VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

WEBOVÁ APLIKACE PRO PLÁNOVÁNÍ SVOZU FRAKCÍ ODPADU

WEB APPLICATION FOR PLANNING WASTE FRACTIONS COLLECTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Jiří Kubowský

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Ladislav Dobrovský

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Jiří Kubowský**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Ladislav Dobrovský**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Webová aplikace pro plánování svozu frakcí odpadu

Stručná charakteristika problematiky úkolu:

V posledních letech byla vyvinuta řada algoritmů pro různé aplikace v problematice nakládání s odpadem, ať jde o rozmístění kontejnerů, výstavbu zařízení na zpracování odpadu, nebo nalezení optimálních tras pro svozová vozidla. Dalším krokem je převedení těchto výsledků do praxe. Moderní webová aplikace by měla splňovat určité požadavky na responsivitu a oddělení kódu na klientovi a serveru pomocí zasílání asynchronních požadavků (AJAX) nebo aktivním duplexním spojením (WebSockets). Aplikace bude postavena na reálných datech a jednotlivé prvky aplikace budou konzultovány s odborníky z Ústavu procesního inženýrství.

Cíle diplomové práce:

Rešerše využití poskytnutého backendu (Flask) a rozšíření o relevantní endpointy a moduly.
Rešerše a zvolení vhodných frontend technologií v kooperaci s dalším studentem.
Vytvoření webové aplikace, která bude soužit uživatelům (jednotlivým obcím, svazům obcí, či firmám) při plánování svozu jednotlivých frakcí v odpadovém hospodářství:
Návrh schematu databáze s ER diagramem, jak společné části aplikace, tak specifické pro problém svozu.
Mockupy pro vstupní i výstupní API.
Propojení s externím výpočetním jádrem.
Uživatelské rozhraní orientované na práci s myší či dotykovou obrazovkou většího formátu (monitor, větší tablet).
Práce s mapovými podklady.

Seznam doporučené literatury:

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.

HAKLAY, M. Openstreetmap: User-generated street maps. IEEE Pervasive Computing [online]. 2008, 7(4), 12 [cit. 2021-10-21]. ISSN 1536-1268.

W3 schools: AJAX Introduction [online]. [cit. 2021-10-21]. Dostupné z: https://www.w3schools.com/xml/ajax_intro.asp

OpenStreetMap documentation: Deploying your own Slippy Map [online]. [cit. 2021-10-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Deploying_your_own_Slippy_Map

Flask-SocketIO [online]. [cit. 2021-10-21]. Dostupné z: https://flask-socketio.readthedocs.io/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem této závěrečné práce je vytvořit webovou aplikaci pro podporu plánování svozu odpadu s důrazem na přívětivé uživatelské rozhraní a variabilitu z pohledu nastavitelnosti okrajových podmínek úlohy. Vývoj aplikace navazuje na vytvořené modely a algoritmy z problematiky nakládání s odpadem, přináší pro koncové uživatele nástroj, který jim umožní nad těmito daty, modely, a algoritmy pracovat. V teoretické části je přiblížena problematika třídění a svozu odpadu. V praktické části je řešen vývoj webové aplikace od základního konceptu přes použité technologie až po detaily samotné implementace.

ABSTRACT

The aim of this thesis is to develop a web application to support the planning of waste collection with an emphasis on a user-friendly interface and variability in terms of configurability of task boundary conditions. The development of the application builds on models and algorithms developed in the field of waste management. The application will be a tool for end users to work with these data, models, and algorithms. In the theoretical part, the issues of waste sorting and collection are presented. The practical part deals with the development of the web application from the basic concept, through the technologies used, to the details of the actual implementation.

KLÍČOVÁ SLOVA

webová aplikace, odpadové hospodářství, komunální odpad, svozu odpadu, svazky obcí

KEYWORDS

web application, waste management, municipal solid waste, waste collection, municipal unions

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

KUBOWSKÝ, Jiří. *Webová aplikace pro plánování svozu frakcí odpadu*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022, 62 s. Diplomová práce. Vedoucí práce: Ing. Ladislav Dobrovský

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Ladislavovi Dobrovskému za odbornou pomoc a vedení při tvorbě této práce. Dále bych rád poděkoval pracovníkům Ústavu procesního inženýrství, jmenovitě Ing. Vlastimírovi Nevrlému, Ph.D., Ing. Radovanu Šomplákovi, Ph.D. a Ing. Veronice Smejkalové za pomoc a součinnost při tvorbě této práce. Také bych chtěl poděkovat Bc. Adamu Vidličkovi za příjemnou spolupráci při vývoji webové aplikace.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

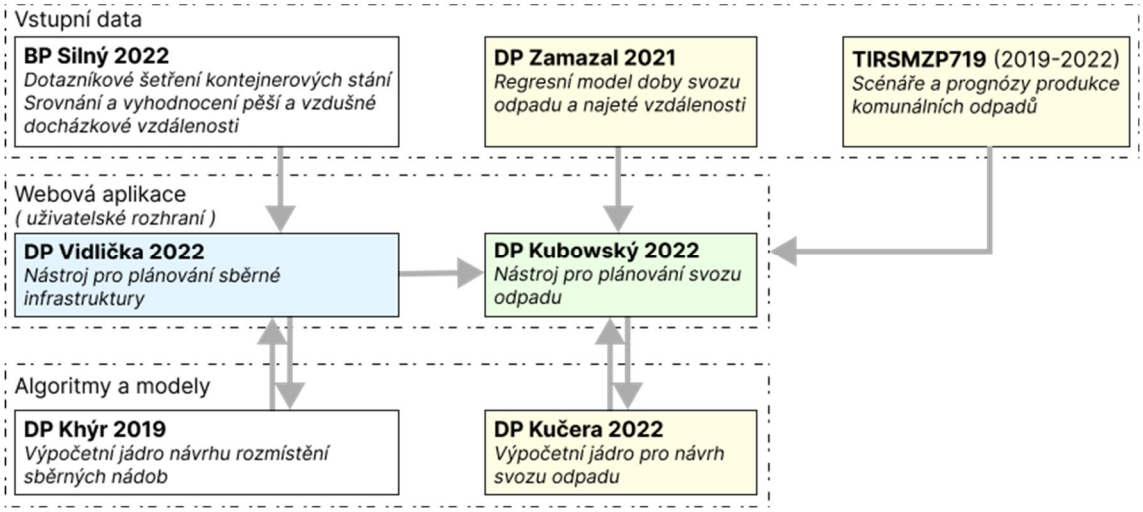
Bc. Jiří Kubowský

OBSAH

1 ÚVOD

Cílem této závěrečné práce je vytvořit webovou aplikaci pro podporu plánování svozu odpadu. S ohledem na komplexnost problematiky bude tato aplikace integrovat existující optimalizační nástroje a statistické analýzy dat. Vzniklá aplikace umožní komunikaci mezi jednotlivými subsystémy. Finálními výstupy aplikace budou přehledné grafové, mapové a tabulkové výstupy. Velký důraz bude kladen na přívětivé uživatelské rozhraní, variabilitu z pohledu nastavitelnosti okrajových podmínek úlohy (tvorba svazků obcí, vozového parku, frekvence svozu, …). Výsledky budou uplatnitelné dlouhodobě z důvodu jejich obecnosti a možnosti nastavovat okrajové podmínky na míru regionu, svazku obcí nebo obci.

Na Ústavu procesního inženýrství (ÚPI) se dlouhodobě vyvíjejí přístupy pro optimalizaci a statistické vyhodnocení dat z odpadového hospodářství. Práce s jednotlivými modely vyžaduje vhodnou softwarovou podporu (často s licencí), erudovanou obsluhu a je často náročná na přípravu dat. Integrace jednotlivých přístupů do komplexního nástroje (webové aplikace) zvyšuje výpočtovou efektivitu a především rozšiřuje využitelnost. Díky webové aplikaci je možné výstupy promítat v aktuálním čase do reálného provozu v rámci nakládání s odpady. Jednotlivé přístupy a dílčí výsledky vznikají v rámci spolupráce ÚPI a Ústavu matematiky prostřednictvím závěrečných studentských prací a výzkumných projektů. Schéma popisující vzniklý komplexní systém (nástroj) pojmenovaný *Popelka* je zobrazeno na obrázku 1.



Obr. 1: Schéma komponent nástroje Popelka

Pozn.: U diplomových prací řešených v roce 2022 se jedná o rok předpokládaného dokončení (nelze zaručit jejich ukončení v tomto termínu). Zeleně je zvýrazněna tato práce, modře práce s překryvem a žlutě práce, jejichž výstupy jsou využívány.

Programovací jazyk bude zvolen dle uvážení a možnosti komunikace s výpočetním jádrem. Vstupní data budou zahrnovat informace o infrastruktuře odpadového

hospodářství konkrétní obce a produkci odpadu. Dopravní infrastruktura může být popsána pomocí mapových podkladů. V rámci obsluhy softwaru bude možné volit množství produkovaného odpadu (změnu od aktuálního trendu produkce, viz [13]) pro uvažované frakce odpadu (SKO, papír, plast, sklo, bioodpad, textil, objemný odpad, kovy, tuky a jedlé oleje). Dalším volitelným parametrem by měla být specifikace vozového parku. Časové podmínky svozu na úrovni obce budou modelovány na základě reálných dat od partnerských svazků obcí (TSMH, KTS Ekologie a DSO Tišnovsko). Svazky obcí bude možné stanovovat libovolně s omezením na celkové množství (nižší stovky) z důvodu výpočtové náročnosti optimalizační úlohy.

Z pohledu ekonomiky budou vyhodnoceny náklady na svoz, jako jsou pohonné hmoty, mzdy a přeprava odpadu do zpracovatelského zařízení. Další funkcionalitou nástroje bude vyhodnocení časového a kapacitního vytížení vozidel v jednotlivých dnech svozového plánu. Úloha bude zaměřena na obce v České republice v plném rozsahu (6258 obcí).

Úloha bude uvažována na meziobecní úrovni, takže každá obec bude agregována do jednoho bodu, který bude odhadnutými parametry (doba svozu v obci, najetá vzdálenost) reprezentovat svoz mezi sběrnými nádobami. Vstupní údaje do modelů návrhu svozu vyžadují odhady okrajových podmínek pomocí regresních modelů [27,18]. Dále je nutné připravit matice vzdáleností a časů, stejně jako parametry související s vozidly, pracovní dobou, sypnou hmotností odpadu či frekvenci svozu. Tyto všechny informace vstupují do výpočtového nástroje na plánování svozu odpadu. Náročnost tvorby výstupu spočívá ve výpočtové náročnosti, jelikož se jedná o kombinatorický problém.

Spolupráce obcí umožňuje efektivněji plánovat infrastrukturu odpadového hospodářství a řídit celý provozní systém (lepší vyjednávací ceny, omezení marží svozových firem, …). Aby bylo možné se touto cestou vydat, je nutné disponovat podpůrnými výpočtovými nástroji. Tato závěrečná práce si klade za cíl implementovat podpůrný systém, který umožní grafický a tabulkově ekonomický pohled na tvorbu svazků obcí na základě plánování a vyhodnocení svozu odpadu.

Hlavní cíle závěrečné práce zahrnují implementaci a návrh struktury webové aplikace obsahující následující body:

- možnost výběru libovolné množiny obcí pro vytvoření svazku
- specifikace způsobu soustřeďování jednotlivých frakcí komunálního odpadu, volbu frekvence svozu a výběr očekávané produkce odpadu
- komunikace s výpočtovým jádrem pro návrh svozových plánů
- grafické zobrazení svozových tras na meziobecní úrovni pro vybrané odpadové proudy a obce
- vyhodnocení technických a ekonomických parametrů svozu odpadu pro zvolený svazek obcí
- odhad nákladovosti celého systému

Výsledný software bude možné využívat pro libovolně zvolené svazky obcí ze všech obcí v České republice. Dále bude možné volit frakce komunálního odpadu a pro ně změny v produkci oproti základnímu scénáři. Volitelná bude i kapacita svozových vozidel. Obecností softwaru je zajištěna jeho široká aplikovatelnost a představuje tím jedinečný nástroj pro návrhy svozových plánů v odpadovém hospodářství na meziobecní úrovni.

Meziobecní spolupráce umožňuje optimalizovat materiálové (odpadové) a finanční toky, sdílet fixní náklady obecních a regionálních systémů, realizovat úspory z rozsahu, snižovat transakční náklady řízení a další. Tyto benefity však nemusí být na první pohled patrné a je proto předmětem práce zobrazit a vyčíslit ekonomické náklady, které plynou z meziobecní spolupráce a jejím prostřednictvím optimalizovat obecní a regionální systémy nakládání s odpady.

Udržitelnost existujícího svazku a případné zapojení nového člena do svazku je nutné posoudit z pohledu ekonomického přínosu. Slučování obcí do větších svazků má také přímý dopad na koncová zařízení pro zpracování odpadu. Softwarová podpora pro plánování svozu v rámci svazků obcí by přispěla samotným svazkům při rozhodování o případné expanzi svazku do dalších obcí, tvorbě nových uskupení a současně zpracovatelskému zařízení pro plánování provozu.

Samotný vývoj webové aplikace je společným projektem této práce a práce řešící plánování sběrné infrastruktury[1], části a komponenty specifické pro jednotlivé diplomové práce byly řešeny nezávisle, společné části jsme řešili v kooperaci a rozdělili jsme si jejich vytvoření.

[1] VIDLIČKA, Adam. Webová aplikace pro plánování rozmístění sběrných míst odpadu. Diplomová práce.

2 PROBLEMATIKA TŘÍDĚNÍ A SVOZU ODPADU

Nakládání s odpady je veřejná služba občanům, kterou zajišťuje místní samospráva. Aktuálně obce České republiky obvykle využívají služeb nabízených specializovanými svozovými společnostmi, často fungujícími na nadnárodní úrovni. Pozice samostatné obce je pro vyjednávání s velkými korporátními společnostmi nevýhodná a výsledkem jsou často vysoké náklady na nakládání s odpady v dané obci [6]. Po vzoru odpadových svazků obcí ze zahraničí vznikají také obdobná uskupení na území České republiky. Příkladem dobré praxe může být skupina sdružení pro nakládání s KO ARGE – Österreichischer Abfallwirtschaftsverbände z Rakouska [1]. Výhody pro členy svazku plynou z šetření administrativních nákladů, spolupráce v rámci svazku, úspora nákladů na svoz atd. Historická data ukazují pozitivní dopad na způsob nakládání s odpady při spolupráci obcí v rámci svazku COVAR 14 v Itálii [3]. V regionu svazku COVAR 14 byl za tři roky existence svazku zaznamenán nárůst vyseparovaného komunálního odpadu z 23 % na 63 %. Cíle a kompetence jednotlivých svazků jsou rozdílné. V některých případech vznikají svazky za účelem sdílení informací, zastupování zájmů členských obcí, vedení jednání se svozovými společnostmi a zpracovatelskými zařízeními. Existují však i svazky obcí, které převzaly veškerou zodpovědnost za sběr a dopravu odpadu. Takové svazky obcí jsou vybaveny vlastními sběrnými nádobami a vozovým parkem [12], příp. jsou i provozovateli zpracovatelských zařízení [8].

Podle zkušeností svazků obcí České republiky stále roste zájem obcí o zakládání nových svazků zajišťujících svoz odpadu, nebo přidání se k již fungujícím svazkům. Při každém zapojení nové obce nebo dokonce vytvoření nového svazku je nutné vyřešit nastavení vhodné infrastruktury a vyhodnocení proveditelnosti při současných technických parametrech (počet zaměstnanců, vozový park apod.). Sdružování do svazků také umožní snáze implementovat potřebné změny odpadového hospodářství Evropské unie s cílem podpořit materiálové využití komunálního odpadu [10] a minimalizovat jeho skládkování [9]. Cíle EU jsou již implementovány do legislativy České republiky [15].

Hlavním cílem svazků obcí je zajistit svoz odpadu z obcí do místa jeho zpracování. Odpad vyprodukovaný občany je shromažďován na sběrných místech, odkud je svážen v rámci obce a následuje meziobecní/meziměstská přeprava až do zařízení ke zpracování odpadu. Optimalizací svozových tras je možné dosáhnout snížení nákladů a dopadů na životní prostředí [7]. Této problematice se věnovalo kromě odborných publikací i více výzkumných projektů v České republice. Projekt Výzkumný a vývojový projekt IT Cluster 2016-2019 [5] se věnoval optimalizaci svozu odpadu se zaměřením na snížení celkových nákladů. Svoz odpadů s cílem snižování nákladů byl řešen také v projektu Modelování logistiky odpadů v městských aglomeracích [2]. V článku [11] byl zkoumán více-typový svoz tříděného odpadu na delší časový horizont (měsíc). K řešení byla využita třífázová heuristika skládající se z geografického přiřazení k vozidlům, rozdělení jednotlivých frakcí odpadu ke konkrétním dnům měsíce a z plánování svozových tras. V současnosti se praktikuje svoz více komodit najednou, jejichž charakteristické vlastnosti to umožnují (převážně pro různé barvy skla). Jedná se

o specifickou úlohu, obzvláště pokud se uvažuje flexibilní velikost jednotlivých nádob. Přehled existujících a nově vyvíjených variant řešení problémů svozu odpadu byl představen v rozsáhlé rešerši [14]. Často jsou zmiňovány heuristické přístupy. Nové modely berou v úvahu různé cíle a metriky výkonnosti. Ty mohou integrovat vyhodnocení svozu s dalšími taktickými rozhodnutími. Jejich cílem by mělo být také zachytit detaily, které představují zásadní aspekty moderních dopravních řetězců. Rešerše [14] tak ukazuje na současné nedostatky, poslední směřování vývoje a hlavně nové výzvy při návrhu a vyhodnocování svozových úloh. Většinou však představené přístupy řeší pouze formulaci modelu, vývoj a implementaci heuristiky, či konkrétní případovou studii. Sofistikovanější a obecný softwarový nástroj umožňující modelování a vyhodnocování svozu na meziobecní úrovni není mimo vysoce nákladové komerční řešení tzv. „na míru“ k dispozici.

