Univerzita Pardubice
Dopravní fakulta Jana Pernera

Organizace stání a posuvných operací na budoucí vlečce Hrochostroj v Hradci Králové

Bc. Antonín Holub

Diplomová práce
2025

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2024/2025

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Antonín Holub**
Osobní číslo:	**D23478**
Studijní program:	**N1041A040008 Technologie a management v dopravě**
Specializace:	**Dopravní management, marketing a logistika**
Téma práce:	**Organizace stání a posuvných operací na budoucí vlečce Hrochostroj v Hradci Králové**
Zadávající katedra:	**Katedra dopravního managementu, marketingu a logistiky**

Zásady pro vypracování

Cílem diplomové práce je vytvoření internetové aplikace v jazyce html, php, javascript a mySQL pro plánování organizace stání a posuvných operací železniční strojů a vozů na vlečce v souladu s platnými předpisy. Budou vizualizovány výstupy a klíčové informace pro přehledné zobrazení. Součástí práce bude zhodnocení ekonomických aspektů a efektivity návrhu v porovnání se stavem současné organizace.

Rozsah pracovní zprávy:	**50-60 stran**
Rozsah grafických prací:	**dle doporučení vedoucí/ho**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam doporučené literatury:

dle pokynů vedoucí/ho práce

Vedoucí diplomové práce:	**doc. Ing. Jiří Křupka, PhD.** Katedra dopravního managementu, marketingu a logistiky
Datum zadání diplomové práce:	**31. října 2024**
Termín odevzdání diplomové práce:	**7. května 2025**

L.S.

doc. Ing. Ladislav Řoutil, Ph.D.
děkan

Ing. Pavla Lejsková, Ph.D.
vedoucí katedry

V Pardubicích dne 24. dubna 2025

Prohlašuji:

Práci s názvem Organizace stání a posuvných operací na budoucí vlečce Hrochostroj a.s. v Hradci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 5. 5. 2025

Bc. Antonín Holub v. r.

Rád bych poděkoval vedoucímu práce doc. Ing. Jiřímu Krupkovi, PhD., za vstřícný přístup a cenné rady při zpracovávání diplomové práce a Luďku Suchánkovi za odborné konzultace při vytváření internetové aplikace.

ANOTACE

Diplomová práce se zabývá procesem plánování a řízení stání a posuvných operací na budoucí vlečce společnosti Hrochostroj a.s. v Hradci Králové.

Cílem práce je návrh a implementace systému pro plánování a řízení stání a posuvných operací na vlečce.

V rámci diplomové práce bude navržena webová aplikace, která umožňuje plánování a vizualizaci pohybu železničních vozů a strojů na vlečce s ohledem na platné předpisy a technické podmínky. Aplikace je založena na kombinaci programovacích jazyků HTML, PHP, JavaScript a databázového systému MySQL.

V rámci práce bude provedeno srovnání navrženého řešení s aktuálním stavem organizace stání a posuvných operací. Bude provedeno ekonomické zhodnocení nově navrženého systému.

KLÍČOVÁ SLOVA

Plánování, stání, posuvné operace, vlečka, železniční doprava, webová aplikace, HTML, PHP, JavaScript, MySQL, optimalizace, efektivita, bezpečnost

TITLE

Organization of standing and sliding operations on the future Hrochostroj siding in Hradec Králové

ANNOTATION

The thesis deals with the process of planning and management of standing and sliding operations on the future Hrochostroj a.s. siding in Hradec Králové.

The aim of the thesis is the design and implementation of a system for planning and controlling the standing and sliding operations on the siding.

Within the scope of the thesis, a web application will be designed that allows planning and visualization of the movement of railway wagons and machines on the siding with respect to the applicable regulations and technical conditions. The application is based on a combination of HTML, PHP, JavaScript programming languages and MySQL database system.

The proposed solution will be compared with the current state of the organization of standing and sliding operations. An economic evaluation will be performed of the newly proposed system.

KEYWORDS

Planning, standing, sliding operations, siding, rail transport, web application, HTML, PHP, JavaScript, MySQL, optimization, efficiency, safety

OBSAH

ÚVOD

Efektivní řízení provozu na železničních vlečkách představuje jednu z významných výzev současné železniční dopravy, zejména v prostředí průmyslových areálů, kde je kladen důraz na přesnost, plynulost a bezpečnost pohybu drážních vozidel.

Tato diplomová práce se zabývá návrhem a implementací webové aplikace *Vlečkostroj*, která slouží k plánování a organizaci stání a posuvných operací na budoucí vlečce společnosti Hrochostroj a.s. v Hradci Králové. Aplikace využívá programovací jazyky HTML, JavaScript, PHP, MySQL a algoritmy pro rozhodování o umístění strojů na jednotlivé koleje. Součástí řešení je i možnost offline provozu prostřednictvím technologie PWA.

Cílem práce je návrh funkčního systému a jeho ekonomické a provozní zhodnocení v porovnání s dosavadní praxí.

1 ŽELEZNIČNÍ VLEČKA

Železniční vlečka je dopravní cesta určená pro drážní vozidla, která slouží výhradně potřebám jejího provozovatele nebo jiného podnikatele. Vlečka je zaústěna do celostátní nebo regionální dráhy, případně do jiné vlečky. Tento druh dráhy hraje důležitou roli především v oblasti přepravy zboží a materiálů mezi průmyslovými, logistickými nebo jinými areály a železniční infrastrukturou.

Podle § 3 odst. 1 písm. d) zákona č. 266/1994 Sb., o dráhách, je vlečka definována jako „železniční dráha, která slouží vlastní potřebě provozovatele nebo jiného podnikatele a je zaústěna do celostátní nebo regionální dráhy nebo jiné vlečky“. Veřejně přístupné vlečky jsou otevřeny pro využití jakémukoliv dopravci za podmínek rovného přístupu, zatímco veřejně nepřístupné vlečky slouží pouze potřebám vlastníka vlečky nebo jiným oprávněným subjektům.

Železniční doprava na vlečkách tvoří klíčový prvek železniční infrastruktury, zejména v kontextu průmyslové a logistické obsluhy. Vlečky, jakožto železniční tratě připojené k hlavní síti, slouží primárně k zajištění dopravních potřeb konkrétních podniků, areálů či logistických center. Jejich hlavním cílem je usnadnit překládku a přepravu materiálů, zboží nebo surovin mezi průmyslovými subjekty a hlavní železniční sítí. Provoz na vlečkách je zpravidla uzpůsoben specifickým požadavkům jednotlivých uživatelů, čímž nabízí vysokou míru logistické flexibility.

Vlečky jsou technicky méně náročné než hlavní tratě. Mají obvykle nižší rychlostní limity a jednodušší konstrukční parametry. Traťová zařízení, například výhybky, jsou přizpůsobena specifickým potřebám provozu. Důležitou roli hrají nakládací a vykládací místa, která mohou být vybavena jeřáby, rampami nebo jinými zařízeními pro manipulaci se zbožím.

1.1 Legislativní rámec provozu na vlečce

Provoz vleček v České republice je regulován zákonem o dráhách č. 266/1994 Sb. a navazujícími předpisy. Zákon č. 266/1994 Sb., o dráhách, stanoví základní podmínky pro stavbu, provoz a účel železničních drah, včetně vleček. Určuje práva a povinnosti provozovatelů drah a drážní dopravy a upravuje vztahy mezi vlastníky vleček, dopravci a dalšími subjekty [1]. Vyhláška č. 177/1995 Sb., stavební a technický řád drah, definuje technické parametry, které musí vlečka splňovat, a zároveň určuje podmínky pro stavbu a úpravy vleček [2]. Zákon č. 367/2019 Sb., novela zákona o dráhách, zpřesňuje a aktualizuje pravidla provozu na vlečkách, včetně povinností provozovatelů veřejně nepřístupných vleček umožnit přístup dalším dopravcům za rovných podmínek [3].

Vlečky nejsou považovány za veřejné dráhy, a proto podléhají specifickým pravidlům. Vlastníci vleček (většinou průmyslové nebo logistické podniky) odpovídají za jejich provoz, údržbu a bezpečnost. Provoz na vlečce může zajišťovat buď vlastník vlečky nebo smluvně zajištěný dopravce, který splňuje podmínky pro provozování železniční dopravy. Každá vlečka musí mít vlastní provozní řád, který podrobně stanovuje způsob organizace dopravy, povinnosti zaměstnanců a šetření mimořádných událostí.

