastní `.vimrc`, kopírovat do schránky atp., v málo případech je nekonzistentní, což je většinou způsobeno kolizí funkčností obou přístupů, autorova zkušenost).

Dalších integrovaných prostředí pro vývoj je mnoho (některých i populárnějších než ty uvedené). Nicméně cílem předchozího výběru nebylo uceleně pojednat o vývojových prostředích a textových editorech, ale uvést relevantní prostředí k technologickému rámci tematiky řídicí a webové aplikace. Samozřejmě dále existují mnohá další vývojová prostředí, Netbeans a Eclipse typicky pro Javu, přes další JetBrains IDE, doménově specifická prostředí (vestavěné systémy, FPGA ad.), další opensource IDE na githubu až po Doom Emacs.*

1.5 Inteligentní technologie

*Ještě je namístě podotknout, že často vývojová prostředí můžou sloužit spíše jako *frontendy* pro různé další nástroje (analyzátory, *lintery* atp. často opensourcové), jako např. LSP, clangd, clang-tidy, ESlint a mnoho dalších.

1.5.1 Inteligentní domácnost

Autoři publikace *Inteligentní technologie* vymezují pojem inteligentní domácnosti (*smart home*) jako: *„rezidence, která využívá zařízení připojená k internetu pro dálkové sledování a správu různých zařízení a systémů (např. vytápění, osvětlení).“*[41]

Typicky zahrnuje technologie jako: inteligentní osvětlení (vzdálené ovládání, detekce osob v místnosti, úprava a regulace osvětlení dle dostupnosti světla atp.), inteligentní termostaty (plánování a monitorování domácí teploty, používající algoritmy strojového učení pro přizpůsobení chodu ad.), inteligentní zámky a otevírání vrat, bezpečnostní kamery, pohybové senzory (identifikace rozdílů mezi návštěvníky, zvířaty atp., detekce podezřelého chování), inteligentní elektrické zásuvky (vyhodnocování přetížení, vzdálené odpojení atp.), inteligentní vodoměry (detekce poruch jako promrzající potrubí, vzdálené vypnutí vody ad.). Dále autoři (Vargic et al.) zmiňují spotřebiče jako inteligentní TV, kávovary a ledničky.[41]

1.5.1.1 Výhody

Mezi výhody tzv. *inteligentního* přístupu patří: funkčnost vzdáleného sledování (zapomenutý zapnutý spotřebič, otevřené vrata atp. – klid duše), ohled na starší osoby, komfort (přizpůsobování, nastavitelnost, automatizace), efektivita (nakládání s energií, vodou atp.), automatizace vytápění a chlazení (plánování, předtápění atp.), inteligentní zavlažovací systémy (zavlažování trávníku pouze v případě potřeby, s minimem potřebné vody).[41]

1.5.1.2 Nevýhody

Mezi nevýhody se řadí zejména: vnímaná složitost a komplexita takových systémů (nicméně výrobci se typicky zaměřují na snižování složitosti a zlepšování UX), nedostatečná interoperabilita (neužívání standardizovaných nebo zaměnitelných protokolů, odlišné technologie odlišných výrobců – standardizační organizace a aliance adresují tuto nevýhodu), zabezpečení (riziko infiltrace systému, vloupání, přístup k systémům jako odemykání dveří atp., toto může způsobit velkou škodu), související ochrana dat uživatelů (zvláště při používání cloudových řešení atp., shromažďování údajů výrobci).[41]

1.5.1.3 Centrální jednotka

Autoři dále uvádějí pojem centrální jednotky – *inteligentního domácího rozbočovače*. Jedná se o HW zařízení, které funguje *„jako ústřední bod systému inteligentní domácnosti a je schopno sbírat data, zpracovávat data a bezdrátově komunikovat. Slučuje všechny rozdílné aplikace do jedné inteligentní domácí aplikace, kterou mohou majitelé domů ovládat vzdáleně. Nejznámější rozbočovače mají většinou hlasem aktivované systémy, obsahují virtuální asistenty,*

které se učí a personalizují inteligentní domácnost podle preferencí a vzorů rezidentů. Obsahují algoritmy pro strojové učení, které umožňují aplikacím domácí automatizace přizpůsobit se jejich prostředí.“[41]§

Například Google Home (Google Assistant ad.), Amazon Echo (Alexa), HomePod (Siri).[41]

1.5.1.4 Typické schopnosti senzorů a pohonů

- U senzorů
  - monitorování vnějších podmínek (teplota, tlak, vlhkost, rychlost větru atp.)
  - monitorování vnitřních podmínek (teplota, tlak, vlhkost, CO2 atp.)
  - detekce otevření/zavření dveří nebo oken
  - detekce pohybu
  - monitorování spotřeby energií/zdrojů (elektřina, voda, plyn)
- U pohonů*
  - obecné elektrické spínače a akutátory
  - ovládání otevření/zavření okna a žaluzií
  - vytápění, chlazení, klimatizace, zvlhčovaní a jiné úpravy vzduchu
  - osvětlení
  - monitorování spotřeby energií/zdrojů (elektřina, voda, plyn)
  - alarmy, hlásiče, zabezpečovací prvky ad.[41]

1.5.1.5 Alternativní inteligentní systémy

Samozřejmě je možné si systém pro centrální jednotku vytvořit vlastní silou (s využitím desek jako Raspberry Pi, na Linuxové stanici atp.). Existují také opensource systémy pro tento účel, například: OpenHab (Java), Homeassistant (Python), Calaos (server v C++), Domoticz (C++), dále MisterHouse a OpenMotics.[41,42,43]

1.5.2 Internet of Things

§Publikace ovšem pomíjí decentralizované systémy, jako je KNX. Takové systémy naopak centrální řízení nemají (nemusí mít) a typicky zařízení v KNX síti fungují nezávisle. To jim avšak nebrání v případné integraci na nějaké zmíněné *centrální jednotky*, pokud se tomu projekt přizpůsobí.

**Akčních členů*, dle KNX terminologie

1.5.2.1 Základní přehled

Internet of Things (IoT, česky *internet věcí*) je pojem, se kterým, dle autorů publikace *Enabling Things to Talk, kap. Introduction to the Internet of Things*, v roce 1999 přišel Kevin Ashton. Od té doby se stává čím dál více rozšířeným. Koncept IoT usnadňuje konvergenci trhů, jako jsou: trh se systémy pro domácí automatizaci, výroba a distribuce energie, logistika, automobilový průmysl a telekomunikace.[44]

Internet věcí je kombinací technologického vývoje (levnější a účinější HW, přístupný SW, otevřené standardy atp.) s lidskou touhou být informován o všem, co se děje v bezprostředním i vzdáleném okolí. Je logickým důsledkem transformace výpočetní síly jednoho stroje do širšího prostředí: *the environment as an interface*. Autoři zmiňované publikace nahlíží na IoT jako na nezastavitelný a transformační trend.[44]

Vize IoT mají ideologický i technický charakter, hovoří se o propojenosti všeho se vším, jeho využití ve vládních institucích, logistice, inteligentních domácnostech, chytrých městech (*smart cities*), chytrých továrnách, maloobchodním prodeji a ve zdravotnictví.[44]

Rámec IoT naplňuje sada technologií a zařízení. Je to celý systém koncových zařízení, protokolů, senzorů, kamer, síti a sběrnic a SW funkcionalit. Konkrétně například technologie související s RFID (Radio Frequency Identification), sítě 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) nebo ZigBee.[44]

1.5.2.2 KNX IoT

Jedná se o poměrně aktuální záležitost (viz také základní informace o KNX v sekci 1.7.3.1), publikace standardu KNX IoT: KNX IoT Point API a KNX Information Model jsou z června roku 2023.*[45]

Standard poskytuje (mimo jiné) tzv. KNX IoT Point API, ve kterém specifikuje chování KNX IoT zařízení. Toto API mapuje data KNX aplikační vrstvy na REST model. K přenosu se používá CBOR/JSON reprezentace dat. Klient tak použitím tohoto rozhraní může zapisovat a číst hodnoty nebo se přihlásit k odběru (subscribe) tzv. Point events (události ze sběrnice – switch on/off atp.).[46]

Standardizuje terminologii jako MaC (Management Client), kterého může představovat např. webová aplikace. Základní čtyři funkcionality KNX IoT Point API představují:

Discovery Stará se o nacházení zdrojů a zařízení. Používá se CoAP (CoRE) protokol (odlehčená varianta HTTP).

*Standard je veřejně dostupný, nicméně je poměrně nepřímočaré s k němu dostat (Google vyhledá relevantní výsledek až při hesle „3_10_3 KNX IoT Information Model“ – na obranu i „KNX IoT download“ – kde se dá prokliknout na stažení standardu na SharePointu.) KNX Information Model je bravurně zdokumentován rigorozní ontologií, na druhou stranu co znamená je otázkou, která může zůstat nejasná i po několika průchodech textem.

S-Mode Messaging Bezpečný vzor/protokol pro skupinovou komunikaci, kdy producent posíla zprávu konzumentům o změně v doméně. Přes MaC pak lze konfigurovat tyto skupinové komunikace.

Point Read, Write, Publish, Subscribe čtení dat přes CoAP GET a zápis přes PUT/POST, možnost přihlášení se (subscribing) ke chtěným událostem.

Security OSCORE (aplikační vrstva – pro S-Mode messaging, r/w/pub/sub) nebo (D)TLS (transportní vrstva).[46]

Dále je veřejně k dispozici technologický rámec (*stack*) k vývoji KNX IoT zařízení. Většina kódu je napsána v čistém C a je vystaven na GitHubu pod Apache-2.0 licencí.[47]

1.5.3 Inteligentní budovy

Ze sekce inteligentních domácností se dostáváme do inteligentních budov. Nové budovy se dnes staví s ohledem na energetickou náročnost. Celá odvětví technologie se věnují tzv. inteligentním budovám*, tedy budovám, které používají moderní systémy automatizace jak pro provozní a komfortní účely, tak i pro účely nízké spotřeby energie. Dále se často setkáváme s pojmem udržitelnosti nebo šetrnosti výstaveb. Zda jsou některé otázky nepřirozeně nadužívané a místy slouží zištným zájmům skupin více nerozebírejme. Nepopiratelným faktem zůstává, že věda uspořádává čím dál sofistikovanější technologie, které nám umožňují využít izolačních kvalit stavebních materiálů se systémy vzduchotechniky a vytápění, zabudovat tepelná čerpadla nebo fotovoltaické panely a používat systémy řízení, které vyhodnocují data ze senzorů, regulují klima uvnitř budovy a jinak s daty zacházejí.[48]

1.5.4 Inteligentní měření

Běžně se jedná o elektroměry (může být i ve smyslu měření vody nebo plynu), které poskytují informace nejen o celkové spotřebě, ale i o průběžných časech spotřeby. Umožňují flexibilitu ve vyúčtování elektřiny, mohou upozornit na výpadky nebo na manipulaci se zařízením. Některé podporují bezdrátovou komunikaci. Inteligentní měřiče byly představeny v roce 2009.[49]

*Terminologie domény automatizačních systémů je poměrně roztěkaná. Přídavné jméno inteligentní u budovy se používá ve smyslu jakési kategorie budov, které zahrnují a kombinují to nejnovější z poznání technologie v oblasti (zejména informační, ale i dalších), nebo jisté autonomie, samosprávy budovy. Typicky kdo užije termín inteligentní budova, si sám dosadí nějaký (typicky kladný) význam v závislosti na jeho potřebách. V anglické literatuře se můžeme setkat s pojmy jako BACS (Building Automation and Control Systems), HVAC (Heating Ventilation and Air-Conditioning), Smart Buildings, ve spíše řídkých případech s Intelligent Buildings (tak se to spíš prezentuje spotřebitelům z marketingového až PR hlediska).

Mimo technologické výdobytky vyvolává také určité obavy v souvislosti s jeho širším přijetím:

- obavy o zdraví
- obavy o bezpečnost (např. možné selhání, požáry)
- otázka soukromí (ochrana osobních dat, zneužití dat, krádeže)
- ekonomická otázka (ve finále ztrátovost, neschopnost nebo nevůle financování, potenciální zdražení energií)[49]

1.5.4.1 Průběhové měření

„Průběhové měření je takové měření, při kterém je kontinuálně zaznamenávána hodnota energie nebo střední hodnota výkonu v měřicím intervalu. Měřicím přístrojem může být podle provedení měřicího zařízení buď samotný elektroměr, nebo externě připojený registrační přístroj. Velmi často se jedná o kombinaci měření průběhového s měřením ostatním, to znamená, že jsou současně využívány příslušné registry (číselníky) energie a výkonu, často jak tarifní, tak i sumární. Registry mohou být nastaveny pro zobrazování stavů, anebo rovnou pro zobrazování spotřeby v daném účtovacím období. Vždy záleží na konkrétním použitém elektroměru a možnostech jeho uživatelského nastavení, které provádí příslušný provozovatel soustavy.“[50]

Vyhláška o měření elektřiny nyní obsahuje pojem průběhového měření a detailněji specifikuje funkcionality měřicích prvků (více v sekci 1.8.1.2) *Vyhláška o měření elektřiny*). Intervaly byly typicky hodinové, nyní se od 1. 7. 2024 počítá s 15ti minutovými.[51]

1.6 Energetická terminologie

1.6.1 Elektřina

1.6.1.1 Složky ceny elektřiny

Regulovaná část a neregulovaná část. Regulovaná část je pevně stanovena a stanovuje ji ERÚ. Neregulovanou část tvoří cena **silové energie** na trhu.