Vznikající svazky obcí ovlivňují i zpracovatelská zařízení odpadu. Aby mohlo zpracovatelské zařízení rychle reagovat na změny, musí mít představu o možných projekcích vývoje z pohledu tvorby nových struktur ve vyjednávacích procesech s vlastníky odpadu. Stejně podstatná je i znalost budoucího vývoje charakteru odpadu z pohledu výhřevnosti a složení. Tyto informace umožní např. pro zařízení na energetické využití odpadu (ZEVO) robustně plánovat svůj provoz, reinvestice a volbu technologických úprav subsystémů. Navíc ZEVO se již dnes potýká s problémem velmi výhřevných odpadů především z průmyslových firem, což snižuje kapacitu a zásadně zasahuje do ekonomiky provozu.

3 POUŽITÉ TECHNOLOGIE

Pro backend (serverová část aplikace) byl zvolen programovací jazyk Python, pro frontend (část aplikace v prohlížeči) jsme se rozhodli použít pouze základní jazyky pro tvorbu webu (HTML, JS, CSS). U backendu se vycházelo z množiny společně známých programovacích jazyků. U frontendu jsme se zaměřili na jednoduchost a minimalizaci zatížení uživatelských zařízení, respektive webových prohlížečů, ve kterých bude aplikace nahlížena.

3.1 BACKEND

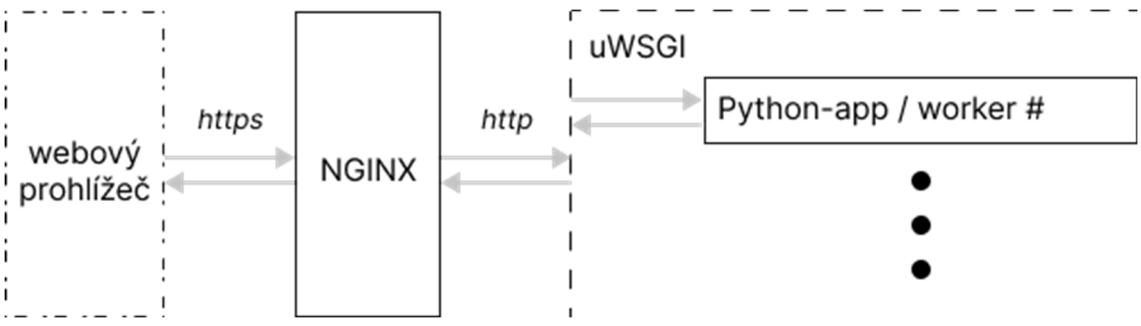
Vedoucím práce byl poskytnut základní backend, ze kterého vychází jádro nové aplikace. Proces přetvoření původního poskytnutého backendu na jádro nové aplikace se skládal z více kroků. Nejprve bylo rozhodnuto, které části budou převzaty zcela nebo jen s drobnými úpravami. Z těchto částí bylo vytvořeno základní jádro, které jsme dále rozšiřovali o nové moduly.

Převzato bylo i použití frameworku Flask, jedná se sice o jeden z méně robustních webových frameworků, ale pro tuto aplikaci je dostačující a jeho výhodou je relativně jednoduchá rozšiřitelnost.

Samotná aplikace, obdobně jako poskytnutý backend, bude využívat uWSGI pro rozložení zátěže ze strany uživatelů mezi více instancí a bude schována za NGINX proxy, se kterou bude komunikovat srze síťový nebo souborový socket. V rámci této práce jsme službu nasadili pod operačním systémem Linux a použili souborovou variantu socketu.

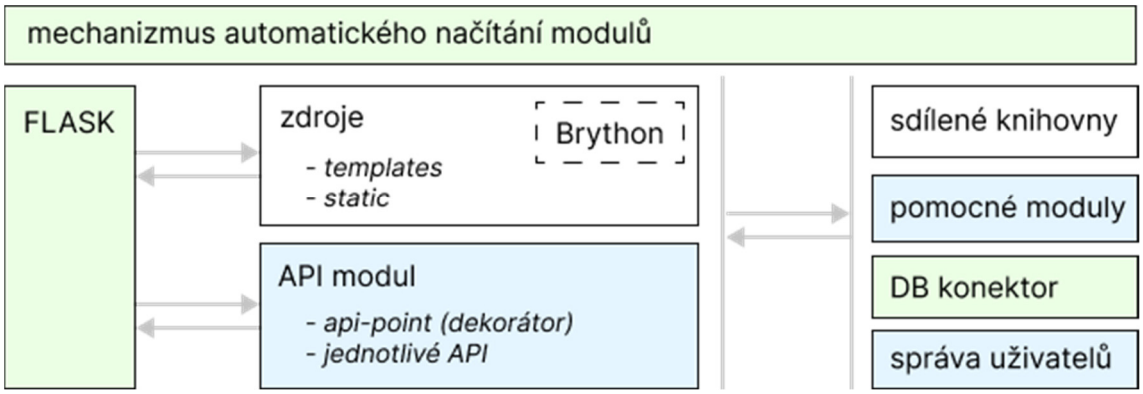
Ze zamýšleného modelu implementace vyplynula potřeba společné paměti pro jednotlivé instance. Pro zajištění dostupnosti dočasných a uživatelských dat, která bude potřeba držet v paměti bez ukládání do klasické databáze, bylo rozhodnuto o využití databázového serveru Redis. Tato služba bude uchovávat potřebná data v paměti a zajistí jejich dostupnost bez ohledu na to která instance aplikace vyřizuje požadavek uživatele. Výhodou této volby je i možnost jednoduše implementovat do backendu frontu úloh RedisQueue pro řešené výpočty.

3.1.1 POSKYTNUTÝ BACKEND



Obr. 2: Implementační model poskytnutého backendu

Aplikace předpokládala nasazení se službami NGINX a uWSGI, jak je naznačeno na obrázku 2. Služba NGINX zpřístupňuje aplikaci klientům a doplňuje komunikaci o SSL/TLS šifrování, komunikace mezi NGINX a aplikací probíhá přes socket. Pomocí uWSGI je spouštěno více instancí aplikace, mezi které je rozložena zátěž uživatelských požadavků. Tento způsob implementace byl převzat i pro naši webovou aplikaci.



Obr. 3: Obecná struktura poskytnutého backendu

Na obrázku 3 je nastíněna obecná struktura aplikace, barevně jsou vyznačeny části, které byly využity při tvorbě jádra vyvíjené aplikace. Zeleně jsou vyznačeny části, které byly převzaty, modře části, které byly převzaty částečně s úpravami. Rozsah úprav převzatých částí je popsán v dalších kapitolách této práce.

3.1.2 Flask

Flask je WSGI framework pro webové aplikace, napsaný v jazyce Python, postavený na knihovně Werkzeug (obsáhlá WSGI knihovna pro webové aplikace). Využívá šablonový engine Jinja (šablonový engine inspirovaný šablonovým systémem frameworku Django). Nenutí nás do specifické implementace nebo použití jen vybraných knihoven, ale naopak nám v tomto ohledu dává svobodu zvolit si podle svých potřeb. Lze ho použít s dalšími rozšířeními i v komplexních aplikacích a je navržen tak, aby bylo co nejjednodušší ho použít v projektu bez jeho předchozí znalosti. Aktuálně se jedná o jeden z nejpopulárnějších frameworků pro webové aplikace v Pythonu. [19]

3.1.3 uWSGI

uWSGI je webový aplikační server implementující protokol uwsgi. Je podporován NGINX a běžně nasazován s webovými aplikacemi vytvořenými v Pythonu. V rámci komunikace s Python aplikacemi je implementováno WSGI podle specifikace PEP 3333, tím je zajištěna podpora běžných Python webových frameworků, do kterých patří i Flask. V základu je již podporováno více programovacích jazyků (Python, Perl, Ruby, …) a protokolů, přičemž podpora dalších jazyků, protokolů a platforem může být dodána díky rozšiřitelné architektuře. [21]

3.1.4 Redis

Redis je open-source úložiště datových struktur v paměti, běžně využívané jako databáze, mezipaměť, zprostředkovatel zpráv a další. Může s daty pracovat pouze v paměti, nebo v případě potřeby i pravidelně ukládat na disk aktuální verzi pro dlouhodobé uložení. Nad uloženými daty lze provádět atomické operace a k stejným datům přistupovat bezpečně z více míst nebo aplikací. [22]

3.1.5 Redis Queue

Redis Queue je jednoduchá Python knihovna řešící aplikační frontu, která využívá Redis server. Předané výkonné metody jsou prováděny v pozadí pomoci workrů. Knihovna je zaměřena na jednoduchost implementace do projektů.

Implementací aplikační fronty je na výběr více, tato konkrétní byla zvolena pro svoji jednoduchost.

3.1.6 NGINX

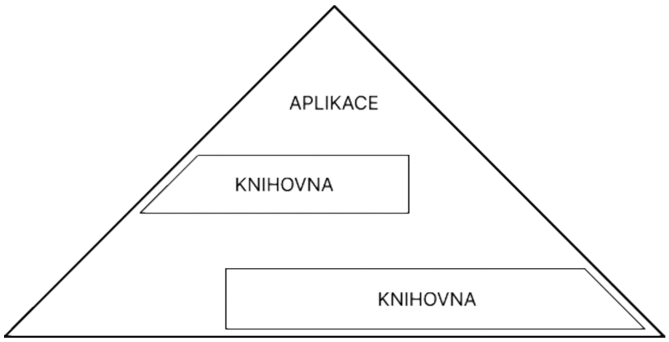
NGINX je open-source webový server s řízením zátěže a funkcionalitou reverzní proxy. Je podporováno více protokolů (HTTP, HTTPS, SMTP, SSL, …). Služba se snaží minimalizovat zatížení HW a zajistit stabilní a rychlé obsloužení klientů. V nejjednodušší variantě může být nasazeno jako samostatný webový server, nebo naopak můžeme využít různých funkcionalit i v kombinaci. Službu lze instalovat na všech základních platformách, verze pro Linux je k dispozici zdarma. [20]

3.2 FRONTEND

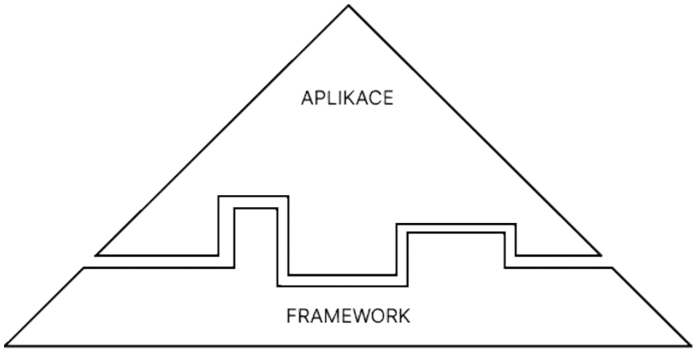
Součástí poskytnutého backendu byl i základní frontend využívající Brython. Tuto část jsme se rozhodli nepoužít a vydali jsme se cestou frontendu postaveného pouze na JavaScriptu. Tato varianta je i při použití knihoven pro tvorbu grafického uživatelského rozhraní méně náročná na koncové zařízení (webové prohlížeče) než využití obdobných knihoven umožňujících použití dalších programovacích jazyků (v tomto případě Python), jejichž kód je pak překládán do klasického JavaScriptu pro prohlížeč za běhu.

Pro tvorbu uživatelského rozhraní jsme se rozhodli nejít cestou GUI frameworku, mezi jehož známější zástupce patří například AngularJS, ale cestou využití knihoven.

Základní rozdíl mezi frameworkem a knihovnou je ve způsobu, jakým se s nimi pracuje. Knihovna jako taková nezasahuje do řízení aplikace a je spíše poslušným dělníkem, který vykonává přidělenou práci a zároveň nás neomezuje v tom, jaké další knihovny budeme chtít použít. Framework nám přináší více možností za cenu toho, že do řízení aplikace zasahuje a poskytuje v některých ohledech méně svobody. Použití frameworku můžeme přirovnat k výstavbě na základech, které vznikly nezávisle na našem projektu.



Obr. 4: Skladba aplikace s knihovnami



Obr. 5: Skladba aplikace postavené na frameworku

V případě frameworku bychom sice měli k dispozici robustnější nástroj, ale také bychom byli více omezeni při psaní vlastního kódu a vytváření jednotlivých uživatelských komponent. I samotný rozsah frontendu webové aplikace nedosahuje rozměrů, pro které by byl framework lepší volbou. Bylo tedy pro nás výhodnější jít cestou knihoven, které máme více pod kontrolou. Konkrétně jsme se rozhodli použít knihovnu React, a to z důvodu její flexibility a relativní jednoduchosti.

Pro mapové podklady jsme se rozhodli použít OpenStreetMap, jejichž využití pro nás bylo nejvhodnější z pohledu dostupnosti a případných licenčních omezení. Pro jejich implementaci do aplikace jsme použili open-source knihovnu Leaflet. Pro propojení s funkcionalitou knihovny React, jsme využili nástavbu React Leaflet, která přináší React komponenty propojující React a Leaflet.

3.2.1 OpenStreetMap

OpenStreetMap je projekt zaměřený na vytváření a šíření geografických dat z celého světa, která je možné dále použít bez komplikovaných právních omezení. Mapové podklady z mapových serverů lze využít v podstatě pro libovolný účel, povinností je pouze uvést autorství OpenStreetMap a přispěvatelů. O aktuálnost dat se stará rozsáhlá komunita a v případě potřeby se lze zaregistrovat a přispět vlastními daty. [26]

3.2.2 Leaflet

Leaflet je open-source knihovna, napsaná v jazyku JavaScript, pro práci s mapovými podklady. Navržena byla se zaměřením na jednoduchost, výkonnost a responzibilitu.

Funkcionalitu je možné rozšířit pomocí pluginů. Jako zdroj mapových podkladů lze použít v podstatě jakýkoli mapový server, v tomto ohledu můžou být omezující spíše podmínky licenčních ujednání konkrétních mapových serverů. Z těchto důvodů se často používá v kombinaci s mapovými podklady OpenStreetMap. [25]

3.2.3 React

React je knihovna pro tvorbu uživatelského rozhraní, napsaná v jazyku JavaScript. V samotném návrhu knihovny se počítá s tím, že v projektu může být použita jen malá část, vybrané části dle potřeby nebo celá knihovna. Jde o značně flexibilní nástroj. React lze kombinovat s klasickým JavaScript kódem a do jisté míry takto i ovlivňovat chování samotných komponent. Pro urychlení uživatelského rozhraní se v Reactu pracuje s virtuální DOM strukturou, se kterou je synchronizována DOM struktura v prohlížeči. Změny se nejprve propíšou v paměti a následně je zajištěna aktualizace v místě změn. Tento přístup zrychluje překreslování v prohlížeči a zlepšuje responzibilitu v případě větších zásahů do GUI. Pomocí knihovny se tvoří React komponenty, které jsou obdobou klasických HTML elementů. Stejně jako u klasických HTML elementů lze tyto komponenty vnořovat do sebe a skládat z nich větší celky. Komponenty lze vytvářet více způsoby, lze je definovat jako třídu nebo jako funkci. [23]

3.2.4 React Leaflet

React Leaflet je rozšíření pro Leaflet, které přináší React komponenty zapouzdřující Leaflet funkcionalitu. Je důležité vést v patrnosti, že tyto komponenty nenahrazují původní Leaflet implementaci, ale pouze ji zabalí do React komponent, které lze použít v uživatelském rozhraní tvořeného pomocí React knihovny. Není tedy potřeba pro tento účel vytvářet vlastní komponenty. [24]

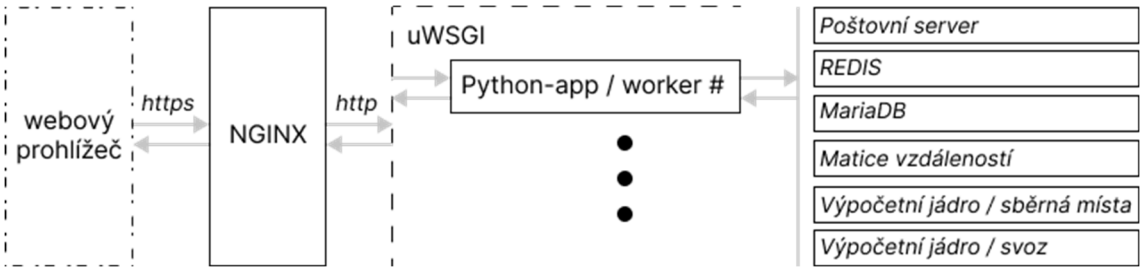
4 NÁVRH APLIKACE

Návrh aplikace vyšel z požadavků na funkcionalitu a strukturu ukládaných dat. Samotný koncept a způsob ukládání a načítání dat prošel během této diplomové práce vlastním vývojem. Původně se počítalo s ukládáním všech dat v databázi, ale během vývoje aplikace se ukázalo jako výhodnější pro část dat použít jiný způsob uložení a následného načítání.

