

Rozšíření webové aplikace pro hodnocení nakládání s odpadem a optimalizaci práce s daty

Bc. Karel Šohaj



ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Bc. Karel Šohaj
Osobní číslo:	A23509
Studijní program:	N0613A140022 Informační technologie
Specializace:	Softwarové inženýrství
Forma studia:	Kombinovaná
Téma práce:	Rozšíření webové aplikace pro hodnocení nakládání s odpadem a optimalizaci práce s daty
Téma práce anglicky:	Extension of a Web Application for Waste Management Evaluation and Data Optimization

Zásady pro vypracování

- Seznamte se s existující webovou aplikací, nastudujte využívané technologie a zpracujte související teoretickou část.
- Navrhněte a implementujte správu uživatelských dat v rámci databázového systému, FE i BE.
- Rozšiřte webovou aplikaci o nové funkcionality a analytickou část.
- Otestujte navržené a implementované funkcionality, analyzujte výsledky testování a proveďte optimalizace s ohledem na výkon a uživatelskou přívětivost aplikace.

Seznam doporučené literatury:

1. CHATURVEDI, Rajneesh, Swati V. CHANDE a Amita SHARMA. Evaluation and Refinement of MVC Web Application Architecture. Journal of Information and Computational Science. 2021, 2021(3). ISSN 1548-7741.
2. HAKLAY, M. a WEBER, P. OpenStreetMap: User-Generated Street Maps. Online. IEEE Pervasive Computing. 2008, roč. 7, č. 4, s. 12-18. ISSN 1536-1268. Dostupné z: <https://doi.org/10.1109/MPRV.2008.80>. [cit. 2024-10-25].
3. ŠOMPLÁK, Radovan; KOPA, Miloš; OMELKA, Marek; NEVRLÝ, Vlastimír a PAVLAS, Martin. Multi-level stratification of territories for waste composition analysis. Online. Journal of Environmental Management. 2022, roč. 318. ISSN 03014797. Dostupné z: <https://doi.org/10.1016/j.jenvman.2022.115534>. [cit. 2024-10-25].
4. W3 schools: AJAX Introduction. Online. Dostupné z: https://www.w3schools.com/xml/ajax_intro.asp. [cit. 2021-10-21].
5. OpenStreetMap documentation: Deploying your own Slippy Map. Online. Dostupné z: https://wiki.openstreetmap.org/wiki/Deploying_your_own_Slippy_Map. [cit. 2021-10-21].
6. Flask-SocketIO. Online. Dostupné z: <https://flask-socketio.readthedocs.io/>. [cit. 2021-10-21].

Vedoucí diplomové práce: **Ing. Dušan Hrabec, Ph.D.**
Ústav matematiky

Konzultant diplomové práce: **Ing. Vlastimír Nevrlý, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **27. října 2024**
Termín odevzdání diplomové práce: **2. června 2025**



L.S.

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

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

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení diplomové práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – diplomovou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na diplomové práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato diplomová práce se zabývá rozšířením a optimalizací modulu webové aplikace Popelka, která slouží k hodnocení míry separace komunálního odpadu na úrovni obcí v České republice. Cílem bylo doplnit modul o nástroj pro zadání cílové míry separace a výpočet potřebného navýšení potenciálu jednotlivých složek odpadu, a zároveň implementovat správu uživatelských dat včetně jejich perzistentního uložení. Součástí práce bylo také přepracování struktury modulu, optimalizace práce s daty a revize uživatelského rozhraní.

Klíčová slova: webová aplikace, odpadové hospodářství, modelování míry separace odpadu, optimalizace, databáze, MariaDB, Python, Flask, Redux, Next.js, React, TypeScript

ABSTRACT

This thesis focuses on the extension and optimization of a module within the Popelka web application, which is designed to evaluate the level of municipal waste separation at the level of municipalities in the Czech Republic. The aim was to add a tool for entering a target separation rate and calculating the required increase in the potential of individual waste fractions, as well as to implement the management and persistent storage of user data. The work also included restructuring the module, optimizing data handling, and revising the user interface.

Keywords: web application, waste management, waste separation modelling, optimization, database, MariaDB, Python, Flask, Redux, Next.js, React, TypeScript

Na tomto místě bych rád poděkoval vedoucímu diplomové práce Ing. Dušanu Hrabcovi, Ph.D., a konzultantovi diplomové práce Ing. Vlastimíru Nevrlému za jejich odborné vedení, cenné rady a vstřícný přístup při tvorbě této práce. Dále děkuji Ing. Jaroslavu Pluskalovi, Ph.D., za poskytnuté podklady a Ing. Adamu Mrkvičkovi za ochotu podělit se o své znalosti a zkušenosti. Zejména bych však chtěl poděkovat své manželce za neuvěřitelnou trpělivost a podporu, kterou mi po celou dobu studia poskytovala, a která byla mou hlavní oporou zejména v jeho závěru. Děkuji také své dceři za pochopení, že jsem jí nemohl věnovat tolik času. A ostatním členům rodiny za veškerou pomoc.

OBSAH

ÚVOD	13
I TEORETICKÁ ČÁST	15
1 ODPADOVÉ HOSPODÁŘSTVÍ	16
1.1 PRÁVNÍ RÁMEC A REGULACE	16
1.1.1 Regulace na úrovni Evropské unie	17
1.1.2 Právní rámec v České republice	19
1.2 KATEGORIE A DRUHY ODPADŮ	20
1.3 SHLUKOVÉ SKUPINY OBCÍ	22
1.4 MATEMATICKÝ MODEL PRO MODELOVÁNÍ POTENCIÁLU	23
1.4.1 Popis veličin	25
1.4.2 Výpočty	25
2 WEBOVÁ APLIKACE	30
2.1 MODULY A JEJICH FUNKCIONALITY	30
2.2 BACKEND	32
2.2.1 Python	32
2.2.2 Flask	33
2.2.3 MariaDB	34
2.2.4 Redis a Redis Queue	35
2.2.5 NGNIX	36
2.2.6 uWSGI	36
2.3 FRONTEND	36
2.3.1 React	37
2.3.2 React Router	38
2.3.3 Tailwind CSS	38
2.3.4 Zod	38
2.3.5 Redux, React Redux, Redux Toolkit a Redux DevTools	39
II PRAKTICKÁ ČÁST	41
3 ANALÝZA STÁVAJÍCÍHO ŘEŠENÍ	42

4	NÁVRH ŘEŠENÍ	44
4.1	FUNKČNÍ A NEFUNKČNÍ POŽADAVKY	44
4.2	KONCEPČNÍ NÁVRH MODULU	44
4.2.1	Backend	45
4.2.2	Frontend	46
4.2.3	Databáze	46
4.3	STRUKTURA MODULU	47
4.3.1	Souborová struktura	47
5	IMPLEMENTACE	50
5.1	SPRÁVA STAVU	50
5.1.1	Waste data slice	50
5.1.2	Waste data API slice	53
5.2	DATABÁZE	55
5.2.1	Tabulky konstant produkce odpadů pro shlukové skupiny obcí	56
5.2.2	Tabulka konfigurace vstupů	57
5.2.3	Tabulka scénářů	59
5.3	POMOCNÉ FUNKCE	59
5.4	POMOCNÉ VÝPOČTY A ALGORITMY	60
5.4.1	SuggestionFormUtil	61
5.4.2	ModelingFormUtil	62
5.5	SEKCE MODULU	62
5.5.1	Obecné informace	62
5.5.2	Konfigurace vstupů	63
5.5.3	Návrh	67
5.5.4	Modelování	70
6	TESTOVÁNÍ	73
6.1	FUNKČNÍ TESTOVÁNÍ	73
6.2	UŽIVATELSKÉ TESTOVÁNÍ	74
6.3	ZJIŠTĚNÉ NEDOSTATKY	74
6.4	VÝKONNOSTNÍ TESTOVÁNÍ PŘI OMEZENÉ RYCHLOSTI PŘIPOJENÍ	75

6.4.1	Shrnutí	75
ZÁVĚR		77
SEZNAM POUŽITÉ LITERATURY		79
SEZNAM OBRÁZKŮ		84
SEZNAM TABULEK		85
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK		86

ÚVOD

V současné době je ochrana životního prostředí a udržitelnost klíčovými tématy na globální úrovni. Diskuse o ekologických opatřeních a strategiích směřujících k minimalizaci negativních dopadů lidské činnosti na přírodu jsou stále intenzivnější. Politické reprezentace i odborníci často prosazují různé restriktce, zákazy a omezení s cílem snížit emise skleníkových plynů, omezit používání jednorázových plastů či regulovat průmyslové procesy vedoucí ke znečištění ovzduší a vodních zdrojů. Ačkoli jsou tato opatření nezbytná, představují pouze jednu stranu řešení. Pro skutečně efektivní přechod k udržitelnému hospodářství je klíčové zaměřit se také na druhou stranu rovnice – na efektivní nakládání s odpady, recyklaci a jejich opětovné využití jako cenného zdroje surovin.

Recyklace a odpadové hospodářství jsou dnes považovány za zásadní prvky udržitelného rozvoje. Hlavní myšlenkou je minimalizace produkce odpadu a jeho přeměna na nové vstupní suroviny, což umožňuje nejen snižovat ekologickou zátěž, ale i optimalizovat ekonomické náklady spojené s těžbou a výrobou nových materiálů. Těchto cílů lze dosáhnout pouze prostřednictvím efektivních systémů třídění a recyklace, které jsou podporovány jak legislativními opatřeními, tak moderními technologiemi.

V České republice se systém třídění odpadu v posledních desetiletích výrazně zlepšil, a to natolik, že Česká republika patří mezi špičku evropského žebříčku v třídění. Občané mají široké možnosti třídění díky husté síti kontejnerů. Těch bylo v roce 2022 přes 640 tisíc a průměrná docházková vzdálenost k nim byla kolem 90 metrů. [1]

V dodržování milníků míry separace odpadů hrajou klíčovou roli jednotlivé obce. Ty jsou povinny zajistit místa pro oddělené soustředění komunálního odpadu (KO), zahrnující nebezpečné odpady, papír, plasty, sklo, kovy, biologický odpad a jedlé oleje a tuky. Od 1. ledna 2025 se tato povinnost rozšířila i na textil [2]. Přestože dnešní infrastruktura umožňuje efektivní sběr, klíčovým faktorem zůstává důsledná a správně organizovaná správa dat o odpadech. Obce však často nedisponují žádnými podpůrnými nástroji, které by jim takovou správu a analýzu dat poskytly.

Naštěstí s rozvojem informačních technologií se otevírají nové možnosti v oblasti monitoringu a optimalizace nakládání s odpady. Digitální nástroje, jako jsou chytré senzory v kontejnerech, pokročilé datové analýzy či automatizované systémy řízení svozu odpadu, umož-

ňují efektivnější plánování a zvyšují účinnost celého procesu. Právě v této souvislosti vznikají webové aplikace a digitální platformy zaměřené na zlepšení správy odpadu na úrovni jednotlivých obcí či regionů.

Tato diplomová práce se zaměřuje na rozšíření webové aplikace Popelka, která byla vyvíjena ve spolupráci mezi Ústavem procesního inženýrství (ÚPI), Ústavem matematiky a Ústavem automatizace a informatiky na Vysokém učení technickém (VUT) v Brně. Tento nástroj slouží k hodnocení úrovně separace KO na základních územních jednotkách České republiky a poskytuje důležité analytické informace, které mohou pomoci obcím optimalizovat jejich strategii v nakládání s odpady. Stávající aplikace umožňuje sběr a vyhodnocování dat o separaci odpadu, avšak její funkcionality vyžaduje další rozšíření, zejména v oblasti efektivnější správy dat, analytických nástrojů a uživatelského rozhraní.

Cílem této práce je rozšířit modul aplikace tak aby umožňoval, detailnější a přesnější správu uživatelských dat, přidat pokročilé analytické nástroje a vylepšit vizualizaci získaných informací. Zároveň bude řešena optimalizace výkonu a efektivní komunikace mezi backendem (BE) a frontendem (FE) aplikace. Výsledkem by měl být nástroj, který nejen poskytne lepší přehled o úrovni třídění odpadu, ale také nabídne konkrétní doporučení pro jeho zlepšení a umožní efektivní plánování strategií v oblasti komunálního odpadového hospodářství. Součástí analytických funkcionalit aplikace bude i metodika odhadu potenciálu pro zvýšení míry třídění odpadu ve vybrané obci na základě statistického srovnání s jinými obcemi. Tento přístup bude zohledňovat složení směsného KO a objemného odpadu (OBJ), čímž umožní navrhnout realistický scénář změny separace jednotlivých frakcí odpadu. Výpočet těchto scénářů bude vycházet z odhadovaného potenciálu jednotlivých frakcí odpadu a jeho naplnění s ohledem na aktuální situaci a očekávanou náročnost zvyšování míry separace. Tímto způsobem aplikace nejen podpoří efektivnější řízení nakládání s odpady, ale také pomůže obcím dosáhnout požadovaných legislativních cílů v oblasti odpadového hospodářství.

I TEORETICKÁ ČÁST

1 ODPADOVÉ HOSPODÁŘSTVÍ

Odpadové hospodářství je oblastí úzce spjatou se všemi druhy výrobních procesů. Představuje soubor činností, které se zaměřují na práci s odpadem od začátku do konce. To znamená, jak na nakládání s ním, následnou péči o místa jeho trvalého uložení, tak i na kontrolu těchto procesů. Hlavním cílem je minimalizovat negativní dopady odpadů na životní prostředí a lidské zdraví prostřednictvím efektivního řízení a využívání odpadů jako cenných zdrojů. Tím se odklání od tradičního lineárního modelu "vyrobit–použít–vyhodit" směrem k udržitelnějšímu systému.

Klíčovým konceptem v rámci odpadového hospodářství je hierarchie nakládání s odpady, která stanovuje následující priority. [3]

- **Předcházení vzniku odpadu** – nejvyšší prioritu má snaha zabránit vzniku odpadu prostřednictvím optimalizace výrobních procesů, designu produktů a změn spotřebitelského chování.
- **Příprava k opětovnému použití** – pokud odpad již vznikne, je preferována jeho úprava tak, aby mohl být znovu použit bez nutnosti dalšího zpracování.
- **Recyklace** – přeměna odpadních materiálů na nové produkty nebo suroviny, čímž se snižuje potřeba těžby nových zdrojů.
- **Jiné využití, včetně energetického** – využití odpadu pro výrobu energie nebo jiných forem materiálového využití, pokud recyklace není možná.
- **Odstranění** – nejméně žádoucí možností je ukládání odpadu na skládky nebo jeho spalování bez energetického využití.

1.1 Právní rámec a regulace

Aby mohlo být nakládání s odpady efektivní, musí být podmíněno pevně daným legislativním rámcem, který stanovuje jasná pravidla, cíle a odpovědnosti pro všechny zúčastněné subjekty. Tento rámec vychází jak z národních, tak nadnárodních právních předpisů, které reflektují aktuální výzvy v oblasti udržitelného hospodářství a ochrany životního prostředí. Základním prvkem evropské i české odpadové legislativy je přechod na oběhové hospodářství, jehož cílem je minimalizace produkce odpadu a maximalizace opětovného využití materiálů.

1.1.1 Regulace na úrovni Evropské unie

Evropská unie (EU) dlouhodobě usiluje o přechod k oběhovému hospodářství a začlenění cirkulární ekonomiky do legislativy EU. Základním kamenem pro tento krok se stal právní rámec pro nakládání s odpady Balíček pro oběhové hospodářství (Circular Economy Package – CEP), který byl přijat Evropskou komisí v prosinci 2015. Tento soubor legislativních a strategických opatření klade důraz na snižování množství odpadu, zvyšování míry recyklace a přepracování stávající odpadové legislativy s cílem dosáhnout efektivnějšího využívání surovin.

Balíček CEP zavádí revizi klíčových směrnic a obsahuje závazné cíle. Mezi hlavní směrnice patří:

- **Směrnice EU 2018/850** – omezuje maximální množství KO, které lze skládkovat, na 10 % do roku 2035.
- **Směrnice EU 2018/851** – stanovuje cíle pro recyklaci KO.
- **Směrnice EU 2018/852** – obsahuje opatření týkající se nakládání s obalovými odpady, která stanovují minimální podíl recyklace pro jednotlivé materiály.
- **Směrnice EU 2018/849** – reguluje nakládání s vozidly s ukončenou životností, bateriemi a elektroodpadem.

Balíček CEP nepokrývá pouze oblast odpadového hospodářství, ale zahrnuje i širší koncepci cirkulární ekonomiky, která se zaměřuje na trvanlivost, opravitelnost a recyklovatelnost produktů. Důležitým nástrojem k dosažení těchto cílů je standardizace, která umožňuje jednotné zavádění opatření napříč členskými státy EU a usnadňuje implementaci materiálové efektivity v průmyslových procesech.[4]

V návaznosti na tento balíček byl v březnu 2020 přijat nový akční plán pro oběhové hospodářství (Circular Economy Action Plan – CEAP) jako součást Evropské zelené dohody (European Green Deal). Tento plán představuje strategii pro udržitelný růst a transformaci evropského hospodářství směrem k vyšší konkurenceschopnosti a ekologické udržitelnosti.

CEAP zavádí iniciativy pokrývající celý životní cyklus produktů, počínaje jejich návrhem, přes výrobní procesy, spotřebu, až po nakládání s odpady. Cílem je zajistit, aby zdroje zůstaly v ekonomice EU co nejdéle a aby se minimalizovala produkce odpadu. Plán obsahuje

legislativní i nelegislativní opatření zaměřená na oblasti, kde může akce na úrovni EU přinést skutečnou přidanou hodnotu. Mezi hlavní cíle patří:

- **Učinit z udržitelných produktů standard v EU** – zavedení legislativních opatření, která zajistí, že produkty budou navrženy s ohledem na trvanlivost, opravitelnost a recyklovatelnost.
- **Posílit postavení spotřebitelů a veřejných zadavatelů** – poskytnout spotřebitelům informace a nástroje pro informovaná rozhodnutí, a podpořit udržitelné nákupní praktiky ve veřejném sektoru.
- **Zaměřit se na sektory s vysokou spotřebou zdrojů a potenciálem pro oběhovost** – mezi tyto sektory patří elektronika a informační a komunikační technologie (ICT), baterie a vozidla, obaly, plasty, textil, stavebnictví a budovy, potraviny, voda a živiny.
- **Zajistit snížení množství odpadu** – přijetí opatření k prevenci vzniku odpadu a podpora opětovného využití a recyklace materiálů.
- **Podpořit oběhovost na úrovni měst a regionů** – zapojení místních a regionálních orgánů do implementace oběhových strategií.
- **Vést globální úsilí o prosazení oběhového hospodářství** – spolupráce s mezinárodními partnery na podpoře globálního přechodu k oběhovému hospodářství.

Implementace CEAP zahrnuje konkrétní legislativní návrhy a iniciativy, jako je strategie pro udržitelné a oběhové textilie, revize směrnice o obalech a obalových odpadech, či opatření proti jednorázovým plastům. Tyto kroky mají za cíl nejen chránit životní prostředí, ale také posílit ekonomiku prostřednictvím inovací a vytváření nových pracovních míst v oblasti zelených technologií.

Přijetí CEAP představuje významný krok směrem k udržitelnější budoucnosti, kde jsou zdroje efektivně využívány a environmentální dopady minimalizovány. Tento rámec slouží jako základ pro národní strategie členských států, včetně České republiky, které musí tyto principy integrovat do svých legislativních a regulačních opatření. [5]

1.1.2 Právní rámec v České republice

Česká republika prošla od počátku 90. let 20. století významným vývojem v oblasti odpadového hospodářství, který byl ovlivněn jak vnitrostátními potřebami, tak závazky vyplývajícími z členství v Evropské unii.

První komplexní právní úprava nakládání s odpady byla přijata v roce 1991 prostřednictvím zákona č. 238/1991 Sb., o odpadech. Tento zákon stanovil základní pravidla pro nakládání s odpady a definoval povinnosti pro fyzické i právnické osoby. Vstupem do Evropské unie se Česká republika zavázala implementovat evropskou legislativu týkající se odpadového hospodářství. To vedlo k přijetí nového zákona č. 541/2020 Sb., o odpadech, účinného od 1. ledna 2021. Tento zákon klade důraz na hierarchii nakládání s odpady a stanovuje následující minimální hodnoty separace KO v obcích.