Regulovanou část tvoří:

- **distribuční poplatky** dle odebraného množství (stanovené jednotlivými distributory)
- fixní platba za jistič podle jeho hodnoty
- systémové služby pro zajištění rovnováhy mezi výrobou a spotřebou (provoz ČEPS)
- poplatek operátorovi trhu OTE

- příspěvky na obnovitelné zdroje[52]

Distribuční poplatky činní na regulované části největší podíl. Část za silovou elekřinu (neregulovaná) putuje k dodavateli. Regulovaná část jde k distributorovi.[53]

Pro rok 2024 (dle zprávy ERÚ) je regulovaná část stanovena na 39.2 % z celkové ceny (bez daní) u domácností a malých podnikatelů, 29 % u velkoodběratelů na hladině vysokého napětí a 21 % na hladině velmi vysokého napětí. Oproti roku 2023 se zvýšila.[54]

„Růst regulovaných složek cen elektřiny i plynu má společného jmenovatele. Stojí za ním především vyšší náklady na provoz soustav, do kterých se promítají vyšší ceny elektřiny a plynu na volném trhu. Ty bezprostředně ovlivňují náklady na krytí technických ztrát v soustavách a v případě elektroenergetiky také ceny systémových služeb, které zajišťují udržování výkonové rovnováhy.“[54] (z tisk. zprávy ERÚ)

1.6.1.2 Elektrizační soustava

Elektrizační soustava zahrnuje všechny aspekty výroby, přenosu, transformace a distribuce a spotřeby elektřiny. Včetně přípojek, vedení, měřicí, řídicí, zabezpečovací, informační a telekomunikační techniky.[55]

1.6.1.2.1 Přenosová soustava Přenosová je část elektrizační soustavy, která tvoří přenosovou cestu pro napájení velkých stanic nebo uzlů, zpravidla vyššího napětí: Zvláště vysoké napětí (400 kV), velmi vysoké napětí (220 kV, část 110 kV). *„Představuje páteřní rozvedení výkonu z velkých elektráren po celém území ČR.“*[55]

1.6.1.2.2 Distribuční soustava Distribuční soustava je konečná část elektrizační soustavy, sloužící pro dodávku elektrické energie odběratelům. Patří sem okružní a paprsková síť, tedy regionální a lokální distribuční soustavy pro rozvod a užití elektrické energie.[55]

1.6.2 Trh s elektřinou

1.6.2.1 Spotové ceny

V běžné řeči se můžeme setkat s tzv. *přechodem na spot*. Typicky se tím míní změna způsobu vyúčtování elektřiny. Oproti fixní sazbě, kde si odběratel smluvně ujedná fixní cenu za elektřinu s distributorem, u které nezávisí (zjednodušeně, vyjma speciálních tarifů ad.) na době odběru energie, spotová cena je určena momentální cenou komodity silové energie na trhu. Klasický elektroměr průběžně přičítá spotřebovanou energii a ve stanovených termínech dojde k jeho odečtu. Toto odečtení dosud typicky probíhalo za fyzické přítomnosti, tedy opsáním či jiným zdokumentováním tzv. hodin. Průběhové, *inteligentní*,

elektroměry specifikované legislativou mají funkce typu vzdáleného odečtu, vzdáleného omezení výkonu, a další (viz sekce 1.8.1.2).[50]

Nově tyto smlouvy, ve kterých se cena určuje podle aktuální ceny elektřiny na burze, upravuje přímo zákon, tzv. *dynamické určení ceny* (Energetický zákon § 11ca). Spotřebitel má podle zákona právo smlouvu s dynamickým určením ceny kdykoli vypovědět – výpovědní lhůta je jen jeden měsíc.[56]

1.6.2.2 Denní trh

Na denním trhu (Day-Ahead Market) se obchoduje s elektřinou, která má být dodána následující den. To umožňuje účastníkům připravit se předem na spotřebu energie. Někdy je označován jako spotový trh. Organizuje ho operátor trhu OTE.[57]

Je založen na aukčním modelu. Na základě obdržených nabídek a poptávek elektřiny den dopředu jsou pak stanoveny ceny. Nejdříve se uspokojí poptávka po nejlevnějších zdrojích (OZE ad.). Pokud se trh neuspokojí, párují se dále poptávky s nabídkami s vyšší cenou. Výsledná cena pro všechny se pak odvíjí od nejdražší aktivované nabídky, u tzv. závěrné elektrárny (typicky plynová nebo dražší). Tento typ párování se nazývá **Merit Order**.*[50,52]

1.6.2.3 Vnitrodenní trh

Cílem operací na vnitrodenním trhu (Intra-day Market) s elektřinou je promptní řešení přebytku nebo nedostatku elektřiny. Fungování vnitrodenního trhu tak přispívá vyrovnávání zátěže elektrizační soustavy. Je uzavírán postupně po jednotlivých hodinách. Uzavřené obchody z VT pak OTE zahrnuje do systému vyhodnocování a zúčtování odchylek.

Díky jeho aspektu vyrovnávání zátěže získal na mimořádném významu mj. díky trendu obnovitelných zdrojů energie, u kterých je jejich provoz těžko předvidatelný a často závisí na výkyvech počasí. Uzavírané hodnoty mohou dosahovat záporných hodnot.[50]

Důležité je ještě zmínit, že v rámci výše zmíněných trhů dochází k velmi častému zahraničnímu obchodování a vyrovnávání zátěže.[50]

1.6.2.4 Dodávky dle zátěže

1.6.2.4.1 Base load Dodávka v základním zatížení (Base load) je minimální množství elektřiny, které je neustále vyžadováno během trvání dodávky.[58]

*Hypoteticky se může stát, že poslední volný zdroj za 5 kč/KWh bude naplněn a další v řadě bude už velmi nákladný, třeba i řádově 50 kč/KWh. Pak cena na tu danou dobu bude cena 50 kč/KWh. Takto striktně je to také z důvodu, že se výroba musí přesně rovnat spotřebě. Přílišný odběr by mohl způsobit škody na elektrizační soustavě.

1.6.2.4.2 Peak load Dodávka ve špičce (Peak load) je v době, kdy je poptávka po elektřině na svém maximu, v období od 8.00 do 20.00 hodin.[58]

1.6.2.4.3 Offpeak load Dodávka mimo špičku (Offpeak load) je opakem k dodávce ve špičce.[58]

1.6.2.5 Operátor trhu OTE a.s.

OTE, a.s. (Operátor trhu s elektřinou a plynem) vykonává činnost tržního operátora v souladu se zákonem č. 458/2000 Sb., o podmínkách podnikání a o výkonu státní správy v energetických odvětvích (energetický zákon). Společnost koordinuje krátkodobé i dlouhodobé obchodování s elektřinou a plynem a zajišťuje bilancování mezi výrobou a spotřebou těchto energií.

1.6.2.6 Funkce a činnosti OTE, a.s.

- **Organizace trhu s elektřinou:** OTE řídí operace denního a vnitrodenního trhu s elektřinou, které umožňují efektivní alokaci zdrojů v reakci na měnící se podmínky výroby a spotřeby.
- **Bilancování:** Společnost zajišťuje bilancování dodávek elektřiny, což je nezbytné pro zajištění stability a spolehlivosti distribučního systému.
- **Podpora integrace OZE:** OTE podporuje začleňování obnovitelných zdrojů energie do energetického mixu, což přispívá k zajištění diverzifikace zdrojů.
- **Transparentnost trhu:** Poskytování informací o trhu je zásadní pro spravedlivou a efektivní tržní soutěž. OTE zveřejňuje data o cenách, objemech a dalších relevantních indikátorech.
- **Regulační shoda:** Spolupráce s Energetickým regulačním úřadem a dalšími institucemi zajišťuje dodržování legislativních a regulačních požadavků. A další.[59]

1.7 Vybrané technologie budovy

1.7.1 Vzduchotechnika

1.7.1.1 Zpětné získávání tepla

V budovách používajících vzduchotechniku (sání, úprava a distribuce vzduchu z venku do budovy a odsávání vzduchu z budovy ven) typicky dochází v zimě k odsávání a vývodu ohřátého vzduchu zevnitř budovy ven. Naopak v situaci,

kdy je venku tepleji než uvnitř, se saje teplý vzduch z venku, který se pak smíchává vevnitř a tím tedy ohřívá vnitřní vzduch. Což v létě může být nežádoucí a je nutné teplý vzduch nejdříve ochladit a v zimě naopak ohřát.

Pro využití výstupní nebo vstupní energie ze zavedl mechanismus zpětného získávání tepla (ZZT).[60]

Výměníky ZZT se typicky dělí na dvě kategorie: Rekuperační a regenerační. Liší v způsobu přenosu tepla. Rekuperační výměníky využívají stěnu nebo deskový systém pro přenos tepla z odváděného vzduchu do přiváděného, aniž by došlo k přímému kontaktu vzduchů. Naopak, regenerační výměníky tepla používají rotační disk, který absorbuje teplo z odcházejícího vzduchu a předává ho přiváděnému vzduchu přes kontakt s tímto rotujícím médiem, což umožňuje i přenos vlhkosti.[60]

1.7.2 Vytápění a chlazení

1.7.2.1 Tepelná čerpadla

Dle webu *tzbinfo*: *„Tepelné čerpadlo přeměňuje nízkopotencionální teplo odebrané z okolního prostředí (venkovní vzduch, půda, voda) na teplo vhodné k vytápění a ohřevu teplé vody pomocí komprese vypařeného chladiva.“*[61]

Sestává se ze čtyřech základních částí: výparník, kondenzátor, kompresor a expanzní ventil. Základní druhy jsou (dle přivedených medií) vzduch–voda, vzduch–vzduch (klimatizace), země–voda, voda–voda. TČ země–voda funguje na bázi vrtného pole (jedná se o hloubkové vrty v rozmezí 50–150 m hloubky) nebo plošného kolektoru. TČ země–voda se provozují s akumulační nádrží a také je lze využít k pasivnímu chlazení.[61]

1.7.2.1.1 Regulace Regulace TČ závisí na konkrétních výrobcích. Můžeme se setkat s řízením protokolem Modbus (varianta RTU nebo TCP). Datový model je popsán tabulkami registrů, ze kterých je možno číst nebo do nich zapisovat. Modbus rozlišuje mezi tzv. Discrete Input (1 bit read), Coil (1 bit r/w), Input Register (16 bit read) a Holding Register (16 bit write). 1 byte z modbus rámce je tzv. Function Code (kód funkce), který specifikuje o jakou operaci se bude jednat. 16 bitové hodnotě se říka word.[62]

1.7.3 Sběrnicová komunikace

1.7.3.1 KNX

KNX zařízení jsou vzájemně kompatibilní elektronické prvky navržené a vyrobené v souladu s KNX standardem pro budovy a domácí automatizaci. KNX je decentralizovaný sběrnicový systém zahrnující několik typů přenosových médií, jako je např. Twisted Pair (TP).

Zařízení (snímače a akční členy) jsou napojena na KNX sběrnici a také jsou z ní napájena. Přednost KNX TP je právě jednoduchost v rozvodu, tzn.

na jednoduchém krouceném dvoudrátu běží jak data, tak napájení. KNX TP nedosahuje vysoké propustnosti, lze proto směrovat KNX provoz i po síti jako je Ethernet pomocí rozhraní (brány) KNX TP – KNXNet/IP.[63]

1.7.3.2 MODBUS

MODBUS je komunikační protokol aplikační vrstvy. Umožňuje komunikaci typu klient–server napříč řadou sběrnic a sítí (RS–232, RS–422,RS–485, TCP/-IP over Ethernet, rádiové sítě ad.). Protokol byl navržen již roku 1979. Definuje pojmy PDU (Protocol Data Unit) a ADU (Application Data Unit). PDU se skládá z kódu funkce (1 byte) a následuje datová část. ADU je doplněna o 1B adresu a 2B CRC.

Mezi jeho implementace se řadí MODBUS TCP/ IP a MODBUS Serial Line (RTU a ASCII vysílací režimy).

Aplikační paměť zařízení implementující MODBUS datový model sestává z tzv. diskrétních vstupů, cívek, vstupních registrů a uchovávácích registrů. Protokol dále definuje trojice požadavek, odpověď, chyba pro jednotlivé kódy funkcí.[64]

1.8 Energetický trh

1.8.1 Právní aspekt

Nejprve si ukážeme legislativu související s aktuálním děním na energetickém trhu. Jak už se stalo zvykem, legislativu České republiky často předchází legislativa Evropské unie, ať už v podobě směrnic nebo nařízení. Ne, že by trendy na energetickém trhu apriorně udávala legislativa, to je spíše naopak, nicméně pokud již dojde k plošnému uzákonění některých postupů a technologických opatření, je, v právním státě, vhodné o tom vědět. Jak se taková legislativa tvoří zůstává dále mimo meze naší práce.