Kromě samotné aplikace byly postupně vytvořeny pomocné skripty, které mohou být použity například pro nahrávání dat do databáze.

4.1 ZÁKLADNÍ KONCEPT

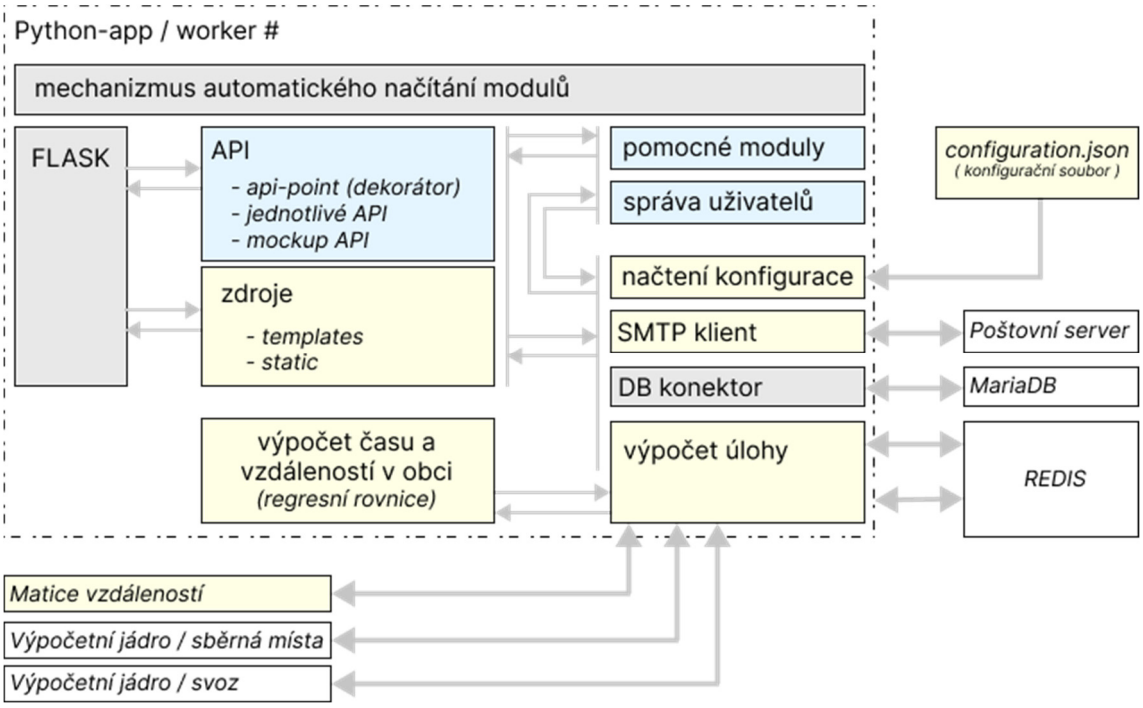
Základní koncept a návrh aplikace vychází z poskytnutého backendu a požadavků na vyvíjenou aplikaci. Model prošel postupně několika úpravami na základě poznatků, potřeb a omezení, které vyplynuly během vývoje. Finální model implementace a struktury aplikace je na naznačen na obrázcích 6 a 7.



Obr. 6: Implementační model aplikace

U aplikace se předpokládá implementace s několika dalšími službami nebo moduly. Následující body popisují služby a moduly, které jsou společné pro obě diplomové práce, nebo jsou specifické pro plánování svozu odpadu:

- **NGINX** – zpřístupnění aplikace do internetu a doplnění šifrování do komunikace mezi aplikací a klientem (webový prohlížeč)
- **uWSGI** – rozložení zátěže ze strany uživatelů mezi více workrů (instancí aplikace)
- **Poštovní server** – je využíván pro e-mailovou komunikaci s uživateli a se zájemci o přístup do aplikace
- **REDIS** – využíván jako „sdílená“ paměť mezi jednotlivými workry, a také pro implementaci aplikační fronty
- **MariaDB** – databáze využívaná pro dlouhodobé ukládání dat aplikace
- **Matice vzdáleností** – služba vytvořená jako součást této práce pro poskytování submatic vzdáleností mezi obcemi (více v kapitole 4.2.2)
- **Výpočetní jádro / svoz** – použité výpočetní jádro, které je výstupem další DP [17]



Obr. 7: Obecná struktura aplikace

Na obrázku 7 je struktura aplikace s barevným odlišením jednotlivých částí podle jejich původu. Šedě jsou části zcela převzaté z backendu poskytnutého vedoucím DP, modře části částečně převzaté a upravené (podrobněji v kapitole 3.3) a žlutě nové moduly.

4.2 DATABÁZE

Na začátku byl stanoven předpoklad, že veškerá dlouhodobá data budou ukládána v relační perzistentní databázi, za tímto účelem byla vybrána databáze MariaDB. Následně byla na základě popisu dat, se kterými se bude pracovat, a sady konkrétních dat poskytnutých formou XML souborů navržena první verze ER (Entity Relationship) diagramu. Ta prošla postupně několika úpravami. Kromě vlivu změnou rozsahu ukládaných dat došlo k úpravě na základě rozhodnutí přesunout z databáze matice vzdáleností mezi obcemi a využít jiný způsob jejich uložení a načítání.

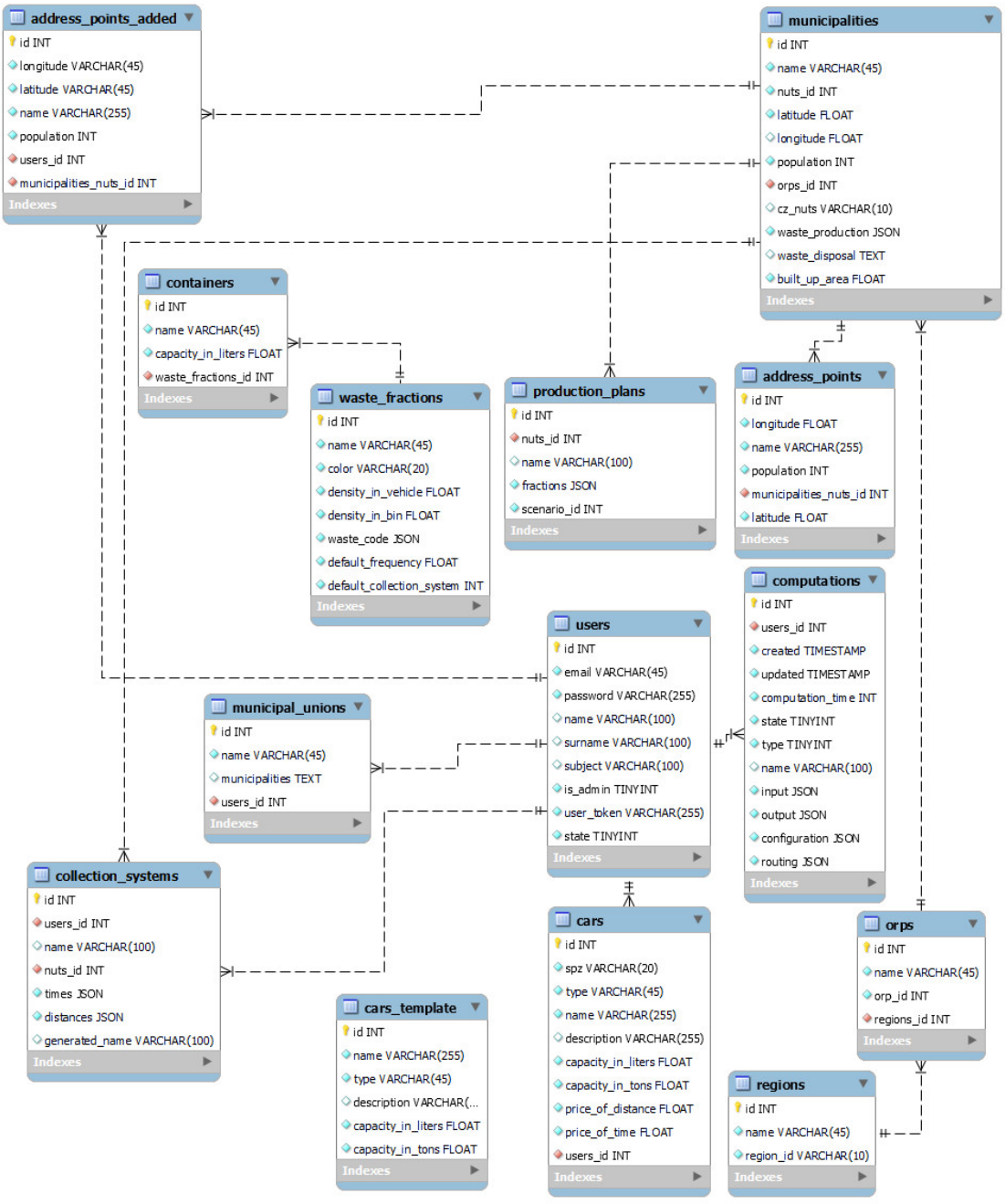
Matice vzdáleností jsou v systému dvě, jedna pro čas přejezdu a druhá pro vzdálenosti v kilometrech. Obě tyto matice jsou předpočítané pro všechny kombinace výchozí a koncové obce, jedná se tedy o čtvercové 6259x6259, počítáme-li i hlavičku. V obou případech se jedná o velký objem dat, ze kterého je při každém výpočtu potřeba jen relativně malá vybraná část. Pro představu, pokud bychom tato data uložili jako CSV soubor, tak máme soubory o velikosti cca 225MB a 260MB. V případě jednoduché databázové tabulky o třech sloupcích (od, do, hodnota) by šlo i v případě, že uvažujeme pouze neduplicitní záznamy, o necelých 20 miliónů záznamů.

Při volbě způsobu načítání a ukládání těchto matic bylo stanoveno několik požadavků:

- minimalizace předávaných/zasílaných dat (pro každý výpočet je relevantní jen vybraná část)
- minimalizace času mezi požadavkem na data a jejich získáním
- ukládání způsobem umožňujícím relativně jednoduchou aktualizaci hodnot

Za účelem zprostředkování konkrétních submatic pro výpočty byla vytvořena jednoduchá služba, se kterou bude aplikace komunikovat přes socket a která bude aplikaci poskytovat požadované submatice.

4.2.1 ER diagram



Obr. 8: ER diagram

Návrh struktury databáze byl vytvořen v kooperaci s přidruženou diplomovou prací[1], část modelu jsme řešili jednotlivě a část společně v závislosti na překryvu dat, se kterými budeme pracovat.

Rozdělení tabulek podle využití

Tab. 1: rozdělení tabulek databáze podle využití

„skupina“	seznam tabulek
sdílená data	regions, orps, municipalities, waste_fractions, production_plans
data uživatelů	users, municipal_unions, computations, cars
sběrná místa	containers, address_points, address_points_added
svoz odpadu	collection_systems, car_templates

4.2.2 Služba zprostředkovávající submatice vzdáleností

Obr. 9: Komunikace mezi aplikací a službou

Aplikace komunikuje se službou přes socket. Součástí služby je konfigurační INI soubor, ve které lze mimo jiné nadefinovat, jestli komunikace bude probíhat přes síťový nebo souborový socket. Požadavek i návrat jsou zasílány formou jednoduché JSON struktury za požití znakové sady UTF-8.

JSON požadavku

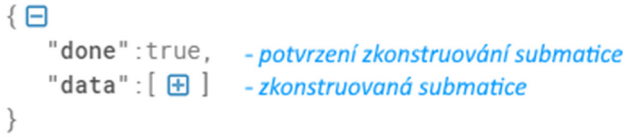
```
{ ⊟
   "nuts":[ ⊞ ],     - seznam NUTS ID vybraných obcí
   "type":"times"    - typ matice
}
```

Obr. 10: JSON pro dotaz na submatici

- **nuts** – seznam unikátních id obcí (nuts id), pro které požadujeme submatici, je možné zadat dvěma různými způsoby, jako seznam hodnot nebo jako textový řetězec s hodnotami oddělenými čárkou
- **type** – typ matice, ze které budou hodnoty načteny, v našem případě jde o matici časů nebo vzdáleností

Jak bude popsáno dále, jednotlivé názvy typů matic odpovídají pro jednoduchost názvům CSV souborů, ze kterých byly inicializovány matice v paměti služby.

JSON odpovědi

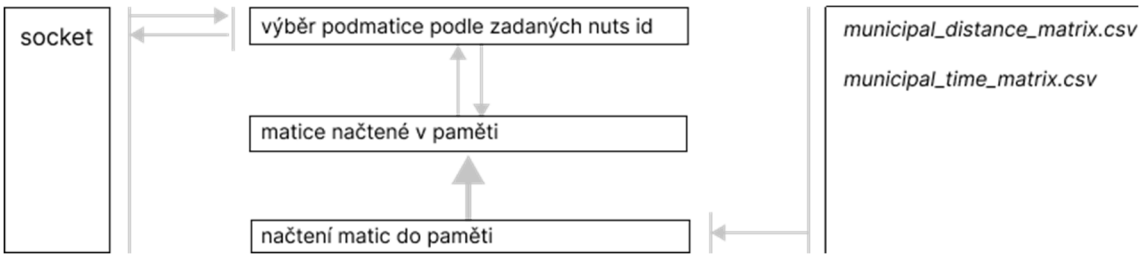


Obr. 11: JSON odpovědi se submaticí

- **done** – logická hodnota potvrzující, jestli se pro požadovaný typ matice a vybraná nuts id podařilo zkonstruovat submatici
- **data** – submatice pro vybraná nuts id včetně vlastní hlavičky nebo null v případě, že submatici nelze zkonstruovat

Vrácená submatice, která je čtvercová a úplná, je v paměti reprezentována jako vektor řádků, každý řádek je vektorem konkrétních hodnot. V prvním řádku a prvním sloupci je hlavička matice formou jednotlivých nuts id, mezi kterými chceme znát konkrétní vzdálenost. V úhlopříčce matice, pro kterou platí, že index řádku je roven indexu sloupce, jsou hodnoty nulové.

Návrh služby



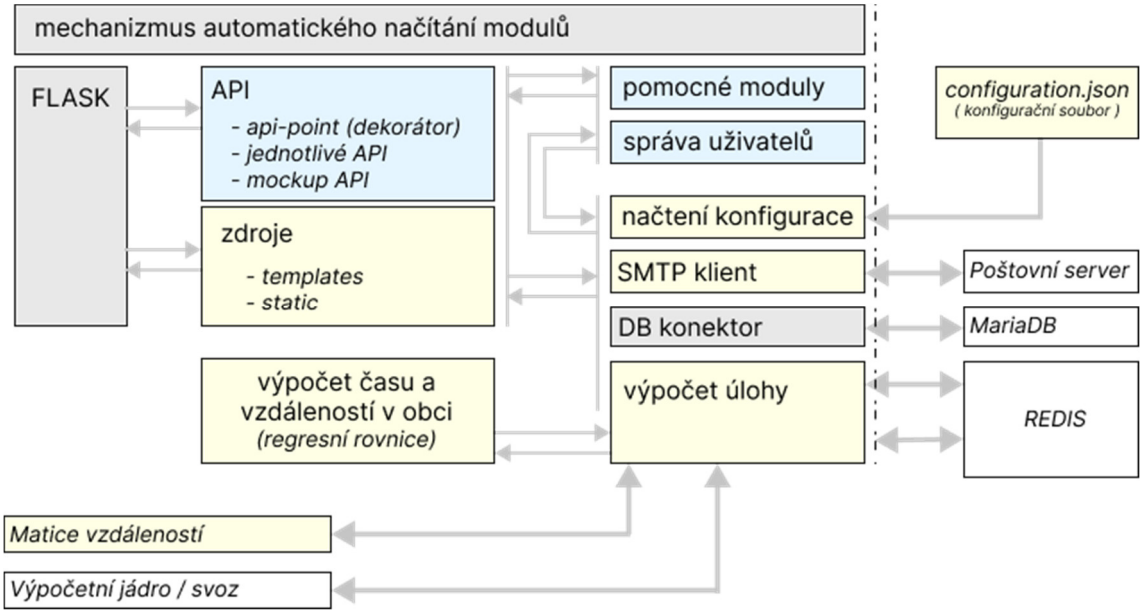
Obr. 12: Schéma služby pro zprostředkování submatic vzdáleností

Základní funkčnost služby je nastíněna na obrázku 12. Pro trvalé uložení matic, se kterými se bude pracovat, byl zvolen formát CSV souborů, které lze v případě potřeby relativně jednoduše aktualizovat nebo rozšířit. V konfiguračním INI souboru, který je součástí služby, je umístěno základní nastavení a seznam matic, které se mají načíst. U každé matice je definován typ (jako součást názvu sekce s využitím tečkové notace) a cesta k CSV souboru s daty matice. Matice jsou při spuštění služby automaticky načteny do paměti pro následný rychlejší přístup k jejich datům. Tento mechanizmus byl navrhnut tak, aby byl v případě potřeby jednoduše rozšiřitelný o další specifické matice.