1.2 Odstavování na vlečce

Organizování a plánování odstavování železničních strojů na železniční vlečce je komplexní proces, který zahrnuje koordinaci technických, logistických a bezpečnostních aspektů. Správné řízení procesu odstavování je klíčové pro zajištění efektivity provozu, minimalizaci rizik a dodržení legislativních požadavků.

Plánování odstavování železničních strojů začíná detailní analýzou dostupných kapacit vlečky. Vlečka je obvykle tvořena omezeným počtem manipulačních a odstavných kolejí, jejichž efektivní využití je nezbytné. Při plánování je třeba zohlednit délku kolejí, jejich dostupnost a potřeby jednotlivých strojů. Pro některé železniční vozy mohou platit specifické požadavky na prostor nebo přístup, například kvůli technické údržbě nebo typu nákladu.

Dalším důležitým krokem je tvorba harmonogramu příjezdů a odjezdů. V praxi to znamená synchronizaci pohybů na vlečce s provozem na hlavní železniční síti. Zpoždění nebo špatné načasování může vést k přetížení vlečky, což má negativní dopad na plynulost provozu i na bezpečnost. Plánování by mělo zahrnovat rezervu pro mimořádné situace, jako jsou technické poruchy, nepříznivé počasí nebo neplánované pohyby.

Specifickým aspektem je řízení technické údržby. Některé železniční stroje potřebují během odstavení pravidelné kontroly nebo drobné opravy. To vyžaduje dostupnost odpovídajících zařízení, například dílen nebo připojení ke zdrojům energie. Plánování údržby musí být koordinováno s provozním plánem, aby nedocházelo k překryvům nebo zbytečným prodlevám.

Organizace odstavování železničních strojů zahrnuje nejen technickou stránku, ale také koordinaci lidských zdrojů, komunikaci s partnery a řízení rizik. Důležitou roli hraje dispečerská kontrola, která zajišťuje plynulý a bezpečný pohyb vozidel na vlečce. Dispečeři sledují aktuální obsazenost kolejí, plánují přidělování kolejových prostor a koordinují pohyby jednotlivých strojů.

Jednou z hlavních výzev je správné řízení kapacit vlečky při sezónních výkyvech nebo mimořádných událostech, jako jsou zpoždění nebo technické poruchy. Moderní technologie jsou nepostradatelným nástrojem v tomto procesu. Elektronické systémy pro sledování

a evidenci umožňují přehledně monitorovat stav vlečky v reálném čase, identifikovat volné kapacity a rychle reagovat na změny v provozu. Tyto systémy také usnadňují dlouhodobé plánování a archivaci dat, což je důležité pro optimalizaci provozu i pro plnění legislativních požadavků. Dalším rizikem je nekompatibilita některých vozidel s vlečkou (např. kvůli rozchodu kolejí nebo technickým parametrům). Aby se těmto problémům předešlo, je klíčová spolupráce s odborníky na železniční dopravu a implementace moderních technologií, jako jsou digitální plánovací nástroje.

Bezpečnost je dalším klíčovým prvkem organizace. Každý pohyb vozidel na vlečce musí probíhat v souladu s pravidly bezpečnosti práce, které vychází ze zákona č. 266/1994 Sb., o dráhách a v dalších souvisejících předpisech. Provozovatelé vleček musí například zajistit, aby byly koleje pravidelně kontrolovány, a aby byl personál řádně vyškolen k manipulaci s vozidly a obsluze vlečky.

1.3 Programy pro železniční dopravu

Programy pro plánování a organizaci železniční dopravy jsou klíčovými nástroji pro efektivní řízení železničního provozu. S narůstající poptávkou po železniční dopravě, a to jak osobní, tak nákladní, a s rostoucí složitostí železničních sítí se stávají softwarové nástroje nezbytnými pro zajištění bezpečnosti, efektivity a spolehlivosti provozu. Tyto programy pomáhají optimalizovat rozvrhy, přidělovat zdroje a reagovat na mimořádné situace.

Systémy pro tvorbu jízdních řádů

Tvorba jízdních řádů je základem plánování železniční dopravy. Moderní systémy využívají algoritmy optimalizace a simulace k vytvoření harmonogramů, které zohledňují dostupnost vlakových souprav, personálu a kapacity tratí. Příkladem takového softwaru je HASTUS-Rail [4], který je široce využíván pro plánování jízdních řádů a optimalizaci nasazení zdrojů. Dalšími příklady jsou systémy jako Trapaze [5] nebo IVU.rail [6].

Systémy pro řízení kapacity infrastruktury

Železniční síť má omezenou kapacitu, a proto je klíčové efektivně spravovat dostupné trasy a kapacitu stanic. Programy jako Railsys [7] a OpenTrack [8] umožňují simulaci a analýzu kapacitních požadavků. Tyto nástroje také podporují identifikaci úzkých míst v infrastruktuře a navrhují způsoby, jak zvýšit efektivitu provozu.

Systémy pro plánování nákladní dopravy

Nákladní doprava vyžaduje specializované plánovací nástroje, které zohledňují různé typy nákladu, překládku a časovou flexibilitu. Rail Freight Planner je příkladem softwaru, který pomáhá optimalizovat přepravu zboží a minimalizovat prázdné jízdy [9].

Systémy pro řízení provozu a operativní plánování

Operativní řízení zahrnuje každodenní řízení železničního provozu a řešení neplánovaných událostí, jako jsou zpoždění nebo výpadky infrastruktury. Systémy jako ControlGuide nabízejí reálné sledování provozu, automatizaci přesměrování vlaků a nástroje pro komunikaci s personálem [10].

Moderní železniční softwary nabízejí klíčové funkce, které zásadně přispívají k efektivitě a spolehlivosti železniční dopravy. Mezi tyto funkce patří především optimalizace, která využívá pokročilé algoritmy ke snižování zpoždění, nákladů na provoz a emisí. Další významnou funkcí je simulace, jež umožňuje testování různých scénářů provozu bez jakéhokoli dopadu na reálný provoz, což přispívá k lepšímu plánování a rozhodování. Prediktivní analýza, založená na technologiích strojového učení, dokáže předpovídat možné problémy, jako jsou zpoždění nebo technické poruchy, což usnadňuje preventivní opatření. Klíčová je také integrace dat, která zajišťuje propojení různých systémů, například informací o vlakových soupravách, infrastruktuře či meteorologických podmínkách, a tím zlepšuje koordinaci a rozhodovací procesy.

Použití těchto softwarů přináší řadu výhod. Patří sem zvýšení efektivity provozu, což se projevuje lepším využitím infrastruktury a zdrojů. Dochází také ke snížení provozních a údržbových nákladů, zlepšení bezpečnosti a zvýšení spokojenosti cestujících díky přesnějším a spolehlivějším jízdním řádům. Neméně důležitým benefitem je pozitivní dopad na životní prostředí – lepší optimalizace tras a provozu vede ke snížení spotřeby energie a emisí.

Budoucí trendy v oblasti železničního plánování a řízení směřují k větší integraci umělé inteligence a technologií IoT (Internet of Things). Tyto technologie umožní například zavedení autonomních vlaků, adaptivního plánování či dynamického řízení kapacity v reálném čase. Klíčovým prvkem budoucího vývoje bude také evropský systém řízení železničního provozu (ETCS), který harmonizuje signalizační a řídicí systémy napříč evropskými státy a usnadňuje mezinárodní spolupráci v oblasti železniční dopravy. Tyto inovace přispějí ke zvýšení efektivity, bezpečnosti a udržitelnosti železniční dopravy v globálním měřítku.

2 PROGRAMOVACÍ JAZYKY

Tvorba webových stránek je komplexní proces, který spojuje různé technologie s cílem vytvořit funkční, vizuálně přitažlivé a uživatelsky přívětivé prostředí. Mezi klíčové nástroje moderního webového vývoje patří HTML, CSS, PHP, JavaScript a databázové systémy, jako například SQL nebo MySQL. Každá z těchto technologií plní specifický účel a jejich vzájemná integrace umožňuje vytvářet robustní a interaktivní webové aplikace.

2.1 HTML

HyperText Markup Language (HTML) je značkovací jazyk, který slouží k vytváření struktury a obsahu webových stránek. Jeho hlavním účelem je organizace obsahu stránky do jednotlivých prvků, jako jsou nadpisy, odstavce, obrázky, seznamy, odkazy, tabulky nebo formuláře. HTML je jedním ze základních kamenů pro vytváření stránek v systému World Wide Web a umožňuje publikaci dokumentů na internetu [11].