- **60 %** - 2025.
- **65 %** - 2030.
- **70 %** - 2035.

Pro naplnění cílů stanovených evropskými směrnici a národní legislativou byl vypracován Plán odpadového hospodářství České republiky (POH ČR). Tento strategický dokument stanovuje rámec pro nakládání s odpady na národní úrovni a definuje strategie vývoje POH v ČR. Dále mimo jiné zdůrazňuje potřebu rozšiřování třídících a recyklačních kapacit, investic do moderních technologií a posilování ekonomických pobídek pro obce a podniky. [6]

Na konci roku 2024 Ministerstvo životního prostředí předložilo do mezirezortního připomínkového řízení návrh nového POH ČR na období 2025–2035. Tento plán formuluje dlouhodobou perspektivu a konceptuální rámec pro udržitelné nakládání s odpady s horizontem do roku 2035, v kontextu závazných cílů vymezených v evropské a národní legislativě. Plán důsledně vychází z principu hierarchie odpadového hospodářství, kladoucí důraz na předcházení vzniku odpadů, maximální recyklaci a využití odpadů. Součástí plánu jsou také Program předcházení vzniku odpadů a Program předcházení vzniku potravinového odpadu. Plán rovněž reflektuje oblast kritických surovin v souladu s požadavky nařízení, kterým se stanoví rámec pro zajištění bezpečných a udržitelných dodávek kritických surovin.

POH ČR 2025–2035 je zpracován na období 11 let a v případě zásadních změn podmínek bude provedena jeho aktualizace. [7]

Obce, které hrají klíčovou roli v systému odpadového hospodářství, jsou podporovány finančními nástroji, například prostřednictvím Státního fondu životního prostředí ČR (SFŽP ČR), který nabízí dotace na projekty v oblasti recyklace a modernizace sběrných dvorů. Dále existuje systém EKO-KOM, který provozuje kolektivní financování třídění a recyklace obalových odpadů, čímž pomáhá obcím financovat sběrné systémy a vzdělávací aktivity v oblasti třídění odpadu.

1.2 Kategorie a druhy odpadů

Komunální odpad představuje významnou složku celkového odpadového hospodářství. Podle zákona č. 541/2020 Sb., o odpadech, je KO definován jako veškerý odpad vznikající činnostmi v domácnostech či jiných útvarů na území obce. Nejčastěji pak zahrnuje papír, sklo, kovy, plasty či biologický odpad. Tato definice zahrnuje odpady z domácností a jim podobné odpady z nevýrobních činností. [6]

V kontextu aplikace a především vyvíjeného modulu pro modelování potenciálu, zavedli pracovníci ÚPI tři skupiny KO. Rozdělení má za cíl přesnější stanovení KO pro účely modelování různých scénářů sběru odpadu a jeho následné separace.

1. **Množství odpadu ze systému sběru obce** – zahrnuje celkový objem odpadů shromážděných prostřednictvím služeb zajišťovaných obcí. Tento údaj zahrnuje veškerý KO pocházející z domácností, veřejných institucí a komerčních provozoven, který je následně předán k dalšímu zpracování nebo likvidaci v souladu s platnými předpisy a obecními odpadovými systémy.
2. **Netypické zdroje odpadu v obecním sběru** – zahrnují odpady vzniklé na území obce, které však nepocházejí z běžného provozu domácností ani standardních obecních činností. Tyto odpady vznikají z méně obvyklých zdrojů, jako jsou například hřbitovy, veřejné parky nebo jiná specifická místa.
3. **Mimosystémové zdroje odpadu** – zahrnují odpady, které jsou ukládány do zařízení pro nakládání s odpady, aniž by byly součástí obecního systému sběru. Tyto odpady

nejsou zahrnuty v rámci smluvních vztahů obce se zpracovateli odpadu a mohou pocházet od subjektů, které nejsou zapojeny do obecního systému nakládání s odpady.

Samotné položky KO je dále možné dělit podle materiálu, z kterého jsou vyrobeny. I toto rozdělení je přesně nadefinované a zanesené v zákoně, a to vyhláškou č. 8/2021 Sb., která detailně definuje jednotlivé kategorie a kódy odpadů. Tento právní předpis umožňuje přesné třídění, evidenci a řízení odpadových toků, což je zásadní pro efektivní recyklaci a minimalizaci environmentálního dopadu. [8]

Díky přesné katalogizaci odpadů mohou jak komunální orgány, tak odpadové společnosti a průmyslové podniky lépe identifikovat a řídit jednotlivé složky odpadů, čímž se zvyšuje efektivita celého systému nakládání s odpady. Tento systematický přístup zároveň usnadňuje dodržování legislativních požadavků a zajišťuje transparentnost při plnění regulačních povinností na národní i evropské úrovni.

Další rozdělení katalogových položek KO, které se uplatňuje nejen ve vyvíjené aplikaci, je rozdělení do čtyř skupin podle charakteru recyklace a původu dané položky.

1. **Recyklovatelné odpady** – odpady, které lze efektivně oddělit od ostatních odpadů, následně upravit na druhotné suroviny a využít v nových výrobních procesech. Díky tomu, slouží jako plnohodnotná náhrada primárních surovin, které jsou často získávány z neobnovitelných zdrojů. Jejich využití nejen snižuje závislost na těžbě přírodních materiálů, ale zároveň významně redukuje spotřebu energie a dalších výrobních vstupů, čímž přispívá k udržitelnějšímu hospodaření se zdroji.
2. **Směsný komunální odpad - SKO** – odpad, který zůstává po vytrídění recyklovatelných odpadů z KO nebo odpady vzniklé v domácnostech a veřejných prostorách, které již nejsou dále tříděny. Nejčastěji se jedná o obaly od potravin, či zbytky potravin. Ačkoliv se může zdát, že se jedná o zbytek KO po vytrídění recyklačních odpadů, tvoří většinu samotného objemu KO.
3. **Objemný odpad - OBJ** – odpad, který vzhledem ke svému objemu nebo hmotnosti nelze uložit do běžných sběrných nádob. Typicky sem patří například rozbitý nábytek, koberce či další vybavení domácností. Pro správnou likvidaci tohoto odpadu je možné využít sběrné dvory, případně mobilní sběr, pokud jej daná obec organizuje.

4. **Ostatní KO** – odpady, které nelze zařadit mezi recyklovatelné složky, SKO ani OBJ.

Tato skupina může zahrnovat specifické nebezpečné materiály, ať už rozpouštědla nebo kyseliny, dále barvy či pesticidy.

Katalogový přehled všech položek KO je pak uveden v tabulkách 1.1 a 1.2.

Tabulka 1.1 Katalogový přehled KO

Katalogové číslo	Název	Barva kontejneru
20101	Papír a lepenka	Modrá
20102	Sklo	Bílá (bílý), Zelená (barevné)
0108	Biologicky rozložitelný odpad z kuchyní a stravoven	Hnědá
20110	Oděvy	Bílá
20111	Textilní materiály	Bílá
20125	Jedlý olej a tuk	Khaki
20138	Dřevo	-
20139	Plasty	Žlutá
20140	Kovy	Červená
20201	Biologicky rozložitelný odpad	Hnědá
20301	SKO	Černá
20307	OBJ	Oranžová

(zdroj: vlastní)

[9][10]

1.3 Shlukové skupiny obcí

Pro přesný odhad potenciálu navýšení míry recyklace SKO je vhodné vycházet ze zkušeností a dat z obcí s podobnými charakteristikami. Právě z tohoto důvodu vznikly shlukové skupiny obcí, které umožňují porovnávat data o třídění a nakládání s odpady mezi obcemi se srovnatelnými socioekonomickými a demografickými faktory. Rozdělení obcí do homogenních skupin umožňuje přesnější identifikaci složení odpadu, lepší využití zdrojů, efektivnější plánování a podporu udržitelného rozvoje.

Jedním z klíčových faktorů pro výpočet v samotném modulu je kategorie dané obce. Tato kategorie určuje, do jakého výběru odhadovaného poměru jednotlivých recyklovatelných složek obsažených ve SKO obec spadá. Díky zařazení do konkrétní shlukové skupiny lze přesněji odhadovat složení odpadu a zlepšovat plánování svozu a třídění. Tento přístup přispívá k optimalizaci celého systému nakládání s odpady a zvyšuje jeho efektivitu.

Tabulka 1.2 Katalogový přehled ostatní KO

Katalogové číslo	Název
200113	Rozpouštědla
200114	Kyseliny
200115	Zásady
200117	Fotochemikálie
200119	Pesticidy
200126	Olej a tuk neuvedený pod číslem 200125
200127	Barvy, tiskařské barvy, lepidla a pryskyřice obsahující nebezpečné látky
200128	Barvy, tiskařské barvy, lepidla a pryskyřice neuvedené pod číslem 200127
200129	Detergenty obsahující nebezpečné látky
200130	Detergenty neuvedené pod číslem 200129
200131	Nepoužitelná cytostatika
200132	Jiná nepoužitelná léčiva neuvedená pod číslem 200131
200137	Dřevo obsahující nebezpečné látky
200141	Odpady z čištění komínů
200199	Další frakce jinak blíže neurčené
200203	Jiný biologicky nerozložitelný odpad
200302	Odpad z tržišť
200303	Uliční smetky
200399	KO jinak blíže neurčené

(zdroj: vlastní)

Kompletní rozdělení vychází ze studie na téma stratifikace území pro analýzu složení odpadů provedené Ústavem procesního inženýrství Fakulty strojního inženýrství VUT v Brně a Katedry pravděpodobnosti a matematické statistiky Matematicko-fyzikální fakulty Univerzity Karlovy [10]. A pro matematický model na jejím základě byly definovány 4 kategorie.

- **Velké město**
- **Střední město**
- **Vesnice**
- **Nejmenší obce**

1.4 Matematický model pro modelování potenciálu

Jádrem modulu pro hodnocení nakládání s odpadem je modelování potenciálu KO. Jeho matematický model byl důkladně popsán v Diplomové práci Marka Niedoby [11]. Modul slouží

k vyhodnocení dosažených cílů v oblasti separace recyklovatelných složek KO. Umožňuje simulovat změny v systému odpadového hospodářství s ohledem na identifikovaný potenciál. Hlavním zdrojem dat je projekt TIRSMZP719, v rámci kterého bylo mezi lety 2019 a 2022 analyzováno složení odpadu ve více než 700 případech. Model odhaduje složení odpadu v konkrétní obci na základě podobnosti s jinými obcemi podle klíčových charakteristik, jako je například velikost.

Konkrétně tedy vypočtený potenciál separovaného odpadu, například skla, pak udává celkové množství separovaného odpadu skla ze SKO, kterého je možné dosáhnout vyšší mírou třídění a následné recyklace.

Tématem této diplomové práce je rozšíření stávajícího modulu o návrh efektivního scénáře pro zvýšení míry třídění KO. To znamená v porovnání s předešlou prací, aby naopak bylo možné zadat cílové procento požadované míry separace KO a model sám dopočítal potřebné zvýšení potenciálního třídění KO. Za tímto účelem byl pracovníky z ÚPI VUT v Brně poskytnut matematický model, jehož hlavním autorem je pan Ing. Jaroslav Pluskal, Ph.D.

Stěžejní myšlenkou modelu je, že pokud obec aktuálně dosahuje míry separace daného odpadu blízko maximální hodnotě jeho potenciálního růstu, bude se již tato hodnota jen velmi těžko navyšovat. Naproti tomu, opačný případ, pokud obec aktuálně nedosahuje míry separace daného odpadu. Pak lze předpokládat výrazný nárůst separace KO, při realizaci třídění a recyklace tohoto odpadu.

Model využívá lineární funkci, která zajišťuje monotónní růst míry třídění mezi jednotlivými frakcemi. To znamená, že frakce s vyšší stávající mírou třídění zůstane relativně lépe tříděna i po navýšení u ostatních frakcí. Klíčovým parametrem je průsečík s osou y , který umožňuje výpočet procentuálního navýšení třídění. Tento přístup zajišťuje, že výsledný scénář bude realistický a v souladu s definovanými omezeními.

Celý model včetně definice veličin, použitých rovnic a mezivýpočtů je podrobně popsán v následující kapitole, aby bylo možné lépe porozumět implementační části modulu, která je dále rozpracována v praktické části této práce.

1.4.1 Popis veličin

Pro správné fungování modelu je klíčové nastavit vstupní parametry na základě lokálních podmínek a odborného posouzení. Veškeré veličiny potřebné pro výpočet matematického modelu jsou uvedeny v tabulce 1.3.

1.4.2 Výpočty

Obtížnost zvýšení třídění odpadu

Tento parametr umožňuje modelu zohlednit rozdílnou obtížnost zvýšení míry třídění u různých frakcí odpadu a přizpůsobit výsledek specifickým podmínkám dané lokality.

Výpočet kritéria pro odpad

Kritérium pro jednotlivé recyklovatelné frakce komunálního odpadu je určeno jako podíl celkového potenciálu pro zvýšení třídění a vstupního parametru, který lze upravit na základě odborného posouzení. Matematický zápis je uveden v rovnici (1.1). Tato úprava umožňuje zohlednit obtížnost navýšení třídění, ale současně může ovlivnit celkové množství odpadu v rámci potenciálu. Protože model předpokládá uzavřený systém, kde tento objem odpadu představuje nové zdroje, může při hodnotách menších než jedna, dojít k jeho překročení.

$$G_i = \frac{m_i^{SKO} + m_i^{OBJ} + (1 - H)m_i^N}{F_i} \quad (1.1)$$

Výpočet Y-průsečíku

Průsečík lineární funkce je určen jako podíl požadovaného zvýšení míry třídění recyklovatelných frakcí a součtu kritérií G_i vypočtených pro jednotlivé frakce komunálního odpadu. Požadované zvýšení míry třídění je stanoveno jako součin cílové míry třídění a celkové produkce odpadu po zohlednění prevence jeho vzniku. Od této hodnoty se následně odečte současná produkce tříděného odpadu. Hodnota průsečíku K funguje jako měřítko, které je následně použito k výpočtu absolutního zvýšení třídění pro jednotlivé frakce odpadu. Matematický zápis je uveden v rovnici (1.2).

$$K = \frac{H \sum_{i \in I} ((1 - P_i)m_i^P) - \sum_{i \in I^R} m_i^P}{\sum_{i \in I^R} G_i} \quad (1.2)$$

Výpočet absolutního zvýšení produkce odpadu

Míra potřebného zvýšení třídění pro každou recyklovatelnou frakci je zapsána v rovnici (1.3). A je definována jako násobek průsečíku K a kritéria odpadu G_i . Tato míra vyjadřuje, jak velké zvýšení třídění je nutné pro dosažení požadované míry třídění.

$$N_i = G_i K \quad (1.3)$$

Rozdělení množství odpadu

Po určení absolutního zvýšení míry třídění N_i je dalším krokem rozdělení tohoto navýšení mezi SKO, OBJ a nové zdroje vzniklé v důsledku zavedení nových opatření. Tento krok je klíčový pro zajištění realistického scénáře, protože model musí respektovat fyzikální omezení daná maximálním potenciálem z existujících zdrojů definovaný jako $m_i^{SKO} + m_i^{OBJ}$. Rozdělení je zapsáno v rovnici (1.4). Podstatnou součástí je i expertní parametr X_i , který určuje poměr mezi tříděním z existujících zdrojů (SKO a OBJ) a nově vznikajícím odpadem.

1. **Vyšší hodnota X_i** – priorita je dána tříděním z již existujících zdrojů.
2. **Nižší hodnota X_i** – priorita je dána tříděním z nově vznikajících zdrojů.

$$T_i = \min \left(\frac{N_i X_i (m_i^{SKO} + m_i^{OBJ})}{X_i (m_i^{SKO} + m_i^{OBJ}) + (1 - H) m_i^N}, m_i^{SKO} + m_i^{OBJ} \right) \quad (1.4)$$

Rozdělení množství odpadu mezi SKO a OBJ

Jakmile je určeno celkové množství tříděného odpadu T_i , je nutné toto množství dále rozdělit mezi jednotlivé typy odpadu SKO a OBJ. Tento krok zahrnuje dva samostatné výpočty, které určují nejprve množství OBJ v rovnici (1.5) a následně jednoduchý dopočet pro SKO v rovnici (1.6). Kde důležitou součástí výpočtu je další expertní parametr Y_i , který určuje poměr mezi SKO a OBJ.

1. **Vyšší hodnota Y_i** – priorita je dána tříděním směsného odpadu.

2. **Nižší hodnota** Y_i – priorita je dána třídění objemného odpadu.

$$T_i^{OBJ} = \min \left(\frac{T_i m_i^{OBJ}}{Y_i m_i^{SKO} + m_i^{OBJ}}, m_i^{OBJ} \right) \quad (1.5)$$

$$T_i^{SKO} = T_i - T_i^{OBJ} \quad (1.6)$$

Rozdělení množství odpadu na nově vzniklý

Nyní ještě zbývá výpočet množství nově vzniklého odpadu z dodatečných opatření pomocí rovnice (1.7). Pokud by model nezohledňoval nové zdroje odpadu prostřednictvím vztahu s požadovanou mírou třídění, mohl by dojít k nerealistickému nadhodnocení zvýšení třídění. Mechanismus zahrnující faktor $1-H$ ve jmenovateli zajišťuje, že nově vzniklý odpad bude adekvátně zohledněn při výpočtu požadovaného zvýšení třídění.

$$T_i^N = \frac{N_i - T_i}{1 - H} \quad (1.7)$$

Výpočet výsledné produkce odpadu

Posledním krokem je výpočet celkového množství vytríděného odpadu pro každou jednotlivou frakci pomocí rovnice (1.8). Tento krok spojuje všechny předchozí části výpočtu – zahrnuje jak tříděný odpad získaný z SKO (rovnice (1.6)) a OBJ (rovnice (1.10)), tak i z nových zdrojů (rovnice (1.11)). Výsledná produkce odpadu zároveň zohledňuje existující míru prevence a respektuje fyzická omezení daná maximálním potenciálem třídění.

Základní logika tohoto kroku spočívá v tom, že skutečná produkce tříděného odpadu nesmí překročit maximální dosažitelnou hodnotu, která je definována součtem aktuální produkce, potenciálu třídění ze směsného a objemného odpadu a potenciálu z nových opatření.

$$W_i = \min \left((1 - P_i) m_i^P + T_i^{SKO} + T_i^{OBJ} + T_i^N, m_i^P + m_i^{SKO} + m_i^{OBJ} + m_i^N \right) \quad (1.8)$$

$$W_{SKO} = (1 - P_{SKO})m_{OBJ}^P - \sum_{i \in I^R} T_i^{SKO} \quad (1.9)$$

$$W_{OBJ} = (1 - P_{OBJ})m_{OBJ}^P - \sum_{i \in I^R} T_i^{OBJ} \quad (1.10)$$

$$W_{OW} = (1 - P_{OW})m_{OW}^P \quad (1.11)$$

Tabulka 1.3 Popis matematického modelu

Množiny	Popis
$i, j \in I$	Množina katalogových čísel KO
$I_R \subset I$	Podmnožina katalogových čísel KO, které představují recyklovatelné složky
$I_M \subset I$	Podmnožina katalogových čísel KO, které představují ostatní složky
Konstanty	Popis
p_i^x	5.-100. percentil produkce odpadu i na obyvatele, [t/1000 obyvatel]
S_i^{SKO}	Podíl odpadu i v SKO, [-]
S_i^{OBJ}	Podíl odpadu i v OBJ, [-]
Parametry	Popis
F_i	Obtížnost zvýšení třídění odpadu i , [-]
H	Požadovaná míra třídění, [-]
m_i^P	Skutečná produkce odpadu i ze všech sběrných systémů, [t]
m_i^{SKO}	Potenciál zvýšení třídění odpadu i z SKO ($A_i + B_i$), [t]
m_i^{OBJ}	Potenciál zvýšení třídění odpadu i z OBJ (C_i), [t]
m_i^N	Potenciál zvýšení třídění odpadu i z dodatečných opatření (D_i), [t]
P_i	Procentní míra předcházení odpadu, [-]
X_i	Poměr mezi tříděním a novým zdrojem způsobeným dodatečnými opatřeními, [-]
Y_i	Poměr mezi tříděním SKO a OBJ, [-]
Proměnné	Popis
G_i	Kritérium pro odpad i vstupující do výpočtu y-průsečíku lineární funkce, [-]
K	Y-průsečík lineární funkce, [-]
N_i	Absolutní zvýšení produkce odpadu i potřebné k dosažení požadované míry třídění, [t]
Q_i	Dodatečné předcházení odpadu i k dosažení požadované míry třídění bez překročení definovaných potenciálů, [t]
T_i	Množství odpadu i získané tříděním SKO nebo OBJ, [t]
T_i^{SKO}	Množství odpadu i získané tříděním SKO, [t]
T_i^{OBJ}	Množství odpadu i získané tříděním OBJ, [t]
T_i^N	Množství odpadu i získané z dodatečných opatření z nových zdrojů, [t]
W_i	Konečné množství odpadu i potřebné k dosažení požadované míry třídění, [t]

(zdroj: vlastní)

2 WEBOVÁ APLIKACE

Jak již bylo uvedeno v úvodu, aplikace je určena k hodnocení úrovně separace komunálního odpadu na základních územních jednotkách České republiky. Analýza odpadového hospodářství patří na Ústavu procesního inženýrství VUT v Brně k dlouhodobě řešeným tématům. Postupem času se k prvním vyvinutým algoritmům začaly přidávat další funkcionality, avšak původní řešení založené na formuláři v Excelu začalo narážet na své funkční limity. Tento přístup nejenže omezoval možnosti rozšiřování analytických nástrojů, ale rovněž nevyhovoval požadavkům na uživatelskou přívětivost a efektivní práci s daty.