Právních předpisů vyjadřujících se na toto téma je mnoho. V dnešní době jsou typicky ovlivněny *harmonizační* snahou Evropské unie o jednotná pravidla a jednotný trh v jejím rámci. K zásadním právním úpravám patří *Energetický zákon*[65], nyní novelizovaný a v částečné účinnosti (1. 4. 2024). Energetický zákon zejména stanovuje podmínky podnikání a podmínky výkonu státní správy v energetických odvětvích. *„Zákon obsahuje normy jak veřejnoprávní (výkon státní správy), tak i soukromoprávní povahy (podmínky podnikání).“*[66]

Dalším podstatným předpisem je *Vyhláška o měření elektřiny*[51], která pojednává o typech měření, rozdělených do několika skupin. Skupiny jsou rozděleny hlavně dle množství odběru a místa připojení do distribuční soustavy a s tím souvisejícího napětí. Z vyhlášky vyplývají povinnosti pro distributory v následující době vyměnit běžné elektroměry, v jisté kategorii C, za průběhové elektroměry splňující technické požadavky určené vyhláškou.

Jak se shodují i někteří právníci, u kterých jejich práce kompilací právních předpisů a jejich výkladů začíná a končí, oblast energetiky je natolik rozsáhlá, že je stěží možné popsat všechny aspekty v rozumném rozsahu. Nicméně přes to je důležité se vybranými aspekty zabývat, jelikož energetická politika je klíčovou politikou Evropské unie a její výstupy se zprostředkovaně dotýkají veškerého průmyslu a domácností.

Jakkoli některé právní prameny a inspirace sahají hluboko do historie, zejména do období Pax Romana –*„renezance studia římského práva se stala nejvýznamnějším fenoménem tvorby kontinentálního právního systému“* – v oblasti energetiky se jedná o relativně mladou právní disciplínu.[67*] Nicméně jak pojednává diplomová práce (G. Blahoudkové) *Energetické právo České republiky* z roku 2019, není snadné energetické právo definovat. Samozřejmě zmiňuje také jeho velkou propojenost s technologickým pokrokem posledního století.[68] Jakkoli můžeme definování čehokoli označit jako nesnadné, v přihlédnutí k propojenosti energetiky s oblastmi životního prostředí, automobilismu, obnovitelných zdrojů, ekonomie a mezinárodních vztahů, snadno dojdeme k podobnému úsudku.

1.8.1.1 Energetický zákon

Energetický zákon tvoří pro Českou republiku právní rámec pro podnikání a výkon státní správy v energetických odvětvích, kterými se míní elektroenergetika, plynárenství a teplárenství. V průběhu let byl několikrát novelizován a upravován, a to dosud (k 25. 4. 2024) 42krát, pokud jako základ bereme Zákon č. 458/2000 Sb, a je prováděn 221 předpisy včetně dnes již zrušených vyhlášek a sdělení Energetického regulačního úřadu (ERU).[65,69]

Energetický zákon udává existenci Energetickému regulačnímu úřadu jako dalšímu orgánu státní správy (konkrétně § 17). Zavádí instituty operátora trhu, oprávněného zákazníka, provozovatele přepravní soustavy a distribuční soustavy. Zákon do sebe zapracovává také tematiku ochrany životního prostředí, udržitelnosti a rozvoje v odvětví energetiky.[65]

Prameny energetického zákona sahají jak do české (československé) historie (např. Zákon č. 79/1957 Sb. a 67/1960 Sb. ad.), tak do mezinárodního práva. Znění zákona jsou ovlivněna účastí České republiky (dříve Československa) v organizacích nebo iniciativách jako jsou EEC, GATT, ECT, UNFCCC ad.[68] Podrobnější rozbor pramenů zákona a vlivů na něj zůstává mimo meze naší práce.*

Novely měly často formu tzv. energetických balíčků, v souvislosti se *zimním energetickým balíčkem* z roku 2019 píše J. Pokorný ve svém článku *Evropské*

*Samozřejmě můžeme zmiňovat ještě novější odvětví práva blíže informatice, jako jsou zpracovávání a ochrana osobních údajů, síťová bezpečnost a konkrétně předpisy NIS1, NIS2, DORA aj., kterým se aktuálně dostává regulace.

*Úvod do energetického práva, jeho prameny a organizace působící v rámci energetického práva je popsán ve VÍCHA, Ondřej. *Základy horního a energetického práva.* Praha: Wolters Kluwer, 2015. 228 stran. ISBN 978-80-7478-919-9 na stranách 72–88.

energetické právo: Vybrané novinky zimního energetického balíčku:

„Je otázkou, zda by si tato část právního řádu České republiky nezasloužila novou právní úpravu, reagující pružněji na technologický vývoj a pravděpodobný rozvoj dalšího sektoru ekonomiky v souvislosti s poskytováním služeb v energetice. Přestože širší odborná veřejnost sleduje zprávy o otevírání bateriových úložišť ve vzdálených státech i okolních státech EU, málokdo si uvědomuje, že komerční provoz takových zařízení v ČR není z důvodu technické zaostalosti, ale chybějící právní úpravy jejich komerčního provozování. Což je zřejmě jeden z nejmarkantnějších nedostatků stávajícího energetického zákona, nicméně dalších příkladů by šlo nalézt celou řadu. Je tedy otázkou, zda namísto novelizace mnohokrát novelizovaného předpisu nepřejít k sepsání celého nového, moderního energetického zákona. Podle autora jednoznačně ano.“[70]

Od roku 2019 se do aktuálního znění (Zákon č. 469/2023 Sb.) ještě provedlo 13 úprav.[69] Nutno říci, že některé úpravy jsou pouze mechanického charakteru a užitná zátěž je minimální, nicméně důležitá pro konzistenci provázaných právních předpisů. Z Elektronické Sbírky zákonů a mezinárodních smluv také můžeme vyčíst informace o budoucích zněních (1. 7. 24, 1. 1. 25 a 1. 1. 27). Otázka autora předešlého citátu, zda novelizovat již mnohokrát novelizované nebo sepsat zcela nový zákon, zatím zůstává zodpovězena ve prospěch první teze.

Zákon mimo jiné upravuje podmínky pro vstup na trh, licencování provozovatelů energetické infrastruktury, podmínky pro výstavbu a provozování energetických zařízení a další. Časté novelizace jsou dané tím, že je třeba, aby reagoval na změny v technologii, tržních podmínkách a legislativním prostředí – dnes zejména na činnost unijní politiky. Novely zahrnují např. oblast obnovitelných zdrojů energie, zlepšování energetické účinnosti nebo komunitní energetiku.[70,65]

1.8.1.2 Vyhláška o měření elektřiny

Vyhláška č. 359/2020 Sb. o měření elektřiny ve znění vyhlášky č. 375/2023 Sb. (nové znění nemění nic zásadního pro měření typu C) stanovuje pravidla a technické požadavky pro měření elektřiny v rámci České republiky. Tato vyhláška specifikuje typy měření (od nejvyššího typu k nejnižšímu A, B, C), způsob jejich provádění, technické parametry měřicích zařízení a povinnosti subjektů zapojených do procesu měření a vyúčtování elektřiny.[51]

Z § 5 odst. 2 písm. a) tedy *„(2) Alespoň měřením kategorie C1 nebo kategorie C2 musí být měřena elektřina a) odebíraná z distribuční soustavy na napěťové hladině do 1 kV s přímým měřením a ročním odběrem elektřiny v odběrném místě, v připojené distribuční soustavě nebo výrobně elektřiny přesahujícím 6 MWh“* vyplývá právě povinnost provozovatelům zajistit průběhové měření těmto odběratelům, zmiňovaná v úvodu. Vyhláška ještě zmiňuje: *„(3) Není-li technicky možné nebo ekonomicky únosné elektřinu podle odstavce 2 písm. b) nebo c) měřit měřením kategorie C1 nebo C2, musí být měřena měře-*

ním typu B nebo měřením kategorie C3.“ Nicméně měření typu B je v podstatě striktnější a C3 měření se oproti C1 nebo C2 liší v tom, že nemá funkci technického blokování spotřebičů, ani dálkového odpojení, připojení nebo omezení výkonu (u C1). Průběžný záznam střední hodnoty činného výkonu a komunikační rozhraní je nutný.

Nejbenevolentnějším je *„ostatní měření kategorie C4, které může být průběhové a může být s dálkovým přenosem údajů.“* Kam spadají všichni ostatní odběratelé nepokryti v kategoriích A, B, C1, C2 a C3. U *„C4 je zpracování a přenos údajů prováděn nejméně jedenkrát za rok“*, čili nyní pořád běžná praxe. Co je ještě zajímavé dodat, tak že měření C4 je také možné uplatnit *„u odběrného místa s odběrem elektřiny z distribuční soustavy, ve kterém není připojena výrobna elektřiny, kde není technicky nebo ekonomicky možné instalovat měřicí zařízení, které využívá měření podle odstavce 2, § 3 nebo 4,“* tedy i u odběratelů nad 6 MWh aj., pokud nejsou také výrobnami – například mající fotovoltaickou elektrárnu. To poskytuje jakýsi právní manipulační prostor, avšak prokazovat, že instalace nebyla technicky či ekonomicky možná, bude praktické nejspíš u minority případů.

Měřicí zařízení musí splňovat technická specifika stanovená vyhláškou. Vyhláška detailně specifikuje požadavky na:

- Přesnost měření, která musí odpovídat třídě přesnosti stanovené pro daný typ zařízení,
- ochrana proti manipulaci s měřicím zařízením, včetně zabezpečení dat a softwaru,
- kompatibilita s komunikačními systémy pro dálkový odečet a integraci s dalšími energetickými systémy.

Vyhláška klade povinnosti na:

- Provozovatele distribuční soustavy, který musí zajišťovat instalaci, údržbu a pravidelnou kalibraci měřicích zařízení,
- odběratele, kteří musí umožnit přístup k měřicím zařízením pro účely kontroly a údržby,
- energetické dodavatele, kteří jsou odpovědní za správné vyúčtování spotřeby na základě dat z měření.[51]

Byť se nabízí další zajímavé oblasti k probádání např. figurující v konceptu klimaticko-energetické unie (komunitní energetika, bateriová úložiště, Lex OZE aj.) a bylo by užitečné se o nich dozvědět více také z jiných zdrojů, než jakými jsou prodejci fotovoltaických elektráren, dodavatelé nebo stránky Evropské rady, dovolme si zde zakončit oblast právního aspektu.

1.8.1.3 Shrnutí legislativy

V předchozí rešerši jsme se pokusili nastínit spletitost českého energetického práva a přihlédneme-li k nutnosti orientovat se jak v unijním, tak i mezinárodním právu, dovolit si důsledně shrnout klíčovou legislativu může opravdu jen skupina odborníků a podnikatelů z praxe. Tedy shrnutí je potřeba brát jako nevyčerpávající:

- Směrnice Evropského parlamentu a Rady (EU) 2019/944 ze dne 5. června 2019 o společných pravidlech pro vnitřní trh s elektřinou a o změně směrnice 2012/27/EU,
- Nařízení Evropského parlamentu a Rady (EU) 2019/943 ze dne 5. června 2019 o vnitřním trhu s elektřinou,
- Vyhláška č. 359/2020 Sb., o měření elektřiny ve znění vyhlášky č. 375/2023 Sb.,
- Cenového rozhodnutí Energetického regulačního úřadu č. 7/2023.

A v souvislosti s přechodem na 15minutový měřicí interval (dle souhrnu OTE):

- *Nařízení Komise (EU) 2017/2195, kterým se stanoví rámcový pokyn pro obchodní zajišťování výkonové rovnováhy v elektroenergetice – povinnost (čl. 53) implementovat do tří let od vstupu nařízení v platnost interval zúčtování odchylek o délce 15 minut*
- *Rozhodnutí ERÚ ze dne 29. 6. 2018 o udělení výjimky (derogaci) z tohoto požadavku, maximálně však do 1. 1. 2025*
- *Nařízení Evropského parlamentu a rady (EU) 2019/943 o vnitřním trhu s elektřinou – čl. 8(4) Od 1. ledna 2021 činí interval zúčtování odchylek 15 minut ve všech oblastech plánování, ledaže regulační orgány udělí obecnou nebo individuální výjimku. Obecné výjimky lze udělit pouze do 31. prosince 2024.*
- *Vyhláška č. 359/2020 Sb. o měření elektřiny předpokládá zahájení zasílaní měření v 15 minutové granularitě k 1. 7. 2024*
- *Vyhláška č. 408/2015 Sb. o Pravidlech trhu s elektřinou pracuje s termínem přechodu na 15 minutovou zúčtovací periodu k 1. 7. 2024*[71]

1.8.2 Veřejné rozhraní webových služeb OTE

OTE poskytuje veřejně přístupné webové služby ještě mimo systému OTE CS. Poskytuje SOAP rozhraní s WSDL zpřístupněným na `http://www.ote-cr.`

`cz/services/PublicDataService/wsdl`, které rozhraní popisuje. Jednotlivé služby a formáty jsou dostupné v *Uživatelský manuál webové služby OTE*.