Elementy (značky) v HTML

Zdrojový kód HTML stránek se skládá ze značek (tagů), které jsou uzavřeny ve špičatých závorkách `<a>`. HTML rozlišuje dva typy značek:

- **Párové tagy**: Většina značek je párová a ovlivňuje určitou část dokumentu, která je uzavřena mezi otevírací a uzavírací značkou (např. `<p>Hello world!</p>`). Příkladem jsou `<p>` pro odstavec, `<h1>` až `<h6>` pro nadpisy, `<a>` pro hypertextové odkazy nebo `<strong>` či `<b>` pro tučné písmo.
- **Nepárové tagy**: Tyto značky působí sami na sebe a definují v HTML nějaký prvek nebo mají vztah k celému dokumentu (např. `<img>`). Příkladem jsou `<br>` pro zalomení řádku, `<hr>` pro vodorovnou čáru nebo `<img>` pro vložení obrázku. Nepárové značky nemají odpovídající uzavírací značku.

Všechno ostatní, co se nachází mimo tyto značky je text, který prohlížeč zobrazí. Prohlížeč při zobrazování ignoruje přechody na nový řádek a vícenásobné mezery ve zdrojovém kódu. Pro zalomení řádku se používá značka `<br>` a pro pevnou mezeru entita ` `.

Každý HTML dokument má svou základní strukturu, která by se měla vždy dodržovat. Základní kostru tvoří následující elementy:

- **<!DOCTYPE HTML>**: Označuje typ dokumentu a informuje prohlížeč o použité verzi HTML.

- **<HTML>**: Kořenový element, který ohraničuje celý HTML dokument. Může obsahovat atribut lang, určující jazyk stránky.
- **<HEAD>**: Hlavička dokumentu, která obsahuje metainformace o stránce, jež se ve samotném okně prohlížeče (kromě `<TITLE>`) nezobrazují. Může obsahovat informace o kódování, klíčová slova, autora, odkazy na CSS styly a další.
    - **<TITLE>**: Titulek stránky, který se zobrazuje v horní liště prohlížeče nebo na kartě.
    - **<META>**: Slouží k poskytování různých informací o dokumentu, jako je kódování (charset), klíčová slova (keywords), popis (description) nebo informace pro automatizované vyhledávače (robots). Důležité je nastavení kódování, například na UTF-8 (`<meta charset="utf-8">`) pro správné zobrazení českých znaků.
- **<BODY>**: Tělo dokumentu, které obsahuje samotný viditelný obsah stránky – text, obrázky, tabulky, odkazy, formuláře a další.

Práce s textem

HTML nabízí řadu elementů pro strukturování a zvýraznění textu:

- **Odstavce**: Text se obvykle umisťuje do odstavců pomocí párových značek `<p>` a `</p>`. Koncový element `</p>` způsobí zalomení textu a dodatečnou svislou mezeru mezi odstavci. Odstavce lze zarovnat pomocí atributu align (např. `<p align="left">`).
- **Nadpisy**: Pro nadpisy se používají párové značky `<h1>` až `<h6>`, přičemž `<h1>` je nadpis nejvyšší úrovně a má největší písmo. Nadpisy jsou automaticky zobrazeny tučným písmem a na jejich konci se vytvoří zalomení řádku s dodatečnou mezerou.
- **Zvýraznění písma**: HTML poskytuje značky pro zvýraznění textu, jako `<b>` nebo `<strong>` pro tučné písmo, `<i>` nebo `<em>` pro kurzívu, `<big>` pro větší písmo a `<small>` pro menší písmo. Je možná i kombinace těchto efektů.
- **Citace**: Pro citace se používá párová značka `<BLOCKQUOTE>`, která text odsadí z obou stran. [12][13]

Vývoj HTML stránek

K tvorbě HTML stránek stačí textový editor, pro pohodlnější práci se však doporučují strukturální editory HTML s podporou zvýraznění syntaxe, automatického doplňování kódu, kontroly syntaxe a správy projektů [13].

2.2 CSS

CSS (Cascading Style Sheets) je jazyk používaný k popisu vzhledu a rozvržení webových stránek. Slouží ke stylizaci prvků definovaných v HTML, což umožňuje oddělit obsah webové stránky od jejího vizuálního provedení. CSS je klíčovou součástí moderního webového designu, protože poskytuje vývojářům nástroje pro kontrolu barev, fontů, rozvržení, animací a responzivního designu [11].

CSS funguje na principu pravidel, která definují, jak by měly být vybrané elementy HTML zobrazeny. Každé pravidlo se skládá ze selektoru a deklarací. Selektor určuje, na které HTML elementy se pravidlo vztahuje, zatímco deklarace obsahují jednu nebo více vlastností a jejich hodnot.

Selektory mohou být různé, například mohou odkazovat na konkrétní HTML elementy (např. body, h1, p), na elementy s určitou třídou (class) nebo ID (id). Třídy umožňují přiřadit stejný styl více elementům, zatímco ID by mělo být v dokumentu unikátní. Kromě základních selektorů existují také pseudotřídy (např. `:hover`, `:first-letter`), které aplikují styly pouze za určitých okolností [14].

Deklarace uvnitř složených závorek {} definují vzhled vybraných elementů. Každá deklarace se skládá z názvu vlastnosti a hodnoty vlastnosti, přičemž každá deklarace je ukončena středníkem ; [13]. Například pravidlo body `{background-color: #CCC; font-family: 'Trebuchet MS', 'Geneva CE'; padding: 50px; }` definuje barvu pozadí elementu body na šedou, použité písmo a vnitřní okraje.

Způsoby vkládání CSS do HTML

Existují tři hlavní způsoby, jak propojit CSS s HTML dokumentem:

- **Externí soubory CSS:** Jedná se o nejčastěji doporučovaný způsob, kdy jsou CSS pravidla uložena v samostatném souboru s příponou .css. Tento soubor je pak připojen k HTML dokumentu pomocí elementu `<link>` v sekci `<head>` [13]. Výhodou tohoto přístupu je jednotný vzhled celého webu a snadná údržba, jelikož změna stylu v jednom souboru se projeví na všech propojených stránkách.
- **Vnitřní styly:** CSS pravidla mohou být také umístěna přímo v HTML dokumentu uvnitř elementu `<style>` v sekci `<head>`. Tento způsob je vhodný pro jednotlivé stránky, které vyžadují unikátní styly [13].
- **Inline styly:** Styly mohou být definovány přímo u jednotlivých HTML elementů pomocí atributu style. Tento způsob by měl být používán minimálně, jelikož zhoršuje čitelnost kódu a porušuje princip oddělení struktury a vzhledu [14].

Dědičnost a kaskáda

CSS využívá mechanismy dědičnosti a kaskády. Dědičnost znamená, že některé vlastnosti CSS se automaticky přenášejí z rodičovských elementů na jejich potomky. Například barva písma nastavená pro element body bude implicitně platit pro všechny textové elementy uvnitř něj, pokud není pro tyto potomky definována jiná barva.

Kaskáda určuje, jak se řeší konflikty mezi různými CSS pravidly, která se vztahují na stejný element. Při rozhodování o tom, které pravidlo bude aplikováno, se bere v úvahu specifičnost selektoru a pořadí pravidel v CSS. Pravidla s vyšší specifičností mají vyšší prioritu, a pokud mají pravidla stejnou specifičnost, použije se to, které je v CSS definováno později [13]

Význam CSS pro moderní web

CSS je klíčovou technologií pro tvorbu moderních webových stránek. Jeho využití přináší řadu výhod:

- **Oddělení vzhledu od struktury:** Zvyšuje přehlednost kódu HTML a usnadňuje jeho údržbu.
- **Konzistentní vzhled:** Umožňuje snadno zajistit jednotný vizuální styl napříč všemi stránkami.
- **Snadná úprava vzhledu:** Změny v CSS se promítnou na všech propojených stránkách, což usnadňuje redesign webu.
- **Přístupnost:** Správně napsané CSS může zlepšit přístupnost webových stránek pro uživatele s různými potřebami.
- **Responzivní design:** CSS umožňuje vytvářet webové stránky, které se přizpůsobují různým velikostem obrazovek a zařízením.