Pro testování a případné nasazení s rozšířenou sadou matic je obsluhující třída připravena pracovat v režimu „přímého čtení“, ve kterém se data nepřednačítají do paměti, ale jsou vždy čtena přímo z odpovídajících CSV souborů. To nám umožní minimalizovat nároky na paměť za cenu častějšího přístupu na disk. Pro přechod do tohoto režimu je nutná úprava parametru ve zdrojovém kódu služby.

Služba poslouchá na socketu a jakmile na něj přijde validní JSON struktura specifikující požadavek na submatici konkrétního typu, je z dat načtených v paměti vyrobena odpovídající submatice a zaslána jako součást JSON odpovědi.

4.3 BACKEND



Obr. 13: částečné schéma aplikace (zaměřeno na backend)

Backend je postaven na serveru Flask a je navržen modulárně, aby ho v případě potřeby bylo možné relativně jednoduše rozšiřovat. Pro ukládání dlouhodobých dat se využívá databáze MariaDB a pro ukládání dočasných dat REDIS, který se také používá pro implementaci aplikační fronty.

V průběhu vývoje aplikace se postupně přešlo z modelu „uživatel na pozvání“ na model „schvalované registrace“, to se stalo jedním z důvodů, proč musí aplikace umět zasílat e-maily. Pro tento účel bylo rozhodnuto implementovat SMTP klientský modul. Z pohledu hodnocení e-mailových zpráv kvůli eliminaci spamu, má vliv server přes který se zprávy zasílají, jak mají konstruovanou hlavičku a další parametry. Implementace vlastního poštovního serveru by tedy byla spíše zdrojem komplikací, a nejlepší bude použít některého ze zavedených velkých serverů skrze zřízenou e-mailovou schránku.

Zdroje aplikace patří pod frontend a jsou popsány v kapitole 4.4.

4.3.1 Převzaté moduly

Mezi moduly zcela převzaté z backendu poskytnutého vedoucím práce patří databázový konektor a mechanizmus automatického načítání modulů.

Databázový konektor je třída postavená na knihovně pymysql, jejíž implementace byla pro účely této aplikace dostačující.

Mechanizmus automatického načítání modulů je souhrnným názvem pro kód rozšiřující každý __init__.py soubor pro automatické načtení všech vnořených modulů. Následně není nutné volat import pro každý z nich, ale postačí pouze import hlavního modulu.

4.3.2 Základní konfigurace

Pro uložení záhladní konfigurace byl zvolen formát JSON uložený v souboru *configuration.json*, tento soubor je při spuštění aplikace načítán pomocí modulu zajišťujícího načtení základní konfigurace. Pokud soubor z nějakého důvodu neexistuje, je automaticky vytvořen s defaultními hodnotami.

```
{
   "dbhost":"127.0.0.1",        - adresa DB serveru
   "dbuser":"...",              - DB uživateské jméno
   "dbpassword":"...",          - DB uživatelské heslo
   "dbdatabase":"popelka",      - název databáze
   "session_timeout":1,         - platnost uživivatelské session v hodinách
   "mockup":true,               - zapnutí mockup api pro API server
   "smtphost":"smtp.gmail.com", - adresa SMTP serveru
   "smtpport":465,              - SMTP port
   "smtpuser":"...",            - SMTP uživatelské jméno
   "smtppassword":"..."         - SMTP uživatelské heslo
}
```

Obr. 14: Struktura konfiguračního souboru

4.3.3 Pomocné moduly

Pomocné moduly byly z části převzaty a z části vytvořeny. Převzaty byly pomocné metody pro práci s JSON strukturou a úpravu relativních url pro webové rozhraní. Doplněny byly pomocné metody pro uživatelské rozhraní (generování náhodných hesel, …).

4.3.4 Správa uživatelů

Jako základ byl použit modul z backendu poskytnutého vedoucím práce, ten byl následně zjednodušen a upraven pro potřeby aplikace. Struktura uživatelských práv byla zjednodušena a přepracována. Nově mají uživatelé pouze přidělena konkrétní oprávnění a nemají oproti původnímu modelu uživatelskou skupinu, samotná struktura třídy reprezentující uživatele byla přepracována, aby obsahovala potřebné údaje.

```python
class acc_user:
    name: str
    surname: str
    subject: str
    login: str
    id: int
    isadmin: bool
    rights: set
    token: str

    def __init__(self, name, surname, subject, login, id, isadmin, rights, token):
        self.name = name
        self.surname = surname
        self.subject = subject
        self.login = login
        self.id = id
        self.isadmin = isadmin
        self.rights = rights
        self.token = token
```

Obr. 15: Třída reprezentující jednoho uživatele

Správa uživatelů je propojena s API mechanizmem, zde jsou oproti původnímu modulu jen drobné úpravy vycházející z upravené struktury uživatelských práv a odebrání uživatelských skupin, upraven byl také název cookies, ve které se ukládá dočasný „token" přihlášeného uživatele. Každý příchozí API dotaz se zkontroluje na přítomnost cookie „utoken". Pokud je načtena, ověří se její platnost a načtou se data případného přihlášeného uživatele. Informace o uživateli jsou předány v rámci vyhodnocování volané API nebo dotazu na konkrétní „stránku".

Propojení správy uživatelů s API rozhraním bylo rozšířeno o mechanizmus, který automaticky maže z paměti aplikace „tokeny" u kterých nebyla v poslední hodině potvrzena platnost. Tu lze potvrdit přes API „keep-alive". Tato API je po přihlášení do aplikace mezi frontendem a backendem automaticky volána v definovaných intervalech.

4.3.5 Výpočet úlohy

Modul pro výpočet úlohy je ve své podstatě rozdělen na dva samostatné celky, část řešící plánování sběrných míst, která je součástí překrývající se DP, a část řešící plánování svozu, na kterou se zaměřuje tato práce.

Zadání vygenerované na základě vstupu uživatele ve webovém rozhraní se ve formátu JSON předá na backend přes API rozhraní. Následně požadavek na výpočet přebírá modul pro výpočet úlohy, samotné zpracování úlohy má 2 fáze, každá o několika krocích.

Modul pro výpočet úlohy využívá regresních rovnic, zajišťuje doplnění zadání od uživatele o údaje, které jsou pro výpočet důležité a nejsou zadávány uživatelem. Vyžádá si od oddělené služby submatice vzdáleností a dopočítá pomocí regresních rovnic kolik v každé obci stráví vůz času a jakou urazí vzdálenost. K samotnému výpočtu je využito výpočetní jádro, které je do aplikace vloženo jako uzavřený Python modul, výpočet úlohy přes aplikační frontu zavolá asynchronně metodu jádra, vstupem je zadání formou JSON struktury, návrat metody jádra je také ve formátu JSON. Zadání i vypočítaný výsledek se ukládají v databázi.

Fáze 1, přišlo zadání od uživatele:

- Na základě zadání od uživatele je vytvořeno rozšířené zadání pro výpočetní jádro:
  - načtení dalších hodnot potřebných pro výpočet z databáze
  - načteny submatice vzdáleností mezi obcemi (čas, vzdálenost)
  - vypočten čas a vzdálenost v obci pro každý „uzel" pomocí regresních rovnic
  - vytvoření rozšířeného zadání (JSON)
- Do databáze je uložen záznam o probíhajícím výpočtu včetně zadání (i rozšířeného).
- Přes aplikační frontu je zavolán výpočet nad výpočetním jádrem.

Fáze 2, byl dokončen výpočet úlohy:

- V databázi je záznam výpočtu doplněn o výsledky a aktualizován jeho stav.
- Systému je předána informace, že se má notifikovat v GUI uživatel o dokončeném výpočtu.

4.3.6 Výpočet času a vzdálenosti v obci

Pro výpočet celé úlohy a naplánování co nejoptimálnějšího svozu odpadu pro zvolený svazek obcí, je vzhledem k agregaci obcí do jednoho bodu, potřeba co nejpřesněji odhadnout jakou vzdálenost budeme muset najet a kolik času v obci strávíme při svozu konkrétní frakce odpadu. Tento odhad vychází ze systému rozmístění sběrných míst, velikosti obce, populace a dalších parametrů. Pro tento účel používá aplikace regresní rovnice.

V případě, že není dostupné rozmístění sběrných míst, spočítají se výchozí odhady pro počet sběrných míst v obci podle zvoleného systému sběru (funkce *RoundIntegerUp* zaokrouhlí hodnotu na celé číslo nahoru):

$$CentralizovanyMist = RoundIntegerUp\left(\frac{pocetAdresnichMist}{800}\right) \quad (1)$$

$$DistribucniMist = RoundIntegerUp\left(\frac{pocetAdresnichMist}{80}\right) \quad (2)$$

$$DoorToDoorMist = pocetAdresnichMist \quad (3)$$

Pro modelování času byl využit zobecněný lineární model (Generalized linear model – GLM), konkrétně s gama rozdělením (gama regrese) s identickou spojovací funkcí. Na první pohled z diagnostických grafů nebylo vidět zásadní porušení předpokladů pro použití dané metody. Využité prediktory jsou uvedeny a popsány v tabulce 2.

Tab. 2: Popis prediktorů

prediktor(y)	popis
odpadPAP, odpadPLA, odpadSKL, odpadSKO	Proměnné pro papír, plast, sklo a SKO nabývající hodnot 0/1. Pro svoz konkrétního typu odpadu bude nejvýše zapnuta jedna proměnná (pokud se bude modelovat svoz pro bioodpad, budou všechny 0 – stejný postup je využit i pro ostatní typy odpadu nevyskytující se v rovnici).
zastavkySKO	Počet zastávek pro sběr SKO, které mají být v obci svezeny (pokud se sváží jiný odpad, tak zastavkySKO=1).
runDist	Součet diferencí vzdáleností mezi jednotlivými zastávkami (využita funkce Haversine).
areaPerKm	Plocha konvexního obalu všech zastávek v obci (svážených bodů) podělena vzdáleností runDist.
zastavenaCLK	Zastavěná plocha a nádvoří (zdroj ČSÚ).

Výslednou dobu svozu v hodinách lze pak odhadnout na základě rovnice 4. Rovnice na první pohled vypadá stejně jako pro lineární regresi, avšak koeficienty byly odhadnuty jiným způsobem.

$$\begin{aligned} dobaSvozu = \; & 1{,}052 \cdot 10 \; - 3{,}629 \cdot odpadPAP - 3{,}403 \cdot odpadPLA \\ & + 2{,}875 \cdot 10^{-1} \cdot odpadSKL - 3{,}967 \cdot odpadSKO + 7{,}057 \\ & \cdot 10^{-2} \cdot zastavkySKO + 4{,}549 \cdot 10^{-3} \cdot runDist - 4{,}406 \\ & \cdot 10^{-5} \cdot areaPerKm \; + \; 1{,}288 \cdot 10^{-3} \cdot zastavenaCLK \end{aligned} \quad (4)$$

Koeficient determinace podle Nagelkerkeho vychází 0,917 a RMSE (root-mean-square error) dosahuje hodnoty 11,261. Model dosahuje relativně vysoké přesnosti a je pro zvolenou aplikaci dobře uplatnitelný.

Odhad najeté vzdálenosti je prováděn na základě následujících vztahů pro centralizovaný, distribuční a door-to-door systém (výsledná vzdálenost je v metrech):

$$CentralizovanyKm \; = \; RoundIntegerUp\left(\frac{pocetObyvatel}{2000}\right) \cdot \; 1000 \quad (5)$$

$$DistribucniKm \; = \; pocetAdresnichMist \cdot \frac{300}{80} \quad (6)$$

$$DoorToDoorKm = \; pocetAdresnichMist \cdot 50 \quad (7)$$

4.3.7 API

API modul rozšiřuje Flask o URL obsluhující jednotlivé API. Pro příchozí dotazy je povolena pouze metoda POST, při každém dotazu je ověřován uživatel a pokud to API vyžaduje, jsou ověřována potřebná uživatelská práva. Odpověď na API je vždy JSON struktura, parametry zaslané v rámci požadavku jsou také předávány jako JSON struktura.

/api/<api-point>

API modul kromě standardních návratů definovaných API umí pracovat i s mockup daty, pokud je v konfiguraci aplikace povolen mockup režim. Mechanizmus implementující mockup API byl navržen a vytvořen jako rozšíření základní obsluhy API za účelem umožnit relativně jednoduchou rozšiřitelnost o testovací data v návratech jednotlivých API. V případě příchozího dotazu ověří dostupnost mockup dat, následně provede návrat těchto dat nebo standardní vyhodnocení API, v případě že nejsou definována. Definice mockup dat je navržena pro jednoduchou rozšiřitelnost, potřeba je pouze přidat do složky „mockup“ definici návratových dat formou JSON souboru pojmenovaného podle API. Mechanizmus mockup API umožňuje testování funkcionalit i v případě, že standardní

vyhodnocení API není ještě implementováno. V rámci jednoduchosti nejsou u mockup dat kontrolována uživatelská práva, ta jsou navázána až na reálnou implementaci konkrétní API.

Obr. 16: Návratový JSON pro API

Pozn.: parametr data je nepovinný a není u některých API přítomen

Tab. 3: API / operace nad databází

API	přístup	parametry	návrat
/api/editindbbyid editace nebo přidání záznamu do databáze	pouze admin	• **table** – název tabulky • **fields** – hodnoty jednotlivých sloupců formou JSON objektu (vlastnost je názvem sloupce) • **rowid** – „new“ nebo unikátní ID záznamu	• **success** – informace o dokončení • **id** – null nebo unikátní ID nového záznamu
/api/delindbbyid smazání záznamu z databáze	pouze admin	• **table** – název tabulky • **rowid** – unikátní ID záznamu	• **success** – informace o dokončení

Tab. 4: API / vozový park

API	přístup	parametry	návrat
/api/car_updateoradd editace nebo přidání auta do vozového parku přihlášeného uživatele	přihlášený uživatel	• **fields** – hodnoty jednotlivých polí formou JSON objektu (vlastnost je názvem pole) • **rowid** – „new“ nebo unikátní ID záznamu	• **success** – informace o dokončení • **id** – null nebo unikátní ID nového záznamu
/api/car_del smazání auta z vozového parku přihlášeného uživatele	přihlášený uživatel	• **rowid** – unikátní ID záznamu	• **success** – informace o dokončení

*Pozn.: Názvy ukládaných hodnot odpovídají názvům sloupců v tabulce **cars***

Tab. 5: API / seznamy

API	přístup	parametry	návrat
/api/listoffractions seznam všech frakcí odpadu	přihlášený uživatel		• **success** – informace o dokončení • data – seznam frakcí s jejich parametry
/api/municipalitiestree seznam všech obcí	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam všech obcí a stromová struktura Kraj-ORP-Obec
/api/listofcars seznam všech aut vozového parku přihlášeného uživatele	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam aut a jejich parametrů
/api/listofproductionplans seznam scénářů produkce pro vybrané obce	přihlášený uživatel	• **municipalities** – seznam NUTS ID vybraných obcí	• **success** – informace o dokončení • **data** – seznam scénářů produkce a jejich parametrů
/api/listofcollectionsystems seznam systémů sběru pro vybrané obce (systémy vycházející z plánování rozmístění sběrných míst)	přihlášený uživatel	• **municipalities** – seznam NUTS ID vybraných obcí	• **success** – informace o dokončení • **data** – seznam systémů sběru
/api/listofaddresspoints seznam adresních míst vybrané obce	přihlášený uživatel	• **municipality** – NUTS ID vybrané obce	• **success** – informace o dokončení • **data** – seznam adresních míst a jejich parametrů
/api/listofregions seznam všech regionů	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam regionů a jejich parametrů
/api/listoforps seznam všech ORP	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam ORP a jejich parametrů
/api/listofcontainers seznam všech kontejnerů pro frakce odpadu	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam kontejnerů a jejich parametrů
/api/listofcartemplates seznam vzorových aut	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam vzorových aut a jejich parametrů