Vzhledem k rostoucímu objemu informací, které je nutné zpracovávat a analyzovat, bylo nezbytné přejít na robustnější systém, který by umožňoval efektivní správu dat, vizualizaci výsledků a lepší uživatelský komfort. Proto padlo rozhodnutí vyvinout webovou aplikaci, která integruje několik modulů s různými funkcionalitami do moderního a intuitivního uživatelského rozhraní. Webová platforma umožňuje centralizované zpracování velkého množství dat, jejich průběžnou aktualizaci a přístup k analýzám odkudkoliv, čímž významně zvyšuje efektivitu práce s daty v oblasti odpadového hospodářství.

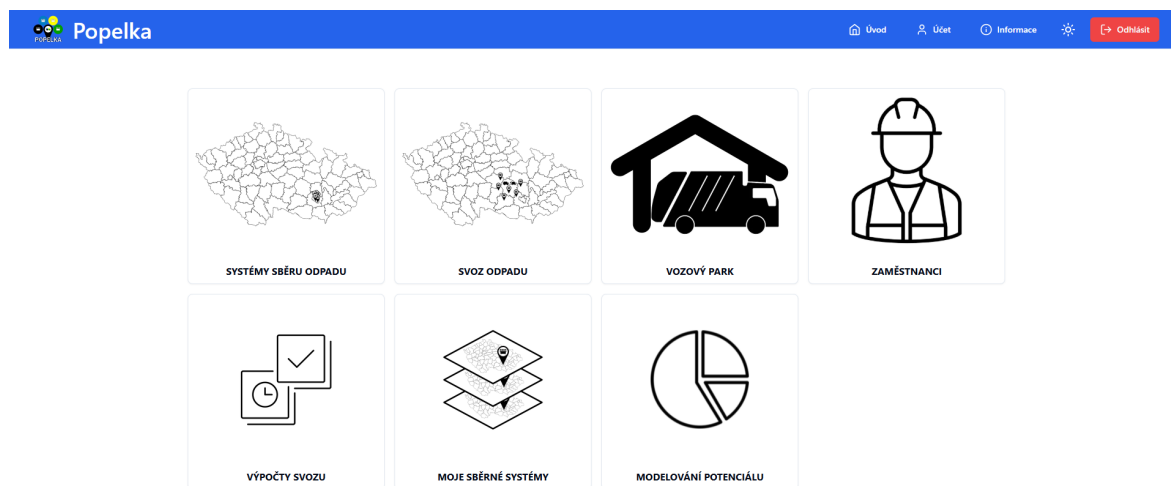
Tato změna přinesla řadu výhod, včetně lepší optimalizace výpočetních procesů, snadnější integrace nových funkcionalit a možnosti přizpůsobit aplikaci specifickým potřebám uživatelů. Díky implementaci pokročilých analytických nástrojů nyní systém umožňuje nejen sledování produkce a separace odpadu, ale také optimalizaci svozových tras, vyhodnocování docházkových vzdáleností k odpadovým nádobám či prediktivní modelování budoucích trendů v oblasti nakládání s odpady. Webová aplikace tak nejen překonává limity původního řešení, ale zároveň poskytuje moderní a efektivní nástroj pro řízení odpadového hospodářství na úrovni obcí a měst.

Přechod ve vývoji na webovou aplikaci, ale vyžadoval pečlivý výběr vhodných nástrojů pro BE i FE, které by umožnily škálovatelnost, rychlou odezvu systému a snadnou integraci dalších funkcionalit v budoucnu. Použité technologie a knihovny budou popsány v následujících kapitolách.

2.1 Moduly a jejich funkcionality

Jak již bylo řečeno, dlouhodobý vývoj aplikace přináší řadu vylepšení nejen na funkční úrovni aplikace, ale také té vizuální. Ta se vyvíjí a přizpůsobuje ruku v ruce té funkční.

Díky tomu a přechodu na moderní technologie se podařilo převést uživatelské prostředí do moderního responzivního designu. Hlavní stránka, která je dostupná po přihlášení, obsahuje dlaždicový seznam všech dostupných modulů a její aktuální podoba je zobrazena na obrázku 2.1.



Obrázek 2.1 Hlavní stránka aplikace se seznamem modulů

(zdroj: vlastní)

Přehled všech modulů s jejich funkcionalitou je popsán níže.

- **Systémy Sběru Odpadu** – umožňuje nastavení a správu systémů sběru odpadu pro jednotlivé části obce či nastavení jejich shluků. Dále je zde možné vytvářet nové sběrné systémy, spravovat adresní body a neaktivní místa, stahovat a nahrávat data pro sběrná místa.
- **Svoz Odpadu** – umožňuje konfiguraci a výpočet trasy včetně rozvrhu svozu odpadu na základě zadaných parametrů. Další možností je nastavení pravidel pro výpočet svozu, výběr vhodného vozidla a definice typu odpadu. Nechybí ani správa a zobrazení map s vybranými obcemi a zařízeními, stejně jako kontrola navržených tras a možnost jejich případné úpravy.
- **Vozový Park** – umožňuje správu vozového parku a jejich specifická nastavení týkající se sběru odpadu. Samozřejmostí jsou standardní funkcionality pro přidání, editaci nebo smazání vozidla. Z dalších parametrů vozidla je pak možné definovat například jeho kapacitu převáženého odpadu nebo typ daného převáženého odpadu.

- **Zaměstnanci** – umožňuje správu dat zaměstnanců, jejich přiřazování k vozidlům a generování záznamů o provozu. Stejně jako u předchozího modulu i zde se správa jednotlivých záznamů provádí přidáním, editací nebo smazáním zaměstnance.
- **Výpočty svozu** – umožňuje zobrazení a správu všech výpočtů souvisejících se svozem odpadu.
- **Moje Sběrné Systémy** – umožňuje konfiguraci a správu specifických sběrných systémů.
- **Modelování potenciálu** – umožňuje analýzu a správu sběrné infrastruktury odpadu pomocí modelace změny v odpadovém sběrném systému na základě stanoveného potenciálu.

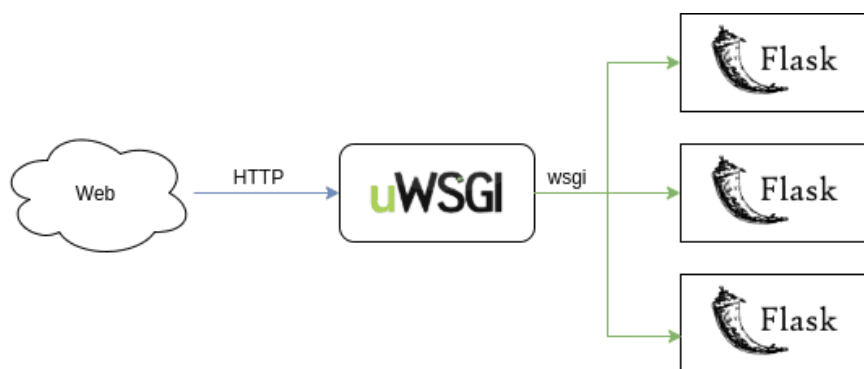
2.2 Backend

Klíčovou vrstvou celého systému, která zajišťuje veškerou logiku, správu dat a komunikaci mezi uživatelským rozhraním a databázovým úložištěm je BE, taktéž nazývaný jako serverová část. Konkrétně se BE stará o zpracování požadavků z FE, provádění výpočtů, validaci dat a jejich ukládání či načítání z databáze. Správně navržený BE zajišťuje stabilní a bezpečný provoz aplikace, efektivní správu zdrojů a škálovatelnost systému podle aktuálních potřeb. [12]

Pro vývoj backendové části této aplikace byl zvolen programovací jazyk Python, který nabízí jednodušší a přímočarou syntaxi. K implementaci webového rozhraní serveru byla využita knihovna Flask. Ta poskytuje flexibilní a lehký framework s důrazem na modularitu a jednoduchost. Flask umožňuje efektivní správu API rozhraní a podporuje integraci s dalšími službami potřebnými pro provoz aplikace jako jsou například databázové systémy. Jejich zástupcem byla pro tento projekt vybrána relační databáze MariaDB, která slouží pro ukládání perzistentních dat a databáze Redis pro ukládání dočasných dat. Celá aplikace nasažena na webovém serveru NGINX doplněna o aplikační server uWSGI jejichž spojení je na obrázku 2.2. [13]

2.2.1 Python

Python je interpretovaný, vysoce úrovněový a multiplatformní programovací jazyk, vyvíjený jako open-source, který se vyznačuje čitelnou syntaxí, rozsáhlou knihovnou modulů a sil-



Obrázek 2.2 Kombinace spojení webové aplikace, NGINX a uWSGI

(zdroj: 14)

nou komunitní podporou. Díky tomu je široce využíván nejen pro rychlý vývoj aplikací, ale i jako skriptovací nástroj pro propojení různých softwarových komponent. Jeho flexibilita umožňuje snadné nasazení na různých operačních systémech bez nutnosti výrazných úprav kódu.

Jedním z hlavních důvodů popularity Pythonu je efektivní cyklus vývoje, který zahrnuje rychlé úpravy, testování a ladění kódu. Absence kompilace zkracuje dobu mezi změnou a jejím otestováním, což vývojářům umožňuje pracovat efektivněji. Python navíc podporuje dynamickou abstrakci a objektově orientovaný přístup, čímž poskytuje silné nástroje pro organizaci a znovupoužitelnost kódu. [15]

2.2.2 Flask

Flask je mikro open-source webový framework pro Python, který umožňuje vytvářet webové aplikace s minimálním množstvím kódu. Je postaven na standardu WSGI (Web Server Gateway Interface), což je specifikace definující rozhraní mezi webovými servery a webovými aplikacemi v jazyce Python. Jinými slovy definuje způsob zpracování HTTP požadavků a odpovědí. Flask dále poskytuje základní nástroje pro směrování URL, podporu pro šablony a další funkce potřebné pro vývoj webových aplikací. Jeho flexibilita a modularita umožňují přidávat pouze ty komponenty, které jsou zapotřebí, což vede k lehčím a efektivnějším aplikacím. [16]

Na druhou stranu právě pro svůj minimalismus je Flask spíše vhodný pro menší projekty nebo jako testovací webový server. Pro produkční nasazení se kvůli absenci komplexního řešení, například z hlediska bezpečnosti, doporučuje kombinace s jinými specializovanými

aplikačními servery, aby poskytl pokročilé funkce, které Flask sám o sobě nenabízí. [17]

2.2.3 MariaDB

MariaDB je open-source relační databázový systém, který vznikl jako odnož MySQL po jeho akvizici společností Oracle. Cílem MariaDB bylo zachovat kompatibilitu s MySQL a zároveň rozšířit její možnosti o nové funkce a lepší výkon. Dnes představuje moderní, výkonnou a bezpečnou alternativu k MySQL s otevřeným vývojovým modelem a širokou komunitní podporou, kde mezi její nejvýznamnější uživatele patří společnosti jako Wikipedia, Google nebo Samsung. [18]

Databázový systém (DBMS) je software určený k ukládání, správě a manipulaci s daty. Poskytuje nástroje pro definici struktury databáze, řízení přístupu k datům a optimalizaci dotazů. Mezi nejčastější typy DBMS patří relační databázové systémy (RDBMS). [19]

RDBMS jsou základem pro ukládání a správu strukturovaných dat v mnoha aplikacích. Organizují data do tabulek, kde každý řádek představuje záznam a každý sloupec atribut tohoto záznamu. Tento model umožňuje efektivní správu velkého množství informací. Pro zajištění integrity a konzistence dat se v relačních databázích používají klíče.

Primární klíč je jednoznačný identifikátor záznamu v tabulce, který nesmí obsahovat prázdnou hodnotu (NULL) a musí být unikátní pro každý záznam. Cizí klíč definuje vztah mezi dvěma tabulkami tak, že hodnota v určeném sloupci jedné tabulky musí existovat jako primární klíč v jiné tabulce. Tím je zajištěna referenční integrita mezi tabulkami. Pro správu a manipulaci s daty slouží standardizovaný programovací jazyk (SQL), který umožňuje uživatelům definovat, manipulovat a dotazovat se na data v databázi pomocí základních operací, jenž jsou často označovány akronymem CRUD. Společným rysem všech operací je nutnost explicitního určení cílové tabulky. [20][19]

- **Create (Vytvořit)** – vkládání nových záznamů do tabulky se provádí pomocí příkazu INSERT. Pokud není vkládána hodnota do všech sloupců tabulky, je nezbytné výslovně specifikovat seznam cílových sloupců, do kterých budou odpovídající hodnoty vloženy. V případě hromadného vkládání více řádků v rámci jediného dotazu je možné seznam sloupců uvést pouze jednou, přičemž jednotlivé záznamy jsou následně zadávány jako oddělené skupiny hodnot.
- **Read (Číst)** – výběr záznamů z tabulky se provádí pomocí příkazu SELECT. Ačko-

liv je možné výběr všech sloupců provést pomocí znaku *, je všeobecně doporučováno pokaždé sloupce explicitně vyjmenovat například z hlediska lepší transparentnosti [21]. Pro selektivní filtrování dat na základě definovaných podmínek se využívá klauzule WHERE, která umožňuje omezení výsledné množiny záznamů dle zadaných kritérií. Dále příkaz podporuje pokročilé operace, jako je seskupování dat (GROUP BY), řazení výsledků (ORDER BY) nebo omezování počtu vrácených řádků (LIMIT).

- **Update (Aktualizovat)** – aktualizace existujících záznamů v tabulce se provádí pomocí příkazu UPDATE. Pro modifikaci dat ve specifikovaných sloupcích, je nezbytné použít klauzuli SET, ve které jsou definovány nové hodnoty pro jednotlivé sloupce. Aby se zabránilo neúmyslné změně všech záznamů v tabulce, je doporučeno vždy specifikovat klauzuli WHERE, která určuje konkrétní podmínky pro výběr aktualizovaných řádků.
- **Delete (Smazat)** – odstranění existujících záznamů z tabulky se provádí pomocí příkazu DELETE. Pro selektivní odstranění řádků, je nutné definovat podmínky v klauzuli WHERE, která určuje konkrétní sadu záznamů určených k odstranění. Pokud klauzule WHERE není uvedena, dojde k odstranění všech záznamů v dané tabulce. V případě potřeby úplného vyprázdnění tabulky lze alternativně použít příkaz TRUNCATE, který je výkonnější, protože neprovádí jednotlivé mazací operace pro každý záznam, ale odstraní veškerý obsah tabulky najednou.

2.2.4 Redis a Redis Queue

Redis je open-source úložiště dat uchovávaných v paměti, které se nejčastěji využívá jako distribuovaná mezipaměť. Redis je známý jako NoSQL databáze a jeho úložiště má podobu klíč-hodnota. Jelikož ukládá data v paměti, zajišťuje vysokou rychlost, spolehlivost a výkon. Tento přístup umožňuje efektivně zpracovávat velké objemy dat v reálném čase, což jej činí ideální volbou pro caching databázových dotazů, minimalizaci zatížení relačních databází a zrychlení odpovědí na často opakované dotazy. [22]

Redis podporuje mechanismus front (Redis Queue), který umožňuje implementaci asynchronního zpracování úloh. Jedná se o Python knihovnu, která poskytuje efektivní mechanismus pro zařazování úloh do fronty a jejich následné zpracování na pozadí. Tento přístup je široce využíván v systémech, kde Redis slouží jako prostředník mezi komponentami apli-

kace. [23]

2.2.5 NGINX

NGINX je open-source určený pro webový server, reverzní proxy, caching a load balancing. Je široce využíván pro distribuci webových aplikací. Díky své asynchronní, Master-Slave událostmi řízené architektuře dokáže efektivně obsluhovat velké množství současných připojení s minimálními nároky na systémové prostředky. Tento přístup mu umožňuje poskytovat vyšší stabilitu a předvídatelný výkon i při vysokém zatížení, což je klíčová výhoda oproti tradičním vícevláknovým serverům, jako je Apache. NGINX je navíc optimalizován pro zpracování statického obsahu, vyvažování zátěže a práci s moderními webovými technologiemi, jako jsou HTTP/2, WebSockets, TLS/SSL nebo i POP3 a SMTP. [24]

2.2.6 uWSGI

uWSGI je open-source aplikační server, který slouží jako most mezi webovými servery, jako je Nginx, a backendovými aplikacemi, zejména těmi postavenými na Pythonu pomocí rozhraní WSGI. Díky své modulární architektuře podporuje širokou nabídku protokolů, včetně HTTP a umožňuje škálovatelné nasazení webových aplikací s optimalizovaným využitím systémových zdrojů. Díky Master-Worker architektuře dokáže spustit několik instancí jedné aplikace, mezi které pak rozděluje příchozí uživatelské požadavky. [25]

2.3 Frontend

Frontend, taktéž nazývaný jako klientská část, představuje uživatelské rozhraní webové aplikace, které je zodpovědné za interakci mezi uživatelem a backendovým systémem. Zajišťuje vizualizaci dat, dynamické prvky a uživatelskou přívětivost aplikace. Pro vývoj frontendové části byl zvolen React.js ve spojení s TypeScriptem pro typovou kontrolu. React je moderní knihovna pro tvorbu komponentově orientovaných uživatelských rozhraní. Stylování aplikace je realizováno pomocí Tailwind CSS, což je moderní utility-first framework, který umožňuje rychlé a flexibilní definování stylů přímo v komponentách.

Aplikace komunikuje s backendem prostřednictvím REST API, zatímco správu globálního stavu zajišťuje Redux, v kombinaci s React Redux a Redux Toolkit, které umožňují efektivní centralizované řízení dat v aplikaci. Pro navigaci mezi jednotlivými částmi aplikace je využit React Router, jenž zajišťuje plynulou navigaci mezi stránkami. Důležitou

součástí aplikace je také vizualizace geografických dat, kterou zprostředkovává knihovna Leaflet, používaná k vykreslování interaktivních mapových prvků. [13][26]

2.3.1 React

React je open-source JavaScript knihovna vyvinutá společností Facebook (nyní Meta). Na rozdíl od frameworků, jako je například Angular, které poskytují komplexní řešení pro strukturu aplikace a často implementují architekturu Model-View-Controller (MVC), se React zaměřuje pouze na View část této architektury. To znamená, že React nenutí specifickou strukturu aplikace, což poskytuje větší flexibilitu při výběru dalších nástrojů a knihoven podle potřeb projektu. [27]

Hlavním rysem Reactu je komponentový přístup, kde uživatelské rozhraní je rozděleno do jednotlivých, znovupoužitelných částí. Tyto komponenty mohou být definovány jako funkční nebo třídové. Funkční komponenty jsou jednoduché funkce, které přijímají vstup (props) a vracejí React elementy. Třídové komponenty jsou založeny na ES6 třídách a poskytují více funkcionalit. V současnosti jsou více preferovány funkční komponenty, protože v nich je možné využít tzv. React Hooks. Jedná se o speciální funkce, umožňující připojit se k životnímu cyklu komponenty a spravovat tak její stav nebo vedlejších efekty. Při volání Hooks, je však důležité dodržovat 2 základní pravidla.