CSS hraje nezastupitelnou roli při vývoji moderních webových stránek. Jeho správné využití nejen zlepšuje vizuální stránku webu, ale zároveň přispívá k lepší uživatelské zkušenosti, přístupnosti a efektivitě vývoje. Ovládnutí CSS je proto nezbytným předpokladem pro každého, kdo se chce věnovat tvorbě profesionálních a funkčních webových aplikací [11][13].

2.3 Javascript

„JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, který se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky.“ [11] Obvykle se používá k ovládání různých interaktivních prvků, jako jsou tlačítka, textová políčka, formuláře a také k tvorbě animací a efektů obrázků.

Syntaxe JavaScriptu patří do rodiny jazyků C/C++/Java. Co se týče syntaxe, nejvíce se podobá jazyku PHP. Novější formou JavaScriptu je asynchronní JavaScript, označovaný jako AJAX (Asynchronous JavaScript and XML). AJAX je soubor technologií určených k vývoji interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovu načítání.

Aplikace Javascript

Kód JavaScriptu může být zahrnut přímo v souborech HTML nebo PHP. Často je ale výhodnější umístit všechny skripty (funkce) do vlastního souboru s příponou .js a na tento soubor se odkazovat z různých stránek [11].

Oproti jazyku PHP, jehož kód se provádí na serveru a klientovi se odesílá pouze výsledek, JavaScript probíhá na straně klienta (návštěvníka webové stránky). Hlavní výhodou JavaScriptu je, že provádění kódu nevyžaduje dodatečnou komunikaci s webovým serverem, takže při zobrazení webové stránky nedochází k delším prodlevám. [13].

Kód JavaScriptu se zpravidla provede ihned po načtení webové stránky. K vložení JavaScriptu do HTML slouží element `<script>`, který se umisťuje v části `<body>`. Do jednoho HTML dokumentu lze vložit i více sekcí `<script>`. JavaScriptový kód lze také umístit do samostatného souboru s příponou .js a načíst jej pomocí elementu `<script src="nazev_souboru.js"></script>`. Další možností je zapsat jeden nebo více příkazů skriptu jako hodnotu některého z atributů v elementu jazyka HTML, například atribut onclick, který spustí kód po kliknutí myší.

Využití Javascript

JavaScript nabízí širokou škálu funkcí, které umožňují vytvářet interaktivní, dynamické a uživatelsky přívětivé weby. Níže je uvedeno několik hlavních oblastí, ve kterých JavaScript nachází své uplatnění:

- **Události:** JavaScript umožňuje reagovat na různé akce uživatele pomocí událostí. Patří sem například kliknutí myší (`onclick`), najetí (`onmouseover`) či opuštění objektu (`onmouseout`) nebo načtení celé stránky (`onload`). Díky tomu může webová stránka dynamicky měnit svůj obsah nebo chování podle interakce s uživatelem. Pomocí funkcí `setTimeout` a `setInterval` lze také spouštět kód s časovým zpožděním nebo v pravidelných intervalech.
- **Manipulace s DOM (Document Object Model):** JavaScript nabízí přístup k tzv. DOM, což je objektová reprezentace struktury HTML dokumentu. Díky tomu může skript dynamicky měnit obsah, strukturu i styl stránky – například přidávat nové prvky,

měnit text nebo upravovat třídy a styly. Tyto změny se provádějí bez nutnosti znovunačtení celé stránky, což zajišťuje plynulejší a interaktivnější uživatelský zážitek.

- **Integrace API:** Pomocí JavaScriptu je možné komunikovat s externími službami prostřednictvím API (Application Programming Interface). To umožňuje například načítat data z databází, aktualizovat obsah bez znovunačtení stránky nebo propojit web s jinými systémy a platformami. API volání se často provádí asynchronně, což znamená, že stránka zůstává plně funkční i během přenosu dat.
- **Programové konstrukce:** JavaScript poskytuje základní stavební prvky pro psaní logiky aplikace. Mezi ně patří například podmíněné příkazy (`if`) nebo funkce (`function()`), které umožňují vytvářet opakovaně použitelný a přehledný kód.
- **Dialogová okna:** K interakci s uživatelem lze využít funkce pro zobrazování dialogových oken – `alert()` pro jednoduché oznámení, `prompt()` pro získání vstupu od uživatele a `confirm()` pro potvrzovací dotazy. Tyto prvky usnadňují jednoduchou komunikaci mezi stránkou a uživatelem.
- **Ukládání dat v prohlížeči:** JavaScript umožňuje ukládat data přímo v prohlížeči uživatele pomocí objektu localStorage. Ten slouží k trvalému ukládání malého množství dat ve formátu klíč–hodnota. Výhodou je, že uložená data přetrvávají i po zavření a opětovném otevření prohlížeče a jsou tak vhodná například pro uchovávání nastavení nebo informací o uživateli.

Využití JavaScriptu v těchto oblastech výrazně rozšiřuje funkčnost webových stránek a umožňuje vytvářet bohatší uživatelský zážitek [13][15].

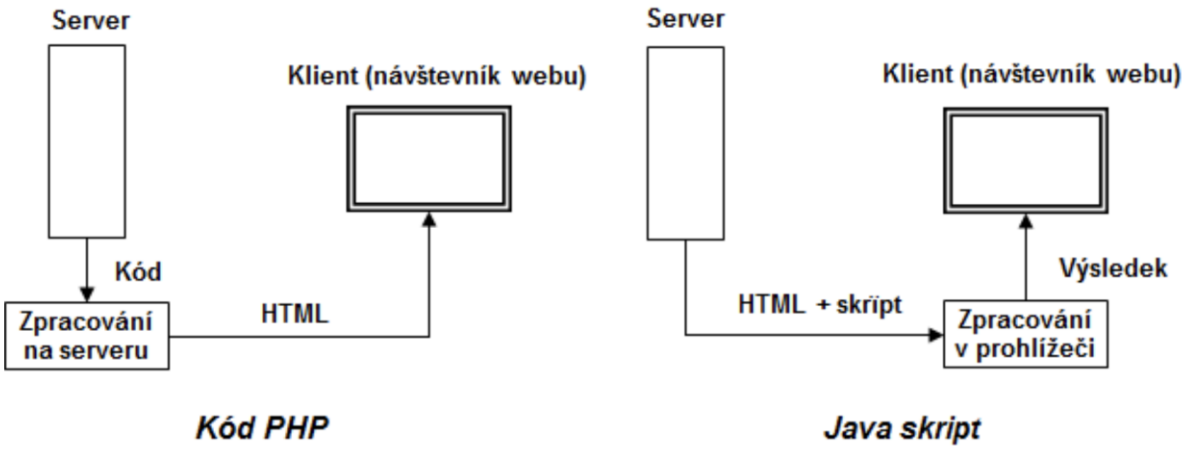
2.4 PHP

„PHP (původně Personal Home Page, nyní obvykle rekurzivně Hypertext Preprocessor) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek.“ [11]

Díky své jednoduchosti, flexibilitě a rozsáhlé podpoře se stal jedním z nejpoužívanějších jazyků pro tvorbu dynamických webových stránek, práci s databázemi, správu uživatelských relací, manipulaci se soubory nebo pro zpracování dat z formulářů. Díky podpoře většiny známých databázových systémů, jako jsou MySQL, PostgreSQL nebo SQLite, je PHP ideální volbou pro aplikace, které vyžadují robustní správu dat. Navíc dokáže snadno integrovat externí API, což umožňuje komunikaci s dalšími službami a aplikacemi.

PHP se liší od JavaScriptu, který se spouští až při načítání stránky v prohlížeči. PHP kód se zpracovává na serveru, kde jsou stránky uloženy, a pouze výsledek se odesílá klientovi.

Tento rozdíl je zobrazen na obrázku 1Obrázek 1. Tímto způsobem je výsledek zobrazení stránky stejný bez ohledu na prohlížeč. Nevýhodou je menší interaktivita, protože zpracování probíhá vždy až po odeslání požadavku na server.



Obrázek 1 Práce kódu PHP a JavaScriptu (zdroj: [13])

Soubory s PHP kódem mají příponu .php, kdy vlastní kód se zapisuje mezi značkami `<?php` a `?>`.

Využití PHP

PHP nabízí podmíněné příkazy pro větvení programu. Syntaxe příkazů `if`, `else` a `elseif` je podobná jako v JavaScriptu. V závislosti na platnosti podmínky se provede jeden nebo více příkazů, které mohou být vnořeny do složených závorek {}. Kromě příkazu `if` existuje také příkaz `switch` pro opakované vyhodnocování stejného výrazu. PHP také podporuje podmíněný výraz (ternární operátor) s obecnou syntaxí podmínka `? true : false`.