Řadí se mezi ně např. GetRutList (bez parametrů, vrací všechny registrované účastníky trhu), GetDamIndexE (vrací indexy krátkodob. obchodu za zadané období od do), GetDamPriceE (vrací objem zobchodované energie a cenu po hodinách z denního trhu, parametry od do, inEur), GetDamAllE (vrací výsledky zúčtování společného denního trhu ČR-SR, parametry od do, inEur), GetImPriceE (vrací ceny a objem za vnitrodenní obchody s elektřinou, parametry od do), GetImPriceG (vrací ceny a objemy vnitrodenního tru s plynem, parametry od od) ad. Služby končící na G (Gas) se týkají plynu a služby končící na E se týkají elektřiny.[72]

Kapitola 2
Analýza a návrh prototypu

2.1 Sběr požadavků

Pro specifikaci požadavků na řídicí aplikaci použijeme FURPS+ model. Dále provedeme SWOT analýzu zaměřenou na volbu vývoje programu vlastní silou oproti koupi od dodavatele.

2.1.1 FURPS+ analýza

Na základě sběru požadavků od zadavatele a jejich analýzy jsou identifikovány následující požadavky:

Funkčnost Systém má průběžně stahovat aktuální data spotových cen, na jejichž základě poskytovat informace vybraným systémům budovy. V bodech:

- Z veřejného rozhraní OTE stahovat data o denním trhu (SOAP rozhranní).
- Poskytovat stavovou veličinu pro ostatní řídicí systémy v podobě výhodnosti energetické energie (např. velmi nevýhodná, nevýhodná, neutrální, výhodná, velmi výhodná). Oproti tomu může jít veličina *potřeby vytápět* nebo jinak spotřebovávat elektřinu. Předpokládá se, že vytápění a chlazení bude mít největší podíl na spotřebě v budově, tedy je přípustné výsadně se na něj zaměřit.
- Rozhraní pro stavovou veličinu ceny elektřiny.
- Aplikace koncipována tak, aby umožňovala rozšíření o další rozhraní v budoucnu.

Použitelnost Prototyp může disponovat webovým či jiným grafickým rozhraním. Dle výběru technologií a prostředků pro aplikaci může jednoduché

webové rozhraní vyplynout jako nejvhodnější z jiných alternativ. Lze volit z konzolového (CLI) rozhraní, konfiguračních (.ini atp.) souborů, REST rozhraní ad.

Vystavení stavové veličiny a dalších dat aplikace (data z denního trhu atp.) pro strojové zpracování. Pro KNX prvky může být vystaveno přímo KNX rozhraní (TCP tunelování přes router nebo KNXNet multicast). REST se předběžně jeví jako vhodný pro další obecné využití.

Spolehlivost Vzhledem k tomu, že se bude jednat o prototyp, neklademe na tento atribut (zatím) velkou váhu. Také nebude možnost prototyp během psaní práce řádně ozkoušet z časových důvodů. Počítá se tedy s určitou mírou nedokonalosti a s laděním běhu programu již za provozu. Tyto nedostatky si můžeme dovolit, jelikož nepřisuzujeme programu kritickou důležitost pro řádný chod budovy. Řešení si klade za cíl poskytnout funkce navíc k řídicím systémům budovy od dodavatelů s výhledem na energetické úspory a výpadek v programu by neměl zásadně ohrozit komfort či provoz budovy.

Výkon Jedná se o interní řídicí systém a nepředpokládá se, že by na moderním hardware – tedy typově na běžném pracovním serveru s dostupnou operační pamětí v řádech gigabyte a procesoru s taktem 3 GHz a výše – při vhodné volbě softwarové technologie mělo docházet k potížím s výkonem. Není třeba řešit škálování.

Podporovatelnost Program má běžet v prostorách budovy na hardwarovém serveru, který má potřebná rozhraní k dispozici. Vzhledem k řídící povaze programu a navázanosti na fyzická rozhraní nepřipadají hostovaná či cloudová řešení v úvahu. Lokalizace buď v češtině nebo v angličtině (pro prototyp stačí napevno).

Očekává se běh na operačním systému Linux a doinstalování potřebných knihoven. Při implementaci je možno využít funkce specifické pro platformu. Podpora pro Windows a další systémy není žádoucí. Vzhledem k potřebným nízkoúrovňovým rozhraním není žádoucí program sestavovat do kontejnerů nebo virtuálních strojů.

2.1.2 SWOT analýza

Následující SWOT analýza nahlíží na možnost dalšího vývoje řídicí aplikace vlastní silou zadavatele, vůči alternativě pořízení aplikace či celého systému od dodavatele.

Silné stánky Zadavatel je zavedená softwarová firma, je možné si dodatečné části aplikace dopsat vlastní silou. V bodech:

- zkušenosti zadavatele v oblasti vývoje informačního software

- zkušenosti zadavatele v oblasti vývoje uživatelských rozhraní
- otevřenost zadavatele věnovat se činnosti mimo jeho specializaci
- serverové zázemí pro běh software
- aplikace není kritická pro řádné fungování společnosti
- společnost zadavatele je dlouhodobě zavedená

Slabé stránky Omezená znalost vývoje řídicích systému, jelikož se zadavatel specializuje na informační systémy. V dalších bodech:

- momentální plošná neobeznámenost vývoje řídicích aplikací
- vývoj informačních systémů podléhá odlišným zákonitostem než u řídicích systémů, včetně volby programovacích jazyků a návrhových postupů

Příležitosti Integrace řídicí aplikace mezi ostatními systémy, byť třeba od dodavatele. Do budoucna možné aplikace rozšiřovat, přidávat funkčnosti, případně uživatelské rozhraní. Zdrojový kód zůstane k dispozici a ve správě zadavatele. V bodech:

- zhotovení vlastního řídicího systému na míru s minimálními dodavatelskými náklady
- zdrojový kód v kompletní správě nevázaný na dodavatelské licence
- možnost doplňovat další funkčnosti jak bude potřeba
- zabývání se tvorbou řídicích systémů budov v budoucnu, nový podnikatelský záměr
- doškolit se v oblasti vývoje řídicích aplikací nebo najmout nové zaměstnance, další předmět podnikání
- zvyšující se poptávka po systémech řízení budov v blízké budoucnosti
- výběr takové technologie (knihovny, frameworku), která dobře odstíní úskalí související s nízkoúrovňovým vývojem
- propojení aplikace s fotovoltaickou elektrárnou, možnost další optimalizace s bateriovým úložištěm
- propojení aplikace se systémem na řízení venkovních žaluzií, se systémy osvětlení, chodu vzduchotechniky, bateriového úložiště a jeho nabíjením a vybíjením
- dlouhodobý sběr dat z jednotlivých tech. komponent pro budoucí analýzy (průtoky, koncentrace CO2, vlhkost, teplota, pohyb aj.)

Hrozby V bodech:

- nutnost uvolnění kapacit zadavatele, které by se věnovaly vývoji řídicí aplikace
- riziko lock-inu ve vlastním produktu, může se stát, že v budoucnu nebude, kdo by aplikaci dál vyvíjel nebo udržoval, pokud firma bude směřovat vývojovou činnost jiným směrem
- riziko volby nevhodné technologické základny (včetně programovacích prostředků, knihoven a frameworků) pro vývoj aplikace, zapříčiněné zejména malou zkušeností v oblasti
- riziko výběru funkční, ale do budoucna neudržované knihovny nebo další závislosti (technologie mimo hlavní proud)
- riziko výběru knihovny, jejíž licence zabraňuje komerční využití nebo požaduje zveřejnění zdrojového kódu (např. GPLv2) a to i v marginálním užití či pouhé interakci přes rozhraní
- riziko volby knihovny sice vhodně odstiňující nízkoúrovňové implementační detaily, či podporující moderní programovací styly, která není zcela korektně implementovaná (z chybami), které vývojový tým nebude schopen odhalit nebo opravit (spolehnutí na závislost, které se nerozumí do detailu, neboť to je její účel)
- konkurenční vývoj cenově dostupné nebo opensourcové technologie, která obsáhne podobné funkčnosti a stane se tak výhodnější si jí pořídit než udržovat vlastní

Vzhledem k silným stránkám můžeme mitigovat některé hrozby. Špatný technologický výběr nebo kroky vedle ve výběru vývojové platformy společnost ustojí, nejedná se o její hlavní činnost, ani kritickou aplikaci pro její fungování. Navíc je společnost zadavatele otevřená k uvolnění zdrojů pro vývoj v této doméně a získání odborných informací.

Licenční rizika přetrvávají a při případném zájmu aplikaci komercializovat bude nutné podrobit zkoumání použitých knihoven a případně navrhnout alternativy.

Konkurenční vývoj volně dostupné technologie může společnosti naopak benefitovat jako zdroj inspirace pro vývoj. Může se stát, že na trhu bude technologie s větší funkcionalitou, nicméně to neznehodnocuje přínosy řešení vytvářeného přímo na míru vlastních potřeb.

2.2 Kontext

Společnosti a osoby, které budují nové výstavby, jsou dnes postaveny před otázky řízení a automatizace budov (HVAC, MaR, BACS, EZS, PBZ atd.). Spoléhání na externí dodavatele systému může nést svá rizika.*

Dejme tomu, že jsme v situaci firmy nebo osoby, která chce stavět novou budovu. I bez různých *inteligentních* systémů se jedná o velký úkol. Nemusíme mít čas nebo dostatečnou vůli, abychom se věnovali návrhu budovy a procesu organizace její výstavby. Není nepravděpodobné, že podlehneme tlaku technologie a budeme stranit řešením, která jsou inteligentní a moderní oproti zastaralým a překonaným. To, jak se skutečně v oblasti orientujeme a do jaké míry jsme schopni identifikovat naše potřeby, se promítne do schopnosti tato řešení od sebe rozlišit.

Čím méně alternativ budeme znát, tím spíše dáme na názor vnímaného odborníka nebo podlehneme trendu. V souvislosti s technologiemi jako IoT je tento efekt umocněn, a místy může přinést i bezpečnostní rizika.**

Čistě dodavatelská řešení mohou být omezující, pokud máme zájem aktivně se podílet na řízení budovy. V tomto případě je namístě dohodnout se s dodavatelem, aby dodal takové řešení, které umožňuje nějaký typ vnějšího programového řízení standardizovaným protokolem.

O inteligentní systémy v budovách je zájem, zejména pro jejich schopnosti zvyšovat komfort a snižovat náklady spojené s provozem. Existence denního trhu a spotových cen společně s bateriovými a jinými úložišťmi vytváří unikátní příležitosti optimalizace spotřeby, a tak i snižování nákladů za energie.

2.3 Vize řešení

Vizi funkcionalit a schopností aplikace a budoucího stavu po jejím zavedení můžeme koncipovat dvěma způsoby. Buď se nakloníme k popisu ideálního cíle, tedy komplexního systému řízení, automatizace a monitoringu budovy, anebo se omezíme v rámci možností prototypu. Vzhledem k tomu, že není reálné v blízké době komplexní systém sestavit, začneme postupně od jednodušších funkcionalit prototypu. Nastíněna jsou možná rozšíření o další funkcionality. V obecnějším slova smyslu se o řídicí aplikaci hovoří také jako o *systému*.

Denní trh s elektřinou se jeví jako vhodná volba pro optimalizaci, jelikož jako jeden z mála trhů umožňuje přesné rozplánování na následující den. Navíc do činnosti OTE spadá i distribuce dat (v rámci transparentnosti) z těchto trhů

*Dodavatel vždy nemusí být stávajícím expertem v oboru. Nebo mohl být, ale rutinní aplikace jeho řešení zapříčinila snížení jeho relevance. Držet krok s trhem je schopnost pouze měnící se skupiny dodavatelů, kteří svá řešení jsou schopni neustále vyvíjet a přizpůsobovat. Jakmile toto přestanou dělat, na volném trhu jsou odsouzeni ke ztrátě jejich postavení.[73]

**Je na místě, abychom předtím než investujeme své úspory za např. inteligentní systémy se zamysleli nad tím, zda to opravdu povede k nejefektivnějšímu uspokojení našich potřeb. To koneckonců platí obecně, a také dále zůstává spíše teorií.

a tedy je možné tyto data veřejné a dostupně získávat.

OTE poskytuje veřejné SOAP rozhraní pro tržní data, které lze číst z jakékoli klientské aplikace. Vhodnou knihovnou toto naimplementujeme poměrně snadno.