1. **Volání Hooks pouze na nejvyšší úrovni** – tedy na nejvyšší úrovni funkce komponenty, nikoli uvnitř smyček, podmínek nebo vnořených funkcí.
2. **Volání Hooks pouze z funkcionálních komponent Reactu** – případně z vlastních Hooks, nikoli z běžných funkcí JavaScriptu nebo třídových komponent.

K vykreslování React využívá JSX (JavaScript XML), jedná se o syntaxi kombinující JavaScript s HTML strukturou. JSX usnadňuje práci s uživatelským rozhraním, protože umožňuje psát HTML-like kód přímo v JavaScriptu, přičemž každý JSX element je převeden na volání funkce `React.createElement()`.

Pro optimalizaci výkonu využívá React virtuální Document Object Model (DOM). Což je kopie skutečného DOM, která je uložena v paměti a umožňuje rychlé provádění změn bez přímé interakce se skutečným DOM. Při změně stavu aplikace React nejprve vytvoří novou verzi virtuálního DOM, kterou následně porovná s předchozí verzí a provádí pouze

diferenciální aktualizace změněných částí DOM. Tento přístup eliminuje nutnost překreslování celého dokumentového modelu, což je výpočetně náročný proces. Dalším důležitým aspektem DOM je jeho schopnost standardizovat názvy a vlastnosti událostí napříč různými webovými prohlížeči, což zajišťuje konzistentní chování aplikace. [28][29]

2.3.2 React Router

React Router je knihovna pro směrování (routing) v React aplikacích. V tradičních webových aplikacích každá změna URL adresy vede k odeslání nového HTTP požadavku na server, který vrátí kompletní HTML stránku. Oproti tomu, Redux Router využívá klientského směrování, které umožňuje měnit URL adresu bez nutnosti znovunačtení celé stránky ze serveru. A to pomocí nástrojů pro definování více cest (routes) a komponent, které se mají renderovat při odpovídajících URL, což zajišťuje konzistentní a intuitivní navigaci v aplikaci. [30]

2.3.3 Tailwind CSS

Tailwind CSS je moderní utility-first framework, který poskytuje sadu předem definovaných CSS tříd pro rychlou a flexibilní tvorbu uživatelských rozhraní. Na rozdíl od tradičního přístupu k psaní CSS, kde se vytvářejí samostatné stylové soubory s třídami aplikovanými na HTML prvky, Tailwind umožňuje aplikovat styly přímo v HTML pomocí utility tříd, což eliminuje nutnost psát vlastní CSS a urychluje vývoj. Díky Just-In-Time (JIT) kompilaci generuje pouze potřebné styly, čímž minimalizuje velikost finálního CSS souboru a zlepšuje výkon aplikace. [31]

2.3.4 Zod

Zod je moderní knihovna pro validaci datových struktur, navržená pro integraci s TypeScriptem a vysokou typovou bezpečnost. Hlavním přínosem je možnost deklarovat typové schéma a validační logiku na jednom místě, bez nutnosti duplikace typů.

Schémata v Zodu jsou definována pomocí kompaktní a srozumitelné syntaxe, přičemž knihovna podporuje široké spektrum datových struktur – od jednoduchých primitiv až po složené objekty, zanořená pole či výčtové typy. Samozřejmostí je možnost vytvářet i pokročilá validační pravidla, včetně podmíněných kontrol, transformací hodnot nebo přizpůsobení chybových hlášení. [32]

2.3.5 Redux, React Redux, Redux Toolkit a Redux DevTools

Redux je knihovna, která slouží jako správa globálního stavu JavaScriptové aplikace. Je založena na principu jednosměrného toku dat a centralizovaného úložiště. Tím nabízí možnost, jak mohou jednotlivé komponenty bezpečně přistupovat a měnit stav aplikace. [33]

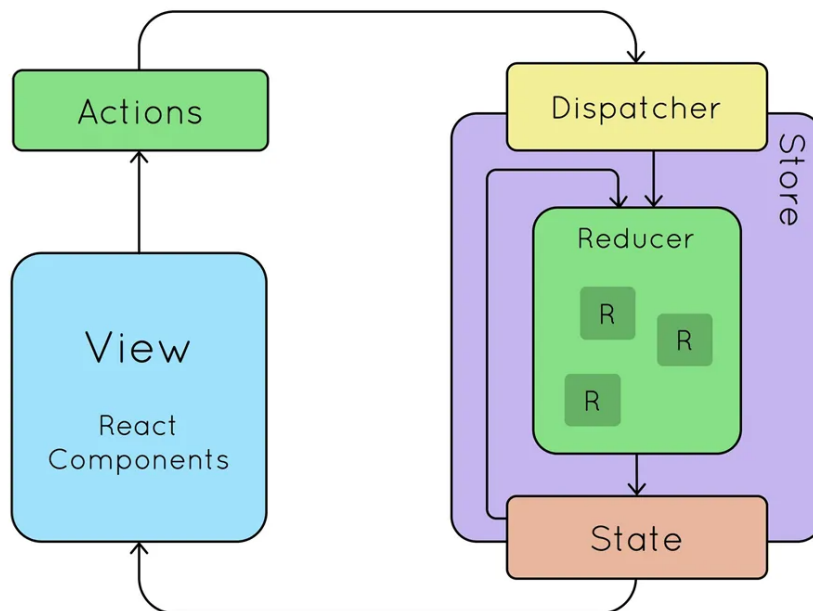
Způsob fungování nejlépe vystihují následující základní principy. [34]

1. **Jednoznačný zdroj pravdy** - globální stav aplikace je uložen ve storu. Tento přístup usnadňuje tvorbu univerzálních aplikací, protože stav ze serveru lze serializovat a následně na klientovi deserializovat bez nutnosti dalšího kódování.
2. **Stav je pouze pro čtení** - Jediný způsob, jak změnit stav, je vyslat action, tím je zajištěno, že ani pohledy, ani síťová volání nemohou přímo měnit stav. Všechny změny jsou centralizovány a probíhají v přísném pořadí, což eliminuje potenciální přepisování si stavu.
3. **Změny jsou prováděny pouze funkcemi** - Pro specifikaci, jak je stav transformován, se používají čistě reducers. Ty pak vrací nové objekty stavu místo mutace předchozího stavu.

Na obrázku 2.3 je zachycena architektura Reduxu, která se skládá z několika klíčových komponent, reflektujících základní principy. [35]

1. **Store** – jediné úložiště stromové struktury, které centralizuje stav aplikace a poskytuje metody pro přístup k tomuto stavu a odesílání akcí.
2. **Action** – je obyčejný JavaScriptový objekt, který popisuje událost, jež by měla způsobit změnu stavu aplikace. Každá akce musí mít vlastnost type, která určuje typ akce, a může obsahovat další data ve vlastnosti payload.
3. **Reducer** – obyčejná funkce, která přijímá předchozí stav a akci, na jejímž základě pak určuje, jak bude výsledný stav aplikace transformován.
4. **Slice** – je koncept zavedený v Redux Toolkit, který umožňuje rozdělit stav aplikace na menší části. Jedná se o nástrojovou sadu s funkcemi, kde každý slice obsahuje počáteční stav, reducers a akce pro danou část stavu. Tento přístup zjednodušuje správu

stavu tím, že seskupuje související logiku na jednom místě, což vede k lepší organizaci a čitelnosti kódu. [36]



Obrázek 2.3 Princip Redux pipeline

(zdroj: 37)

Hlavní výhody Reduxu by se daly shrnout do následujících bodů. [38]

1. **Předvídatelnost** – díky centralizovanému a neměnnému stavu je chování aplikace konzistentní a snadno testovatelné.
2. **Centralizace** – stav aplikace je uložen na jednom místě.
3. **Laditelnost** – nástroje jako Redux DevTools umožňují sledovat, kdy, kde, proč a jak se stav aplikace změnil, což výrazně usnadňuje ladění a sledování změn stavu. Redux DevTools je rozšířením prohlížeče, které stačí například pro Google Chrome nainstalovat klasicky přes Chrome Web Store.
4. **Flexibilita** – Redux funguje s jakoukoli UI vrstvou a má rozsáhlý ekosystém doplňků, které lze přizpůsobit potřebám vývoje.

II PRAKTICKÁ ČÁST

3 ANALÝZA STÁVAJÍCÍHO ŘEŠENÍ

Stávající modul je rozdělen do několika sekcí, přičemž základní struktura se skládá ze dvou hlavních částí. První část představuje stránku s obecnými informacemi, zatímco druhá část obsahuje formulář určený pro modelaci potenciálu. Toto rozdělení se odráží i ve vizuální podobě submenu, kde je položka pro obecné informace zarovnána vlevo, zatímco ostatní položky jsou zarovnány vpravo. Jednotlivé sekce modulu jsou pak popsány a analyzovány níže.

- **Obecné informace** – po přechodu do modulu Modelování potenciálu se uživateli zobrazí základní informace o samotném produktu a možnosti stažení původního Excel souboru, z něhož modul vychází. Informace jsou graficky rozděleny do několika bloků znázorněných jako kartičky. Ačkoliv nápad takového uspořádání není špatný, jeho provedení je trochu nešťastné. Nejen, že kartičky mají různou velikost, ale jejich pozice nadržuje žádný formát. Každá je tedy v jiné horizontální i vertikální úrovni. Což bohužel působí trochu chaoticky a kazí to první dojem po příchodu do samotného modulu.
- **Konfigurace** – je vstupní sekce celého formuláře a nástroje jako takového. Rozdělení částí je opět realizováno pomocí grafické reprezentace kartiček. Bohužel i zde bez větší koncepce nějakého jednotného formátování. Nejdůležitější částí je formulářový prvek parametrů obce, bez jehož vyplnění není možné překlíknout na další sekce formuláře. Do další části je možné doplnit volitelné informace, s tím, že každý vstup musí být ručně potvrzen. Zadané vstupy se pak trochu zbytečně duplicitně zobrazují vedle v samostatné kartě s možností jejich resetu.
- **Vstupy - Produkce Odpadu Obce** – jak už název sekce napovídá, slouží pro zadání veškerých vstupních dat, týkajících se odpadové produkce. Celá sekce je pak ještě rozdělena do čtyř podsekcí, které reprezentují kategorie odpadů definovaných v kapitole 1.2. Všechny podsekce mají stejné rozvržení, kde v levé polovině se nachází tabulka pro zadání objemu každého typu odpadu. V pravé části je pak text s informacemi a definicí dané kategorie odpadu.
- **Modelování** – hlavní sekce a jádro celého modulu. V této části jsou zobrazeny vý-

sledky aktuálního třídění odpadu na základě dat vložených v předchozí sekci. Po vizuální stránce je obsah rozdělen do tří bloků, kde prostřední z nich slouží pro simulaci a sledování změn ve vývoji výsledného třídění odpadu. Bohužel i zde není grafické zpracování úplně ideální. Vhodnost zvoleného barevného pozadí daných prvků zůstává na individuálním posouzení. Naproti tomu, nesoulad v úrovni jednotlivých řádků tabulky posledního bloku vůči předchozím dvěma, vytváří dojem nesourodosti.

- **Export** – samostatná sekce, kde se v podstatě nachází pouze tlačítko pro export vypočítaných dat do Excel souboru.

Přestože je uživatelské rozhraní modulu logicky rozvrženo, z analýzy vyplývá několik oblastí, které by bylo vhodné optimalizovat. Tyto úpravy by mohly zlepšit uživatelskou zkušenost a zvýšit efektivitu práce s modulem. Níže je uveden seznam klíčových návrhů na zlepšení.

- **Vizuální uspořádání** – sjednocení velikosti kartiček, případně jejich odstranění, a zarovnání prvků v jednotlivých sekcích.
- **Barevné schéma a grafické prvky** – sladění barevného schématu, úprava kontrastu a velikosti textu. Ačkoliv celá aplikace podporuje tmavý režim, styly jednotlivých prvků modulu tomu nejsou plně adaptovány, a dochází tak například ke změně barvy textu na světlou se zachováním světlým pozadím, a tím k nečitelnosti daných textů.
- **Kontrola vstupů** – zlepšení validace vstupních dat a zajištění konzistence. Například v modelaci formulářový prvek pro zadání procenta inkrementuje hodnotu po tisících, což okamžitě zobrazí chybovou hlášku.
- **Přeskupení rozvržení** – nabízí se otázka, jestli je podmíněné vyplnění konfigurace pro aktivaci zbylých sekcí nutné. Nebo zda by jen stačilo zbylé sekce nainicializovat s defaultními hodnotami. Stejně tak, stojí za zvážení, zda sekce pro zadání vstupů produkce odpadů, musí být ještě rozdělena na další subsekce.

4 NÁVRH ŘEŠENÍ

Návrh na rozšíření modulu webové aplikace probíhal v několika krocích. Na základě zadání diplomové práce, diskuzí s vedoucími diplomové práce a pracovníky ÚPI byly nejprve definovány klíčové funkční požadavky. Následně byl vypracován koncepční návrh modulu, který popisuje strukturu a uspořádání jednotlivých částí modulu. Tento návrh vychází ze stávajícího řešení, které bylo analyzováno v předchozí kapitole, a reflektuje identifikované nedostatky a oblasti pro zlepšení.

4.1 Funkční a nefunkční požadavky

Vzhledem k tomu, že na modul není vázáno žádné speciální oprávnění, je přístupný všem registrovaným uživatelům. V návrhu tedy vystupuje pouze jeden aktér – uživatel, pro něhož jsou nové funkční požadavky vyjmenovány níže.

- Jednotlivé sekce budou přístupné bez nutnosti vyplnění předchozí sekce.
- Parametry musí být validovány a uloženy hned po zadání bez manuálního potvrzení.
- Modul musí umožnit uložení a načtení parametrů pro budoucí použití.
- Systém musí zobrazit potvrzení o úspěšném uložení nebo chybovou hlášku při selhání.
- Nová sekce bude umožňovat zadání požadované míry separace a dopočítá potřebnou hodnotu zvýšení potenciálu pro její dosažení.
- Výsledky výpočtů musí být vizuálně prezentovány v přehledné tabulce.

Nefunkční požadavky zůstávají shodné s požadavky definovanými pro stávající aplikaci. Tento návrh nový modul z hlediska nefunkčních požadavků nijak nerozšiřuje, proto nejsou nefunkční požadavky v této části detailně rozpracovány.

4.2 Koncepční návrh modulu

Nový návrh je založen na třívrstvé architektuře, která zahrnuje FE, BE a databázovou vrstvu. V rámci návrhu bylo provedeno přepracování struktury modulu s cílem usnadnit navigaci, zvýšit efektivitu práce s daty a zlepšit uživatelskou zkušenost.

4.2.1 Backend

BE je implementován pomocí frameworku Flask v programovacím jazyce Python. Důvody pro volbu této technologie byly podrobně popsány v teoretické části. Struktura BE je navržena v souladu s principy vývoje Flask aplikací. Hlavní soubor `app.py` slouží jako vstupní bod aplikace. Jednotlivé endpointy jsou definovány v samostatných souborech podle jejich logických celků umístěných ve složce `api`. Toto rozdělení zajišťuje přehlednost a usnadňuje údržbu a rozšiřování systému.

Flask komunikuje s FE prostřednictvím REST API. Jednotlivé endpointy přijímají požadavky ve formátu HTTP requestů (např. GET, POST, PUT, DELETE), zpracovávají je a vracejí odpovědi ve formátu JSON. K serializaci dat je použita funkce `jsonify` nebo `orjsonify`, což zajišťuje konzistentní formát odpovědí a efektivní přenos dat. Tento přístup umožňuje snadnou integraci s frontendovou částí aplikace a jednotný způsob zpracování dat.

V rámci implementace byly nové endpointy navrženy s jednotnou strukturou v samostatném souboru v souladu s aktuálním uspořádáním. Základem každého endpointu je definice cesty, metody a funkce, která přichází požadavek zpracovává. Pro definici endpointů se ve Flasku používají dekorátory. Dekorátor `@app.route()` umožňuje definovat, na jakou HTTP metodu bude endpoint reagovat. Kromě toho může být v dekorátoru definována cesta pomocí parametru. Dekorátor zároveň umožňuje provést kontrolu přihlášení uživatele, v případě, že uživatel není ověřen, bude přesměrován na přihlašovací stránku.

V aplikaci je implementován vlastní dekorátor `@point()` viz ukázka v kódu 4.1, který rozšiřuje základní funkčnost o kontrolu oprávnění na základě zadaného názvu. Tento dekorátor vytváří interní slovník, kde je klíčem název funkce a hodnotou samotná funkce. Po provedení přidružené funkce je odpověď generována pomocí modulu umístěného ve složce `utils`, který obaluje funkcionalitu knihovny `orjson` a funkci `make_response` z Flasku. Tento přístup zjednodušuje vytváření odpovědí z API pointů, protože umožňuje efektivní serializaci dat do formátu JSON jediným voláním funkce, čímž se eliminuje nutnost opakovaného psaní kódu pro zpracování odpovědí. Pro komunikaci s databází je v aplikaci implementován modul konektoru, který zjednodušuje proces připojení k databázi. Tento modul je realizován jako třída obalující funkce knihovny `pymysql`.

```
1 @point()
2 def getSomeData(parametry):
```

```
3  # Vykonání potřebného kódu
4  data = db.query("query request")
5
6  # Navrácení dat s příznakem úspěšného vyřízení dotazu
7  return utils.json.orjsonify({'success': True, 'data':
    data})
```

Výpis 4.1 Schéma endpointu (zdroj: vlastní)

4.2.2 Frontend

FE je postaven na knihovně React, přičemž jako programovací jazyk je použit TypeScript namísto standardního JavaScriptu s tím, že komponenty budou implementovány podle zavedené struktury v rámci React ekosystému. TypeScript je nadstavbou jazyka JavaScript, což znamená, že veškerý kód se musí zkompilevat zpátky do JavaScriptu. Na druhou stranu, TypeScript přináší výhody v podobě statické typové kontroly, která umožňuje odhalit chyby již v době kompilace a přináší lepší čitelnost kódu. Předem nadefinované typy mohou být kromě proměnných a parametrů i návratové hodnoty funkcí viz ukázka v kódu 4.2. Díky tomu je na první pohled jasné, jakých hodnot může výsledek z funkce nabývat.

```
1 public someFunction(): boolean | undefined {
2     // Vykonání potřebného kódu
3
4     return success;
5 }
```

Výpis 4.2 Schéma funkce (zdroj: vlastní)

Pro FE se dále využívá framework Next.js, který rozšiřuje základní funkcionalitu Reactu o podporu server-side renderingu (SSR) a client-side renderingu (CSR). Tento přístup umožňuje kombinovat předgenerování stránek na serveru pro rychlejší načítání obsahu s možností dynamické aktualizace dat na straně klienta. Výsledkem je vyšší výkon aplikace a zlepšení uživatelské zkušenosti díky rychlejším odezvám a plynulejším přechodům mezi stránkami. Zároveň má SSR i pozitivní vliv na search engine optimization (SEO).

4.2.3 Databáze

Pro správu databázových schémat a dat je v projektu využíván nástroj HeidiSQL, který poskytuje přehledné grafické rozhraní pro práci s databází. HeidiSQL umožňuje snadné vytváření a úpravy tabulek, definici indexů, spouštění SQL dotazů a správu uživatelských oprávnění.

nění. Tento nástroj byl zvolen kvůli své jednoduchosti, podpoře MariaDB a efektivnímu návrhu databázové struktury.

4.3 Struktura modulu

Na základě analýzy stávajícího řešení a námětů na vylepšení, došlo k reorganizaci celého modulu do následujících sekcí.