Tab. 6: API / svazky obcí

API	přístup	parametry	návrat
/api/listofunions seznam svazku obcí, které si přihlášený uživatel uložil	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam svazků (součástí svazku je seznam NUTS ID obcí)
/api/saveunion uložení nového svazku obcí pro přihlášeného uživatele	přihlášený uživatel	• **name** – název svazku • **data** – seznam NUTS ID vybraných obcí oddělených čárkou	• **success** – informace o dokončení • **id** – unikátní ID uloženého svazku

Tab. 7: API / uživatelé

API	přístup	parametry	návrat
/api/login přihlášení	veřejné	• **login** – login (e-mail) • **pw** – heslo	• **success** – informace o dokončení • **token** – základní token pro vygenerování ověřovacího
/api/logout odhlášení	veřejné	• **cookie-utoken** – token uživatele který má být odhlášen	• **success** – informace o dokončení
/api/ustatus ověření uživatele	veřejné	• **cookie-utoken** – token uživatele který má být ověřen	• **success** – informace o dokončení • **data** – seznam čekajících notifikací, nový token, …
/api/resetpasswd obnova hesla	veřejné	• **login** – login (e-mail) pro který se má heslo obnovit	• **success** – informace o dokončení • **timelock** – je obnova aktuálně blokována? (povoleno jednou za 10 minut)
/api/register registrace	veřejné	• **mail** – e-mail • **name** – jméno • **surname** – příjmení • **subject -** subjekt	• **success** – informace o dokončení
/api/dataofuser data uživatelského účtu přihlášeného uživatele	přihlášený uživatel		• **success** – informace o dokončení • **data** – data uživatelského účtu
/api/confirmuser schválení uživatele	pouze admin	• **id** – unikátní ID schvalovaného uživatele	• **success** – informace o dokončení
/api/listofusers seznam uživatelů (pokud není nastaven filtr, jsou vráceni všichni uživatelé daného typu)	pouze admin	• **type** – typ uživatelských účtů (schválen/neschválen) • **filter** – kolekce unikátních ID vybraných uživatelů	• **success** – informace o dokončení • **data** – seznam uživatelů a jejich základních dat

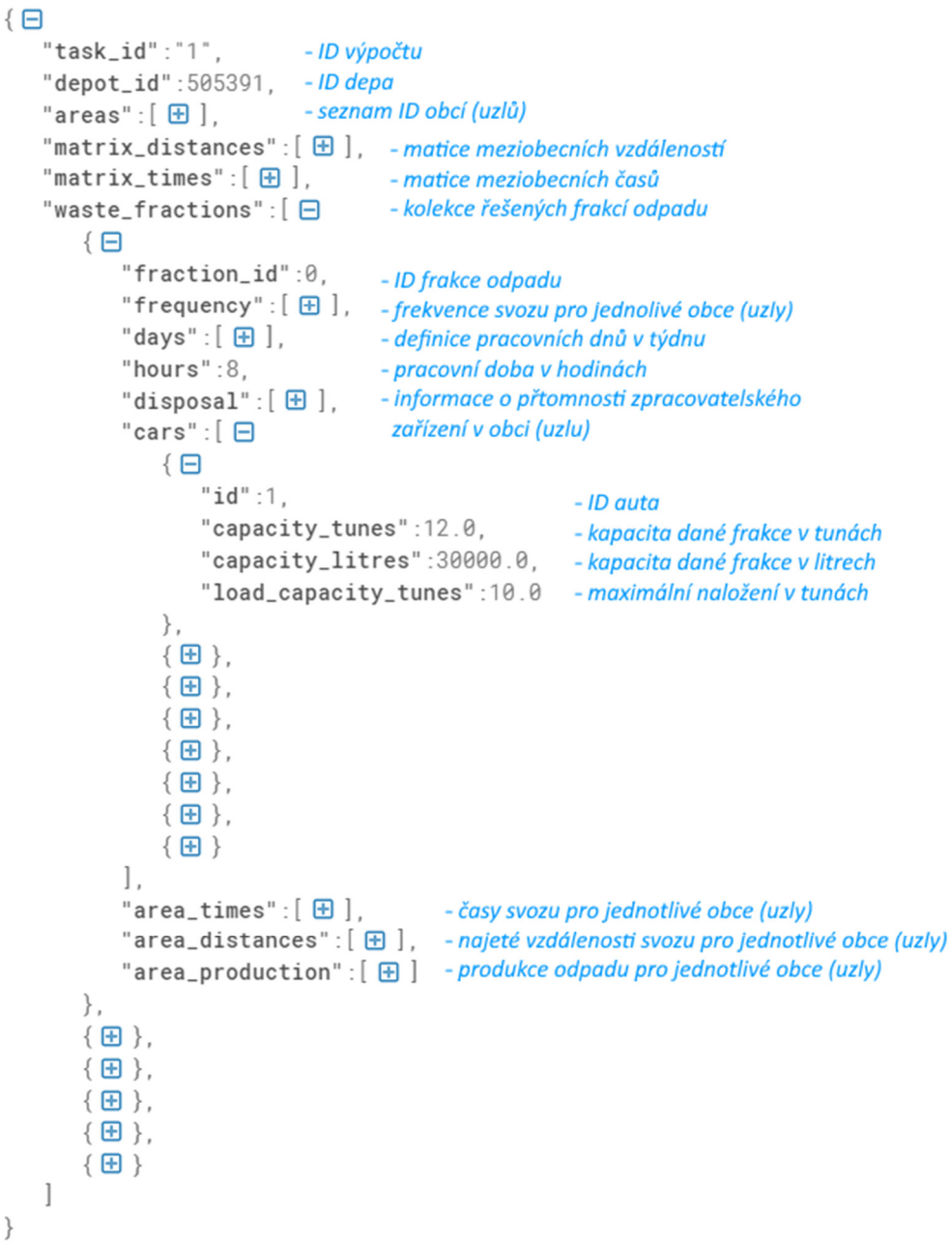
Tab. 8: API / výpočet

API	přístup	parametry	návrat
/api/executecomputation zahájení výpočtu nad uloženým zadáním	přihlášený uživatel	• **id** – unikátní ID plánu (výpočtu)	• **success** – informace o dokončení
/api/savecomputation uložení rozpracovaného plánu (výpočtu) svozu/sběru	přihlášený uživatel	• **fields** – hodnoty jednotlivých polí formou JSON objektu (vlastnost je názvem pole) • **id** – „new“ nebo unikátní ID záznamu	• **success** – informace o dokončení • **id** – unikátní ID uloženého plánu (výpočtu)
/api/listofcomputations seznam uložených plánů (výpočtu) svozu/sběru	přihlášený uživatel		• **success** – informace o dokončení • **data** – seznam plánů (výpočtů) a jejich dat
/api/onecomputation data uloženého plánu (výpočtu) svozu/sběru	přihlášený uživatel	• **id** – unikátní ID vybraného plánu (výpočtu)	• **success** – informace o dokončení • **data** – data konkrétního plánu (výpočtu)

4.3.8 Komunikace s výpočetním jádrem

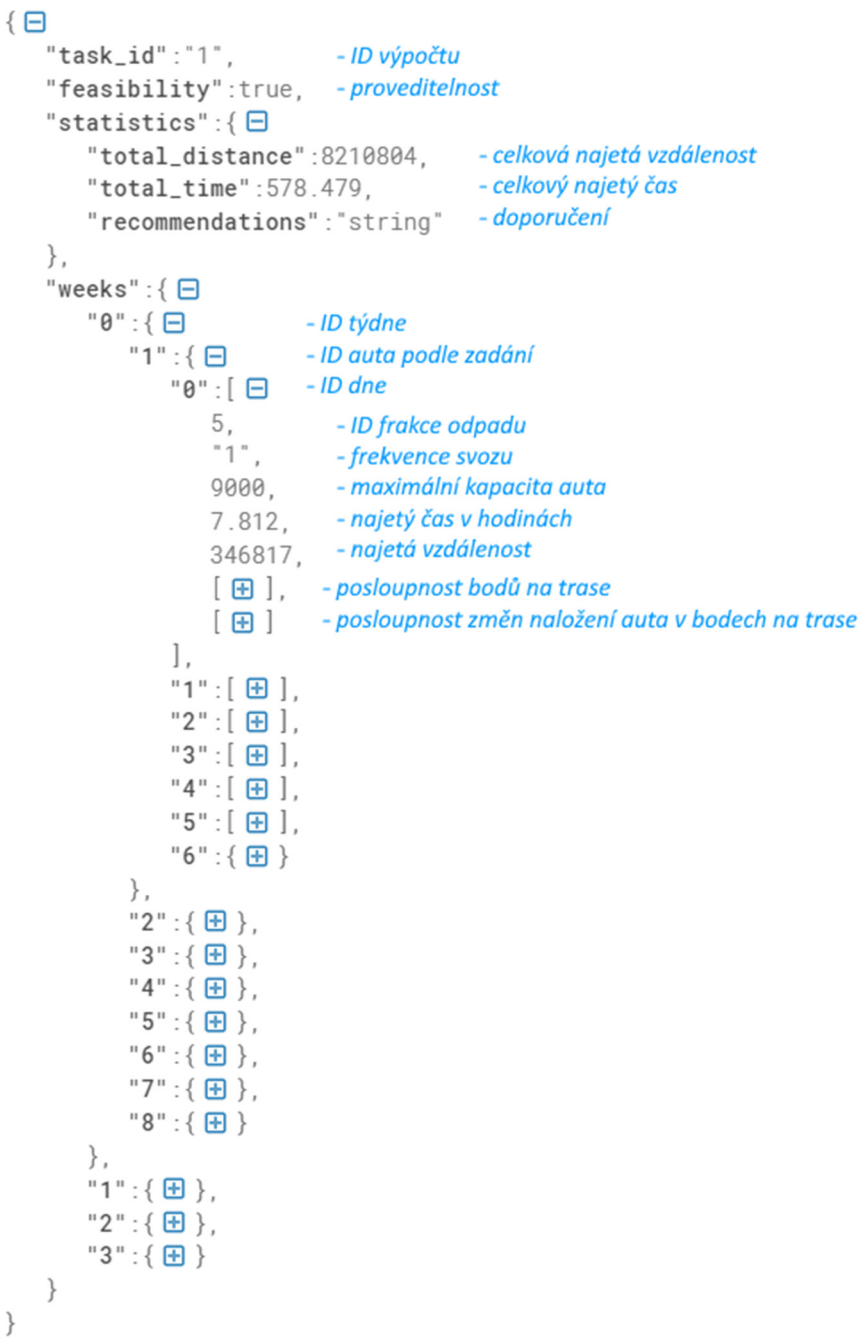
Původně byla u obou výpočetních jader předpokládána implementace formou nezávislých služeb s API rozhraním. Během tvorby aplikace jsme se rozhodli výpočetní jádro pro plánování svozu [17] implementovat jako Python modul. Z původního návrhu se zachovalo předávání zadání pro výpočet a výsledků formou JSON struktury. Modul aplikace pro výpočet úlohy, zadá volání metody výpočetního jádra do aplikační fronty s odpovídajícím zadáním. Po dokončení výpočtu je jeho výsledek dále zpracován.

Během testování integrace výpočetního jádra bylo třeba řešit i použití vhodné verze Pythonu, výpočetní jádro nepracovalo správně pod Pythonem 3.10, konkrétně měla na testovacím stroji (OS Linux) problém knihovna *ortools*. V této konkrétní implementaci jsme tedy nebyli schopni úspěšně dokončit výpočet, z toho důvodu bylo testováno i nasazení pod jinými verzemi Pythonu. Problematickým se z testovaných verzí ukázal pouze Python 3.10, což vedlo ke změně používané verze na Python 3.9.



Obr. 17: JSON se zadáním pro výpočetní jádro

Pozn.: U parametrů area_times, area_distances a area_productions se může vyskytovat hodnota 0, znamená to, že se daná frakce odpadu v konkrétní obci nesváží, tedy se její produkce a "cena" svozu v této obci neuvažují.

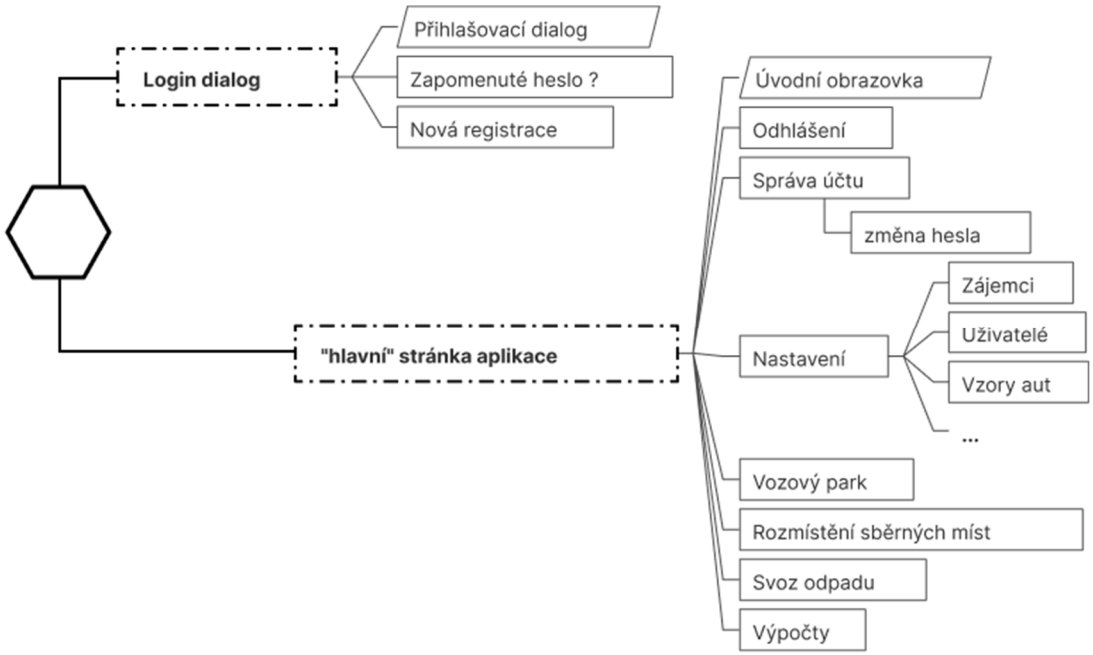


Obr. 18: JSON s výsledky výpočtu

4.4 FRONTEND

Návrh rozložení uživatelského rozhraní vycházel z předpokladu, že bude zobrazováno na obrazovkách většího formátu a uživatel bude pracovat s myší a klávesnicí. Aplikace je navržena jako „jednostránková“, využívající funkcionality React komponent pro změnu zobrazovaného obsahu a provolávání API v pozadí pro načítání dat. Načtení potřebných skriptů a zdrojů je tedy potřeba provést jen jednou při načtení stránky. Výjimkou z tohoto pravidla je přechod mezi hlavní stránkou aplikace a přihlašovacím dialogem, vyžadující vždy načtení všech zdrojů. Všechny zdroje komponent byly vytvořeny přímo pro tuto

aplikaci, stejně jako několik pomocných scriptů. Aplikace také zahrnuje několik standardních knihoven třetích stran vycházejících z použitých technologií.



Obr. 19: Rozdělení uživatelských „obrazovek“

Hlavní obrazovka aplikace je tvořena z komponent. Kromě komponent specifických pro jednotlivé DP aplikace obsahuje i komponenty, které jsou pro tuto a překrývající se DP společné. Jejich vytvoření jsme si rozdělili a následně spolupracovali na jejich úpravách, aby naplňovaly společné potřeby.

Pro práci s mapovými podklady je použita knihovna Leaflet s využitím mapových podkladů OpenStreetMap. V základu využíváme komponenty z knihovny React Leaflet, které jsme rozšířili o další vlastní komponenty vytvořené na míru našim potřebám.

Mezi pomocné skripty patří kolekce pomocných funkcí (API komunikace, notifikace, cookies, …), práce se sdílenými daty v paměti nebo implementace HTML5 dialogu.

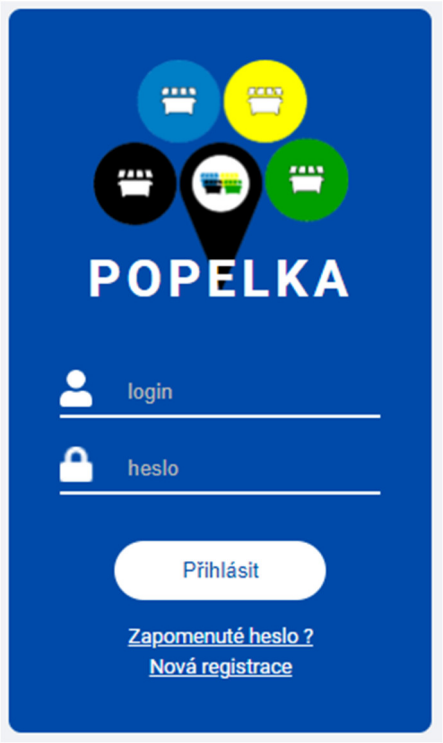
4.4.1 React komponenty

React komponenty vytvořené pro webovou aplikaci se v základu dají rozdělit podle využití a podle autorství. Komponenty vytvořené čistě kolegou z překrývající se DP[1] vynecháme. Zaměříme se na komponenty vytvořené jako součást této práce a komponenty, na kterých jsme pracovali společně.