Základní funkcionalita (veličina výhodnosti ceny): Na základě dat z denního trhu provede systém výpočet k zjištění výhodnosti ceny v dané hodině (či do budoucna jiném časovém intervalu). Spočítá průměrnou cenu (mean) za daný den a dle ní hodnotí výhodnost jednotlivých hodinových cen. Ke každé tak spočítá jednu hodnotu. Pokud bude v danou hodinu cena vyšší než průměr, bude hodnotu dané hodiny penalizovat a naopak, pokud bude cena nižší. Výstupem je pro každý den pole 24 hodnot reprezentujících cenovou výhodnost pro danou hodinu. Toto mohou být například 1bytové proměnné nabývající pěti hodnot -2, -1, 0, 1, 2 ve smyslu velmi nízká cena, nízká cena, neutrální, vysoká cena, velmi vysoká cena. Alternativně to mohou být procentuální hodnoty vychýlení oproti průměru.

1. rozšíření (REST): Pro využití spočtených dat tyto hodnoty systém zpřístupní přes RESTové rozhraní pro ostatní systémy budovy. To také umožní paralelní vývoj dalších rozšíření, které nebudou muset být součástí jádra. Také pro rozprostření funkcionalit do několika programových celků, což může zvýšit testovatelnost.

2. rozšíření (systémová rozhraní): Systém podporuje rozhraní jako MODBUS a KNX pro komunikaci se systémy budovy.

3. rozšíření (alternativní veličina potřeby spotřebovávat a vnitřní faktory): Společně s veličinou výhodnosti ceny se zavede kontraveličina *potřeby spotřebovávat energii*. Postavena do protipólu s výhodností ceny poskytuje řešení pro situace, kdy je cena vysoká, ale je třeba vytápět nebo silně chladit. Bez zakomponování této veličiny, nebo obdobného systému, v takových případech hrozí významné snížení komfortu, pokud budou systémy vytápění a chlazení pozastaveny vlivem veličiny výhodnosti ceny. To souvisí s tím, že systém dokáže sbírat data ze senzorů budovy, jako je teplota v místnosti, vlhkost a CO2 a zohlednit je ve výpočtu veličiny. Výsledná veličina vzniká rozdílem (příp. dalšími úpravami) veličiny výhodnosti ceny a potřeby spotřebovávat energii a lépe vystihuje situaci (může být propojeno s plánem cílových teplot). Toto dává základ sofistikovanějším výpočtům, které zahrnují i fyzikální vztahy.

4. rozšíření (uživatelský přístup): Je dán základ uživatelskému přístupu k systému v podobě webové aplikace. V budoucnu umožní také přístup k dalším systémům, jako např. řízení žaluzií, osvětlení ad.

5. rozšíření (plán vytápění na následující dny): Systém umožňuje uživatelům definovat cílové teploty pro např. 15minutové intervaly na následující dny. Pokusí se o dodržení těchto specifikovaných hodnot v rámci uživatelem definovaných rozmezí (např. $\pm$ 1 stupeň), ve kterých bude spotřebu optimalizovat. S tím souvisí znalost akumulačních a dalších schopností místností v budově a způsobů regulace.

Ve spojením s denním trhem, předpovědí počasí a plánem cílových teplot tak systém může např. předchladit budovu.

6. rozšíření (bateriová úložiště): Systém interaguje s prvky bateriového úložiště v budově (a typicky fotovoltaickou elektrárnou). Pokud cena bude dostatečně nízká a pokryje cenu uložení, systém uloží energii do baterií. V souvislosti s plánem teplot a cenou elektřiny na následující den (dále třeba předpověď slunečního svitu aj.) lze také rozhodnout o spotřebování/úspoře energie tento den. Část energie systém uskladní na další den, pokud bude cena vysoká. Pokud nízká, systém se pokusí zpracovat uloženou energii (např. dílčím předtopením / předchlazením budovy v noci atp.).

7. rozšíření: (objektivní funkce / AI) Systém používá rigorozní objektivní/cenovou funkci pro výpočet veličiny. Ta zohledňuje jak vnitřní faktory, tak vnější faktory jako je např. sluneční svit, rychlost větru, data z vnitrodenního trhu ad. Významnost jednotlivých faktorů lze porovnat regresní nebo jinou analýzou na historických datech. Pro stanovení výstupu lze zakomponovat učící se algoritmy penalizující vstupní parametry nebo jiné AI.

Výstupem je jedna hodnota vyjadřující *cenu* energie. Když vzroste nad stanovenou hladinu, systém např. přes MODBUS uvede tepelná čerpadla do úsporného módu.

2.4 MoSCoW klasifikace

V souvislosti s vizí řešení jsou identifikovány prvky implementace rozdělené do čtyřech kategorií dle jejich priority (pro prvotní implementaci prototypu).

Must have (nutné)

- implementace SOAP rozhraní pro získávání dat o cenách z portálu OTE
- výpočet veličiny výhodnosti ceny pro daný den

Should have (měly by být)

- REST rozhraní pro zpřístupnění veličiny

Could have (možná budou)

- prototyp webové aplikace (frontend) přistupující na REST rozhraní
- vývojová podpora pro MODBUS nebo KNX (v rozsahu zanesení knihovny do projektu)
- implementace každodenního stahování

Won't have (nebudou)

- analýza rizik časově závislých chyb (přístupy z vícero rozhraní najednou ke sdílené proměnné)

- výpočet veličiny potřeby spotřebovávat
- specifikace cílových teplot a rozmezí optimalizace
- a ostatní navržená rozšíření

Vidíme, že většina navržených funkcionalit a rozšíření se zařadila do kategorie *Won't have*. Tyto funkcionality jsou typicky velmi komplexní. Pro jejich implementaci by bylo třeba daleko podrobnější specifikace. Také nemohou být implementovány z důvodů vysoké pracnosti, nejspíše vyžadující vývojový tým. Tyto funkcionality lze zařadit do budoucího rozvoje současného řešení.

Implementace SOAP rozhraní bude možná skrze nástroj gSOAP (pro jeho schopnosti automatického generování hlavičkových souborů z WSDL a C++ Proxy třídám). Pro REST rozhraní knihovna Pistache (zejména díky jednoduchosti prvotního vývoje a také implementace v C++17). Pro KNX komunikaci je možné do projektu zařadit knihovnu libknxnet.

2.5 Volba vývojové platformy

Pro vývoj jádra aplikace je zvolen jazyk **C++**. Zváženy byly alternativy jako Rust a Java. Rust by napomohl v přístupu k rizikům práce s pamětí, nicméně za cenu vynucení místy kompletně odlišného přístupu. C++ pořád zůstává jedním z nejtvárnějších jazyků vůbec, což samozřejmě přichází se svými riziky.

V kombinaci s C++ je zvoleno vývojové prostředí **Qt Creator.** Oproti nejbližší alternativě CLion je, alespoň ze zkušenosti, daleko šetrnější ke zdrojům PC. To za cenu mírných ztrát v kontextové analýze a refaktoringu. Qt Creator v sobě integruje analyzátory jako clangd, cland-tidy aj. podobně jako CLion (CLion místy dokáže lépe interpretovat jejich výstupy a to v přívětivějším stylu).

Jako build systém je vybrán **CMake**, který Qt Creator přirozeně podporuje. Alternativou byl qmake, nicméně Qt ekosystém (byť v projektu není, vyjma Qt Creatoru, přímo využit) od qmake (od Qt verze 6) ustupuje ve prospěch CMake. CMake se tedy jeví jako do budoucna udržitelnější volbou.

Pro frontend k sytému byl vybrán framework **Svelte** společně s nástrojem, metaframeworkem SvelteKit. Na základě rešerše vyšel jako nejvhodnější pro účely prototypu. Zejména pro nízké technické zatížení v rámci prvotní fáze vývoje. Dále pro jeho výhody rychlosti a v základních funkcionalitách jednodušší syntaxe.

Jako verzovací systém lze využít **Git** pro všechny moduly aplikace. SvelteKit generuje předdefinovaný .gitignore v rámci generace adresářové struktury projektu (npm create svelte@latest app).

Projekt si neklade za cíl nezávislost vůči platformě operačního systému. Domovským OS je stanoven Linux. Vývojová stanice používá distribuci Fedora 40.

V souvislosti na to je potřeba na vývojové stanici mít k dispozici všechny závislosti a balíčky spojené s výše uvedenými technologiemi.

Pro přehled vybraných technologií viz tab. 2.1.

Část	Typ	Vybraný nástroj
Jádro	Programovací jazyk	C++
	Vývojové prostředí	Qt Creator
	Build systém	CMake (Ninja, Makefile)
Web app.	Framework	Svelte
	Metaframework	SvelteKit
	Vývojové prostředí	Webstorm nebo VS Code
	Build systém	pnpm, vite

Tabulka 2.1 Souhrn technologického rámce prototypu, zdroj: autor

2.6 Návrh prototypu

Jak vidíme z obrázku 2.1, aplikace je rozdělena do komponenty CPP CORE, plnící roli jádra systému. Jádro vystavuje REST rozhraní s daty, na které je také napojena webová aplikace (REST FRONTEND). Přerušované čáry značí komunikační rozhraní pro další systémy a prvky.

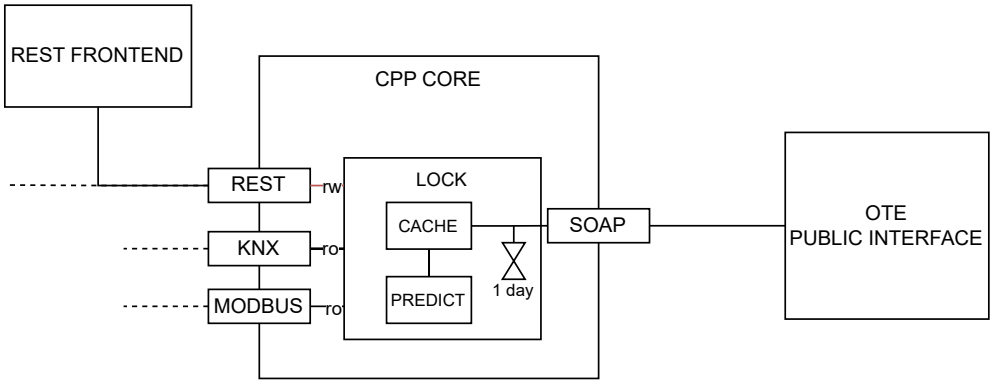
Jádro obsahuje prvek CACHE, do kterého se uloží stažená data, aby je nebylo nutné často opakovaně stahovat. Z CACHE jsou data (o denním trhu atp.) pro webovou aplikaci. Data se přes SOAP rozhraní z OTE stáhnout do CACHE jednou denně automaticky (po čase zveřejnění denního trhu na následující den) nebo na vyžádání webovou aplikací. Je zde potenciální hrozba časově závislé chyby, zejména díky vícevláknové povaze REST serveru (možno ze začátku omezit na 1 vlákno nebo dále implementovat zámek pro přístup ke sdílenému zdroji).

REST, KNX a MODBUS značí podporovaná rozhraní. Přes REST rozhraní lze činnost jádra ovládat a vyvolávat funkce na vyžádání, proto je na schématu označeno `rw` (read write), jelikož jako jediné v tomto návrhu dokáže změnit data v CACHE. KNX a ostatní rozhraní přistupují pouze v režimu read. Toto omezení bude do budoucna nutné zrušit, pokud budeme chtít vyčítat například senzorická data.

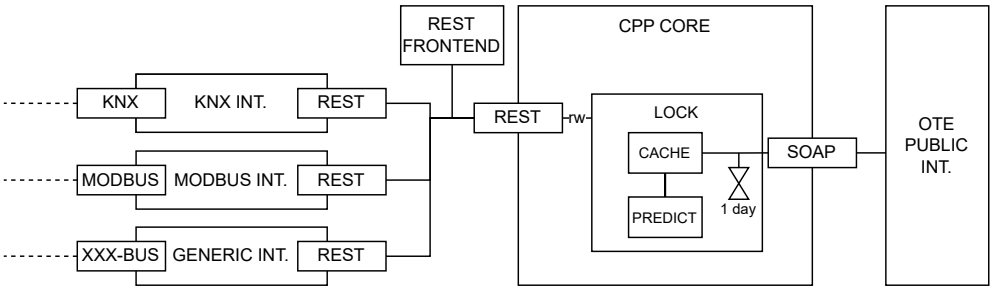
Druhé schéma se liší v oddělení systémových rozhraní do dalších modulů. S jádrem by se komunikovalo pouze skrze REST rozhraní na vnitřní straně. Takové řešení může být do budoucna rozšiřitelnější a snadněji udržovatelné, nicméně více pracné.

Prvek	Knihovna
SOAP	gSOAP
REST	Pistache
KNX	libknx
JSON	nhloman::json

Tabulka 2.2 Souhrn použitých knihoven návrhu prototypu, zdroj: autor



Obrázek 2.1 Schéma prototypu systému v. 1, zdroj: autor



Obrázek 2.2 Schéma prototypu systému v. 2 (REST), zdroj: autor

Kapitola 3
Implementace prototypu

V rámci technologií doporučených výše uvedeným návrhem jsou reálně v prototypu využity knihovny zobrazené v tab. 3.1. Je implementováno schéma podobné tomu na obr. 2.1. Výstupní veličina byla rozšířena z pěti na sedm hodnot, viz. kód 3.3. A metoda řešící výpočet cenové veličiny je zobrazena v kódu 3.5 a v související funkci v kódu 3.6, která zařazuje možné vstupní hodnoty mezi sedm hodnot.* Vracená JSON struktura s veličinou je viditelná z výpisu kódu 3.7. Příkaz pro vygenerování kódu z WSDL je uveden ve výpisu 3.1. Funkcionalita stažení dat z SOAP rozhraní OTE je ukázaná ve výpisu 3.2.