- **Obecné informace** – souhrn informací o modulu.
- **Konfigurace vstupů** – jednotný formulář pro zadání veškerých vstupních dat, nutných pro modelaci.
- **Návrh** – nová sekce pro výpočet návrhu na cílové využití teoretického potenciálu.
- **Modelování** – simulace a vyhodnocení míry separace odpadu.

Reorganizace modulu byla navržena s ohledem na uživatelskou volnost při procházení jednotlivých sekcí. Uživatel může přistupovat do kterékoli části modulu bez nutnosti vyplnění předchozích vstupních údajů. Tento přístup umožňuje získat lepší přehled o celkové podobě a fungování systému již v počáteční fázi práce s modulem. Zároveň však rozdělení reflektuje jednotlivé fáze práce s daty a jejich modelováním.

4.3.1 Souborová struktura

V rámci reorganizace modulu bylo nutné přizpůsobit také souborovou strukturu, aby odpovídala konvencím používaným ve frameworku Flask a Next.js.

V BE byla změna realizována vytvořením nového souboru, který byl přidán v souladu se stávající strukturou aplikace na relativní cestu `/backend/api/wasteData.py`.

Ve FE bylo přizpůsobení složitější z důvodu použití Next.js, který řídí routing prostřednictvím file-based routingu. Všechny podadresáře v adresáři `/app` se automaticky stávají routami, přičemž každý adresář reprezentuje odpovídající část URL cesty. Každý adresář musí obsahovat vstupní soubor `page.tsx`, který definuje obsah a logiku příslušné stránky. Proto byl adresář modulu umístěn na relativní cestě `/frontend/app/modelovani-potencialu`. Použití českého názvu adresáře bylo zvoleno záměrně, aby generovaná URL adresa reflektovala českou terminologii aplikace.

Výjimkou v routingu jsou však adresáře, jejichž název začíná podtržítkem jako například `_components`. Ty jsou při generování rout automaticky ignorovány a nejsou tedy dostupné jako samostatné URL adresy. Díky tomu je možné organizovat pomocné soubory nebo sdílené komponenty v rámci projektu bez rizika, že by byly automaticky mapovány na URL cestu.

Struktura adresáře modulu odpovídá konvencím používaným v Next.js a React ekosystému viz obrázek 4.1.



Obrázek 4.1 Zjednodušená souborová struktura modulu modelování potenciálu

(zdroj: vlastní)

- **page.tsx** – vstupní soubor modulu, který Next.js automaticky mapuje na základní URL cestu.
- **_ui** – adresář obsahující jednotlivé komponenty.
- **_ui-common** – adresář pro společné komponenty používané napříč jednotlivými sekcemi modulu.
- **_helpers** – adresář obsahující pomocné funkce.
- **_utils** – adresář obsahující pomocné výpočty a algoritmy.
- **podadresáře pro jednotlivé sekce** – každá sekce modulu (např. konfigurace vstupů) je implementována ve vlastní složce.

Další inovací bylo rozšíření definic rout v aplikaci prostřednictvím centrálního souboru s enumem, který obsahuje základní cesty pro jednotlivé sekce modulu. Tento přístup umožňuje jednoznačné mapování URL adres na odpovídající komponenty reprezentující jednotlivé sekce jako například `/modelovani-potencialu/konfigurace-vstupu`.

Navigace mezi stránkami je realizována pomocí systémové komponenty `Link` z `Next.js`, jejíž užití je uvedeno v ukázce kódu 4.3. Komponenta umožňuje klientské přechody mezi jednotlivými sekcemi bez nutnosti opětovného načítání stránky ze serveru. Tohoto chování je dosaženo díky tomu, že `Next.js` předběžně načítá potřebné soubory JavaScriptu a statická data spojená s cílovou stránkou ještě před tím, než uživatel přejde na novou stránku. Po kliknutí na odkaz pak `Next.js` provede přechod na stránku na straně klienta (client-side). Tento mechanismus zajišťuje rychlejší načítání obsahu a plynulejší přechody mezi stránkami.

```
1   <Link
2       href={Route.MODELOVANI_POTENCIALU}
3       data-state={pathName === Route.MODELOVANI_POTENCIALU
4           ? "active" : "inactive"}
5   >   Obecné Informace
6   </Link>
```

Výpis 4.3 Příklad odkazu na sekci Modelování potenciálu (zdroj: vlastní)

5 IMPLEMENTACE

V této sekce bude popsána vlastní implementace celého modulu.

5.1 Správa stavu

Správa stavu v rámci modulu vychází ze zavedeného mechanismu správy stavu celé aplikace, který je řízen pomocí knihovny Redux (podrobně popsáno v kapitole 2.3.5). V rámci rozšíření modulu nebyl důvod tuto volbu měnit, protože Redux poskytuje jednotné a efektivní řešení pro správu stavu napříč celou aplikací. Hlavními faktory pro zachování této implementace byla snadná integrace s existujícím řešením, konzistence v práci se stavem a možnost snadného budoucího rozšíření nebo úpravy logiky modulu.

Jako alternativa k Reduxu připadalo v úvahu použití tzv. prop drilling – tedy předávání dat z rodičovské komponenty na potomky prostřednictvím props. Tento přístup by na první pohled mohl působit jako jednodušší a rychlejší řešení, avšak při složitější hierarchii komponent naráží na zásadní problém. Pokud je mezi rodičovskou komponentou a cílovou komponentou více mezilehlých komponent, musí každá z nich přijmout a předat props, i když je sama přímo nevyužívá. Tento přístup vede ke zbytečnému nárůstu počtu props v jednotlivých komponentách, což zvyšuje složitost kódu, zhoršuje čitelnost a ztěžuje jeho budoucí údržbu.

5.1.1 Waste data slice

Co se ale změnilo byl přístup ke stavu modulu jako takového. V původní implementaci bylo pro správu stavu modulu vytvořeno dvanáct slice souborů. Tento přístup vedl nejen ke zdvojnásobení velikosti objektu ve store a zároveň narušení struktury jeho rozvržení, ale obecně k velkému množství kódu, kde muselo být vše explicitně nadefinováno. Původním záměrem totiž bylo umožnit práci s velkým množstvím stavových dat, která se sice mohla v různých částech aplikace opakovat, ale z hlediska logiky zpracování se významově lišila. Například pro každý typ potenciálu nebo pro vstupní údaje jednotlivých zdrojů odpadu byly vytvořeny samostatné slice soubory.

Naproti tomu je nově stav modulu spravován prostřednictvím jednoho slice souboru s názvem `wasteDataSlice.ts`. Změna vychází z celkové refaktORIZACE modulu a optimalizace práce s daty. Celý stav je nadefinován pomocí jednoho rozhraní (interface) `IState 5.1`.

Kde každá sekce je reprezentována samostatným objektem uvnitř struktury stavu, což umožňuje přehledné oddělení logiky a snadnou manipulaci s daty.

```
1 interface IState {
2     configurationInput: IWasteConfigInputPayload;
3     suggestion: {
4         percentage: number;
5         theoPotentialGoalsDefs: Record<PotentialDef, {[
6             wasteCode: number]: number}>;
7     };
8     modeling: {
9         potentialDef: PotentialDef;
10        theoPotentialGoalsDefs: Record<PotentialDef, {[
11            wasteCode: number]: number}>;
12    };
13    loadedDataInfo: ILoadedDataInfo;
14 }
```

Výpis 5.1 Interface stavu Waste data slice (zdroj: vlastní)

Klíčovou myšlenkou bylo zvýšení úrovně abstrakce nad daty, což by eliminovalo nutnost vytváření extra slice souborů. Pro sjednocení datové struktury tedy byly zavedeny specifické enumy. První z nich PotentialDef (viz 5.2) reprezentuje jednotlivé definice teoretických potenciálů, s nimiž modul pracuje.

```
1 export enum PotentialDef {
2     A,
3     B,
4     C,
5     D,
6 }
```

Výpis 5.2 Schéma funkce (zdroj: vlastní)

A druhý WasteCode (viz 5.3) obsahuje katalogová čísla KO. Tento enum je rozšířen o položku s hodnotou 999999, která reprezentuje agregovanou položku pro ostatní odpady.

```
1 export enum WasteCode {
2     PAPER = 200101,
3     GLASS = 200102,
4     BIO_WASTE_KITCHEN = 200108,
5     CLOTHING = 200110,
6     TEXTILE_MATERIALS = 200111,
7     EDIBLE_OIL_AND_FAT = 200125,
8     WOOD = 200138,
9     PLASTICS = 200139,
10    METALS = 200140,
11    BIO_WASTE = 200201,
12    MMW_SKO = 200301,
13    BW_OBJ = 200307,
14    OTHERS = 999999,
15 }
```

Výpis 5.3 Schéma funkce (zdroj: vlastní)

Základem je pak objekt typu `Record`, který slouží jako slovník – klíčem je hodnota z `enumu` definujícího potenciál a hodnotou je `indexová signatura` (`index signature`), který reprezentuje dvojici klíče a hodnoty. Klíčem je katalogové číslo KO a hodnotou je číselná hodnota představující velikost potenciálu v procentech pro dané katalogové číslo KO.

Výhodou této kombinace je typová kontrola a datová úplnost na úrovni vnějšího objektu, protože objekt typu `Record` vyžaduje explicitní definici všech hodnot z `enumu`. Zatímco `indexová signatura` umožňuje dynamickou inicializaci a flexibilní mapování dat. Samotné užití `indexové signatury` s klíčem katalogového čísla KO a hodnotou s typem podle kontextu podle použití, je pak hlavní strukturou práce s daty napříč celým modulem.

Další změnou byla samotná implementace `slice`. V původní implementaci, bylo všech dvanáct `slice` sdruženo do jedné pomocí funkce `combineSlices`. Každá `slice` pak byla nadefinována standardním způsobem, ale bez použití `selectoru`. Využíval se tedy přímý přístup přes `rootReducer`, což znamenalo, že při jakékoliv změně ve struktuře stavu bylo potřeba následné úpravy i v komponentách. V nové implementaci se exportují explicitně `action` a `selector`, tím se umožňuje lepší zapouzdření a snižuje závislost na struktuře. Implementace je naznačena ve zjednodušeném kódu 5.4.

```

1 interface IState {
2   ...
3 }
4
5 export const NAMESPACE = "wasteData";
6
7 export const DEFAULT_STATE: IState = {
8   ...
9 };
10
11 export const wasteDataSlice = createSlice({
12   name: NAMESPACE,
13   initialState: DEFAULT_STATE,
14   reducers: {
15     resetState: () => {
16       return DEFAULT_STATE;
17     },
18     setConfigurationInputs: (state, action: PayloadAction
19       <Partial<IWasteConfigInputPayload>>) => {
20       state.configurationInput = {
21         ...state.configurationInput,
22         ...action.payload,
23       };
24     },
25     setSuggestionPercentage: (state, action:
26       PayloadAction<number>) => {
27       state.suggestion.percentage = action.payload;
28     },
29     ...
30   },
31   selectors: {
32     configurationInput: (state) => state.
33       configurationInput,
34     ...
35   },
36 });

```

```
35 export const wasteDataAction = wasteDataSlice.actions;  
36  
37 export const wasteDataSelector = wasteDataSlice.selectors;
```

Výpis 5.4 Implementace Waste data slice (zdroj: vlastní)

5.1.2 Waste data API slice

V rámci rozšíření modulu bylo nutné zajistit efektivní komunikaci mezi BE a FE. Pro tento účel byla použita knihovna Redux Toolkit Query (RTKQ), která umožňuje spravovat stav získaný z API přímo v Redux state. Její další výhodou je automatická správa načítání, cachování nebo synchronizace stavu.

Definice základního rozhraní pro připojení k API je nadefinována v objektu `apiSlice` pomocí metody `createApi` z knihovny RTKQ. Tento objekt zajišťuje základní konfiguraci, jako je definice kořenové adresy (`baseUrl`) a přidávání hlaviček (`headers`) pro HTTP požadavky.

Pro přidání nových endpointů bylo využito metody `injectEndpoints`, která umožňuje rozšíření existujícího `apiSlice` bez nutnosti vytvářet nový objekt. Tento přístup umožňuje sdílet základní konfiguraci pro HTTP požadavky a zároveň modularizovat jednotlivé endpointy. Rozšíření o nové endpointy je nadefinováno v souboru `wasteDataApiSlice.ts`. Soubor byl uložen v souborové struktuře dle konvencí, a jeho relativní cesta je tedy dle očekávání `/frontend/redux/api/wasteDataApiSlice.ts`.

V naznačené implementaci 5.5, je znázorněno přidávání jednotlivých endpointů v podobě objektů, které nastavují konkrétní API volání. Kromě názvu endpointu, pod kterým se generuje odpovídající exportní hook, jsou důležitými i následující prvky.

- **query** – funkce, která definuje detaily požadavku.
- **transformResponse** – funkce, která transformuje odpověď z API před jejím uložením do Redux store.
- **providesTags a invalidatesTags** – RTKQ umožňuje označit odpověď nadefinovaným tagem. Pokud se pak provede mutace na stejný tag, RTKQ automaticky invaliduje cache a provede nové načtení dat.

```

1 export const wasteDataApiSlice = apiSlice.injectEndpoints({
2   endpoints: (builder) => ({
3     getWasteConfigInputs: builder.query<IWasteConfigInput
4       [], void>({
5         query: () => ({
6           url: "/getwasteconfiginputs",
7           body: {jsonify: true},
8         }),
9         transformResponse: (response:
10           UserWasteConfigInputsApiResponse) => {
11             if (isErrorResponse(response)) {
12               return [];
13             }
14             return response.data;
15           },
16         providesTags: ["WasteConfigInputs"],
17       }),
18     saveWasteConfigInput: builder.mutation<
19       IApiResponseToast, IWasteConfigInput>({
20         query: (payload) => ({
21           url: "/savewasteconfiginput",
22           body: payload,
23         }),
24         invalidatesTags: ["WasteConfigInputs"],
25       }),
26     ...
27   }),
28 });
29 export const {
30   useLazyGetWasteConfigInputsQuery,
31   useSaveWasteConfigInputMutation,
32 } = wasteDataApiSlice;

```

Výpis 5.5 Implementace Waste data API slice (zdroj: vlastní)

Pro načtení dat z jednotlivých endpointů v komponentě se v RTKQ používají hooky, které se podle způsobu spuštění požadavku dělí na dva typy. Pokud je v názvu hooku uvedeno slovo "Lazy", jako v příkladě 5.6, požadavek se odešle až po explicitním zavolání vrácené funkce. Což umožňuje dynamické načítání dat na základě uživatelské akce nebo jiných podmínek. Naproti tomu druhý typ, viz příklad 5.7, provede odeslání požadavku automaticky při načtení komponenty. Po prvním načtení dat RTKQ již automaticky spravuje jejich stav.

```

1 const [getWasteConfigInputs, { data, isLoading, isError }] =
  useLazyGetWasteConfigInputsQuery();

```

Výpis 5.6 Hook s lazy načtením (zdroj: vlastní)

```

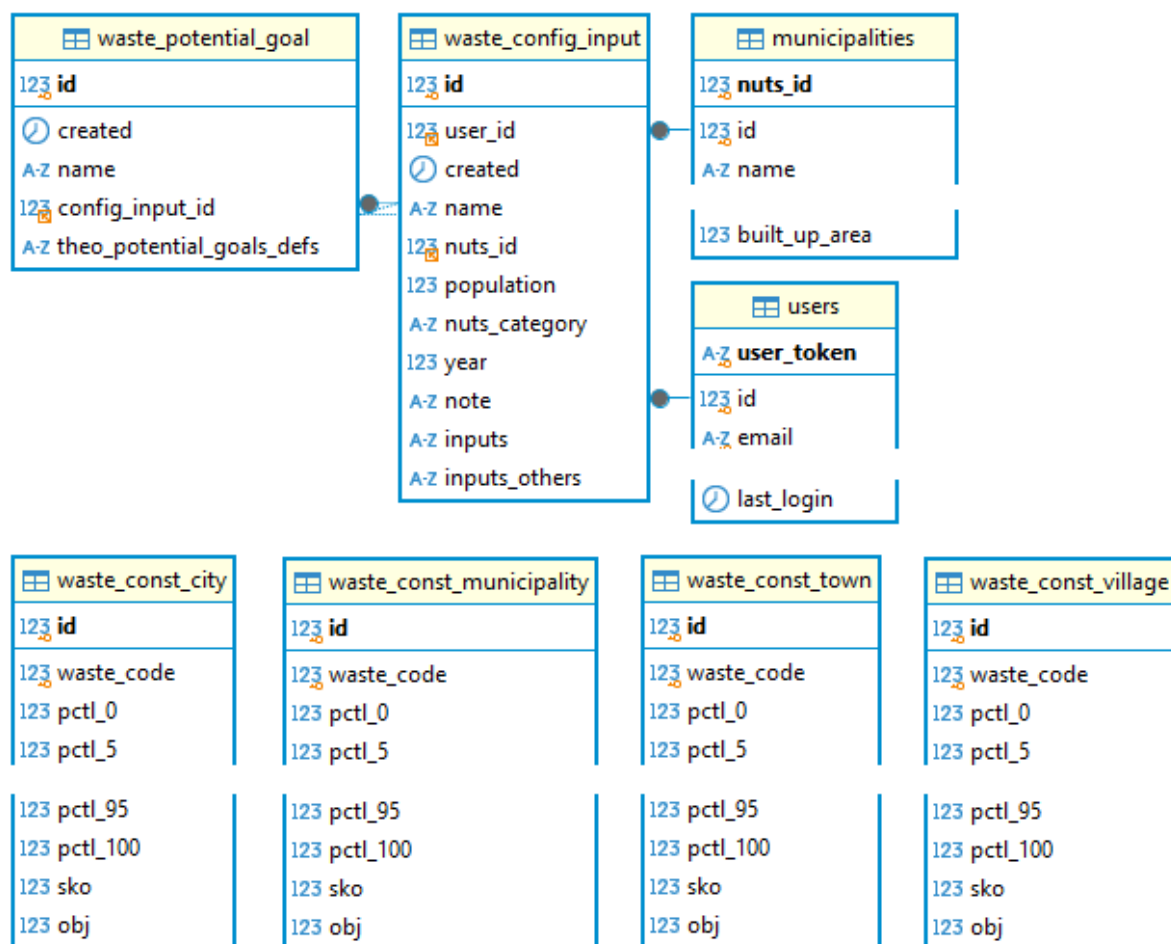
1 const { data, isLoading, isError } =
  useGetWasteConfigInputsQuery();

```

Výpis 5.7 Hook s okamžitým načtením (zdroj: vlastní)

5.2 Databáze

V rámci rozšiřování modulu o možnost ukládání dat do perzistentní paměti, bylo nezbytné navrhnout a implementovat databázové řešení s integrací do již existující databáze, které zajistí efektivní a konzistentní ukládání, správu a dotazování nad strukturálně členěnými daty. Zjednodušený diagram nových a závislých tabulek je na obrázku 5.1.



Obrázek 5.1 Zjednodušený diagram rozšíření stávající databáze

(zdroj: vlastní)

Pro správu relační databáze byl zvolen nástroj HeidiSQL, který představuje bezplatné open-source řešení umožňující efektivní správu databází prostřednictvím grafického uživatelského rozhraní (GUI). Tento nástroj se vyznačuje přehledným a intuitivním prostředím, což výrazně usnadňuje běžné operace, jako je tvorba a úprava tabulek, spouštění SQL dotazů či správa databázových schémat. Přestože je vhodný i pro méně zkušené uživatele, poskytuje širokou škálu pokročilých funkcí, včetně vizuálního návrhu dotazů, možnosti exportu a importu dat.

Z hlediska znakové sady a pravidel pro porovnávání textových hodnot (collation) je ve všech dosavadních tabulkách využívána znaková sada s pravidlem `utf8mb4_general_ci`, které zajišťuje podporu pro plný rozsah znaků včetně speciálních a mezinárodních symbolů, a zároveň poskytuje dobrý výkon při běžných porovnávacích operacích.

S ohledem na tuto praxi je znaková sada `utf8mb4_general_ci` zvolena také jako standardní nastavení pro veškeré nově navržené tabulky uvedené v této kapitole a popsané níže, pokud není explicitně uvedeno jinak.