PHP se často používá v kombinaci s databází MySQL. Pro připojení k MySQL databázi v PHP se používají funkce jako `mysqli_connect` (modernější varianta) nebo starší funkce jako `mysql_connect`. Po připojení je nutné vybrat databázi pomocí funkce `mysqli_select_db` a nastavit znakovou sadu pomocí funkce `mysqli_set_charset`. K odesílání a zpracování SQL dotazů slouží funkce jako `mysqli_query` a pro získání výsledků dotazu funkce jako `mysqli_fetch_array`.

PHP se také hojně využívá ve světě redakčních systémů. Populární platformy jako WordPress, Joomla nebo Drupal jsou postaveny na PHP a tvoří základ pro miliony webových stránek po celém světě. Jazyk je také nedílnou součástí mnoha e-commerce řešení, jako jsou Magento nebo WooCommerce, které umožňují vytvářet a spravovat online obchody. Jeho jednoduchá syntaxe a široká komunita vývojářů z něj dělají ideální volbu jak pro začátečníky, tak pro zkušené programátory.

Nevýhody PHP se často spojují s některými jeho historickými omezeními a výzvami, které mohou ovlivnit vývoj a správu aplikací. Jedním z hlavních problémů je bezpečnost, zejména u starších aplikací, které mohou být náchylné k zranitelnostem, pokud nejsou pravidelně aktualizovány a správně spravovány. Dalším aspektem je nečitelnost kódu, která v minulosti vznikala kvůli nedostatku jasných standardů, což vedlo ke špatně strukturovaným a obtížně udržovatelným projektům. PHP také čelí konkurenci moderních programovacích jazyků, jako jsou Python, Node.js nebo Ruby, které jsou v některých případech považovány za pokročilejší, efektivnější nebo lépe přizpůsobené specifickým typům aplikací. Tyto faktory mohou vést k tomu, že někteří vývojáři dávají přednost jiným technologiím.

2.5 MySQL

MySQL je jeden z nejpopulárnějších relačních databázových systémů na světě. Tento systém, který je open-source a založený na jazyce SQL (Structured Query Language), slouží k ukládání, správě a organizaci dat v relačním modelu. MySQL je široce využíván v různých oblastech od malých webových aplikací až po velké podnikové systémy, a to díky své flexibilitě, výkonnosti a robustnímu ekosystému.

MySQL, stejně jako většina databázových systémů, funguje na principu relační databáze, což znamená, že data jsou organizována do tabulek, které se skládají z řádků (záznamů) a sloupců (atributů). Každý sloupec musí mít unikátní jméno a každá tabulka by měla obsahovat primární klíč, což dělá každý záznam v databázi unikátní. Samotná tabulka musí mít také jedinečný název, jelikož v jedné databázi nemůže existovat více tabulek se stejným názvem. V tabulce se mohou vyskytovat i sloupce obsahující odkazy (cizí klíč), které odkazují na primární klíč jiné tabulky v databázi, což slouží k propojení více tabulek v rámci jedné databáze [11].

Základní příkazy MySQL

Mezi základní SQL příkazy patří:

- **SELECT**: Používá se k získání dat z databáze.
- **INSERT INTO**: Využívá se k vložení nových dat do databázové tabulky.
- **UPDATE**: Slouží k aktualizaci již vložených dat.
- **WHERE**: Definuje podmínky, které musí splňovat příkaz.
- **INNER JOIN**: Umožňuje vytvářet propojení mezi více tabulkami na základě primárních a cizích klíčů.

Pro přístup k datům v databázi musí být databázový systém na serveru nepřetržitě spuštěn [11].

Velkou výhodou MySQL je jeho výkonnost a schopnost zvládat velké objemy dat. Díky optimalizacím, jako je indexování, cachování a podpora transakcí, dokáže MySQL poskytovat rychlé odpovědi i při vysoké zátěži. To z něj činí ideální volbu pro aplikace, které vyžadují rychlý přístup k datům, například e-commerce platformy, systémy pro správu obsahu nebo sociální sítě.

Další silnou stránkou MySQL je jeho kompatibilita a podpora různých platforem. MySQL běží na většině operačních systémů, jako jsou Windows, Linux nebo macOS, a je snadno integrován s mnoha programovacími jazyky, včetně PHP, Pythonu, Javy nebo Ruby. Tato flexibilita umožňuje jeho použití v širokém spektru projektů a technologií.
MySQL je také známý svou bezpečností a spolehlivostí. Podporuje autentizaci uživatelů, šifrování dat a různé úrovně přístupových práv, což zajišťuje, že data jsou chráněna před neoprávněným přístupem. Navíc nabízí mechanismy pro zálohování a obnovu dat, což minimalizuje riziko jejich ztráty [16].

2.6 PWA

Progressive Web Applications (PWA) představují moderní přístup k vývoji webových aplikací, který spojuje výhody tradičních webových stránek a nativních mobilních aplikací. Tento přístup umožňuje vývojářům vytvářet aplikace, jež jsou rychlé, responzivní, bezpečné a dostupné na jakémkoliv zařízení, aniž by uživatelé museli cokoliv instalovat z aplikačních obchodů. Hlavními charakteristikami PWA jsou responzivita, offline funkčnost, rychlost, možnost instalace na plochu, bezpečnost a automatické aktualizace na pozadí. Díky responzivnímu designu se PWA přizpůsobí různým velikostem obrazovek, zatímco technologie Service Worker umožňuje fungování aplikací i bez aktivního připojení k internetu. Tato technologie ukládá klíčové zdroje, jako jsou HTML, CSS, JavaScript, obrázky a data, do mezipaměti, což zajišťuje hladký chod aplikace i v offline režimu.

PWA jsou optimalizovány pro rychlé načítání a nabízejí uživatelům spolehlivou funkci i při pomalém připojení. Uživatelé si navíc mohou aplikace instalovat přímo na domovskou obrazovku svého zařízení, kde se spouštějí v samostatném okně bez adresního řádku prohlížeče, což vytváří dojem nativní aplikace. Bezpečnost PWA je zajištěna komunikací prostřednictvím HTTPS, která chrání aplikace před útoky typu man-in-the-middle. Aktualizace probíhají automaticky na pozadí, takže uživatelé mají vždy k dispozici nejnovější verzi aplikace.

Za vývojem PWA stojí několik klíčových technologií. Service Worker je nezbytný pro offline funkčnost a správu mezipaměti, zatímco Application Shell Architecture zajišťuje rychlé načítání aplikací oddělením statické části od dynamického obsahu. Web App Manifest je soubor

ve formátu JSON, který obsahuje metadata o aplikaci, jako jsou název, ikona, barvy a URL pro spuštění, a umožňuje prohlížečům nabídnout instalaci aplikace. Dalšími důležitými technologiemi jsou Push API a Notifications API, které umožňují odesílat oznámení i v případě, že aplikace není aktivní.

Proces vývoje PWA zahrnuje několik klíčových kroků. Nejprve je třeba definovat cíle aplikace a naplánovat její funkce a uživatelské rozhraní. Při návrhu je důležité vytvořit responzivní design, který zajistí správné zobrazení aplikace na různých zařízeních. Implementace Service Worker je dalším klíčovým krokem, protože umožňuje offline funkčnost a správu požadavků na síť. Web App Manifest se pak umisťuje do kořenového adresáře aplikace a jeho správná konfigurace je klíčová pro možnost instalace. Optimalizace výkonu zahrnuje minimalizaci velikosti zdrojových souborů, optimalizaci obrázků a zajištění rychlého načítání aplikace, například s pomocí nástrojů jako Google Lighthouse nebo webpack. Před nasazením je nutné aplikaci důkladně otestovat na různých zařízeních a v různých prohlížečích a zajistit její nasazení na serveru podporujícím HTTPS [17].

PWA mají několik výhod. Patří mezi ně nižší náklady na vývoj oproti nativním aplikacím, snadnější distribuce bez nutnosti schvalování v aplikačních obchodech a široká dostupnost na všech zařízeních s moderním prohlížečem. Na druhé straně mají i omezení, například omezený přístup k některým nativním funkcím zařízení nebo závislost na podpoře PWA v prohlížečích. Přesto PWA představují budoucnost webového vývoje, protože nabízejí flexibilitu a uživatelskou přívětivost při nižších nákladech. Vytvoření kvalitní PWA však vyžaduje důkladné plánování, precizní implementaci a optimalizaci výkonu, aby aplikace splňovala očekávání uživatelů a mohla konkurovat nativním aplikacím [18].