Popis	Použitá knihovna
Převod WSDL speficikace na C++ kód	gSOAP v. 2.8.132 (wsdl2h, soapcpp2)
Implementace REST rozhraní v C++	Pistache v. 0.2.9
Implementace (de)serializace JSON pro C++	JSON for Modern C++ v. 3.11.3
Implementace KNX rozhraní přes multicast (KNXNet/IP) pro C++	libknxnet v. 1
Implementace INI konfigurace pro C++	mINI v. 0.9.15

Tabulka 3.1 Knihovny použité u prototypu verze 1, zdroj: autor

*Mean multiplier (o jaký násobek je aktuální hodnota dražší než průměr z daného dne), se kterým se počítalo v rámci základní funkcionality, byl ještě doplněn o tzv. Z–score (*standardní skóre*, což je rozdíl aktuální ceny s průměrem podělený směrodatnou odchylkou). V případech, kdy je průměr denní ceny blízko nuly, by u prvního přístupu vycházely obrovské násobky. V prototypu jsou implementovány obě dvě veličiny.

```
wsdl2h -c++17 ote_public.wsdl
soapcpp2 -c++17 -j -C ote_public.h
```

Výpis kódu 3.1 Vygenerování C++ kódu z WSDL veřejného rozhraní OTE, zdroj: dokumentace gSOAP, autor

```
void RestServer::downloadDayAheadHours(std::string startDate,
                                       std::string endDate)
{
    PublicDataServiceSoapProxy ote;

    _ns1__GetDamPriceE req;
    _ns1__GetDamPriceEResponse resp;

    req.StartDate = startDate;
    req.EndDate = endDate;
    int ret = ote.GetDamPriceE(&req, resp);
    if( ret == SOAP_OK )
    {
        for(auto & item : resp.Result.Item)
        {
            double price;
            if(item.Price != nullptr)
            {
                price = std::stod(*item.Price); //knowingly throws
            }
            else
            {
                std::cerr << "Price was empty for: "
                          << item.Date
                          << " " << item.Hour << "\n";
            }
            m_cached_elements.emplace_back(item.Date,
                                           item.Hour, price);
        }
    }
    else
    {
        ote.soap_stream_fault(std::cerr);
    }
}
```

Výpis kódu 3.2 C++ funkcionalita stahování cen z denního trhu OTE, zdroj: autor

Prototyp poskytuje konfigurační soubor pro parametry REST serveru (IP, port) v `config.ini`. Ten je při startu zpracován za pomocí knihovny mINI v metodě loadConfig().

Blokující čekání REST serveru pak započne v metodě restServe(), tím se zahájí obsluha REST rozhraní.

```
enum class PriceClass : int8_t
{
    extreme_low = -3,
    very_low = -2,
    low = -1,
    neutral = 0,
    high = 1,
    very_high = 2,
    extreme_high = 3
};
```

Výpis kódu 3.3 Výčet veličiny cenové výhodnosti, zdroj: autor

```
struct ComputedVariable
{
    std::string date;
    int hour;
    double price;
    double mean_multiplier;
    double z_score;
    PriceClass price_class; // -3,-2,-1,0...
};
```

Výpis kódu 3.4 Struktura vypočtené hodnoty, zdroj: autor

Implementované funkcionality prototypu verze 1 jsou popsány tab. 3.2.

3.1 Možná rozšíření

V bodech shrneme možná rozšíření implementovaného prototypu.

- programové ovládání venkovních žaluzií (přes sytém KNX)
- programové ovládání osvětlení budovy (KNX a DALI)
- vyčítání dat o spotřebách energie jednotlivých technologických celků (teplota, vlhkost, CO2, průtoky VZT atd.)

```
void RestServer::computeVariables() //overwrites last computation
{
    m_predict.computed_variables.clear();
    long int sum = 0;
    for (const auto & e : m_cached_elements) {
        sum += e.price;
    }
    double mean = m_cached_elements.empty() ? 0
                : sum / m_cached_elements.size();

    double var_sum = 0;
    for (const auto &e : m_cached_elements) {
        var_sum += (e.price - mean) * (e.price - mean);
    }
    double var = m_cached_elements.empty() ? 0
                : var_sum / m_cached_elements.size();
    double sd = std::sqrt(var);

    for (const auto & e : m_cached_elements) {
        ComputedVariable variable;
        variable.date = e.date;
        variable.hour = e.hour;
        variable.price = e.price;
        variable.mean_multiplier = (mean == 0) ? 0 : e.price / mean;
        variable.z_score = (sd == 0) ? 0 : (e.price - mean) / sd;
        variable.price_class = computePriceClass(variable.z_score);

        m_predict.computed_variables.push_back(variable);
    }
    m_predict.mean = mean;
    m_predict.sd = sd;
}
```

Výpis kódu 3.5 Metoda zpracovávající data z denního trhu, zdroj: autor

```
PriceClass computePriceClass(double z_score)
{
    if (z_score > 1.7 || z_score < -1.7) {
        return (z_score >= 0) ?
                  PriceClass::extreme_high
                  : PriceClass::extreme_low;
    }
    if (z_score > 0.85 || z_score < -0.85) {
        return (z_score >= 0) ?
                 PriceClass::very_high
                 : PriceClass::very_low;
    }
    if (z_score > 0.35 || z_score < -0.35) {
        return (z_score >= 0) ?
                 PriceClass::high
                 : PriceClass::low;
    }
    return PriceClass::neutral;
}
```

Výpis kódu 3.6 Funkce klasifikace standardního skóre (Z–score), zdroj: autor

```
{
  "stats": {
    "mean": 831,
    "sd": 735.2858860632034
  },
  "variables": [
    {
      "date": "2024-04-01",
      "hour": 1,
      "mean_multiplier": 1.412635379061372,
      "price": 1173.9,
      "price_class": 1,
      "z_score": 0.46634922075809465
    },
    {
      "date": "2024-04-01",
      "hour": 2,
      "mean_multiplier": 1.2972202166064981,
      "price": 1077.99,
      "price_class": 0,
      "z_score": 0.3359101604988095
    }]
}
```

Výpis kódu 3.7 JSON výstup z REST rozhraní, zdroj: autor, zkráceno

Popis funkcionality C++ části	Stav implementace
CMake build systém a specifikace CMakeLists.txt	Vytvořen základ extensibilního CMakeLists.txt. Využití tzv. modern CMake přístupu. Vybrané knihovny v podadresářích mají vlastní CMakeLists.txt, který se do hlavního přidává příkazem add_subdirectory. CMakeLists.txt společně s Qt Creatorem funguje bezchybně. Využívá výhod paralelní kompilace.
Parsing config.ini souboru	Konfigurační soubor je zpracován v metodě loadConfig() s kontrolou vstupů.
REST rozhraní	REST rozhraní je implementováno pomocí knihovny Pistache. Jsou definovány cesty pro získání dat z CACHE, získání dat PREDICT, stažení dat denního trhu OTE na vyžádání, kalkulace veličiny (základní metodou průměru přes tabulku intervalů) na vyžádání a status.
KNX rozhraní	KNX funkcionalita je zařazena do projektu (libknxnet) a zohledněna v CMakeLists.txt. Přímo se zatím nevyužívá, je připravena na vývoj.
MODBUS rohzraní	Není implementováno, v budoucnu lze uvažovat o libmodbuspp.
JSON serializace	Implementována pomocí knihovny nlohmann/json.

Tabulka 3.2 Implementované funkcionality C++ části prototypu verze 1, zdroj: autor

Popis funkcionality webové části	Stav implementace
Build systém a adresářová struktura	Je využita kostra projektu poskytovaná metaframeworkem SvelteKit. Jako build systém se používá pnmp. Pro stažení balíčků pro spuštění stačí spustit pnpm install a pro spuštění webové aplikace ve vývojářském módu pnpm run dev.
Napojení na REST rozhraní C++ části	Jsou definovány vazby na REST rozhraní, konkrétně získání dat o denním trhu uložených v CACHE jádra, jejich stažení, získání seznamu veličin výhodnosti ceny a získání statusu C++ serveru. Tyto akce lze spouštět pomocí tlačítek.
Graf dat denního trhu	Není implementován, možno uvažovat o Charts.js
UI knihovna	Není definována, prototyp je bez stylů. Možno uvažovat o Bits UI.

Tabulka 3.3 Implementované funkcionality webové části prototypu verze 1, zdroj: autor

- logování a správa přístupů k technologickým celkům (sytémům budovy), mj. pro dodržení bezpečnostní politiky firmy
- specifikace vhodných datových struktur pro ukládání hodnot do dat. úložiště (perzistence dat)

Další rozšíření nad rámec funkcionalit řídicího systému.

- na základě shromážděných dat o technologiích budovy vyhodnotit rentabilitu jednotlivých tech. celků
- ověřit deklarované účinnosti jednotlivých komponent systémů (tep. čerpadel, VZT, ZZT aj.)

Závěr

Tato bakalářská práce se zaměřila na neustále vyvíjející se oblast energetiky a informačních technologií. Cílem bylo vytvoření prototypu aplikace, která umožňuje na základě dat z veřejného rozhraní operátora trhu s energiemi, konkrétně z denního trhu, poskytovat data o výhodnosti ceny elektrické energie. Tyto výstupy předkládá skrze REST rozhraní, které je agnostické vůči koncovým zařízením.

Práce disponuje základním přehledem technologií používaných v moderních budovách. V částech se věnuje tématu inteligentních budov a inteligentních domácností. Dále se v krátkém rozsahu zmiňuje také o IoT, a s tím souvisejícího KNX IoT, jehož koncept je velmi aktuální.

Věnuje se také popisu použitých metod klasifikace atributů software, zmiňuje FURPS+ analýzu a známou SWOT analýzu, u které naráží na podrobné zdroje o její historii (a o jejím předchůdci, SOFT analýze).

Dále byla provedena nezanedbatelná rešeršní činnost v oblasti energetické terminologie, dalšího rozměru probírané tematiky. Za zmínku stojí kapitola věnovaná legislativnímu rámci, jejíž snahou bylo předložit vybrané aspekty energetických zákonu pro hlubší porozumění zkoumané domény.

V souvislosti se spotovým – průběhovým – měřením zmiňuje připravovaný přechod na 15minutové měřicí intervaly. Jedná se o nemalou změnu, která se bude týkat mj. i funkcionality prototypu, jelikož ten aktuálně zpracovává hodinová data.

Implementovaný prototyp sestává ze sady technologií, které se práce pokouší přiblížit ve své rešeršní části. Za zmínku stojí nástroj gSOAP, který dokáže z WSDL (což je jazyk pro popis rozhraní webových služeb, který se mj. hojně využívá v aplikacích státní správy) automaticky vygenerovat kód pro C++ (včetně implementace tříd). Zmiňovaná platforma pro vývoj webových aplikací Svelte, ve které je implementována webová část prototypu, vychází v uvedeném porovnání lépe, nežli například široce rozšířená technologie React.

Součástí bakalářské práce jsou také výpisy z kódu C++ části prototypu, které představují jeho podstatné funkcionality, jako je např. výpočet stavové

veličiny, v budoucnu využitelný pro systémy MaR a vytápění a další.

V částech práce (zejm. 2.3 Vize řešení a 3.1) jsou také nastíněny možná rozšíření funkcionalit prototypu do budoucna, a také rozšíření nad rámec řídicího software. Zadavatel může na prototyp dále navázat. A pokud výstupy z práce osloví i další zájemce v oblasti řízení inteligentních budov, bude to jen potěšující.

Příloha A

JavaScript Framework Benchmark

Jedná se o `krausest/js-framework-benchmark`, konkrétně běh pro Chrome 124 na platformě OSX. Detailnější informace lze naleznout na: `https://krausest.github.io/js-framework-benchmark/2024/table_chrome_124.0.6367.91.html`.