5.2.1 Tabulky konstant produkce odpadů pro shlukové skupiny obcí

Jak je uvedeno v kapitole 1.3, každá shluková skupina obcí je charakterizována specifickým souborem parametrů, obecně označovaných jako konstanty. Ty jsou nově uloženy v databázové vrstvě a to samostatně pro každou skupinu ve formě oddělené tabulky.

Tabulky jsou pojmenovány podle jednotného vzoru `waste_const_NázevSkupiny`, kde *NázevSkupiny* přesně odpovídá hodnotám enumů používaného v aplikační logice. Toto pojmenování zajišťuje konzistenci mezi databázovou a aplikační vrstvou a umožňuje jednoznačné přiřazení relevantních záznamů na základě příslušnosti obce ke konkrétní skupině.

Pro sloupce reprezentující hodnoty percentilů byly zvoleny názvy ve formátu `pctl_x`, kde *x* označuje konkrétní percentil, např. 5, 25 nebo 95. Tato struktura odpovídá matematickému modelu, který pracuje s rozdělením teoretického potenciálu dle stanovených percentilových hladin. Na obrázku 5.2 je pak znázorněna zkrácená struktura jedné z těchto tabulek.

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	id	INT	11	☒	☐	☐	AUTO_INCREME...
2	waste_code	INT	6	☒	☐	☐	Žádná hodnota
3	pctl_0	DECIMAL	13,9	☒	☐	☐	'0.000000000'
4	pctl_5	DECIMAL	13,9	☒	☐	☐	'0.000000000'
22	pctl_95	DECIMAL	13,9	☒	☐	☐	'0.000000000'
23	pctl_100	DECIMAL	13,9	☒	☐	☐	'0.000000000'
24	sco	DECIMAL	13,9	☒	☒	☐	NULL
25	obj	DECIMAL	13,9	☒	☒	☐	NULL

Obrázek 5.2 Zkrácená struktura tabulky konstant shlukové skupiny pro město

(zdroj: vlastní)

Sloupce tabulky jsou stručně popsány níže.

- **id** INT UNSIGNED AUTO_INCREMENT — sloupec jehož typ je nastaven na kladné celé

číslo a hodnota na auto inkrement od 1 s krokem 1. Představuje klasické řešení pro zajištění jednoznačné identifikace záznamů v rámci tabulky.

- **Katalogové číslo odpadu (waste_code) MEDIUMINT UNSIGNED** — sloupec obsahuje hodnoty odpovídající šesticiferným katalogovým číslům odpadu (viz tabulka 1.1). Datový typ je omezen na kladná čísla s přesností odpovídající očekávanému rozsahu, MEDIUMINT. Z důvodu integrity dat je na tomto sloupci nastaven unikátní klíč, čímž je zajištěno, že každé katalogové číslo se v dané tabulce vyskytuje nejvýše jednou.
- **Sloupce s konstantami** — všechny sloupce jsou typu DECIMAL desetinné číslo s maximálním počtem třinácti cifer. Z toho s přesností až devět desetinných míst. A to z důvodu zajištění maximální přesnosti pro výpočty. Nicméně, do budoucna je vhodné zvážit, zda je takto vysoká přesnost nutná. Poslední dva sloupce mají navíc příznak NULL (nulový). Což znamená, že jejich hodnota není při vkládání nového záznamu vyžadována.

5.2.2 Tabulka konfigurace vstupů

Tabulka *waste_config_input*, jejíž schéma je na obrázku 5.3, slouží pro uložení vstupních hodnot zadaných uživatelem na začátku procesu návrhu a modelování. Každý uživatel může mít uloženo několik konfigurací pro různé obce v několika letech. Umožňuje uchovávat více různých konfigurací pro jednotlivé obce a roky, přičemž každá kombinace obce a roku je pro jednoho uživatele jedinečná. Tato unikátnost je zajištěna pomocí kombinovaného unikátního klíče zahrnujícího sloupce *user_id*, *nuts_id*, *year*.

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí	Porovnání
1	id	INT	11	☒	☐	☐	AUTO_INCREME...	
2	user_id	INT	11	☐	☐	☐	Žádná hodnota	
3	created	DATETIME		☐	☒	☐	NULL	
4	name	VARCHAR	50	☐	☒	☐	NULL	utf8mb4_gene...
5	nuts_id	INT	11	☐	☐	☐	Žádná hodnota	
6	population	MEDIUMINT	7	☒	☐	☐	Žádná hodnota	
7	nuts_category	VARCHAR	13	☐	☐	☐	Žádná hodnota	utf8mb4_gene...
8	year	SMALLINT	4	☒	☐	☐	Žádná hodnota	
9	note	VARCHAR	255	☐	☒	☐	NULL	utf8mb4_gene...
10	inputs	LONGTEXT		☐	☒	☐	NULL	utf8mb4_bin
11	inputs_others	LONGTEXT		☐	☒	☐	NULL	utf8mb4_bin

Obrázek 5.3 Struktura tabulky konfigurace vstupů

(zdroj: vlastní)

Tabulka má vazby na existující tabulku *uživatelů* (*users*) a *obcí* (*municipalities*) prostřednictvím cizích klíčů, což umožňuje jednoznačné propojení s aplikačními entitami. Primární klíč sloupce *id* slouží pro jednoznačnou identifikaci záznamu při jeho úpravě či mazání, zatímco složený klíč s *user_id* na první pozici umožňuje efektivní indexování a filtrování záznamů podle konkrétního uživatele.

Jedná se tak o klíčový prvek celého modulu, protože tabulka je vstupním bodem výpočetního procesu a zároveň slouží jako základ pro následné modelování a vytváření scénářů.

- **id** INT UNSIGNED AUTO_INCREMENT – primární klíč tabulky.
- **user_id** INT – cizí klíč odkazující na entitu v tabulce *users*. Umožňuje přiřadit každou konfiguraci konkrétnímu uživateli.
- **created** DATETIME NULL – časová značka vytvoření konfigurace. Hodnota je volitelná (může být NULL), slouží primárně pro orientaci v seznamu konfigurací a jejich řazení.
- **name** VARCHAR(50) NULL – uživatelem zadaný název konfigurace. Zvolený typ umožňuje zadat text o proměnné délce, maximálně však 50 znaků.
- **nuts_id** INT - cizí klíč odkazující na entitu v tabulce *municipalities*. Reprezentující vybranou obec, ke které se konfigurace vztahuje.
- **population** MEDIUMINT UNSIGNED – počet obyvatel dané obce. Hodnota je v aplikaci omezena horní hranicí 1 700 000 předem stanoveným limitem. Proto svým rozsahem stačí MEDIUMINT.
- **nuts_category** VARCHAR(13) – textová reprezentace shlukové skupiny obce podle enumu používaného v aplikaci. Maximální délka 13 znaků odpovídá nejdelší hodnotě enumu.
- **year** SMALLINT UNSIGNED – rok, ke kterému se konfigurace vztahuje. Maximální možná hodnota je v aplikaci omezena na 9999. Proto svým rozsahem tentokrát stačí SMALLINT.
- **note** VARCHAR(255) NULL – volitelné textové pole pro poznámku k dané konfiguraci. Sloupec akceptuje až 255 znaků a není povinný.

- **inputs, inputs_others** LONGTEXT NULL – sloupce slouží pro uložení serializovaných objektů JSON odpovídajících vstupním údajům z formulářových sekcí. Proto je tentokrát jazyková sada nastavena na utf8mb4_bin, která umožňuje jen základní binární porovnání. Oba sloupce umožňují hodnotu NULL, čímž je možné uložit konfiguraci i bez vyplněných konkrétních dat o množství odpadu.

5.2.3 Tabulka scénářů

Tabulka s názvem *waste_potential_goal* (schéma je na obrázku 5.4) slouží pro uložení tzv. scénářů využití teoretického potenciálu, tedy uživatelem definovaných hodnot využití jednotlivých definic potenciálu pro konkrétní konfiguraci obce. Tato relace je realizována pomocí cizího klíče *config_input_id*, který vytváří vztah 1:N mezi konfiguracemi a scénáři.

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí	Porovnání
1	id	INT	11	☒	☐	☐	AUTO_INCREME...	
2	created	TIMESTAMP		☐	☒	☐	NULL	
3	name	VARCHAR	50	☐	☒	☐	NULL	utf8mb4_gene...
4	config_input_id	INT	11	☒	☐	☐	Žádná hodnota	
5	theo_potential_go...	LONGTEXT		☐	☒	☐	NULL	utf8mb4_bin

Obrázek 5.4 Struktura tabulky scénářů

(zdroj: vlastní)

Níže je uvedena struktura této poslední tabulky.

- **id** INT UNSIGNED AUTO_INCREMENT – primární klíč tabulky.
- **created** DATETIME NULL – časová značka vytvoření scénáře.
- **name** VARCHAR(50) NULL – uživatelem zadaný název scénáře.
- **config_input_id** INT – cizí klíč odkazující na entitu v tabulce *waste_config_input*.
- **theo_potential_goals_defs** LONGTEXT NULL – sloupec s utf8mb4_bin jazykovou sadou slouží k uložení serializovaného objektu ve formátu JSON, který obsahuje procentuální hodnoty využití jednotlivých definic teoretického potenciálu.

5.3 Pomocné funkce

Jsou funkce, které obvykle slouží k usnadnění konkrétních úloh nebo procesů v rámci aplikace jako je například formátování dat nebo validace vstupů. Pro jejich umístění je v projektu vyhrazen adresář *_helpers*. V aktuální implementaci je zde dostupná pomocná funkce

roundNumber (viz výpis 5.8), která slouží k zaokrouhlení číselné hodnoty na požadovaný počet desetinných míst.

```
1 export function roundNumber(num: number, decimalPlaces:
   number): number {
2   if (!num) {
3     return 0;
4   } else if (num !== 0 && num % 1 !== 0) {
5     return parseFloat(num.toFixed(decimalPlaces));
6   } else {
7     return num;
8   }
9 }
```

Výpis 5.8 Funkce zaokrouhlení čísla na daný počet desetinných míst (zdroj: vlastní)

5.4 Pomocné výpočty a algoritmy

Jsou uloženy v adresáři `_utils` a slouží jako výpočetní jádro celého modulu zaměřené výhradně na matematické operace a algoritmické zpracování dat. Tím se liší od adresáře `_helpers` sdružujícího pomocné funkce obecného charakteru.

V rámci revize a nově zvoleného přístupu k návrhu modulu byla původní implementace výpočetních algoritmů kompletně refaktorizována. Původní řešení představovalo doslovný přepis matematických vzorců, který sice zajišťoval přímou návaznost na teoretický model, avšak vedl k nadměrnému množství velmi podobného kódu.

Nově jsou základem výpočetního jádra dvě specializované třídy. Tento způsob implementace byl zvolen s ohledem na potřebu centralizovat související výpočetní funkce a inicializovat pomocí konstruktoru pomocné objekty, které budou k dispozici napříč každé instance třídy.

- **SuggestionFormUtil** – třída pro výpočet návrhu míry separace KO, která je novým rozšířením výpočetní části modulu podle matematických výpočtů popsanych v teoretické části v kapitole 1.4.2.
- **ModelingFormUtil** – třída pro výpočet potenciálu míry separace KO, která je refaktorem původního algoritmu z předchozí implementace.

Pro podpůrné mezi výpočty, potřebné v hlavních třídách, jsou k dispozici celkem tři pomocné třídy.

- **CurrentStateUtil** – třída pro výpočet aktuálního stavu míry separace KO na základě zadaných vstupních hodnot.
- **PotentialDefUtil** – třída pro výpočet hodnot jednotlivých definic potenciálů míry separace.
- **GoalStateUtil** – třída pro výpočet výsledného množství KO při naplnění teoretického potenciálu.

Implementace všech výpočtů je tedy prováděna na straně klienta jako součást FE bez nutnosti odesílání požadavků na BE API. To může být trochu překvapením, ale vzhledem k tomu, že samotné výpočty jsou založeny na základních aritmetických operacích bez dalších náročných algoritmů, bylo by zbytečné přesouvat výpočet na server a riskovat prodlevu v odezvě. Tím spíš, že díky CSR v prostředí Next.js, jsou výpočty prováděny okamžitě při změně vstupních dat, bez nutnosti opětovného načítání stránky nebo komunikace se serverem.

5.4.1 SuggestionFormUtil

Tato třída slouží jako hlavní výpočetní nástroj pro sekci Návrh, čemuž odpovídá i struktura jejího rozhraní. To je navrženo s ohledem na charakter výstupních dat a jejich způsobu využití v rámci dané sekce. Třída poskytuje celkem tři metody, z nichž dvě jsou určeny pro přípravu dat pro vizualizaci a třetí zajišťuje pomocný výpočet využitelný v sekci Modelování.

- **getCurrentState** – metoda pro výpočet aktuálního stavu celkového množství míry separovaného KO, přepočet množství v kilogramech na jednoho obyvatele obce a vyhodnocení výsledků v porovnání s podobnými obcemi. Výsledky porovnání mohou nabývat hodnot z následujícího výčtu.
 - Podprůměr
 - Mírný podprůměr
 - Průměr
 - Mírný nadprůměr
 - Nadprůměr

- **getSuggestionState** – metoda pro výpočet navrhované míry separace a jejich výsledných změn vůči aktuálnímu stavu. První část zahrnuje výpočet množství odpadu, přepočet na jednoho obyvatele a stanovení procentuální změny. Následně je vypočten dopad navržených opatření na množství separovaného odpadu a očekávané množství přepočtené na jednoho obyvatele.
- **getTheoPotentialGoalDefs** – metoda pro výpočet jednotlivých definic potenciální míry separace, které jsou následně využívány jako vstupní parametry pro výpočty v sekci Modelování.

5.4.2 ModelingFormUtil

Tato třída slouží jako hlavní výpočetní nástroj pro sekci Modelování. Stejně jako v předchozím případě vychází struktura jejího rozhraní z charakteru výstupních dat a jejich využití v rámci příslušné sekce. Třída poskytuje dvě metody, obě určené pro přípravu dat k vizualizaci.

- **getCurrentState** – metoda pro výpočet aktuální míry separace KO, identifikaci minimálního množství odpadu z netypických zdrojů a přepočet celkového množství odpadu na jednoho obyvatele.
- **getModelState** – metoda pro výpočet modelovaných potenciálů míry separace KO a výsledného množství odpadu při jejich naplnění. Nejprve se pro každou definici potenciálu vypočítá množství odpadu dostupné k recyklaci a jeho očekávaný objem při zadaném procentu cílového využití. Ve druhé části je následně provedeno součtové zhodnocení celkového recyklovaného množství a jeho přepočet na jednoho obyvatele.

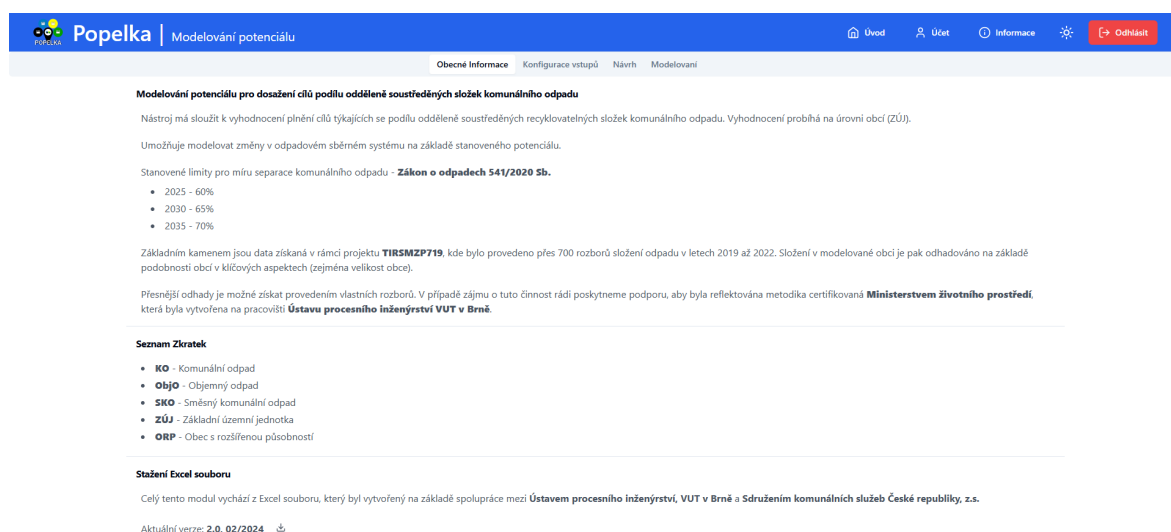
5.5 Sekce modulu

V následujících kapitolách jsou detailně popsány jednotlivé sekce modulu a jejich způsob implementace. Jejichž stručný přehled je uveden v kapitole 4.3.

5.5.1 Obecné informace

Tato sekce je tzv. přistávací stránkou celého modulu (obrázek 5.5). To znamená, že je tato sekce nastavena jako výchozí pro zobrazení po kliknutí na odkaz modulu v hlavním menu 2.1 celé aplikace. Charakter sekce je čistě informativní – neobsahuje žádné interaktivní prvky

s výjimkou možnosti stažení původního Excelového souboru, ze kterého návrh modulu vychází.



Obrázek 5.5 Stránka s obecnými informacemi

(zdroj: vlastní)

Přestože tato sekce neumožňuje uživateli měnit žádná nastavení ani ovlivňovat stav aplikace, obsahuje klíčové informace potřebné pro práci s modulem. Patří mezi ně například přehled stanovených limitů míry separace komunálního odpadu pro nadcházející roky či vysvětlení zkratk používaných v jednotlivých částech modulu.

Rozvržení textových prvků bylo navrženo tak, aby veškerý obsah bylo možné zobrazit bez nutnosti posouvání stránky. Zároveň bylo dbáno na přehlednost, logickou strukturu a vizuální vyváženost celého rozhraní.

5.5.2 Konfigurace vstupů

Je sekce zobrazena na obrázku 5.6. Představuje první interaktivní část modulu a slouží jako centrální rozhraní pro zadávání vstupních dat prostřednictvím sady formulářů. Zároveň obsahuje akční prvky pro načítání, ukládání a správu uživatelských konfigurací. Uspořádání stránky odráží její funkční rozdělení a je členěno do dvou hlavních oblastí – levého navigačního panelu a pravé části s jednotlivými formuláři.

Levé menu

Levá část stránky má fixní pozici i výšku odpovídající celé dostupné výšce viewportu. Ob-

Sekce konfigurace

- Vlastnosti Obce
- Produkcce odpadu obce
- Produkcce ostatního odpadu obce

Načtená data

Konfigurace vstupů:

Modelování:

Vlastnosti Obce

Vyberte svoji obec ze seznamu. Upravte předem nadefinovanou hodnotu na přesný počet obyvatel a zvolte kategorii obce. Tyto údaje jsou nezbytné pro výpočty a bez nich není zbytek modulu zpřístupněn. Zbývající položky jako rok a poznámka jsou již volitelné.

Obec: Vybrat obec

Počet obyvatel: 10

Kategorie: Velké město

Rok: 2025

Poznámka:

Produkcce odpadu obce

Zadejte hodnoty produkce odpadu v tunách [t] pro jednotlivé zdroje sběru odpadu.

Název odpadu	TI	Katalogové číslo	TI	Systémové zdroje	TI	Netypické zdroje	TI	Mimosystémové zdroje	TI
Papír		200101		0		0		0	
Sklo		200102		0		0		0	
Biolog. odpad z kuch.		200108		0		0		0	
Oděvy		200110		0		0		0	
Textilní materiály		200111		0		0		0	
Jedlý olej a tuk		200125		0		0		0	
Dřevo		200138		0		0		0	

Obrázek 5.6 Stránka s konfigurací vstupů

(zdroj: vlastní)

sahuje dvě na sobě nezávislá menu, každé reprezentované samostatným vizuálním blokem v podobě kartičky. První z těchto menu slouží k navigaci mezi jednotlivými sekcemi formulářů v pravé části. Po výběru položky dochází k automatickému posunu (scrollování) na začátek příslušného formuláře.

Druhé menu je určeno pro správu vstupních konfigurací. Obsahuje názvy aktuálně načtené konfigurace a scénáře pro modelování. Součástí tohoto bloku jsou také tlačítka s následujícími funkcemi.