3 INTERNETOVÁ APLIKACE VLEČKOSTROJ

Efektivní řízení pohybu a odstavení železničních strojů na vlečkách představuje v současné době jednu z klíčových výzev v oblasti železniční dopravy. Vlečky, jakožto specifická část železniční infrastruktury, slouží primárně k odstavení a manipulaci se železničními stroji, zejména v průmyslových závodech, depech a dalších specializovaných areálech. Efektivní organizace a řízení provozu na vlečkách je zásadní nejen z hlediska bezpečnosti, ale i z hlediska plynulosti provozu, provozních nákladů a kapacity. Nevhodné řízení vlečky může vést k provozním konfliktům, prodlevám, neefektivnímu využití kapacit a v nejhorším případě k nehodám a poškození železniční infrastruktury nebo samotných strojů. Vzhledem k tomu, že provoz na vlečkách často probíhá v prostředí s vysokou hustotou provozu a s omezeným počtem dostupných kolejí, je klíčové, aby byl tento provoz řízen pomocí sofistikovaného softwarového řešení, které umožňuje nejen monitorování aktuální situace, ale i prediktivní plánování a automatizované řízení odstavení strojů.

Jedním z moderních softwarových řešení, které je určeno právě pro tento účel, je program Vlečkostroj, kterým se zabývá autor této práce. Tento program představuje systém pro monitorování, řízení a automatizované plánování odstavení železničních strojů na vlečkách. Hlavním cílem programu je zvýšit efektivitu provozu na vlečkách, eliminovat provozní konflikty a zajistit optimální využití dostupné infrastruktury. Program Vlečkostroj využívá moderní webové technologie, mapové podklady a automatizované algoritmy pro rozhodování o přidělení strojů na konkrétní koleje. Vzhledem k tomu, že se jedná o systém postavený na webových technologiích, je možné jej provozovat v libovolném moderním webovém prohlížeči, přičemž je optimalizován jak pro použití na desktopových počítačích, tak na mobilních zařízeních.

V následujícím textu se podrobně zaměříme na technické aspekty programu Vlečkostroj, jeho architekturu, základní funkcionality, implementaci offline režimu, bezpečnostní opatření a možnosti dalšího rozvoje. Cílem této analýzy je poskytnout ucelený přehled o tom, jakým způsobem tento systém přispívá k optimalizaci provozu na vlečkách a jaké možnosti přináší jeho další rozvoj.

3.1 Představení společnosti Hrochostroj a.s.

Společnost Hrochostroj a.s., založená v roce 2015, je dynamicky se rozvíjející firmou specializující se na údržbu a obnovu železniční infrastruktury. Její vozový park čítá 10 specializovaných strojů, jako jsou podbíječky pro směrovou a výškovou úpravu kolejí

a výhybek, štěrkové pluhy pro úpravu štěrkového lože, čističky kolejového lože s následným odvozem vytěženého materiálu a 2 lokomotivy.

V lednu 2025 společnost oznámila rozšíření své flotily o další lokomotivu EffiShunter 1000 od českého výrobce CZ LOKO a.s., která by měla být v provozu za dva roky.

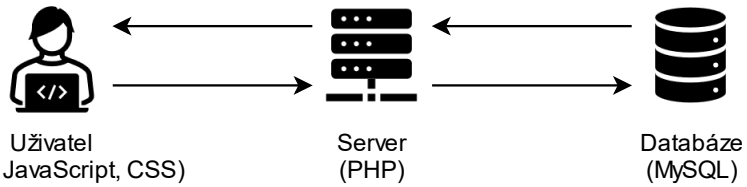
Hrochostroj a.s. je součástí stavebního holdingu enteria a.s., který sdružuje české stavební společnosti s ročním obratem téměř 10 miliard Kč a zaměstnává přibližně 1 400 zaměstnanců.

Enteria a.s. vznikla v roce 2008 jako holdingová společnost, která zastřešila firmu Chládek a Tintěra, Pardubice a.s., včetně jejich akvizic. Cílem bylo vytvořit uspořádání, kde se Chládek a Tintěra, Pardubice a.s. stane sesterskou společností dalších dceřiných firem. Mezi dceřiné společnosti patří například Chládek a Tintěra, Pardubice a.s., Marhold a.s., KVIS a.s. Pardubice, Hrochostroj a.s., INSTAV Hlinsko a.s. a Obalovna Týniště nad Orlicí s.r.o.

Hrochostroj a.s. působí nejen v České republice, ale také na Slovensku, kde se snaží získat osvědčení na svařování kolejnic, aby mohla poskytovat služby pro své partnery, kteří zde vysoutěžili zakázky. Větší expanzi do zahraničí však společnost v současné době neplánuje.

3.2 Technické specifikace systému

Program Vlečkostroj je navržen jako webová aplikace, která umožňuje řízení a monitorování pohybu železničních strojů na vlečkách na základě reálných dat. Technické řešení systému je postaveno na využití webových technologií, konkrétně na kombinaci HTML, JavaScript, PHP a CSS viz obrázek 2. Kromě základní webové architektury využívá systém několik specializovaných knihoven a frameworků, které zajišťují klíčové funkcionality.



Obrázek 2 Schéma systému (autor)

Jedním z klíčových prvků systému je knihovna Leaflet.js, která umožňuje vizualizaci polohy strojů na mapě na základě poskytnutých dat. Leaflet.js je open-source knihovna určená pro práci s mapovými podklady, která poskytuje širokou škálu funkcí včetně možnosti přidávání vlastních vrstev, zobrazení vektorových objektů, přidávání interaktivních prvků na mapu a integrace s různými zdroji mapových dat [19]. Program Vlečkostroj využívá Leaflet.js k vykreslení kolejí vlečky do mapy a zobrazení rozložení vozů na vlečce. Polohy strojů jsou

aktualizovány v pravidelných intervalech, což umožňuje uživateli sledovat polohy strojů v takřka reálném čase.

Aplikace Vlečkostroj využívá jako mapový podklad OpenStreetMap prostřednictvím knihovny Leaflet. Tento přístup zajišťuje detailní a otevřená geografická data, která jsou volně dostupná a pravidelně aktualizovaná komunitou přispěvatelů. Díky použití OpenStreetMap může aplikace zobrazovat přesné umístění kolejí, strojů a dalších prvků bez nutnosti licencovat komerční mapové služby [20].

Dalším důležitým prvkem je knihovna jQuery, která je využívána pro manipulaci s DOM strukturou, obsluhu událostí a zpracování AJAX požadavků. Díky použití jQuery je možné implementovat pokročilé uživatelské rozhraní s dynamickou aktualizací dat bez nutnosti obnovy celé stránky. Tím je dosaženo vysoké plynulosti aplikace a rychlé reakce na změny v provozu [21].

Systém je navržen jako PWA, což znamená, že je možné jej instalovat na zařízení jako nativní aplikaci a používat i v offline režimu. K tomu slouží komponenta Service Worker, která ukládá klíčová data do cache a umožňuje jejich načítání bez nutnosti připojení k internetu. Service Worker zároveň umožňuje přijímání push notifikací a automatickou aktualizaci dat na pozadí.

Datová vrstva systému je založena na relační databázi MySQL, která obsahuje výchozí informace o strojích a kolejích a ukládá informace o poloze strojů, stavech kolejí a provozních událostech.

3.3 Databáze

Každá moderní webová nebo mobilní aplikace potřebuje pro svůj provoz množství dat, která definují její obsah, chování a funkčnost. Aplikace *Vlečkostroj*, sloužící k vizualizaci a správě strojů na železničních kolejích, není výjimkou. V této aplikaci představují data základní prvek, bez kterého by nebylo možné zobrazit aktuální stav strojů nebo vykreslit mapu.

Všechna data jsou uložena v relační databázi MySQL, která běží na interním serveru společnosti Hrochostroj a.s. Struktura databáze je navržena tak, aby byla dostatečně flexibilní a zároveň efektivní z hlediska dotazování. Každý logický celek dat (koleje, stroje, události) je reprezentován samostatnou tabulkou s jednoznačně definovanými klíči a případnými relacemi mezi tabulkami.