A.1 Vysvětlivky

- Svelte 1 – svelte-v5.0.0-next.100
- Vue 1 – vue-v3.4.21
- Preact 1 – preact-classes-v10.19.3
- Angular 1 – angular-cf-v17.3.1
- Angluar 2 – angular-cf-signals-v17.3.1
- Vue 2 – vue-jsx-v3.4.21
- React 1 – react-classes-v18.2.0
- Preact 2 – preact-hooks-v10.19.3
- React 2 – react-hooks-v18.2.0

A.2 Výsledky

Název	Svelte	Vue 1	Preact 1	Angular 1
uncompressed size	14.6	55.9	16.9	133.6
compressed size	5.8	20.5	5.8	41.4
first paint	66.3	115.0	69.6	201.3
geometric mean	2.96	8.46	3.15	17.24

Tabulka A.1 *Transferred size (in kBs) and first paint*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Angular 2	Vue 2	React 1	Preact 2	React 2
uncomp. size	134.9	53.8	142.8	14.2	142.3
comp. size	41.9	19.6	40.3	5.5	40.1
first paint	198.4	104.5	219.8	65.0	229.5
geom. mean	17.28	7.97	17.99	2.86	18.2

Tabulka A.2 *Transferred size (in kBs) and first paint*, pokračování, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Svelte	Vue 1	Preact 1	Angular 1
ready memory	0.5	0.7	0.5	1.3
run memory	2.5	3.7	4.2	4.6
update every 10th row for 1k rows	2.6	3.7	4.2	4.8
creating/clearing 1k rows	0.7	1.1	0.7	2.0
run memory 10k	19.4	28.2	36.1	29.6
geometric mean	1.54	2.25	2.12	3.19

Tabulka A.3 *Memory allocation in MBs*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Ang. 2	Vue 2	React 1	Preact	React 2
ready memory	1.4	0.7	0.9	0.5	0.9
run memory	4.7	4.1	4.4	3.8	4.4
update every 10th row for 1k rows	4.8	4.2	5.0	3.8	5.0
creating/clearing 1k rows	2.1	1.1	1.8	0.8	1.7
run memory 10k	29.8	32.2	32.1	32.7	32.2
geometric mean	3.28	2.42	2.94	1.94	2.91

Tabulka A.4 *Memory allocation in MBs*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Svelte	Vue 1	Preact 1	Angular 1
create rows	39.7	46.7	48.9	51.2
replace all rows	44.8	52.5	56.7	62.8
partial update	18.4	22.9	21.5	19.5
select row	4.9	4.9	5.8	5.4
swap rows	21.7	22.5	165.5	22.9
remove row	16.6	20.3	18.0	17.7
create many rows	409.4	465.7	476.9	527.8
append rows to large tbl.	44.9	52.5	54.9	55.9
clear rows	15.5	20.1	20.9	37.0
weighted geometric mean	1.11	1.32	1.42	1.44

■ **Tabulka A.5** *Duration in milliseconds*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Ang. 2	Vue 2	React 1	Preact	React 2
create rows	52.4	48.1	49.6	48.6	49.6
replace all rows	62.7	54.4	55.2	57.3	58.9
partial update	19.8	31.2	25.7	34.7	24.3
select row	7.2	15.5	7.1	21.7	6.5
swap rows	22.7	32.4	176.6	39.3	177.7
remove row	19.0	22.5	19.4	25.2	20.2
create many rows	530.5	473.5	639.0	483.1	641.2
append rows to large tbl.	56.7	59.5	55.2	57.9	55.0
clear rows	36.7	20.3	19.2	17.9	30.6
weighted geometric mean	1.49	1.53	1.53	1.60	1.61

■ **Tabulka A.6** *Duration in milliseconds*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Svelte	Vue	React
create rows	40.8	47.1	49.7
replace all rows	15.2	20.3	20.3
partial update	18.2	21.5	24.1
select row	4.2	4.1	6.9
swap rows	13.3	13.1	15.5
remove row	31.6	40.8	42.1
create many rows	430.1	471.3	644.1
append rows to large tbl.	47.3	53.1	55.6
clear rows	15.1	18.7	17.1
weighted geometric mean	1.23	1.45	1.60

■ **Tabulka A.7** *Duration in milliseconds*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Svelte	Vue	React
ready memory	0.4	0.7	0.9
run memory	2.7	3.7	4.4
update every 10th row for 1k rows	2.7	3.7	4.9
creating/clearing 1k rows	1.2	1.1	1.8
run memory 10k	20.7	28.1	31.5
geometric mean	1.62	2.15	2.79

■ **Tabulka A.8** *Memory allocation in MBs*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Název	Svelte	Vue	React
uncompressed size	15.5	55.9	143.2
compressed size	6.0	20.5	4.2
first paint	52.2	118.8	241.6
geometric mean	2.00	6.07	13.17

■ **Tabulka A.9** *Transferred size (in kBs) and first paint*, zdroj: *js web frameworks benchmark* (krausest.github.io)

Příloha B

Postup pro sestavení Qt KNX ze zdrojových kódů

Na stránce `https://download.qt.io/` stáhneme potřebné Qt moduly a build systém. Zajímají nás konkrétně soubory:

- `qtknx-everywhere-opensource-src-5.15.12.tar.xz`,
  ten je v `single/`
  na `https://download.qt.io/official_releases/qt/5.15/5.15.12/`
- `qt-everywhere-opensource-src-5.15.12.tar.xz`,
  ten je v `submodules/`

Možné stáhnout .zip místo .tar.xz, pokud vyhovuje více. Potřebujeme to takto rozdělit, jelikož `qt-everywhere-opensource-src-5.15.12.tar.xz` neobsahuje KNX modul, ale obsahuje skripty pro sestavení. Samotný KNX modul je `qtknx-everywhere-opensource-src-5.15.12.tar.xz`. V průběhu času může dojít k aktualizaci a mohou být k dispozici novější zdrojové kódy.
Dále rozbalíme oba dva, z qt odstraníme nepotřebné moduly – složky začínající na qt (nechat qtbase, ostatní možno smazat – můžeme nechat například qtsensors, qtwebsockets aj.)
V rozbaleném `qt-everywhere-opensource-src-5.15.12.tar.xz` budeme mít i build systém. Tam nakopírujeme složku `qtknx-xxx` z `qtknx-everywhere-opensource-src-5.15.12.tar.xz` a přejmenujeme ji na `qtknx` – jako modul. Uvnitř `qtknx/` tedy bude `qtknx.pro`, `src/` atp.
V kořeni dáme `./configure`, vygeneruje se Makefile ad.
Poté `make -j<pocet vlaken>`, tedy např. `make -j8` u 8vláknového PC. Pomocí make se nám sestaví spustitelné binární soubory (to tehdy, když budeme mít dostupný kompilátor a požadované knihovny – jinak se bude třeba dle chybových hlášek a internetu dopátrát k funkční sestavě, takový

návod je zde bohužel mimo meze). Sestavený Qt pak můžeme pomocí `make install` zkopírovat do nějakého adresáře, výchozí je `/usr/local/Qt-xxx`. Takto si vytvoříme námi zkompilovaný Qt SDK, v Qt Creatoru pak přidáme další dev. kit (`/usr/local/Qt-xxx` dle adresáře, kam jsme dali sestavu) a dále již postupujeme stejně, jako bychom používali předkompilovaný commercial (sestavený podléhá GPLv3) balík stažený přes Qt Maintanance Tool.
Pozn. nejspíš by šlo stáhnout pouze build systám a pak si dostahovat jednotlivé moduly (submoduly) samostatně, místo stahování *celého* `qtknx-everywhere-opensource` a následného odstraňování nechtěných modulů.

Příloha C

Novostavba VERA, spol. s r.o.

C.1 Kontext a motivace

Softwarová firma VERA, spol. s r.o. ke dni 14. 5. 2024 staví administrativní budovu. Hrubá stavba je dokončena včetně zateplení vnější obálky a fasády. V současné době probíhají vnitřní práce a instalace technologií (vzduchotechnika, silnoproudé a slaboproudé rozvody, topení atd.)

C.2 Technologie budovy

Na základě jednání s firmou byly identifikovány následující klíčové soubory technologií.

- vrtné pole umístěné pod budovou jako zdroj energie pro vytápění a ochlazování (6 vrtů, hloubka 190 m)
- strojovna s tepelnými čerpadly (3 TČ ecoGEO zapojené v kaskádě)
- způsob vytápění a ochlazování sálavými podhledy (nízkoteplotní topení, vysokoteplotní ochlazování)
- centrální vzduchotechnická jednotka s výkonem 4800 m^3/h
- vzduchotechnická jednotka pro školicí místnost s výkonem 2000 m^3/h
- regulátory průtoků VZT potrubí (regulace na základě koncentrace CO2 v místnost)
- dvě serverovny, každá se ztrátovým výkonem 3 kW (ochlazování fancoilovými jednotkami a klimatizačními jednotkami)

- elektronický zapezpečovací systém budovy
- požárně bezpečnosntní systém budovy
- kamerový a identifikační systém
- řízení a ovládání venkovních žaluzií systémem KNX
- řízení a ovládání osvětlení systémem KNX a DALI
- měřicí a regulační systém (MaR)

Bibliografie

1. YADAV, Shivani; KISHAN, Bal. Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software. *International Journal of Emerging Trends in Engineering Research* [online]. 2020, roč. 8, č. 6, s. 2824–2840 [cit. 2024-05-12]. ISSN 2347-3983. Dostupné z: http://www.warse.org/IJETER/static/pdf/file/ijeter96862020.pdf.
2. SAINI, G. L.; PANWAR, Deepak. *Comparative Study of Open Source Software Quality Models*. Sv. 1 [online]. 2020. [cit. 2024-05-12]. Č. 1. Dostupné z: https://doi.org/10.51682/JISCOM.00101004.2020.
3. DYSON, Jonathan. *Conjoining FURPS+ and MoSCoW to Analyse and Prioritise* [online]. LinkedIn [cit. 2024-05-01]. Dostupné z: https://www.linkedin.com/pulse/conjoining-furps-moscow-analyse-prioritise-jonathan-dyson.
4. GRADY, Robert; CASWELL, Deborah. Managing From the Right Data. In: *Software Metrics: Establishing a Company-Wide Program*. Prentice Hall, 1987, s. 159. ISBN 978-0138218447.
5. GRADY, Robert. Project Management to Maximise Customer Satisfaction. In: *Practical software metrics for project management and Process Improvement*. Prentice Hall, 1992, s. 30–39. ISBN 978-0137203840.
6. VANĚK, Dušan. *Dimenze kvality FURPS+* [online]. Luděk Havlíček, slideplayer [cit. 2024-05-01]. Dostupné z: https://slideplayer.cz/slide/3752047/.
7. *ISO/IEC 25010* [online]. ISO25000.com [cit. 2024-05-10]. Dostupné z: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.
8. KEŘKOVSKÁ, M.; VYKYPĚL, O. *Strategické řízení: teorie pro praxi*. C.H. Beck, 2006. C.H. Beck pro praxi. ISBN 9788071794530. Dostupné také z: https://books.google.cz/books?id=sRbLlNtZDrgC.

9. *SWOT analýza* [online]. ManagementMania [cit. 2024-05-11]. Dostupné z: https://managementmania.com/cs/swot-analyza.
10. SARSBY, Alan. *Swot Analysis: A Guide to Swot for Business Studies Students*. Spectaris Ltd, 2016. ISBN 9780993250422.
11. PUYT, Richard W.; LIE, Finn Birger; WILDEROM, Celeste P.M. The origins of SWOT analysis. *Long Range Planning*. 2023, roč. 56, č. 3. ISSN 0024-6301. Dostupné z DOI: https://doi.org/10.1016/j.lrp.2023.102304.
12. WEICHBRODT, Nico. *libknxnet: KNXnet/IP library for Linux* [online]. GitHub [cit. 2024-05-11]. Dostupné z: https://github.com/envy/libknxnet.
13. STROUSTRUP, Bjarne; SUTTER, Herb. *C++ Core Guidelines: Don't Use a Union for Type Punning* [online]. GitHub [cit. 2024-05-11]. Dostupné z: https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c183-dont-use-a-union-for-type-punning.
14. P, Pieter. *Don't Use Unions or Pointer Casts for Type Punning* [online]. GitHub Pages [cit. 2024-05-11]. Dostupné z: https://tttapa.github.io/Pages/Programming/Cpp/Practices/type-punning.html.
15. *std::variant* [online]. cppreference.com [cit. 2024-05-11]. Dostupné z: https://en.cppreference.com/w/cpp/utility/variant.
16. *Qt KNX* [online]. The Qt Company [cit. 2024-05-11]. Dostupné z: https://doc.qt.io/qt-5/qtknx-index.html.
17. *QKnxNetIpTunnel Class* [online]. The Qt Company [cit. 2024-05-11]. Dostupné z: https://doc.qt.io/qt-5/qknxnetiptunnel.html.
18. *gSOAP Documentation – the gSOAP user guide* [online]. Genivia [cit. 2024-05-07]. Dostupné z: https://www.genivia.com/docs.html.
19. *Pistache Documentation* [online]. Pistache.io [cit. 2024-05-10]. Dostupné z: https://pistacheio.github.io/pistache/docs/.
20. *A list of open-source C++ libraries* [online]. cppreference.com [cit. 2024-05-12]. Dostupné z: https://en.cppreference.com/w/cpp/links/libs.
21. *About CMake* [online]. Kitware, Inc. [cit. 2024-05-11]. Dostupné z: https://cmake.org/about/.
22. *About npm* [online]. npmjs.com [cit. 2024-05-14]. Dostupné z: https://docs.npmjs.com/about-npm.
23. *An Introduction to the npm Package Manager* [online]. Node.js [cit. 2024-05-14]. Dostupné z: https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager/.
24. *Introducion: Motivation* [online]. pnpm [cit. 2024-05-14]. Dostupné z: https://pnpm.io/motivation.