- **Načíst** – funkce pro načtení dříve uložené konfigurace. Nemá žádná omezení. Po kliknutí na tlačítko, se zobrazí seznam dostupných uživatelských konfigurací, ze kterého lze buď to vybrat jednu konkrétní nebo ji také smazat. Po vybrání konkrétní položky se zvolená konfigurace okamžitě načte, a to bez ohledu na aktuální stav formulářů – veškeré zadané hodnoty jsou přepsány nově načtenými daty. Zároveň dochází k jejich zápisu do globálního Redux stavu, čímž jsou zpřístupněny i pro další části modulu, které se vstupními daty dále pracují. V případě smazání položky je nejdříve uživatel vyzván pro potvrzení akce, protože v případě smazání konfigurace dojde i k smazání všech jejich přidružených scénářů. Což vychází z charakteru propojení, kde scénář je závislý na konfiguraci, ale nikoliv naopak.
- **Uložit** – funkce pro uložení nové nebo upravené konfigurace. Podléhá dvěma vali-

dačním omezením. Nejprve je uživateli zobrazeno dialogové okno s formulářem pro zadání názvu. Po potvrzení probíhá kontrola, zda již neexistuje jiná konfigurace daného uživatele se stejným názvem. Druhá validace pak kontroluje, zda již uživatel nemá uloženou konfiguraci pro stejný rok a obec. To vychází z požadovaného omezení, které má zabránit možnosti uložit více záznamů pro daný rok a obec. V obou případech je však uživatel vyzván k potvrzení přepsání již uložené konfigurace, aby ji tímto způsobem mohl editovat.

- **Reset** – funkce pro resetování formulářů na výchozí hodnoty. Nemá přímé omezení, avšak má dopad na další část modulu. Konkrétně dochází k resetování i v sekci Modelování, která funkčně závisí na aktuální vstupní konfiguraci. Na tuto skutečnost je ale uživatel upozorněn prostřednictvím dialogového okna, kde musí akci potvrdit.

Panel formulářů

Zbývající prostor stránky tvoří panel se scrollovatelným obsahem, ve kterém je umístěna sada formulářů pro zadávání vstupních dat. Tyto formuláře obsahují jak povinné, tak volitelné vstupní prvky. Rozlišení mezi nimi není provedeno standardním způsobem (např. hvězdičkou u daných formulářových prvků), ale odlišuje se přítomností výchozí hodnoty. Tedy povinné položky ji mají vždy předvyplněnou, zatímco volitelné zůstávají implicitně prázdné.

Každá část formulářové sekce je realizována jako samostatná komponenta, přičemž validace vstupních dat je zajištěna pomocí knihovny zod.

- **Vlastnosti Obce** – V této části uživatel definuje základní parametry obce. Obec vybírá z rozbalovacího seznamu, který podporuje jak přímé vyhledávání dle názvu, tak hierarchické zobrazení dle krajů a měst. Následně je vybrána kategorie obce podle klasifikace shlukových skupin uvedené v kapitole 1.3. Pole pro počet obyvatel je automaticky předvyplněno na základě zvolené obce, ale jeho hodnota může být uživatelsky upravena. Dále je zadáván rok, pro který je konfigurace vytvářena. Jediným volitelným polem v této části je poznámka, která se v případě exportu propisuje do výstupních dat.

Pro účely výpočtů byly definovány následující limity vstupních hodnot: velikost obce musí být v rozsahu 10 až 1 700 000 obyvatel a rok v rozsahu 0 až 9999. Hodnoty jsou kontrolovány jak z hlediska rozsahu, tak typu – očekává se nezáporné celé číslo. V pří-

padě překročení limitu nebo chybného vstupu je zobrazena validační hláška, a ohraničení daného formulářového pole na chvíli problikne červeně.

Vzhledem k tomu, že formuláře nevyžadují potvrzení pro uložení, které se děje automaticky, je nevalidní hodnota ignorována a ve formulářovém poli zůstává ta předchozí pomocí vlastní komponenty `NumberFormCheckField`. Tento způsob byl vybrán ze subjektivního názoru, že je snazší už při zadávání kontrolovat platný rozsah než pak implementovat dodatečná opatření, jako například deaktivace navigačního menu.

- **Produkce odpadu obce** – Tato sekce slouží k zadání aktuální produkce odpadu v obci. Formulář má podobu jedné tabulky tvořené pomocí sdílené komponenty. Díky tomu je vzhled jednotný se zbytkem tabulek použitých v aplikaci. Tabulka samotná je plně zobrazena bez nutnosti vnitřního posouvání, což přispívá k přehlednosti a komfortnímu zadávání údajů.

Struktura tabulky obsahuje pět sloupců – první dva jsou statické a slouží k informativnímu zobrazení názvu odpadu a jeho katalogového čísla. Do těchto polí nelze zasahovat. Zbývající tři sloupce jsou určeny pro zadávání hodnot produkce odpadu dle původu: systémové, netypické a mimosystémové zdroje. Podrobné vysvětlení tohoto rozdělení je uvedeno v kapitole 1.2.

Vstupy jsou zadávány prostřednictvím číselných polí, která akceptují pouze kladné hodnoty. Aktualizace stavů v aplikaci je navržena tak, aby nedocházelo k nadměrnému přepisování Redux stavu při každém stisku klávesy. Z tohoto důvodu byla zvolena implementace založená na události `onBlur`, která zapisuje hodnotu do stavu až ve chvíli, kdy uživatel opustí dané pole.

- **Vstupy produkce ostatního odpadu obce** – Poslední sekce slouží pro evidenci produkce odpadu pocházejícího výhradně z mimosystémových zdrojů, taktéž definovaných v kapitole 1.2. Struktura i vzhled formuláře jsou totožné s předchozí sekcí, a to ve formě tabulky sdílené komponenty s jednotným zobrazením. Oproti předchozí tabulce však chybí sloupec pro netypické zdroje, jelikož tento zdroj odpadu v této kategorii není.

Způsob zadávání hodnot i jejich validace zůstává nezměněn. To znamená, že se opět používají číselná pole s omezením na nezáporné hodnoty a aktualizace probíhá pro-

střednictvím události `onBlur`. Rozšířením oproti první sekci je automatické sčítání jednotlivých hodnot za každou kategorii odpadu, přičemž výsledné součty jsou uživateli zobrazovány v zápatí tabulky.

Dále byla doplněna možnost alternativního způsobu zadání dat, kdy uživatel může místo jednotlivých hodnot zadat přímo celkovou produkci dané kategorie odpadu. Tato možnost reflektuje běžnou praxi, kdy jsou údaje o této skupině odpadů často vedeny a distribuovány pouze v podobě celkové produkce. V návaznosti na to byla tato varianta zohledněna i v implementaci výpočetního jádra, které pracuje přímo s celkovými hodnotami.

Alternativní varianta se zachováním hodnot v obou formulářích i při jejich přepínání, byla nakonec zamítnuta. A to z důvodu potenciálního zmatení uživatele ohledně toho, které hodnoty jsou aktuálně platné a budou ve výpočtech zohledněny.

5.5.3 Návrh

Tato sekce představuje jednu z klíčových novinek tohoto modulu a plní několik zásadních funkcí. Z hlediska uživatelského přístupu k celému modulu tvoří přirozené přemostění mezi sekcí vstupních dat a následným modelováním. Zároveň však poskytuje nástroj pro výpočet množství jednotlivých recyklovatelných složek KO pro požadované procento výsledné míry separace, a také jeho případné načtení do sekce Modelování.

Rozvržení uživatelského rozhraní je navrženo v souladu s předchozí sekcí. Tedy levý postranní panel slouží k ovládání, zatímco pravá část obrazovky zobrazuje výsledná data. Tím je zachována konzistence uživatelského prostředí a usnadněna orientace v rámci interaktivní části modulu. Sekce je znázorněna na obrázku 5.7.

Levé menu

Levá část rozhraní zachovává stejnou pozici i rozměrové uspořádání jako v předchozí sekci, což podporuje jednotný vzhled aplikace. V tomto případě je však panel tvořen pouze jednou kartičkou. Ta slouží k zobrazení aktuálně aktivní konfigurace a scénáře modelování. Zároveň obsahuje jedno akční tlačítko s funkcí pro přenos dat do další části modulu.

- **Načíst do** – funkce pro načtení návrhu do scénáře v sekci Modelování. Po kliknutí na tlačítko se uživateli zobrazí dialogové okno s přehledem dříve uložených scénářů

Procento požadované míry separace 0									
Současný Stav					Definice Potenciálu			Změna	
Název odpadu	Katalogové číslo	Množství odpadu [t]	Množství KO na obyvatele [kg/os]	Srovnání s podobnými obcemi	Množství odpadu [t]	Množství KO na obyvatele [kg/os]	Procentuální změna množství odpadu	Množství dopadu [t]	Množství KO na obyvatele [kg/os]
Papír	200101	0	0	Podprůměr	0	0	0	0	0
Sklo	200102	0	0	Podprůměr	0	0	0	0	0
Biolog. odpad z kuch.	200108	0	0	Podprůměr	0	0	0	0	0
Oděvy	200110	0	0	Podprůměr	0	0	0	0	0
Textilní materiály	200111	0	0	Podprůměr	0	0	0	0	0
Jedlý olej a tuk	200125	0	0	Podprůměr	0	0	0	0	0
Dřevo	200138	0	0	Podprůměr	0	0	0	0	0
Plasty	200139	0	0	Podprůměr	0	0	0	0	0
Kovy	200140	0	0	Podprůměr	0	0	0	0	0
Biolog. odpad	200201	0	0	Podprůměr	0	0	0	0	0
SKO	200301	0	0	Podprůměr	0	0	0	0	0
Obyč	200307	0	0	Podprůměr	0	0	0	0	0
Ostatní	-	0	0	Podprůměr	0	0	0	0	0

Obrázek 5.7 Stránka s návrhem pro modelování potenciálu

(zdroj: vlastní)

modelování, pokud tedy již nějaké uložil. Ať již uživatel nějaké dříve uložené scénáře má či nemá, jednou z možností pro přenos je vytvořit scénář nový. Ten se nabízí vždy na začátku seznamu. Po výběru konkrétní položky se do Redux stavu uloží vypočtená procenta navýšení recyklovatelných složek KO pro každou definici potenciálu (definice proměnné 5.9), která pak slouží jako vstupní parametr pro následnou sekci.

```
1 private theoPotentialGoalDefs: Record<PotentialDef, {[
    wasteCode: number]: number}>;
```

Výpis 5.9 Inicializace sekce Návrh (zdroj: vlastní)

K přepnutí do ní však nedochází automaticky, a to z důvodu lepší uživatelské přívětivosti. To znamená, že uživatel má tak možnost výsledky návrhu dále upravovat změnou procenta nebo je opakovaně analyzovat bez přerušení aktuální práce.

Panel návrhu

Hlavní část pravé strany rozhraní vyplňuje panel návrhu, který zabírá celý dostupný prostor a slouží k přehlednému zobrazení výstupních dat výpočtu. Komponenta je koncipována jako tabulka s maximálními rozměry bez nutnosti posouvání.

Na rozdíl od předchozích tabulek z předešlé sekce má tato tabulka převážně statický charakter. Slouží totiž primárně pro vizualizaci vypočítaných dat. Jediným interaktivním prvkem

je číselné vstupní pole v záhlaví tabulky, které slouží pro zadání požadovaného procenta cílové míry separace. Vstupní hodnota je validována v rozsahu od 0 do 70 %, což odpovídá omezením matematického modelu popsaného v kapitole 1.4.2.

Z hlediska implementace je tabulka rozdělena do tří částí, přičemž veškeré výpočty a generování dat zajišťuje dříve popsaná třída `SuggestionFormUtil`. Ta je inicializována až po načtení všech potřebných konstant, které jsou uchovány v databázi. To znamená, že při prvním načtení komponenty ještě nemusejí být k dispozici v paměti klientské aplikace. Z tohoto důvodu je vykreslení panelu podmíněno dostupností těchto dat. Proto do doby, než jsou načtena, je uživateli zobrazen indikátor načítání (spinner) s informativní hláškou „Načítání dat“ (viz ukázka v kódu 5.10).

```
1 function SuggestionTab() {
2   const selectedConfigurationInput = useSelector(
3     wasteDataSelector.configurationInput);
4   const {
5     data: wasteConstants,
6     isLoading: wasteConstantsLoading,
7     isError: wasteConstantsError,
8   } = useGetWasteConstantsQuery(selectedConfigurationInput.
9     nuts_category);
10  return (
11    <div className="w-full h-[calc(100%-2.5rem)] flex">
12      <SuggestionTabMenu />
13      {wasteConstantsLoading && (
14        <div className="w-full h-full flex justify-center
15          items-center p-4">
16          <Spinner size="xs" className="mr-5" /> <span>Načítá
17            ní dat</span>
18        </div>
19      )}
20      {wasteConstantsError && <span>Při načítání dat došlo k
21        chybě</span>}
22      {wasteConstants?.length && <SuggestionTabForm
23        wasteConstants={wasteConstants} />}
24    </div>
25  );
26 }
```

Výpis 5.10 Inicializace sekce Návrh (zdroj: vlastní)

Při inicializaci celé komponenty se nejprve předpřipraví data z Konfigurace vstupů, nainicializuje se výpočetní nástroj a provede se výpočet první části, současného stavu. Tyto dílčí kroky jsou optimalizovány pomocí hooku `useMemo`, který uchovává mezivýsledky a zajišťuje, že se výpočty provádějí pouze v případě, že dojde ke změně některé ze závislostí. Díky tomu se první dílčí kroky provedou pouze jednou při načtení komponenty, při dalších změnách jsou přepočítávány pouze zbylé dvě části, teoretický potenciál a navrhovaná změna,

protože v tomto případě hook nepotřebuje žádné závislosti.

- **Současný stav** – tato část zobrazuje přehled aktuální úrovně recyklace pro jednotlivé složky odpadu. Obsahuje tři sloupce. První udává množství aktuálně recyklovaného odpadu, druhý obsahuje přepočet na jednoho obyvatele a třetí nabízí porovnání aktuálních hodnot s podobnými obcemi.
- **Definice potenciálu** – tato část představuje cílové hodnoty recyklace potřebné k dosažení zadané míry separace. První sloupec udává požadované množství odpadu k recyklaci, druhý přepočet na obyvatele a třetí znázorňuje procentuální změnu vůči aktuálním hodnotám.
- **Změna** – poslední část slouží k explicitnímu zobrazení rozdílů mezi aktuálním stavem a cílovým návrhem. Jedná se o rozdíl prvních dvou sloupců z předchozích částí.

5.5.4 Modelování

Představuje poslední a zároveň nejdůležitější sekci v modulu. Jejím účelem je poskytnout uživateli nástroj pro interaktivní simulaci míry separace KO na základě vstupních parametrů definovaných v sekci Konfigurace vstupů.

Koncept i vizuální uspořádání této části zůstává v souladu s ostatními sekcemi modulu a je na obrázku 5.8. Uživatelské rozhraní je opět rozděleno na levou část s ovládacím panelem a pravou část obsahující výpočtový nástroj, který je realizován formou tabulky. Tabulka je navržena tak, aby se maximálně vešla do rámce hlavního obsahu, a bylo tak možné prezentovat všechna data bez nutnosti rolování.

Levé menu

Navigační panel zachovává jednotnou šířku a pozici jako ve všech ostatních sekcích modulu. Obsahuje dvě samostatné kartičky. Z nichž první slouží k přepínání jednotlivých kategorií definic potenciálu, které se promítají do zobrazeného výstupu v tabulce. Druhá kartička zobrazuje název aktuálně načtené konfigurace a aktivního scénáře. Zároveň obsahuje čtyři ovládací tlačítka s následujícími funkcionalitami.

- **Načíst** – funkce pro načtení dříve uložených scénářů modelování. Má jediné omezení, a to aby měl uživatel načtenou konfiguraci. V případě, že jsou k dispozici dříve uložené

FURUS

Modelování potenciálu

Obecné informace

Konfigurace vstupů

Návěh

Modelování

Definice Potenciálu

Zvýšení třídění - obecní SKO

Zvýšení třídění - netyp. zdroje SKO

Dotřídění ObjO

Dodatečná opatření

Načtená data

Konfigurace vstupů:

-

Modelování:

-

Načíst

Uložit

Reset

Export

Současný stav

0 %

Název odpadu	Katalogové číslo	Množství odpadu [t]	Netypické zdroje odpadu v obecním sběru [t]	Minimální množství odpadu z netypických zdrojů [t]	Množství KO na obyvatele [kg/os]
Papír	200101	0	0	0	0
Sklo	200102	0	0	0	0
Biolog. odpad z kuch.	200108	0	0	0	0
Oděvy	200110	0	0	0	0
Textilní materiály	200111	0	0	0	0
Jedlý olej a tuk	200125	0	0	0	0
Dřevo	200138	0	0	0	0
Plasty	200139	0	0	0	0
Kovy	200140	0	0	0	0
Biolog. odpad	200201	0	0	0	0
SKO	200301	0	0	0	0
ObjO	200307	0	-	-	0
Ostatní	-	0	-	-	0

Definice potenciálu

Zvýšení třídění - obecní SKO

Teoretický potenciál [t]	Cílové využití teoretického potenciálu [%]	Očekávaná změna [t]
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
-	-	-
-	-	-
-	-	-

Stav při naplnění potenciálu

0 %

Množství dopadu [t]	Množství KO na obyvatele [kg/os]
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

Obrázek 5.8 Stránka s modelováním potenciálu

(zdroj: vlastní)

scénáře, zobrazí se po kliknutí na tlačítko jejich seznam. Z něho je možné si buďto vybrat položku nebo ji smazat. Po výběru dané položky dojde k načtení jeho dat do tabulky i Redux stavu.

- **Uložit** – funkce pro uložení nového nebo upraveného scénáře. Nemá žádné zásadní omezení. Pro uložení je pouze nutné, jako v předchozím případě, mít načtenou konfiguraci, se kterou bude uložený scénář spárován. Po kliknutí na tlačítko se zobrazí dialogové okno pro zadání názvu po jehož potvrzení je pouze zkontrolována jeho unikátnost v rámci aktivní konfigurace a v případě shody je uživatel dalším dialogovým oknem dotázán, zda chce předchozí scénář přepsat.
- **Reset** – funkce pro resetování tabulky na výchozí hodnoty.
- **Export** – funkce pro export dat z modelování. Po stisknutí tlačítka se spustí stahování souboru ve formátu xlsx obsahující všechna data z celého modulu. Data jsou rozdělena do listů podle jednotlivých sekcí napříč celým modulem.

Panel modelování

I tato poslední část modulu se z hlediska návrhu uživatelského rozhraní drží jednotného designového i funkčního rámce, který je uplatněn v celém modulu. Celý dostupný prostor panelu opět vyplňuje tabulka tak, aby se její obsah nemusel rolovat.

Charakter tabulky je stejně jako v předchozí sekci Návrh, převážně statický, ovšem s tím rozdílem, že v tomto případě tabulka slouží jako nástroj pro aktivní modelování separace odpadu a obsahuje tím pádem ve vybraném sloupci interaktivní prvky v podobě číselných polí.

Výpočty v této sekci zajišťuje výpočetní třída `ModelingFormUtil`, která je strukturálně i funkčně analogická výpočetní třídě `SuggestionFormUtil` použité v sekci Návrh. Pro svou inicializaci vyžaduje sadu konstant, které jsou načítány z databáze. V případě, že tyto konstanty dosud nejsou v paměti aplikace dostupné, je renderování tabulky pozdrženo a uživateli je zobrazen spinner.

Po načtení všech potřebných dat je spuštěna inicializace pomocí hooku `useMemo`. Nejprve jsou na základě aktuální konfigurace načteny vstupní hodnoty. Poté je inicializována instance výpočetní třídy a následně je vypočten aktuální stav, který reprezentuje první část tabulky. Výpočet pro zbylé dvě části je již mimo hook `useMemo`.

- **Současný stav** – tato část zobrazuje reálná data o aktuální úrovni recyklace jednotlivých složek odpadu. Klíčovým ukazatelem je procentuální hodnota uvedená v záhlaví této části, která reprezentuje podíl odděleně soustředěvaných recyklovatelných složek vzhledem k celkové produkci komunálního odpadu.
- **Definice potenciálu** – tato část představuje jádro samotného modelování. Uživatel zde může interaktivně zadávat požadovanou míru využití teoretického potenciálu pro každou jeho definici. Název aktuálně vybrané definici je v záhlaví této části a jejich výběr se provádí pomocí navigačního menu v levém panelu.