Společné komponenty

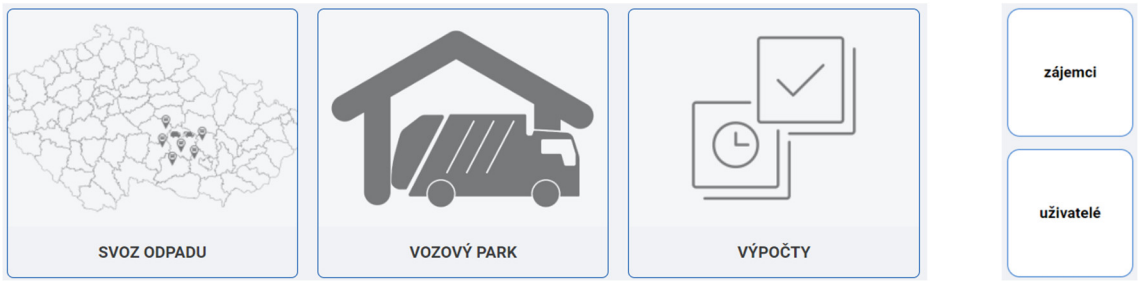
Celé uživatelské rozhraní (pro přihlášeného uživatele) zapouzdřuje centrální komponenta ***index***. Již z její samotné podstaty se jedná o společně utvářenou komponentu, která se během vývoje aplikace a jejího rozšiřování postupně vyvíjela.

Komponenty vytvořené jako součást této DP



Obr. 20: React komponenta ***login***

Komponenta ***login*** (obrázek 20) je hlavní komponentou přihlašovací obrazovky, její součástí jsou i dialogy pro obnovu hesla a novou registraci do systému. V rámci této komponenty je v pozadí řešeno ověření uživatelských údajů vůči přihlašovací API a vyhodnocení jejího návratu přesměrováním do hlavního okna aplikace, nebo animovanou vizualizací neplatných uživatelských údajů.



Obr. 21: React komponenty ***cards*** a ***gridselection***

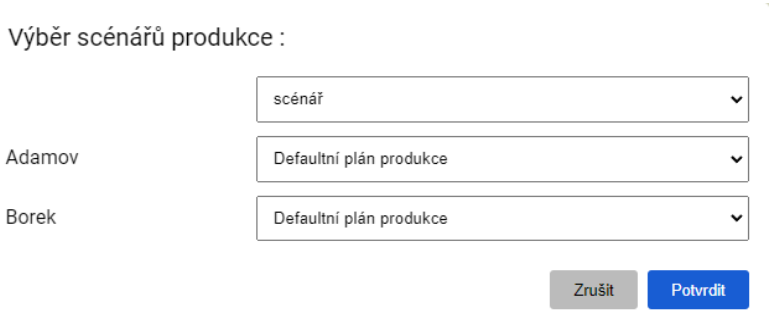
Pozn.: U obou komponent je na obrázku zobrazena jen část položek dostupných v aplikaci.

Na obrázku 21 můžeme vidět dvě komponenty sloužící jako menu v různých úrovních aplikace.

V levé části je komponenta ***cards***, která je použita jako hlavní rozcestník aplikace. Jednotlivé možnosti jsou zobrazovány formou velkých karet zobrazených v mřížce podle dostupného místa pro vykreslení. Každá položka má svůj název a tematický obrázek.

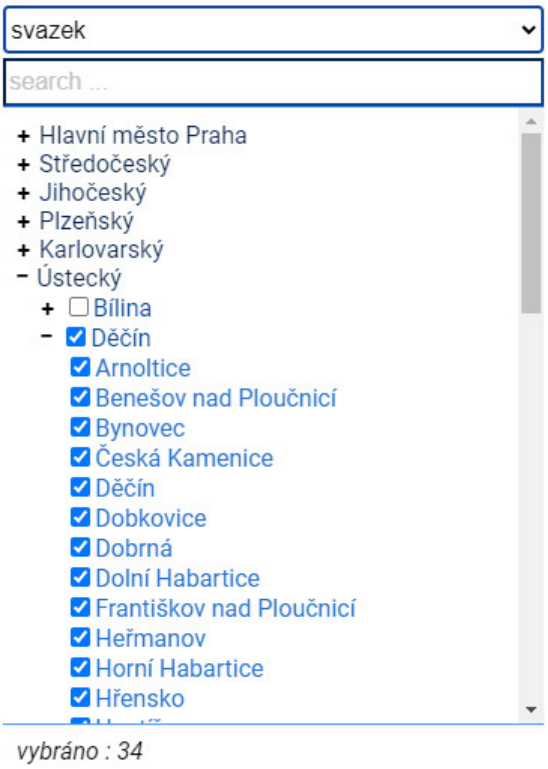
V pravé části obrázku máme komponentu ***gridselection***, která je použita jako rozcestník konkrétních částí aplikace (například jednotlivé sekce nastavení). Je navržena pro případné zobrazení většího počtu položek v relativně omezeném prostoru, z toho vychází i menší rozměry objektů představujících jednotlivé možnosti. Položky jsou zobrazeny jako malé karty v mřížce podle dostupného místa pro vykreslení. Každá položka má svůj název a nepovinně může mít i obrázek, který se zobrazuje nad názvem. Název a případný obrázek jsou centrovány v oblasti karty.

U obou komponent jsou položky zvýrazněny, pokud se na ně najede myší. Samotný výběr se provádí kliknutím na požadovanou položku. Funkce, která se po výběru zavolá, je součástí definice jednotlivých položek.



Obr. 22: React komponenta ***regform*** – příklad využití

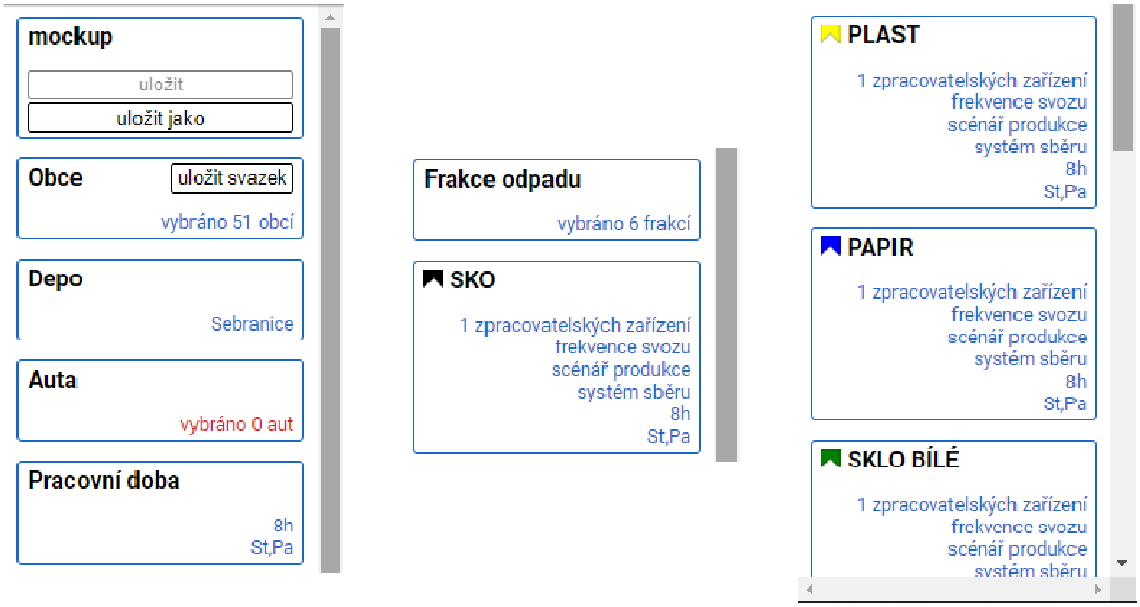
Komponenta ***regform*** zajišťuje vykreslení formulářů pro uživatelské vstupy nebo výběry z možností (příklad na obrázku 22), zobrazení je vždy formou dvou sloupců, tedy popisek-pole. Konfigurace se předává formou seznamu JavaScript objektů, jeden objekt reprezentuje jedno formulářové pole včetně jeho popisku a dalších parametrů. Podporovány jsou seznamy a základní typy formulářových polí kromě typů *checkbox* a *radio*. Pro každé pole lze definovat i funkci, která se provede při každé změně hodnoty.



Obr. 23: React komponenty ***treeselection*** a ***unions***

Na obrázku 23 jsou komponenty ***treeselection*** a ***unions***. Komponenta ***unions*** zapouzdřuje v tomto konkrétním případě komponentu ***treeselection*** a rozšiřuje ji o seznam s výběrem z již uložených svazků obcí přihlášeného uživatele. Při výběru konkrétního uloženého svazku se v komponentě ***treeselection*** vyberou odpovídající obce (předchozí výběr obcí předcházející výběru svazku je zrušen).

Komponenta ***treeselection*** umožňuje výběr ze stromové struktury o jedné až třech úrovních, výběr jen jedné hodnoty nebo N hodnot, a vyhledávání podle názvů položek nejnižší úrovně. Konkrétní položky lze vybírat jen z nejnižší úrovně, pokud je povolen výběr více položek, je možné vybrat celou skupinu přes jejího rodiče (položka předposlední úrovně). Příkladem jednoúrovňového výběru je výběr frakcí odpadu, více úrovňový výběr využívá například výběr obcí (Kraj-ORP-Obec).



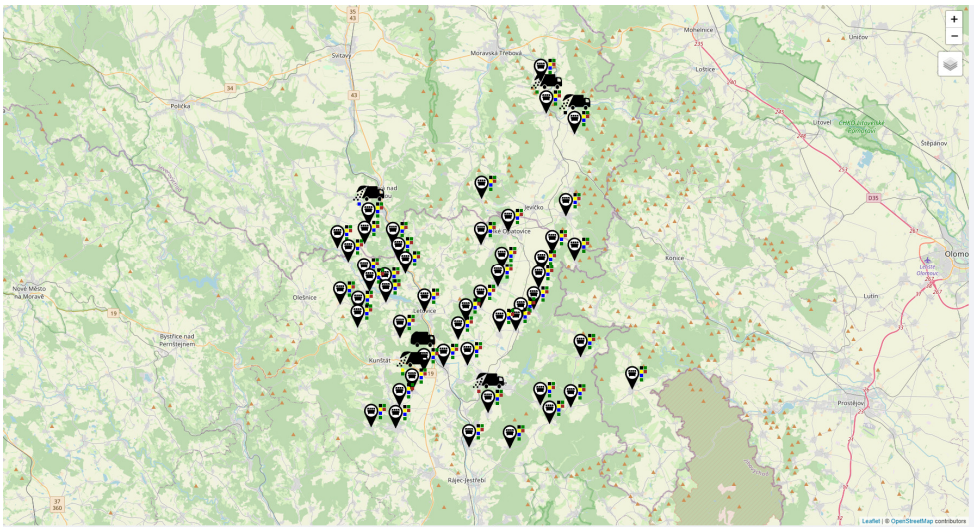
Obr. 24: React komponenta ***collectionPanel***

Pozn.: Komponenta je na obrázku rozdělena na tři části z důvodu její výšky.

Komponenta ***collectionPanel*** (obrázek 24) umožňuje nastavení vstupních parametrů plánovaného svozu, od vybraného svazku obcí až po frekvenci svozu v jednotlivých obcích. V základním zobrazení je vypsán stav aktuálního nastavení (červené hodnoty reprezentují nenastavené parametry), jednotlivé parametry se nastavují formou dialogů, komponenta umožnuje celé zadání uložit pro pozdější editaci bez ohledu na stav rozpracovanosti. Jedná se o jednu ze tří hlavních částí obrazovky plánování svozu odpadu.

Komponenta je vizuálně rozdělena do následujících částí:

- blok s uloženým názvem (nebo informací o neuložených datech) a tlačítky pro uložení
- blok s výběrem obcí, pro které budeme plánovat svoz
- blok s výběrem depa
- blok s výběrem aut, která lze pro svoz použít
- blok s nastavením obecné pracovní doby
- blok s výběrem frakcí odpadu, pro které se má svoz plánovat
- bloky s nastavením pro každou vybranou frakci odpadu



Obr. 25: React koponenty ***collectionMapWrap***, ***collectionMapChangeView*** a ***collectionMap***

Na obrázku 25 je druhá ze tří hlavních částí plánování svozu odpadu, kterou utváří tři doplňující se komponenty. Komponenta ***collectionMapWrap*** zapouzdřuje tento celek, předpřipraví seznam bodů pro vykreslení v mapě, určuje zájmovou oblast v mapě a zajišťuje správné umístění komponenty ***collectionMap***. Komponenta ***collectionMapChangeView*** zajišťuje při zásahu do konfigurace svozu aktualizaci pohledu mapy, aby zobrazovaná oblast měla správné přiblížení a obsahovala všechny vybrané obce (svážené, depo, zpracovatelská místa). Komponenta ***collectionMap*** využívá komponenty ReactLeaflet a knihovnu Leaflet pro zobrazení mapy, vykreslení vybraných obcí a trasy, která je vybrána k zobrazení.

Obce jsou zobrazeny jako ikonky reflektující aktuální nastavení (více v popisu komponent pro vykreslení ikonek). Trasy se v mapě vykreslují barvou odpovídající frakci odpadu, pro kterou jsou naplánovány.

Obr. 26: React komponenty – ikonka obce v mapě

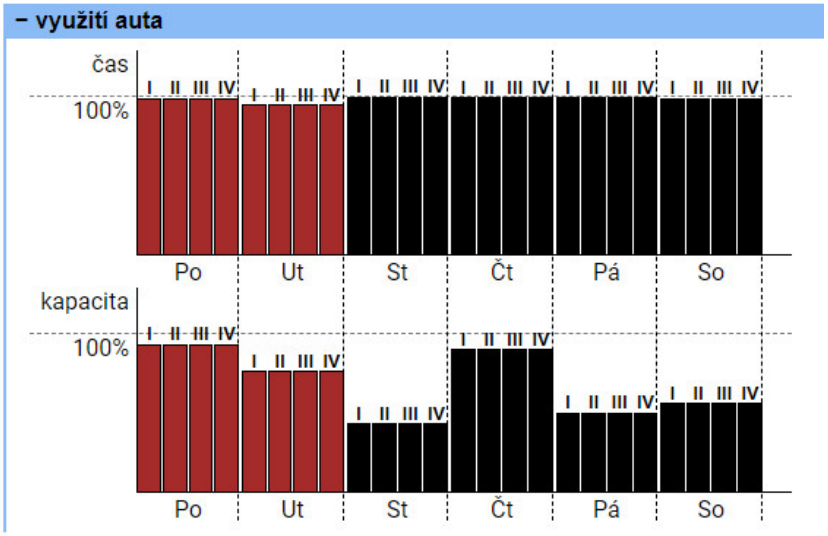
Vykreslení samotných ikonek obcí v mapě zajišťuje komponenta ***collectionRenderMarkers***, ta převezme seznam k vykreslení od komponenty ***collectionMapWrap***, připraví doplňující grafiku vykreslovanou kolem základní ikonky (svážené frakce odpadu, zpracovávané frakce odpadu, depo) a zajistí samotné vykreslení doplněných ikonek pomocí komponent ***municipalityMarker*** a ***municipalityMarkerIcon***.

Příklad výsledné vykreslené ikonky je na obrázku 26. Základní ikonka je vlevo doplněna barevnou mapou zpracovávaných frakcí odpadu, vpravo svážených frakcí odpadu. V horní části mohou být až dvě různé ikonky svozového auta, jedna pro depo a druhá pro přítomnost zpracovatelského zařízení (vykládající vůz).

obec	zpracovatelské zařízení	frekvence svozu	scénář produkce	systém zběru
Bezděčí u Trnávky	ano	svoz 1 za 2 týdny	Defaultní plán produkce	door-to-door
Jaroměřice	ne	svoz 1 za 2 týdny	Defaultní plán produkce	door-to-door

Obr. 27: React komponenta ***simpleTable*** – příklad použití

Na obrázku 27 je příklad využití komponenty ***simpleTable*** pro vykreslení jednoduchých tabulek podle zadání. Součástí nastavení je kromě samotných dat k zobrazení i definice hlavičky tabulky, možnost hlavičku skrýt, doplňující styl pro zobrazení konkrétního sloupce a možnost zobrazit součet hodnot pro vybrané sloupce (pokud jsou v těchto sloupcích číselné hodnoty). Komponenta je využívána při rozšířeném zobrazení zadání svozové úlohy a prezentaci části výsledků svozové úlohy, respektive navrženého plánu.