25. NUSSBAUM, Joshua. *The Svelte compiler: How it works* [online]. DEV Community, 2023 [cit. 2024-05-11]. Dostupné z: https://dev.to/joshnuss/svelte-compiler-under-the-hood-4j20.

26. *Svelte Docs* [online]. Svelte.dev [cit. 2024-05-10]. Dostupné z: https://svelte.dev/docs.

27. TWARDOWSKA, Beata. *Why Svelte is the Next Big Thing in JavaScript Development* [online]. Naturaily, 2023 [cit. 2024-05-14]. Dostupné z: https://naturaily.com/blog/why-svelte-is-next-big-thing-javascript-development.

28. KRAUSE, Stefan. *JavaScript Framework Benchmark - Results for Chrome 124.0.6367.91* [online]. [cit. 2024-05-11]. Dostupné z: https://krausest.github.io/js-framework-benchmark/2024/table_chrome_124.0.6367.91.html.

29. JAMEEL, Sara Arghwan. *Comparison of Modern JavaScript Web Frameworks*. 2023. bcthesis. Tomas Bata University in Zlín, Faculty of Applied Informatics.

30. CROTOFF, Thibaut. *Front-end Frameworks Popularity (React, Vue, Angular, and Svelte)* [online]. GitHub Gist [cit. 2024-05-11]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190.

31. PREACT. *Preact* [online]. GitHub, 2015/ [cit. 2024-05-12]. Dostupné z: https://github.com/preactjs/preact.

32. *SvelteKit Docs* [online]. kit.svelte.dev [cit. 2024-05-11]. Dostupné z: https://kit.svelte.dev/docs.

33. *SvelteKit Docs*. Form Actions [online]. Svelte [cit. 2024-05-11]. Dostupné z: https://kit.svelte.dev/docs/form-actions.

34. *Progressive Enhancement* [online]. Svelte Learning Team, 2024 [cit. 2024-05-11]. Dostupné z: https://learn.svelte.dev/tutorial/progressive-enhancement. Online Tutorial.

35. *Most Popular Technologies: Integrated Development Environment* [online]. Stack Overflow [cit. 2024-05-11]. Dostupné z: https://survey.stackoverflow.co/2023/#section-most-popular-technologies-integrated-development-environment.

36. *VSCodium - Open Source Binaries of VSCode* [online]. VSCodium [cit. 2024-05-11]. Dostupné z: https://vscodium.com/.

37. *License - Visual Studio Code* [online]. Microsoft [cit. 2024-05-11]. Dostupné z: https://code.visualstudio.com/License.

38. PASERO, Benjamin. *VS Code — The Story and Technology Behind One of the World's Most Popular Desktop Apps for Developers*. Technologies Being Used [online]. Tower Blog, 2021-04 [cit. 2024-05-11]. Dostupné z: https://www.git-tower.com/blog/developing-for-the-desktop-vscode/.

39. *IBM Z Open Editor* [online]. Visual Studio Marketplace [cit. 2024-05-11]. Dostupné z: https://marketplace.visualstudio.com/items?itemName=IBM.zopeneditor.

40. *Neovim: Hyperextensible Vim-based Text Editor* [online]. Neovim team [cit. 2024-05-11]. Dostupné z: https://neovim.io/.

41. VARGIC, Radoslav; TRÚCHLY, Peter; PODHRADSKÝ, Pavol. Inteligetní domácnosti. In: *Inteligentní technologie* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2019, s. 8 [cit. 2024-04-16]. Dostupné z: https://techpedia.fel.cvut.cz/single/?objectId=153.

42. CALAOS. *Calaos server* [online]. GitHub [cit. 2024-05-11]. Dostupné z: https://github.com/calaos/calaos_base.

43. DOMOTICZ. *Domoticz: Open Source Home Automation System* [online]. GitHub [cit. 2024-05-11]. Dostupné z: https://github.com/domoticz/domoticz.

44. KRAMP, Thorsten; KRANENBURG, Rob van; LANGE, Sebastian. Introduction to the Internet of Things. In: *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*. Ed. BASSI, Alessandro; BAUER, Martin; FIEDLER, Martin; KRAMP, Thorsten; KRANENBURG, Rob van; LANGE, Sebastian; MEISSNER, Stefan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, s. 1–10. ISBN 978-3-642-40403-0. Dostupné z DOI: 10.1007/978-3-642-40403-0_1.

45. HÄNEL, André. *KNX IoT Downloads* [online]. KNX Association [cit. 2024-05-13]. Dostupné z: https://support.knx.org/hc/en-us/articles/10386532582930-Downloads.

46. *KNX Specification 3-10-5 KNX IoT Point API* [online]. KNX Association [cit. 2024-05-14]. Dostupné z: https://knxcvba.sharepoint.com/:b:/s/KNXPublic/EaHk2E6S10pHtgyQUJBdAdQBtWU-rBSlqHW5mwb5QyEhHw?e=ZQUziN.

47. KNX. *KNX-IOT STACK: KNX-IOT open source stack* [online]. GitHub [cit. 2024-05-14]. Dostupné z: https://github.com/KNX-IOT/KNX-IOT-STACK.

48. *Chytré budovy* [online]. Svaz moderní energetiky [cit. 2024-05-14]. Dostupné z: https://www.modernienergetika.cz/wp-content/uploads/2019/10/chytre-domy-zlom_2019_10_23-final-web.pdf.

49. VARGIC, Radoslav; TRÚCHLY, Peter; PODHRADSKÝ, Pavol. Inteligentní měření a inteligentní energetické systémy. In: *Inteligentní technologie* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2019, s. 34–36 [cit. 2024-04-16]. Dostupné z: `https://techpedia.fel.cvut.cz/single/?objectId=153`.

50. FLÁŠAR, Petr; FOUSEK, Jan; JÍCHA, Tomáš; KABELE, Richard; KANTA, Jan et al. *Úvod do liberalizované energetiky*. Praha: Asociace energetických manažerů, 2016. ISBN 978-80-260-9212-4.

51. *Vyhláška č. 359/2020 Sb., o měření elektřiny ve znění vyhlášky č. 375/2023 Sb.* [online]. Elektronická Sbírka zákonů a mezinárodních smluv, Ministerstvo vnitra České republiky [cit. 2024-04-17]. Dostupné z: `https://www.e-sbirka.cz/sb/2020/359/2024-01-01`.

52. NOVOTNÁ, Kateřina. *Co utváří cenu elektřiny?* [online]. CSRD, 2023 [cit. 2024-05-14]. Dostupné z: `https://csrd.cz/z-ceho-se-tvori-cena-elektriny/`.

53. *Silová elektřina* [online]. Elektřina.cz [cit. 2024-05-11]. Dostupné z: `https://www.elektrina.cz/slovnik/silova-elektrina`.

54. *ERÚ zveřejnil regulované ceny elektřiny a plynu na rok 2024* [online]. Energetický regulační úřad, 2023 [cit. 2024-05-14]. Dostupné z: `https://eru.gov.cz/eru-zverejnil-regulovane-ceny-elektriny-plynu-na-rok-2024`.

55. MUDRUŇKOVÁ, Anna. *Elektroenergetika 1* [online]. VOŠ a SPŠ elektrotechnická Františka Křižíka, 2016 [cit. 2024-05-11]. ISBN 978-80-88058-81-6. Dostupné z: `https://publi.cz/books/260/01.html#11`.

56. *Co pro spotřebitele změnila letošní novela energetického zákona* [online]. Energetický regulační úřad, 2024 [cit. 2024-05-11]. Dostupné z: `https://eru.gov.cz/co-pro-spotrebitele-zmenila-letosni-novela-energetickeho-zakona`.

57. *Denní trh* [online]. OTE, a.s [cit. 2024-04-18]. Dostupné z: `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`.

58. FLÁŠAR, Petr; FOUSEK, Jan; JÍCHA, Tomáš; KABELE, Richard; KANTA, Jan et al. OBCHODOVÁNÍ S ELEKTŘINOU NA ORGANIZOVANÝCH TRZÍCH A DVOUSTRANNÉ OBCHODY. In: *Úvod do liberalizované energetiky*. Praha: Asociace energetických manažerů, 2016, s. 110. ISBN 978-80-260-9212-4.

59. *Základní údaje* [online]. OTE, a.s. [cit. 2024-05-11]. Dostupné z: `https://www.ote-cr.cz/cs/o-spolecnosti/zakladni-udaje`.

60. ZIKÁN, Zdeněk. *Zpětné získávání tepla a větrání objektů* [online]. TZB-info, 2010 [cit. 2024-05-01]. Dostupné z: `https://vetrani.tzb-info.cz/vetrani-s-rekuperaci/6325-zpetne-ziskavani-tepla-a-vetrani-objektu`.

61. *Michal Kapoun* [online]. TZB-info [cit. 2024-05-04]. Dostupné z: https://vytapeni.tzb-info.cz/tepelna-cerpadla/12629-co-je-to-tepelne-cerpadlo-zakladni-casti-druhy.

62. *Modbus Protocol (V1.7) for the Versati Heat Pump Water* [online]. Loxone, GREE ELECTRIC APPLIANCES, INC. OF ZHUHAI [cit. 2024-05-04]. Dostupné z: https://api.library.loxone.com/downloader/file/1099/Modbus%20Protocol%20(V1.7)%20for%20the%20Versati%20Heat%20Pump%20Water.pdf.

63. *Principy systému KNX* [online]. KNX NG CZ [cit. 2024-05-11]. Dostupné z: https://knxcz.cz/images/clanky/KNX-System-Principles_cz.pdf.

64. *Andrea Ronešová* [online]. zcu.cz [cit. 2024-05-15]. Dostupné z: http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf.

65. *Zákon č. 469/2023 Sb., kterým se mění zákon č. 458/2000 Sb., o podmínkách podnikání a o výkonu státní správy v energetických odvětvích a o změně některých zákonů (energetický zákon), ve znění pozdějších předpisů, a další související zákony* [online]. Elektronická Sbírka zákonů a mezinárodních smluv, Ministerstvo vnitra České republiky [cit. 2024-04-19]. Dostupné z: https://www.e-sbirka.cz/sb/2023/469/2024-04-01.

66. *Právní povaha norem v energetickém zákoně* [online]. Energetický regulační úřad, Oddělení legislativní, 2022-04 [cit. 2024-04-19]. Dostupné z: https://eru.gov.cz/pravni-povaha-norem-v-energetickem-zakone.

67. ŽIDLICKÁ, Michaela. Soukromoprávní recepce. In: ŽIDLICKÁ, Michaela; CIPROVSKÝ, Tomáš; DOSTALÍK, Petr. *Vliv římského práva na evropské právní myšlení: pocta Valentinu Urfusovi* [online]. Masarykova univerzita, ELPORTÁL, 2015–2019 [cit. 2024-04-19]. Dostupné z: https://is.muni.cz/do/rect/el/estud/praf/js19/urfus/web/pages/01-soukromopravni-recepce.html.

68. BLAHOUDKOVÁ, Gabriela. *Energetické právo České republiky* [online]. Praha, 2020 [cit. 2024-04-19]. Dostupné z: http://hdl.handle.net/20.500.11956/122061. Dipl. pr. Univerzita Karlova, Právnická fakulta. Vedoucí práce Jakub HANDRLICA.

69. *Historie Zákonu č. 458/2000 Sb.* [online]. Elektronická Sbírka zákonů a mezinárodních smluv, Ministerstvo vnitra České republiky [cit. 2024-04-25]. Dostupné z: https://www.e-sbirka.cz/sb/2000/458?zalozka=historie.

70. POKORNÝ, Jiří. Evropské energetické právo: Vybrané novinky zimního energetického balíčku. *Acta Universitatis Carolinae Iuridica*. 2019, roč. 65, s. 93–100. ISSN 0323-0619. Dostupné také z: https://doi.org/10.14712/23366478.2019.32.

71. *Přechod na 15 minutový zúčtovací a obchodní interval v ČR z pohledu operátora trhu* [online]. OTE, a.s. [cit. 2024-05-14]. Dostupné z: https://www.ote-cr.cz/cs/dokumentace/dokumentace-elektrina/2023_06_21-informacni-webinar-ote-k-prechodu-na-15-min-zuctovaci-periodu-v-cr-2.pdf.

72. *Uživatelský manuál webové služby OTE* [online]. OTE, a.s., 2023 [cit. 2024-04-16]. Dostupné z: https://www.ote-cr.cz/cs/dokumentace/dokumentace-elektrina/uzivatelsky-manual_webove_sluzby_ote_c.pdf.

73. MISES, Ludwig von. *Lidské jednání: pojednání o ekonomii*. Praha: Liberální institut, 2006. ISBN 80-86389-45-6.

Obsah příloh

/
- `cppcore` ..... adresář se zdrojovými kódy v C++
- `sveltewebapp` ..... adresář se zdrojovými kódy ve Svelte
- `bakalarska_prace.pdf` ..... bakalářská práce ve formátu PDF