Zadávané hodnoty jsou omezeny na rozsah 0–100 % a validace je zajištěna již na úrovni formulářové komponenty. Změna hodnoty vyvolává okamžitý přepočtení výsledků pomocí události `onChange`. Hodnoty jsou přímo svázány s objektem z `Redux` stavu, přičemž změny vedou k výpočtu a přerenderování pouze odpovídající části tabulky.

- **Stav při naplnění potenciálu** – tato část pak zobrazuje výsledné hodnoty produkce KO pokud by došlo k naplnění jejich zadaného potenciálu.

6 TESTOVÁNÍ

Cílem této kapitoly je zhodnotit funkčnost nově navrženého a implementovaného modulu na základě průběžného testování, identifikace chyb a návrhu možných optimalizací s ohledem na výkon a uživatelskou přívětivost aplikace.

V průběhu vývoje se nepodařilo plně zprovoznit aplikaci včetně nového modulu na vzdáleném testovacím serveru. Z tohoto důvodu bylo testování prováděno výhradně v lokálním vývojovém prostředí, kde byla dostupná veškerá funkcionalita FE i BE části včetně připojení k databázi.

Pro testování byly využity běžné nástroje webového vývojáře (např. DevTools) a ladicí výstupy frameworků (Next.js, Flask). Základní scénáře byly testovány ručně formou průchodu jednotlivými sekcemi aplikace v různých vstupních stavech.

6.1 Funkční testování

Během testování byly ověřeny následující klíčové funkce.

- **Zobrazení a editace vstupních údajů** – všechny formulářové komponenty akceptují data v předepsaném formátu, validace probíhá korektně a bez pádu aplikace. Až na drobnou výjimku u komponenty `NumberFormCheckField` popsanou v následující podkapitole.
- **Ukládání a načítání dat z databáze** – konfigurace i scénáře se úspěšně ukládají do příslušných tabulek a následně znovu načítají v původní podobě. Po editaci jsou záznamy konzistentní. Smazáním konfigurace se korektně smažou i přidružené scénáře. A naopak smazání daného scénáře neovlivní výchozí konfiguraci.
- **Práce s Redux stavem** – změny ve formulářích se správně propisují do globálního stavu a zároveň vyčítají napříč všemi sekcemi. Přičemž nedochází k nadbytečnému prerenderování, zacyklení nebo ztrátě dat.
- **Výpočtová logika** – jak nástroje návrhu, tak modelování pracovaly s korektně načtenými konstantami a poskytovaly výsledky odpovídající zadaným vstupům podle poskytnutých vzorových dat.

6.2 Uživatelské testování

Pro získání nezávislé zpětné vazby byla aplikace předložena externí osobě, která s modulem pracovala bez předchozí znalosti jeho struktury. Během neformálního testování byla sledována.

- srozumitelnost navigace mezi sekcemi
- logická návaznost kroků
- vizuální uspořádání a čitelnost výstupních dat

Zpětná vazba potvrdila, že aplikace působí jako přehledná a intuitivní, a to i bez předchozího školení. Drobná doporučení směřovala např. k přidání tooltipů s vysvětlením některých pojmů nebo většímu zvýraznění rozdílů mezi režimy zadávání dat.

6.3 Zjištěné nedostatky

V rámci testování bylo identifikováno několik drobných nedostatků.

- **Validace v komponentě NumberFormCheckField** – komponenta rozšiřující funkcionalitu číselného pole o kontrolu rozsahu zadané hodnoty ignoruje zadané hodnoty mimo daný rozsah. Což způsobuje problém při zadání například čísla 111 při s limitem spodní hranice 10. Uživatel sice může původní hodnotu smazat, ale skrz stanovené omezení mu není umožněno zadat požadovanou hodnotu. Tento problém byl opraven úpravou validace tak, aby probíhala až po opuštění pole. Tím uživatel může zadávat hodnotu v libovolném rozsahu, kdy při jeho překročení je upozorněn na neplatnou hodnotu.
- **Sjednocení chybových dialogových oken** – při neuložené konfiguraci se v sekci Modelování při pokusu o načtení scénáře objeví informativní dialogové okno s příslušnou hláškou. Zatímco při pokusu o uložení scénáře se objeví chybové dialogové okno stejně jako v sekci Konfigurace vstupů. Vyřešeno sjednocením na užití chybového dialogového okna pro větší pozornost uživatele.

6.4 Výkonnostní testování při omezené rychlosti připojení

V rámci testování výkonu nově implementovaných sekcí modulu byla jako sledovaná metrika zvolena doba vykreslení hlavních komponent po události *DOMContentLoaded*. Tato událost značí, že základní HTML dokument byl kompletně načten a parsován. Neznamená však, že aplikace je plně interaktivní, protože může stále probíhat stahování a spouštění JavaScriptových zdrojů nezbytných pro funkčnost komponent.

Je důležité zdůraznit, že cílem bylo posoudit rychlost zobrazení samotných modulových komponent, nikoliv celé aplikace jako celku. Do výsledků proto nejsou zahrnuty časy potřebné ke stažení rozsáhlejších bundle souborů (např. *main-app.js*, *layout.js*), které mohou výrazně prodloužit načítání při pomalém připojení.

Měření probíhalo prostřednictvím nástroje DevTools v prohlížeči Google Chrome s použitím profilů omezené propustnosti sítě (tzv. throttling) a jeho výsledky jsou zaznamenány v tabulce 6.1.

Tabulka 6.1 Výsledky testování výkonnosti

Sekce	Slow 4G [s]	Fast 4G [s]	No throttling [s]
Obecné informace	3,44	0,719	0,337
Konfigurace vstupů	3,59	0,694	0,473
Návrh	3,54	0,644	0,403
Modelování	3,56	0,772	0,461

(zdroj: vlastní)

Naměřené hodnoty ukazují, že vykreslení jednotlivých komponent probíhá při běžném připojení (bez omezení) do 0,5 sekundy, což zajišťuje plynulý přechod mezi sekcemi. I při použití profilu Fast 4G zůstává odezva pod hranicí 0,8 sekundy, což lze považovat za plně akceptovatelné. Při profilu Slow 4G se časy pohybují kolem 3,5 sekundy, kde je již vliv snížené přenosové rychlosti a vyšší latence velmi znatelný, ale i tak v limitu použitelnosti. Testy tak potvrzují, že nové komponenty modulu jsou z pohledu uživatelského rozhraní optimalizované a nenarušují běh aplikace jako celku.

6.4.1 Shrnutí

Testování ověřilo funkčnost většiny implementovaných částí a prokázalo stabilitu systému při běžném i zpomaleném síťovém provozu. Identifikované drobné nedostatky byly odstra-

něny a celkově lze novou funkcionalitu hodnotit jako stabilní, logicky navrženou a připravenou k produkčnímu nasazení po doplnění testovacího prostředí.

ZÁVĚR

Tato diplomová práce se zabývala rozšířením webové aplikace Popelka o nástroj pro zadání cílové míry separace a výpočet potřebného navýšení potenciálu jednotlivých složek odpadu. Součástí bylo také přepracování struktury modulu.

Úvodní část diplomové práce se zabývá problematikou odpadového hospodářství a formuluje body zadání této práce.

Celý obsah práce je strukturován do dvou hlavních částí – teoretické a praktické. Na začátku teoretické části je pojednáváno o problematice odpadového hospodářství v rámci České republiky a Evropské Unie. Dále byl představen matematický model a všechny jeho klíčové pojmy, které tvoří základ funkční logiky celého modulu. Ve zbytku teoretické části je nejprve představena webová aplikace Popelka a celý její projekt včetně historie, funkcionalit a popisu jednotlivých technologií.

Praktická část práce je zaměřena na konkrétní řešení jednotlivých úkolů vyplývajících ze zadání. Pro BE byla navržena a implementována struktura databáze pro ukládání uživatelských dat včetně rozšíření API o nové endpointy. Na straně FE byl původní fragmentovaný model správy stavu nahrazen centralizovaným slice modulem. Uživatelské rozhraní bylo přepracováno, aby sekce byly nově uspořádány podle logiky zpracování dat. Vstupy jsou nyní soustředěny do sekce *Konfigurace vstupů* s možností jejich uložení a načtení, přibyla sekce *Návrh* pro výpočet cílové míry separace a sekce *Modelování* byla rozšířena o podporu a správu scénářů. Všechny vizuální prvky byly sjednoceny, zjednodušena interakce s formuláři a upravena validace pro větší uživatelskou přívětivost.

Funkcionalita byla následně testována z hlediska správnosti výpočtů, stability a uživatelské použitelnosti. Výsledky testování ukázaly, že nově přidané části systému jsou stabilní a funkčně správné.

Výsledná podoba modulu tedy naplnila všechny cíle stanovené v zadání diplomové práce. Nově navržené řešení působí jako celek moderně, přehledně a umožňuje uživateli intuitivní ovládání při práci s daty. I když je tak modul připraven k nasazení do reálného prostředí, přesto existuje prostor pro jeho další rozvoj. Například ve formě rozšíření o grafické výstupy pro jednodušší analýzu výsledků, návrhů parametrů na základě podobných obcí nebo optimalizaci klientské části pro provoz v prostředí se zhoršeným připojením.

SEZNAM POUŽITÉ LITERATURY

- [1] MÜLLEROVA, Lucie, 2022. Třídění musí být pohodlné. Online. *EKO-KOM*. 2022. Dostupné z: https://www.ekokom.cz/wp-content/uploads/2022/04/2303_Special_DNES.pdf. [cit. 2025-02-16].
- [2] PROSTŘEDÍ, Ministerstvo, 2020. *Komunální odpady*. Online. Dostupné z: https://www.mzp.cz/cz/komunalni_odpady. [cit. 2025-02-16].
- [3] ZEMĚDĚLSTVÍ, Ministerstvo, 2020. *Odpadové hospodářství a jeho hierarchie*. Online. Dostupné z: <https://mze.gov.cz/public/portal/mze/legislativa/vap95309-458>. [cit. 2025-02-17].
- [4] COMMISSION, European, 2015. *First circular economy action plan*. Online. Dostupné z: https://environment.ec.europa.eu/topics/circular-economy/first-circular-economy-action-plan_en. [cit. 2025-02-22].
- [5] COMMISSION, European, 2023. *Circular economy action plan*. Online. Dostupné z: https://environment.ec.europa.eu/strategy/circular-economy-action-plan_en. [cit. 2025-02-22].
- [6] VNITRA, Ministerstvo, 2020. Zákon č. 541/2020 Sb. Online. *Elektronická Sbírka zákonů a mezinárodních smluv*. 2020. Dostupné z: <https://www.e-sbirka.cz/sb/2020/541?zalozka=text>. [cit. 2025-02-22].
- [7] EKOLOGIE, Redakce Komunální, 2024. Nový plán odpadového hospodářství pro následující roky je v mezipřehledu. Online. *Komunální ekologie*. 2024. Dostupné z: <https://www.komunalniekologie.cz/info/novy-plan-odpadoveho-hospodarstvi-pro-nasledujici-roky-je-v-mezipřehledu>. [cit. 2025-02-22].
- [8] PROSTŘEDÍ, Ministerstvo, 2021. *Vyhláška o Katalogu odpadů a posuzování vlastností odpadů (Katalog odpadů)*. Online. Dostupné z: <https://www.mzp.cz/www/platnalegislativa.nsf/%24%240penDominoDocument.xsp?documentId=26B2B93E9CCDE5B0C125865B002C4914&action=openDocument>. [cit. 2025-02-22].

- [9] PROSTŘEDÍ, Ministerstvo, 2020. *Odpady*. Online. Dostupné z: https://www.mzp.cz/cz/odpady_podrubrika. [cit. 2025-02-22].
- [10] ŠOMPLÁK, Kopa M. Omelka M. Nevrlý Vlastimil. a Pavlas M. R., 2022. Multi-level stratification of territories for waste composition analysis. Online. *Journal of environmental management*. 2022. ISSN 0301-4797. Dostupné z: <https://www.sciencedirect.com/science/article/abs/pii/S0301479722011070>. [cit. 2025-02-22].
- [11] NIEDOBA, Marek, 2024. *Implementace výpočtového modulu do webové aplikace*. Online, diplomová práce. Vysoké učení technické v Brně. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=270378. [cit. 2023-02-22].
- [12] TEAM, Codecademy, [b.r.]. Back-End Web Architecture. Online. *Codecademy*. [b.r.]. Dostupné z: <https://www.codecademy.com/article/back-end-architecture>. [cit. 2025-03-01].
- [13] VIDLIČKA, Adam, 2023. *Webová aplikace pro plánování rozmístění sběrných míst odpadu*. Online, diplomová práce. Vysoké učení technické v Brně. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=256406. [cit. 2023-03-01].
- [14] KARAGIANNAKOS, SERGIOS, 2020. *How to use uWSGI and Nginx to serve a Deep Learning model*. Online. Dostupné z: <https://theaisummer.com/uwsgi-nginx/>. [cit. 2025-04-02].
- [15] FOUNDATION, Python Software, [b.r.]. *What is Python? Executive Summary*. Online. Dostupné z: <https://www.python.org/doc/essays/blurb/>. [cit. 2025-03-01].
- [16] FLASK, [b.r.]. *Flask's documentation*. Online. Dostupné z: <https://flask.palletsprojects.com/en/stable/>. [cit. 2025-03-01].
- [17] KARAGIANNAKOS, Sergios, 2020. *How to use uWSGI and Nginx to serve a Deep Learning model*. Online. Dostupné z: <https://theaisummer.com/uwsgi-nginx/>. [cit. 2025-03-01].

- [18] FOUNDATION, MariaDB, [b.r.]. *About MariaDB Server*. Online. Dostupné z: <https://mariadb.org/about/>. [cit. 2025-03-01].
- [19] S. GILLIS, HUGHES Adam a LUTKEVICH Ben A., 2024. Database. Online. *TechTarget*. 2024. Dostupné z: <https://www.techtarget.com/searchdatamanagement/definition/database/>. [cit. 2025-03-02].
- [20] S. GILLIS, a BRUSH Kate A., 2024. RDBMS (relational database management system). Online. *TechTarget*. 2024. Dostupné z: <https://www.techtarget.com/searchdatamanagement/definition/RDBMS-relational-database-management-system/>. [cit. 2025-03-02].
- [21] ERKEC, Esat, 2022. 5 Best Practices for writing SQL queries. Online. *SQLShack*. 2022. Dostupné z: <https://www.sqlshack.com/5-best-practices-for-writing-sql-queries/>. [cit. 2025-03-02].
- [22] REDIS, [b.r.]. *Redis Docs*. Online. Dostupné z: <https://redis.io/docs/latest/>. [cit. 2025-03-02].
- [23] FOUNDATION, Python Software, [b.r.]. *Redis Queue*. Online. Dostupné z: <https://python-rq.org/>. [cit. 2025-03-02].
- [24] DISSANAYAKE, Kasun, 2021. What is Nginx? Online. *Medium*. 2021. Dostupné z: <https://medium.com/geekculture/what-is-nginx-2edfdad3722b/>. [cit. 2025-03-01].
- [25] UWSGI, [b.r.]. *The uWSGI project*. Online. Dostupné z: <https://uwsgi-docs.readthedocs.io/en/latest/>. [cit. 2025-03-01].
- [26] LAZUARDY, Dyah M. F. S. a. A., 2022. Modern Front End Web Architectures with React.Js and Next.Js. Online. In: *International Research Journal of Advanced Engineering and Science*. Roč. 7, s. 132–141. ISSN 2455-9024. Dostupné z: <https://irjaes.com/wp-content/uploads/2022/02/IRJAES-V7N1P162Y22.pdf>. [cit. 2023-03-01].

- [27] VASANI, Darshan, 2024. Is React a Library or a Framework? Online. *DEV Community*. 2024. Dostupné z: <https://dev.to/dpvasani56/is-react-a-library-or-a-framework-1o19/>. [cit. 2025-03-02].
- [28] SOURCE, Meta Open, [b.r.]. *Welcome to the React community*. Online. Dostupné z: <https://react.dev/>. [cit. 2025-03-02].
- [29] DESHPANDE, Chinmayee, 2024. The Best Guide to Know What Is React. Online. *Simplilearn*. 2024. Dostupné z: <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs/>. [cit. 2025-03-02].
- [30] ROUTER, React, [b.r.]. *React Router*. Online. Dostupné z: <https://reactrouter.com/>. [cit. 2025-03-02].
- [31] INC., Tailwind Labs, [b.r.]. *Rapidly build modern websites without ever leaving your HTML*. Online. Dostupné z: <https://tailwindcss.com/>. [cit. 2025-03-04].
- [32] ZOD, [b.r.]. *Zod dev*. Online. Dostupné z: <https://zod.dev/>. [cit. 2025-03-04].
- [33] REDUX, [b.r.]. *Redux*. Online. Dostupné z: <https://redux.js.org/>. [cit. 2025-03-09].
- [34] REDUX, [b.r.]. *Three Principles*. Online. Dostupné z: <https://redux.js.org/understanding/thinking-in-redux/three-principles/>. [cit. 2025-03-09].
- [35] REDUX, [b.r.]. *Redux Fundamentals, Part 3: State, Actions, and Reducers*. Online. Dostupné z: <https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers/>. [cit. 2025-03-09].
- [36] TARANTO, Jack, 2022. A powerful React + Redux Toolkit pattern (reuseable state slices). Online. *Previous Next*. 2022. Dostupné z: <https://www.previousnext.com.au/blog/powerful-react-redux-toolkit-pattern-reuseable-state-slices/>. [cit. 2025-03-02].
- [37] DVALASHVILI, DAVIT, 2024. *Exploring Redux with React: From Basics to Advanced with Redux Toolkit*. Online. Dostupné z: <https://davitdvalashvili1996.medium.com/exploring-redux-from-basics-to-advanced-with-redux-toolkit-3c3b2feb8b85>. [cit. 2025-04-02].

- [38] AIDASANI, Chanchal, 2023. Redux Devtools - How To Use, Getting Started. Online. *Browsee*. 2023. Dostupné z: <https://browsee.io/blog/redux-devtools-how-to-use-getting-started/>. [cit. 2025-03-02].

SEZNAM OBRÁZKŮ

Obrázek 2.1	Hlavní stránka aplikace se seznamem modulů	31
Obrázek 2.2	Kombinace spojení webové aplikace, NGINX a uWSGI	33
Obrázek 2.3	Princip Redux pipeline	40
Obrázek 4.1	Zjednodušená souborová struktura modulu modelování potenciálu	48
Obrázek 5.1	Zjednodušený diagram rozšíření stávající databáze	55
Obrázek 5.2	Zkrácená struktura tabulky konstant shlukové skupiny pro město	56
Obrázek 5.3	Struktura tabulky konfigurace vstupů	57
Obrázek 5.4	Struktura tabulky scénářů	59
Obrázek 5.5	Stránka s obecnými informacemi	63
Obrázek 5.6	Stránka s konfigurací vstupů	64
Obrázek 5.7	Stránka s návrhem pro modelování potenciálu	68
Obrázek 5.8	Stránka s modelováním potenciálu	71

SEZNAM TABULEK

Tabulka 1.1	Katalogový přehled KO	22
Tabulka 1.2	Katalogový přehled ostatní KO	23
Tabulka 1.3	Popis matematického modelu	29
Tabulka 6.1	Výsledky testování výkonnosti	75

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BE	Backend
CEAP	Circular Economy Action Plan
CEP	Circular Economy Package
CRUD	Create Read Update Delete
CSR	Client-Server Rendering
DBMS	Database Management System
FE	Frontend
GUI	Graphical User Interface
KO	Komunální odpad
MVC	Model View Controller
Obj	Objemný odpad
POH ČR	Plán Odpadového Hospodářství České republiky
RDBMS	Relational Database Management System
RTKQ	Redux Toolkit Query
SEO	Search engine optimization
SKO	Směsný komunální odpad
SSR	Server-Side Rendering
SQL	Structured Query Language
ÚPI	Ústav procesního inženýrství
VUT	Vysoké učení technické