Obr. 28: React komponenty ***collapsibleTab*** *a* ***carGraph***

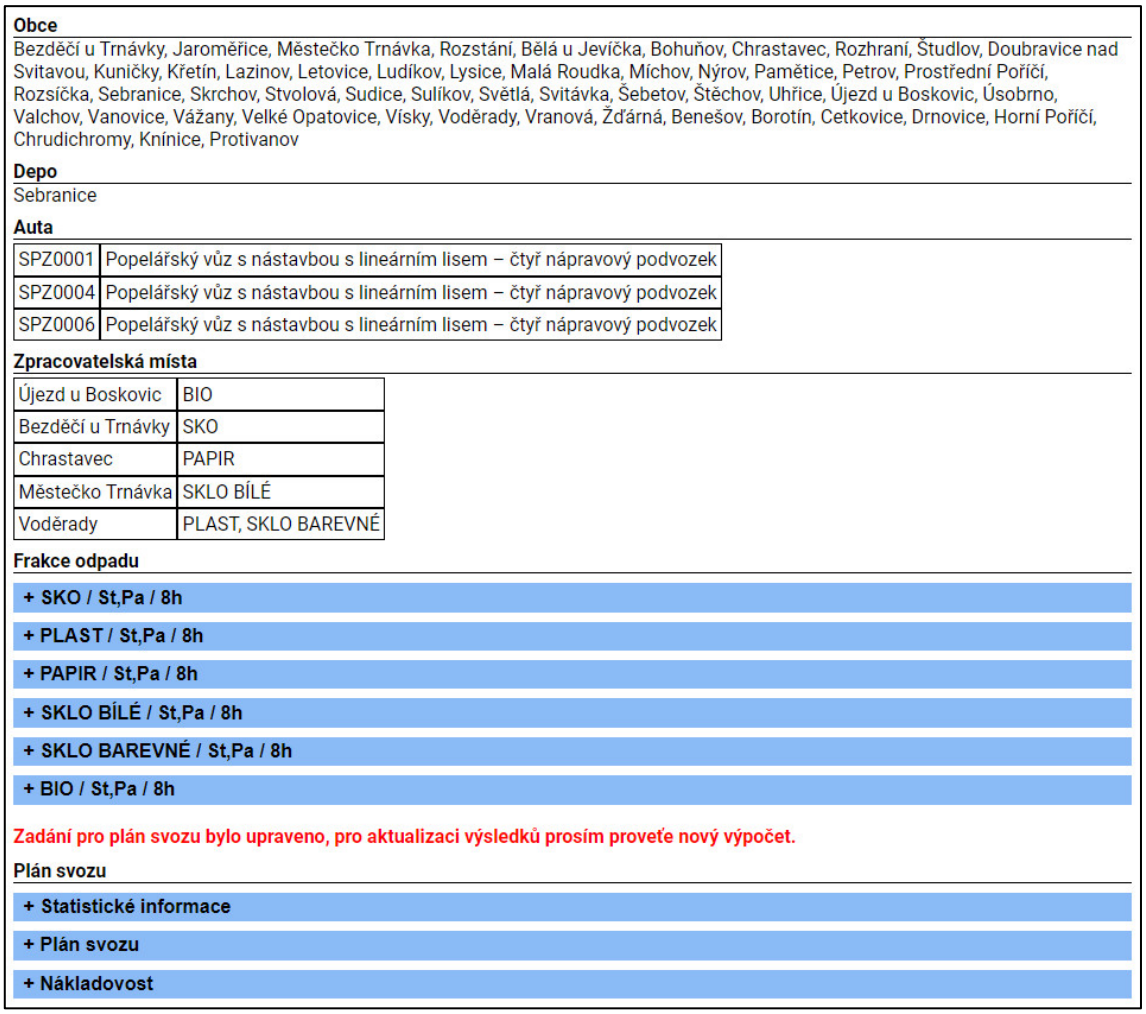
	1. týden	2. týden	3. týden	4. týden
Po	BIO	BIO	BIO	BIO
	Bělá u Jevíčka, Velké Opatovice, Letovice, Skrchov, Stvolová, Rozhraní, Vranová, Rozsíčka, Sulíkov, Petrov, ***Žďárná***	**Bělá u Jevíčka**, Velké Opatovice, Letovice, Skrchov, Stvolová, Rozhraní, Vranová, Rozsíčka, Sulíkov, Petrov, ***Žďárná***	**Bělá u Jevíčka**, Velké Opatovice, Letovice, Skrchov, Stvolová, Rozhraní, Vranová, Rozsíčka, Sulíkov, Petrov, ***Žďárná***	**Bělá u Jevíčka**, Velké Opatovice, Letovice, Skrchov, Stvolová, Rozhraní, Vranová, Rozsíčka, Sulíkov, Petrov, ***Žďárná***
	Žďárná, Městečko Trnávka, Rozstání, Bezděčí u Trnávky, Jaroměřice, Úsobrno, Uhřice, Knínice, Benešov, ***Žďárná***	***Žďárná***, Městečko Trnávka, Rozstání, Bezděčí u Trnávky, Jaroměřice, Úsobrno, Uhřice, Knínice, Benešov, ***Žďárná***	***Žďárná***, Městečko Trnávka, Rozstání, Bezděčí u Trnávky, Jaroměřice, Úsobrno, Uhřice, Knínice, Benešov, ***Žďárná***	***Žďárná***, Městečko Trnávka, Rozstání, Bezděčí u Trnávky, Jaroměřice, Úsobrno, Uhřice, Knínice, Benešov, ***Žďárná***
	Žďárná, Velké Opatovice, Cetkovice, Světlá, Šebetov, Pamětice, Vísky, Lysice, Újezd u Boskovic, Žďárná, **Bělá u Jevíčka**	***Žďárná***, Velké Opatovice, Cetkovice, Světlá, Šebetov, Pamětice, Vísky, Lysice, Újezd u Boskovic, Žďárná, **Bělá u Jevíčka**	***Žďárná***, Velké Opatovice, Cetkovice, Světlá, Šebetov, Pamětice, Vísky, Lysice, Újezd u Boskovic, Žďárná, **Bělá u Jevíčka**	***Žďárná***, Velké Opatovice, Cetkovice, Světlá, Šebetov, Pamětice, Vísky, Lysice, Újezd u Boskovic, Žďárná, **Bělá u Jevíčka**

Obr. 29: React komponenta ***carPlanTable*** – výřez jednoho dne

V prezentaci rozšířeného zadání a navrženého plánu svozu se využívá dalších komponent. Komponenta ***collapsibleTab*** umožňuje rozdělení obsahu do tematických částí, které lze pro přehlednost na klik sbalit či rozbalit (příklad na obrázku 28).

Komponenta ***carGraph*** vykresluje jednoduchý graf využití konkrétního auta. Sloupce mají barvu odpovídající konkrétní frakci odpadu, graf je rozdělen do bloků (jednotlivé pracovní dny) po N sloupcích (podle počtu týdnů, na které je plán navržen). Příklad grafů je na obrázku 28, týdny jsou číslovány římskými číslicemi, velikost sloupců je vztažena k úrovni stoprocentního využití. Pokud by v jeden stejný den bylo sváženo více frakcí odpadu, bude sloupec na výšku rozdělen do více částí s barvami podle svážených frakcí odpadu.

Komponenta ***carPlanTable*** zobrazuje tabulku s plánem tras konkrétního auta (obrázek 29), trasy jsou rozděleny podle jednotlivých týdnů, pracovních dnů a frakce sváženého odpadu. V tabulce lze kliknout na konkrétní trasu a označit ji tak pro vykreslení, vybrána může být jen jedna trasa (trasou se z pohledu výběru myslí jak dílčí úsek, tak celá trasa konkrétní frakce odpadu ve zvolený den) ze všech zobrazených plánů, v tomto ohledu jsou všechny zobrazené instance této komponenty propojeny. U konkrétní trasy je vizuálně odlišeno depo (tučný text) a zpracovatelská místa (tučný text kurzívou).



Obr. 30: React komponenta ***collectionInfo***

Komponenty využívané pro zobrazení a prezentaci svozového plánu uzavírá komponenta ***collectionInfo***. Tato komponenta zapouzdřuje jednotlivé dílčí komponenty do jednoho panelu, který je třetí z hlavních částí rozhraní pro plánování svozu. Také na základě aktuální konfigurace a dostupných dat určuje, které části se zobrazí a které pro konkrétní zobrazení nemají význam.

Pomocné knihovny

V rámci této DP bylo pro frontend vytvořeno několik pomocných knihoven, které jsou částečně využívány i některými komponentami.

Knihovna ***utilities*** je ve své podstatě kolekce pomocných funkcí, například pro parametrizované volání API v pozadí, správu lokálního úložiště a další. Minimalistická knihovna ***notifications*** definuje třídu, která usnadňuje používání mechanizmu notifikací zabudovaného do webových prohlížečů. V knihovně ***memory*** je definována třída pro práci se strukturovanými daty v paměti, s podporou vyžádání překreslení navázaných React komponent. Pro každou skupinu dat se lze definovat vlastní úložiště (z pohledu vnitřní struktury vlastní kořenový objekt).

4.5 POMOCNÉ SKRIPTY

Při tvorbě webové aplikace vzniklo jako vedlejší produkt pár pomocných nástrojů, využívaných během vývoje aplikace. Jedná se o Python konzolové skripty, pro import dat do databáze a přidání uživatele s právy administrátora. Pomocné skripty pracující s databází si načítají nastavení z konfiguračního souboru aplikace přes její modul pro načítání konfigurace.

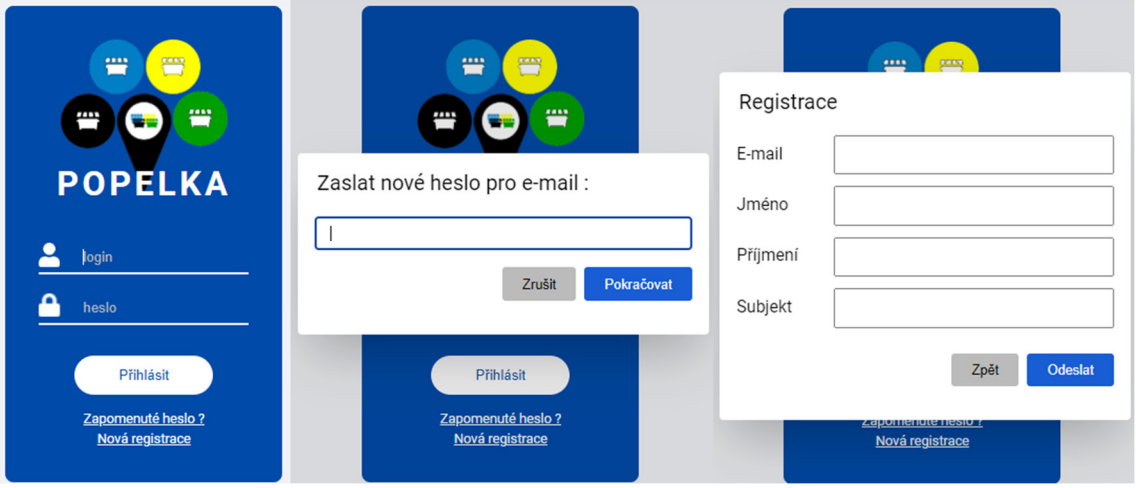
- **console_adduser.py** – skript pro vytvoření nového uživatele s právy administrátora, v konzoli je výzva pro zadání požadovaného loginu (e-mail) a hesla.

- **console_dbinsert.py** – skript pro import dat do databáze, v konzoli je výzva pro zadání zdrojového XLSX (excel) souboru. Skript provádí import dat vždy do jedné tabulky, jejíž název se shoduje s názvem souboru. Předpokládá se, že v prvním řádku obsahují pole názvy sloupců, do kterých se budou zapisovat data. Sloupce se ze souboru čtou po první prázdné pole v prvním řádku (prázdné pole není zahrnuto), řádky se čtou po první prázdné pole prvního sloupce (prázdný řádek není zahrnut), z těchto dat se vytvoří sada SQL příkazů, která se aplikuje nad připojenou databází. Využívá se příkazu REPLACE, který zajišťuje v případě existujícího záznamu se shodnými hodnotami klíčových sloupců pouze aktualizaci hodnot.

Byly vytvořeny i dvě verze skriptu pro import matic vzdáleností do databáze (z XLSX a CSV). Po rozhodnutí přesunout tato data mimo databázi (více v 4.2) byly tyto skripty zrušeny.

5 UŽIVATELSKÉ ROZHRANÍ

V kapitole 4.4 na obrázku 19 je vidět schéma uživatelských obrazovek. Rozložení a rozměry konkrétních částí uživatelského rozhraní předpokládají zobrazení na zařízeních o rozměru většího tabletu nebo na větším monitoru. Doplnění kompatibility s mobilními zařízeními (menší zobrazovací plocha) a další rozšíření mohou být předmětem dalšího vývoje webové aplikace.



Obr. 31: Uživatelské rozhraní - přihlašovací obrazovka a související dialogy

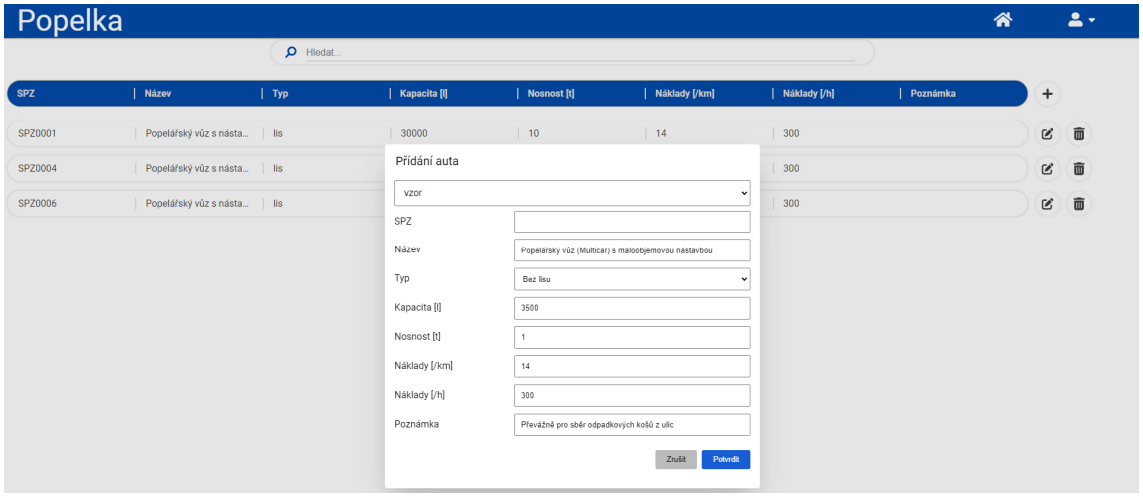
První obrazovka při navštívení webu nepřihlášeným uživatelem je přihlašovací obrazovka. Pokud uživatel nemá v systému účet, může se do systému registrovat. Registrace je v systému schvalována, vyplnění registračního formuláře zavede uživatele do seznamu žadatelů a odešle na jeho e-mail potvrzení jeho žádosti o přístup do systému. Pokud uživatel přístup do systému již má, ale zapomene heslo, může využít jednoduchého dialogu pro obnovu hesla. Pro stejný e-mail jde požádat o obnovu hesla jen jednou za 10 minut.

Obr. 32: Uživatelské rozhraní - hlavní obrazovka aplikace

Po úspěšném přihlášení se uživateli zobrazí hlavní obrazovka aplikace (obrázek 32), obsah aplikace se drobně liší podle uživatelských práv. Správce systému má oproti běžnému uživateli navíc dostupné základní nastavení aplikace. Mezi moduly pro správu patří správa vzorových aut pro svoz nebo správa a schvalování uživatelů.

Součástí aplikace je i systém notifikací, pokud bude dokončen výpočet zadaný uživatelem, připraví se odeslání notifikace, kterou uživatel obdrží, pokud je přihlášen, nebo při prvním přihlášení po dokončení úlohy. Přes notifikaci o dokončeném výpočtu se lze prokliknout na plánování svozu s daty tohoto výpočtu (zadání a výsledky).