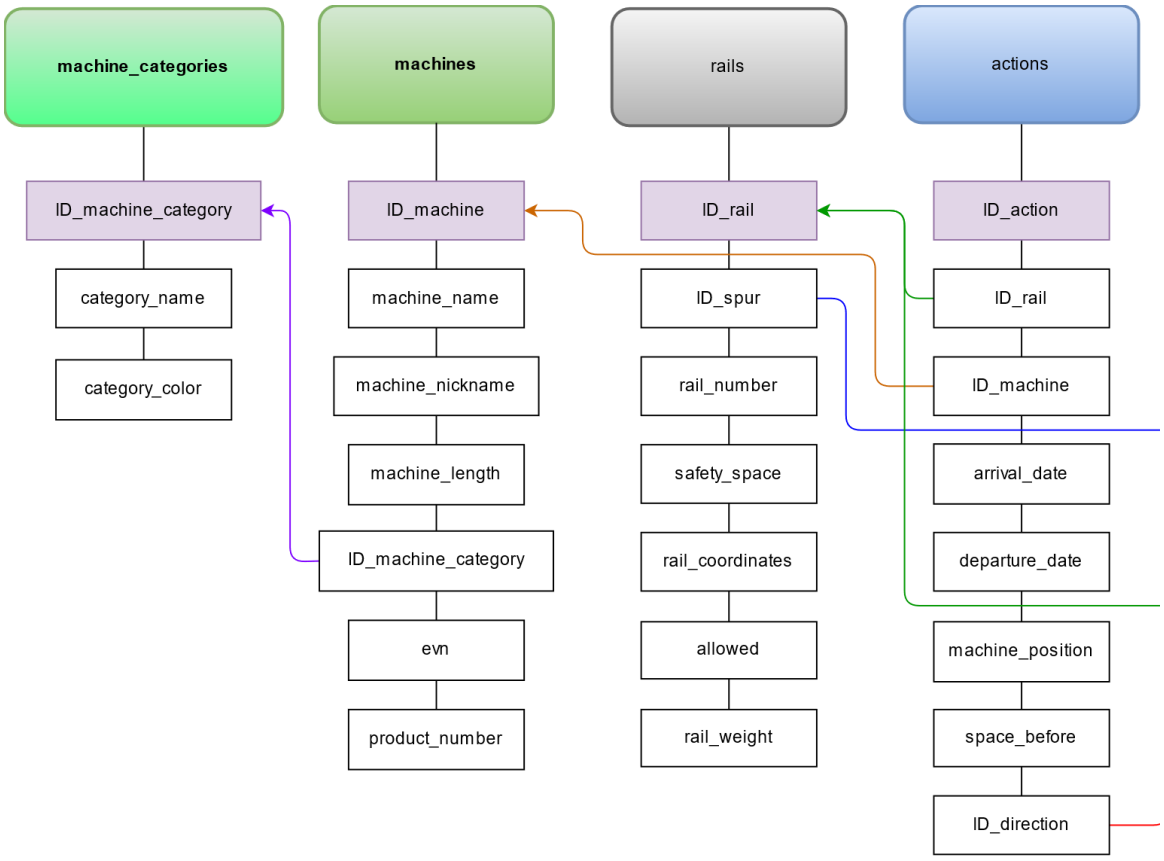
3.3.1 Návrh databáze

Při vývoji aplikace, jejímž cílem je správa a vizualizace železniční vlečky a strojů, hraje klíčovou roli promyšlený výběr dat a odpovídající návrh databáze. Aby bylo možné zajistit

funkční, spolehlivý a efektivní systém, je nezbytné nejprve definovat, jaká data budou pro chod aplikace nezbytná, jaké budou jejich vzájemné vztahy a jakým způsobem budou tato data strukturovaně uložena.

Proces výběru dat nezbytných pro fungování aplikace začal důkladnou analýzou zadání a požadavků na funkčnost celého systému. Bylo nutné přesně vymezit, jaké informace bude aplikace zpracovávat, jaké datové entity bude obsahovat a jaké vztahy mezi těmito entitami musí být zachovány. Analýza byla provedena s cílem navrhnout takovou datovou strukturu, která bude efektivní nejen z hlediska technické realizace, ale i z pohledu budoucí správy, rozšiřitelnosti a možnosti interpretace dat v grafickém rozhraní.

Aby byl celý návrhový proces přehlednější a systematičtější, byl vytvořen databázový diagram, jehož část je zobrazena na obrázku 3 a celé schéma se nachází v příloze A. Diagram slouží jako vizuální znázornění logické struktury databáze. Tento diagram obsahuje názvy jednotlivých tabulek (databázových entit), definice jejich atributů (sloupců) a také propojení mezi tabulkami pomocí relací – například prostřednictvím primárních a cizích klíčů.



Obrázek 3 Část schématu databáze (autor)

3.3.2 Obsah databáze

Databázová struktura aplikace *Vlečkostroj* byla navržena tak, aby umožňovala efektivní ukládání a správu informací o strojích, kolejích, vlečkách, směrech a akcích. Celý systém je postaven na relačním modelu, ve kterém jsou jednotlivé datové entity rozděleny do samostatných tabulek. Tyto tabulky mezi sebou komunikují prostřednictvím jednoznačných identifikátorů (primárních a cizích klíčů), čímž je zajištěna integrita a konzistence uložených dat.

Tabulka *machines*

Tabulka *machines* uchovává informace o jednotlivých strojích. Každý záznam v této tabulce obsahuje informace zobrazené v tabulce 1Tabulka 1.

Tabulka 1 *machines*

Název	Typ	Popis
ID_machine	int	Unikátní identifikátor stroje
machine_name	string	Název stroje
machine_nickname	string	Přezdívka stroje
machine_length	float	Délka stroje
ID_machine_category	int	Identifikátor typu stroje
evn	string	EVN (Evropské číslo vozidla)
product_number	string	Výrobní číslo vozidla

Zdroj: autor

Díky použití cizího klíče *ID_machine_category* je možné ke každému stroji přiřadit konkrétní kategorii, čímž se eliminuje opakování slovních označení a zjednodušuje se práce s větším množstvím dat.

Tabulka *machine_categories*

Tabulka *machine_categories* slouží k evidenci jednotlivých typů strojů a její rozložení je zobrazeno v tabulce 2Tabulka 2.

Tabulka 2 *machine_categories*

Název	Typ	Popis
ID_machine_category	int	Unikátní identifikátor typu stroje
category_name	string	Název kategorie
category_color	string	Barva kategorie

Zdroj: autor

Hlavní funkcí tabulky *machine_categories* je centralizace označení typů strojů, což přispívá k lepší udržitelnosti a přehlednosti databáze.

Tabulka *rails*

Tabulka *rails* obsahuje informace o jednotlivých kolejích, přičemž každý záznam obsahuje informace z tabulky 3Tabulka 3.

Tabulka 3 *rails*

Název	Typ	Popis
ID_rail	int	Unikátní identifikátor koleje
ID_spur	int	Identifikátor vlečky
rail_number	int	Číslo koleje
safety_space	int	Bezpečnostní mezera u výhybky
rail_coordinates	string	Souřadnice bodů koleje
allowed	int	Povolení odstavování na koleji
rail_weight	float	Váha koleje pro TOPSIS

Zdroj: autor

Každá kolej je tak jednoznačně přiřazena k určité vlečce prostřednictvím vazby na tabulku *spurs*. V provozním řádu vlečky jsou stanoveny koleje, které slouží pouze pro posun strojů a je na nich odstavování strojů zakázáno. Pro tyto koleje je v databázi nastavena vlastnost *allowed* na hodnotu 0. Pro speciální koleje, jako například koleje vedoucí do opravárenské haly je vlastnost *allowed* nastavena na hodnotu 2, což značí, že na koleji je možné stroje odstavovat, ale pro fungování programu jsou tyto koleje odděleny a uživatel si musí vybrat, že chce tyto speciální koleje zařadit do výběru kolejí.

Tabulka *spurs*

Tabulka *spurs* uchovává základní údaje o jednotlivých vlečkách a její rozložení je zobrazeno v tabulce 4Tabulka 4.

Tabulka 4 *spurs*

Název	Typ	Popis
ID_spur	int	Unikátní identifikátor vlečky
location	string	Název vlečky
view	string	Souřadnice pro vykreslení vlečky
zoom	float	Přiblížení mapy na vlečku

entry_node	string	Vstupní bod vlečky

Zdroj: autor

Tyto údaje umožňují specifické zobrazení informací o vlečkách a pomáhají správnému vykreslení mapových podkladů a funkcionalitě programu.