5.1 Vozový park

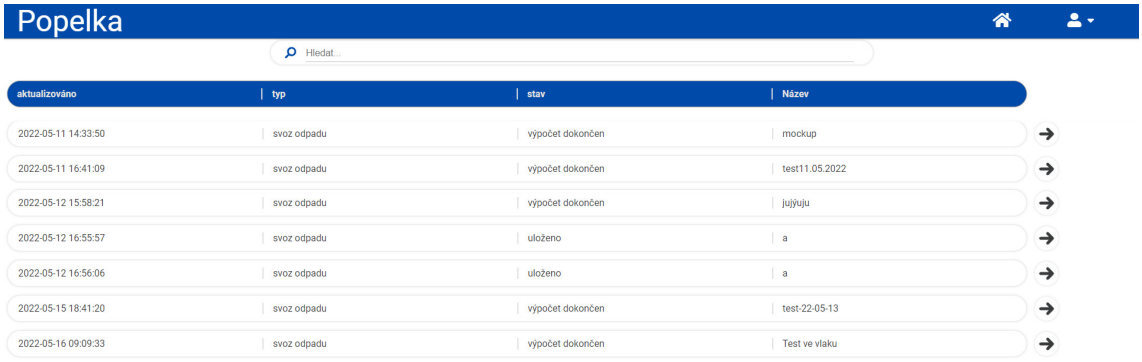


Obr. 33: Uživatelské rozhraní – editace vozového parku

Každému uživateli je umožněno spravovat svůj vlastní vozový park. Auta je možné přidávat, mazat, upravovat a v již uložených vyhledávat. Pro usnadnění je možné při přidávání nových aut vycházet z předem připravených vzorů. Z aut, která si zde uživatel nadefinuje, mu bude umožněno vybírat při plánování svozu. U každého auta se nastavují následující parametry:

- **SPZ** – unikátní identifikace vozidla
- **Název** – název vozidla definovaný uživatelem
- **Typ** – typ vozidla, s lisem nebo bez lisu
- **Kapacita** – kapacita vozidla v litrech
- **Nosnost** – nosnost vozidla v tunách
- **Náklady [Kč/km]** – náklady na ujetý kilometr
- **Náklady [Kč/h]** – náklady na hodinu jízdy
- **Poznámka**

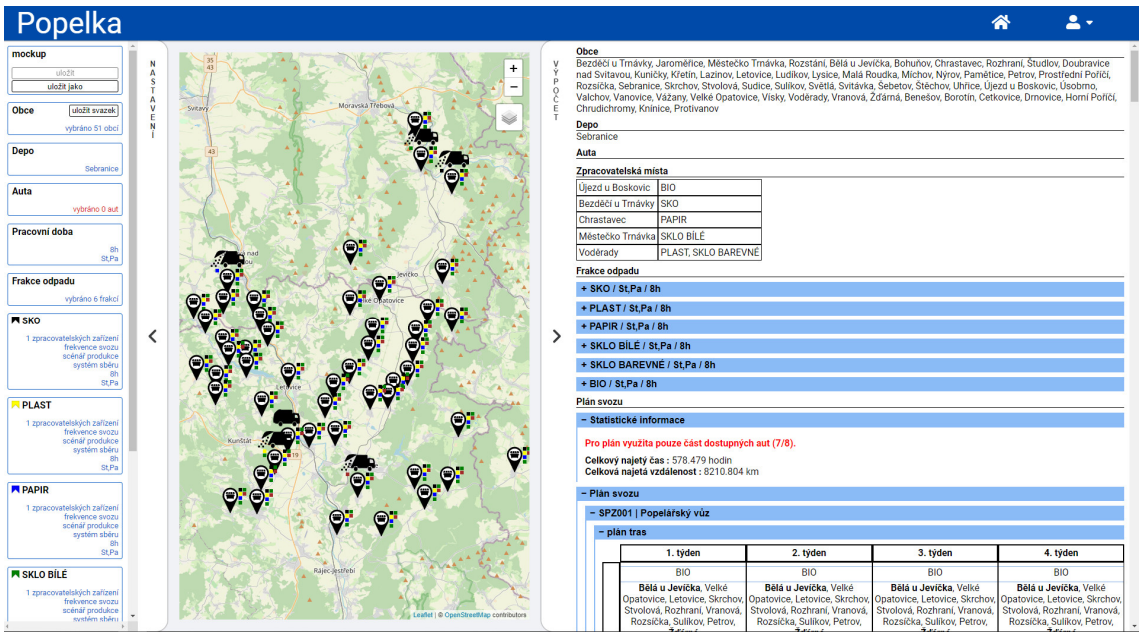
5.2 Výpočty



Obr. 34: Uživatelské rozhraní – přehled výpočtů

V sekci výpočty je seznam uložených plánů svozu a sběru odpadu přihlášeného uživatele. U každého výpočtu je uvedeno, zda se jedná o sběr či svoz, stav úlohy (uloženo, výpočet zadán, probíhá výpočet, výpočet dokončen), čas vytvoření a název, pod kterým byl uložen. Ve výpočtech lze vyhledávat a kliknutím na ikonku šipky konkrétní výpočet otevřít a přejít na odpovídající obrazovku plánování.

5.3 Plánování svozu



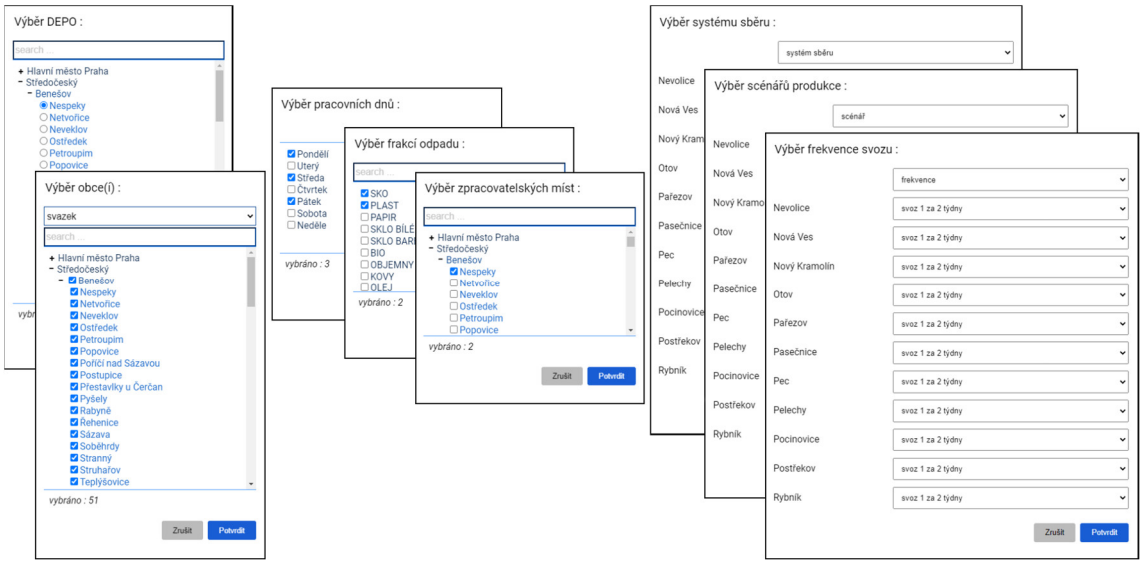
Obr. 35: Uživatelské rozhraní – plánování svozu

Plánování svozu slouží k vytvoření plánu pro svoz zvolených frakcí odpadu ve vybraném svazku obcí. Obrazovka je rozdělena do tří částí, levý panel s nastavením pro výpočet sestavující plán svozu, pravý panel s rozepsaným zadáním a navrženým plánem (pokud je již vypočítán) a mapa, ve které jsou zobrazeny značky reflektující parametry svozu.

V navrženém plánu svozu lze vybrat konkrétní trasu z plánu tras aut, ta je poté vyznačena v mapě barvou odpovídající svážené frakci odpadu. Trasy jsou v mapě zobrazeny zjednodušeně, tedy formou přímých spojnic mezi jednotlivými obcemi. Zobrazována je vždy poslední vybraná trasa (lze vybrat celou trasu nebo konkrétní úsek „prázdné auto“ - zpracovatelské zařízení), výběr lze zrušit kliknutím na poslední vybranou trasu (aktuálně vybraná je v plánu vyznačena).

V jakékoli fázi rozpracovanosti svozového plánu si lze rozpracovaný plán uložit, pokud nebyl plán uložen do spuštění výpočtu, je uživatel vyzván k zadání názvu, pod který se zadání automaticky uloží před spuštěním výpočtu (pokud byl editován již uložený záznam, jsou data aktualizována).

Výpočet je možné spustit pouze v případě, že je zadání kompletní. Nenastavené položky jsou barevně vyznačeny. Nastavení konkrétních parametrů se provádí přes systém dialogů (ukázka na obrázku 36), uživatel otevře odpovídající dialog výběrem požadované položky v levém panelu.



Obr. 36: Uživatelské rozhraní – ukázka dialogů

Po spuštění výpočtu je rozhraní svozu nahrazeno obrazovkou informující o probíhajícím výpočtu (samotný výpočet může trvat několik minut). Po dokončení výpočtu je uživateli opět zobrazeno standardní rozhraní svozu doplněné o vypočítaný svozový plán. Pokud během výpočtu uživatel přejde do jiné části aplikace, může se kdykoli vrátit přes obrazovku uložených výpočtů. Po dokončení každého výpočtu je uživateli zobrazena notifikace, pokud nezakázal v prohlížeči jejich zobrazování. Přes tuto notifikaci se lze prokliknout na detail konkrétního svozového plánu.

6 ZÁVĚR A DALŠÍ VÝVOJ

Tato práce řešila návrh a vytvoření webové aplikace pro plánování svozu různých frakcí odpadu na meziobecní úrovni. V úvodu byly shrnuty cíle, motivace a začlenění této práce do většího celku. Následoval krátký úvod do problematiky třídění a svozu odpadu, rozbor použitých technologií a samotný návrh aplikace. Práci pak uzavírá popis uživatelského rozhraní zaměřený na svoz odpadu.

Již v rámci vytváření návrhu aplikace a při jejím postupném budování docházelo k jejímu dalšímu vývoji (změny v návrhu databáze, rozšiřování ukládaných struktur, úpravy v návrhu uživatelského rozhraní, …). Vývoj vedl i k rozšíření sady služeb ze kterých se aplikace skládá. Kromě pomocných služeb třetích stran, mezi které patří například NGINX a REDIS, vznikly jako součást aplikace tři základní služby, hlavní služba aplikace, služba pro poskytování matic vzdáleností a služba vykonávající úlohy které se hromadí v aplikační frontě.

Vytvořená aplikace přináší základní uživatelské rozhraní pro plánování svozu frakcí odpadu a je možné ho dále rozšiřovat o další funkcionality a moduly. Aplikace je připravena na další případné rozšiřování, backend je možné modulárně rozšiřovat a frontend využívá React komponenty, které z něj v podstatě dělají dále rozšiřitelnou stavebnici sestavenou z jednotlivých komponent.

Mezi omezení aplikace patří její cílení na zařízení s většími obrazovkami a podporou myši, není tedy rozložením přizpůsobena například mobilní zařízením. Aplikace má prostor pro další vývoj z pohledu funkčnosti, uživatelské ergonomie i požadavků v odpadovém hospodářství. Z pohledu požadavků v odpadovém hospodářství lze aplikaci rozšířit například o možnost upravovat navrhnutý plán svozu (přesun tras mezi dny a vozidly, změna pořadí obcí v trase, …). Z pohledu funkčnosti lze aplikaci rozšířit o další nástroje pro správu celého systému (import dat do databáze, …), nebo rozšířit možnosti nastavení plánovaného svozu o další parametry (například preferované vozidlo pro frakci odpadu). Je ale v tomto ohledu třeba pamatovat na to, že aplikace je součástí většího celku a některé úpravy budou vyžadovat zásah i do výpočetního jádra (například rozšíření vstupních parametrů pro výpočet).

Do budoucnosti se předpokládá další vývoj a rozšiřování aplikace i struktury samotných dat, nad kterými se pracuje (rozdělení obcí na městské/obecní části, …) pro větší variabilitu nastavení svozového plánu, dosažení přesnějších výpočtů a optimálnějších svozových plánů.

7 SEZNAM POUŽITÉ LITERATURY

[1] AWV, 2020. Österreichischer Abfallwirtschaftsverbände, [online]. [cit. 2022-03-10]. Dostupné z: https://www.argeawv.at/.

[2] CG932-066-520. Modelování logistiky odpadů v městských aglomeracích, [online]. [cit. 2022-03-10]. Poskytovatel: Ministerstvo dopravy. Dostupné z: https://starfos.tacr.cz/cs/project/CG932-066-520.

[3] COVAR 14, 2020. Consorzio obbligatorio di bacino, [online]. [cit. 2022-03-10]. Dostupné z: https://www.covar14.it/index.php/covar14.

[4] DSO Tišnovsko, 2022. Dobrovolný svazek obcí Tišnovsko, [online]. [cit. 2022-03-10]. Dostupné z: https://www.dsotisnovsko.cz/

[5] EG15_007_0003463. Výzkumný a vývojový projekt IT Cluster 2016–2019. Poskytovatel: Ministerstvo průmyslu a obchodu, [online]. [cit. 2022-03-10]. Dostupné z: https://starfos.tacr.cz/cs/project/EG15_007_0003463.

[6] EKO-KOM, 2020. Ekonomika odpadového hospodářství v roce 2020, [online]. [cit. 2022-03-10]. Dostupné z: https://www.ekokom.cz/ekonomika-odpadoveho-hospodarstvi-v-roce-2020/

[7] Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., Scavino, E., 2018. Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. Waste Management, 71, 31-41. DOI: 10.1016/j.wasman.2017.10.019.

[8] KTS Ekologie, 2021, [online]. [cit. 2022-03-10]. Dostupné z: https://www.kts-ekologie.cz/o-firme.

[9] Směrnice Evropského parlamentu a Rady (EU) 2018/850 ze dne 30. května 2018, kterou se mění směrnice 1999/31/ES o skládkách odpadů.

[10] Směrnice Evropského parlamentu a Rady (EU) 2018/851 ze dne 30. května 2018, kterou se mění směrnice 2008/98/ES o odpadech.

[11] Teixeira J., Antunes A.P., de Sousa J.P., Recyclable waste collection planning—a case study, European Journal of Operational Research, Volume 158, Issue 3, 2004, Pages 543-554, ISSN 0377-2217, https://doi.org/10.1016/S0377-2217(03)00379-5.

[12] Technické služby Malá Haná, 2020, [online]. [cit. 2022-03-10]. Dostupné z: https://www.tsmh.cz/.

[13] TIRSMZP719. Prognózování produkce odpadů a stanovení složení komunálního odpadu [online]. [cit. 2022-03-10]. Poskytovatel: Technologická agentura České republiky. Doba řešení: 1.1.2019 – 30.6.2022. Dostupné z: https://starfos.tacr.cz/cs/project/TIRSMZP719.

[14] Vidal T., Laporte G., Matl P., A concise guide to existing and emerging vehicle routing problém variants, European Journal of Operational Research, Volume 286, Issue 2, 2020, Pages 401-416, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2019.10.010.

[15] Zákon č. 541/2020 Sb. Zákon o odpadech. Účinnost od 1.1.2021.

[16] KHÝR, Lukáš. Matematické modely v oblasti strategického rozhodování [online]. Brno, 2020. 69 s [cit. 2022-05-18]. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/121551. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce Ing. Martin Pavlas, Ph.D.

[17] KUČERA, Jiří. Modelování logistiky meziobecní přepravy odpadu [online]. Brno, 2022 [cit. 2022-05-18]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/141042. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce Ing. Vlastimír Nevrlý, Ph.D.

[18] ZAMAZAL, P. Statistická analýza rozsáhlých dat z průmyslu [online]. Brno, 2021 [cit. 2022-05-10]. 51 s. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132858. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce doc. Ing. Radovan Šomplák, Ph.D.

[19] Flask | The Pallets Projects [online]. [cit. 2022-05-10]. Dostupné z: https://palletsprojects.com/p/flask/

[20] Advanced Load Balancer, Web Server, & Reverse Proxy – NGINX [online]. [cit. 2022-05-10]. Dostupné z: https://www.nginx.com/

[21] The uWSGI project [online]. [cit. 2022-05-10]. Dostupné z: https://uwsgi-docs.readthedocs.io/en/latest/

[22] Redis [online]. [cit. 2022-05-10]. Dostupné z: https://redis.io/

[23] React – A JavaScript library for building user interfaces [online]. [cit. 2022-05-10]. Dostupné z: https://reactjs.org/

[24] React Leaflet [online]. [cit. 2022-05-10]. Dostupné z: https://react-leaflet.js.org/

[25] Leaflet - a JavaScript library for interactive maps [online]. [cit. 2022-05-10]. Dostupné z: https://leafletjs.com/

[26] OpenStreetMap [online]. [cit. 2022-05-10]. Dostupné z: https://www.openstreetmap.org/about

[27] SILNÝ, Marek. Analýza rozmístění sběrných míst a kontejnerů pro komunální odpady [online]. Brno, 2022 [cit. 2022-05-18]. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/140674. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního inženýrství. Vedoucí práce Ing. Vlastimír Nevrlý, Ph.D.

8 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

Seznam zkratek

- ČSÚ – Český statistický úřad
- SKO – Směsný komunální odpad
- TSMH – Technické služby Malá Haná
- ER – Entity Relationship
- NUTS – Nomenclature of territorial units for statistics
- DOM – Document Object Model
- CSV – Comma-separated values
- XLSX – Microsoft Excel Open XML Spreadsheet
- XML – extensible markup language
- JSON – JavaScript Object Notation
- API – Application Programming Interface
- WSGI – Web Server Gateway Interface
- HTTP – Hypertext Transfer Protocol
- HTTPS – Hypertext Transfer Protocol Secure
- SMTP – Simple Mail Transfer Protocol
- SSL – Secure Sockets Layer
- RMSE – Root Mean Square Error

Seznam tabulek

Seznam obrázků