Tabulka *directions*

Směry, ve kterých mohou být stroje orientovány, jsou zaznamenány v tabulce *directions*, která obsahuje informace z tabulky 5Tabulka 5.

Tabulka 5 *directions*

Název	Typ	Popis
ID_direction	int	Unikátní identifikátor směru otočení
ID_spur	int	Identifikátor vlečky
orientation	string	Směr otočení stroje

Zdroj: autor

Tabulka *directions* slouží k centralizaci možných směrů otočení strojů, což přispívá k lepší udržitelnosti a přehlednosti databáze snížením opakujících se vstupů v tabulce *actions*.

Tabulka *actions*

Nejdůležitější tabulkou v rámci celého systému je tabulka *actions*, která eviduje jednotlivé události odstavení strojů. Každý záznam reprezentuje jednu akci a obsahuje informace zobrazené v tabulce 6Tabulka 6.

Tabulka 6 *actions*

Název	Typ	Popis
ID_action	int	Unikátní identifikátor akce
ID_rail	int	Identifikátor koleje
ID_machine	int	Identifikátor stroje
arrival_date	string	Datum a čas příjezdu stroje
departure_date	string	Datum a čas odjezdu stroje
machine_position	int	Pořadí stroje na koleji
space_before	int	Bezpečnostní mezera před strojem
ID_direction	int	Identifikátor směru otočení stroje

Zdroj: autor

Provázání této tabulky s ostatními tabulkami (*machines*, *rails*, *directions*) umožňuje ke každé akci dohledat podrobné informace jako: který stroj byl kdy a kam odstaven, jak byl otočen, zda před ním byla mezera a na jaké vlečce se celá operace odehrála. Nepřímé vazby na tabulku *spurs* jsou zajištěny skrze vazby přes *rails* nebo *directions*.

Tabulka *arrivals*

Poslední tabulkou je tabulka *arrivals*, které je určena k evidenci příjezdů a odjezdů strojů a obsahuje informace zobrazené v tabulce 7Tabulka 7.

Tabulka 7 *arrivals*

Název	Typ	Popis
ID_arrival	int	Unikátní identifikátor příjezdu/odjezdu
ID_actions	string	Pole identifikátorů akcí
ID_rail	int	Identifikátor koleje
arriving_machines	string	Pole strojů, které přijedou/odjedou
directions	string	Pole otočení strojů
spaces_before	string	Pole bezpečnostních mezer
arrival_date	string	Datum a čas příjezdu stroje
departure_date	string	Datum a čas odjezdu stroje
arrived	int	Označení příjezdu
departed	int	Označení odjezdu

Zdroj: autor

Tabulka *arrivals* slouží k záznamu budoucích a historických příjezdů a odjezdů strojů na jednotlivé koleje, což umožňuje funkci plánování těchto akcí do budoucna. Tabulka *arrivals* na rozdíl od ostatních tabulek využívá v některých svých vstupech pole informací, s kterými se pracuje složitěji, ale umožňují ukládat informace hned pro několik strojů či událostí, což je zásadní pro plánování příjezdu či odjezdu více strojů najednou. Vlastnost *arrived* označuje, zdali stroje daného vstupu přijely (hodnota 1 pokud ano) a podobně tak vlastnost *departed* označuje, zda stroje odjely (hodnota 1 pokud ano). Navíc tyto vlastnosti mohou nabývat hodnoty 2 v případě, že byl plánovaný příjezd či odjezd zrušen.

Celková logika databázového modelu umožňuje nejen uchovávat historická i aktuální data o provozu, ale také zobrazovat stroje a koleje v mapovém rozhraní aplikace. Uživatelé

mohou přehledně sledovat polohu strojů, plánovat jejich odstavení s ohledem na dostupnou délku koleje a přehledně analyzovat provozní historii.

Struktura databáze je navržena s ohledem na škálovatelnost a budoucí rozšíření systému. Relační vazby mezi tabulkami zajišťují konzistentní a udržitelný datový model, který tvoří robustní základ pro efektivní řízení provozu na vlečkách

3.4 Offline režim a bezpečnost

Aplikace *Vlečkostroj* je navržena jako PWA, což umožňuje uživatelům pracovat s aplikací i bez připojení k internetu. Offline funkcionalita je zajištěna kombinací skriptu `pwa_script.js`, service worker scriptu `sw.js` a manifestem `manifest.json` [17][18].

Základem offline režimu je technologie service worker, která funguje jako prostředník mezi aplikací a internetem. Tento skript běží nezávisle na samotném webu a umožňuje zachytávat a spravovat síťové požadavky, a tím i zajišťovat, že důležité části aplikace budou dostupné i při výpadku připojení. Ve *Vlečkostroji* je tímto hlavním service workerem soubor `sw.js`, který je zodpovědný za caching statických zdrojů (HTML, CSS, JavaScript, obrázky apod.) a obsluhu dynamických API požadavků. Při instalaci se uloží vybrané prostředky do cache a při spuštění aplikace se požadavky na tyto prostředky získají přímo z ní. V případě požadavků na API (například získávání aktuálních dat o strojích) se worker nejprve pokusí o síťové připojení a pokud není dostupné, vrátí uloženou verzi z cache.

Samotné propojení těchto funkcí s uživatelským rozhraním a logikou aplikace zajišťuje skript `pwa_script.js`. Ten se spouští ihned po načtení stránky a provádí několik důležitých činností. V první řadě registruje service worker (`sw.js`) a zajišťuje jeho aktualizaci. Pokud je dostupná novější verze, uživatel je vyzván k jejímu načtení, čímž se zajistí, že aplikace vždy běží v nejaktuálnější podobě. Dále je zde implementována funkcionalita pro zobrazení stavu připojení, který uživatele informuje o tom, zda je online či offline.

Kromě toho `pwa_script.js` umožňuje instalaci aplikace jako PWA na domovskou obrazovku zařízení. Pokud jsou splněny podmínky pro instalaci, zobrazí se uživateli tlačítko „Nainstalovat aplikaci“. Po jeho potvrzení je aplikace přidána mezi ostatní nativní aplikace zařízení, což zvyšuje její dostupnost a komfort používání. Důležitou funkcí je rovněž podpora periodické synchronizace na pozadí, která umožňuje aplikaci i po zavření aktualizovat data na pozadí.

Pro definování vlastností *Vlečkostroj* jako PWA instalované aplikace slouží soubor `manifest.json` [22], který definuje základní informace o aplikaci – její název, ikony, výchozí

stránku, barevné schéma a další vlastnosti. Toto umožňuje aplikaci chovat se jako nativní, což znamená například spuštění přes ikonu na ploše či otevření v režimu celé obrazovky.

Offline funkcionalita aplikace *Vlečkostroj* tak umožňuje přístup k informacím i v podmínkách, kde není možné se spolehnout na stabilní připojení k internetu. Uživatelé tak mohou v omezeném režimu využívat aplikaci i v terénu nebo v místech s horším připojením.

3.5 Práce s daty pomocí PHP

V této kapitole je popsán způsob získávání a aktualizace dat na straně serveru pomocí PHP skriptů a jejich následné zpracování a zobrazení na straně klienta. PHP soubory slouží jako prostředník mezi klientskou aplikací (uživatelským rozhraním) a databází. Tento přístup umožňuje efektivní práci s daty v reálném čase a zajišťuje, že uživatelské rozhraní bude vždy zobrazovat aktuální informace. Aplikace obsahuje několik PHP souborů, z nichž každý má jasně definovanou funkci.

Soubor `connect.php` (obrázek 4Obrázek 4) je základním prvkem celé webové aplikace, protože slouží k navázání spojení s databází. V kontextu architektury PHP aplikací se jedná o běžně používaný soubor, který centralizuje a zajišťuje opakovatelné a bezpečné připojení ke všem databázovým operacím.

organizace-vlecky - connect.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "vleckostroj";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Obrázek 4 Připojení k databázi vleckostroj v lokálním prostředí (autor)

Ve své podstatě tento soubor obsahuje přihlašovací údaje k databázovému serveru, a to název serveru, název databáze, jméno uživatele a heslo pro přihlášení do databáze. Pomocí

objektu `new mysqli` se tento skript pokouší navázat spojení s databází. V případě chyby se vyvolá výjimka, která obvykle ukončí běh skriptu a zobrazí chybovou hlášku, pokud není výjimka zachycena [23].

Soubory pro získání dat jsou volány při načtení stránky v scriptu `fetch_data.js`, který volá funkci `fetchAllData()`. Tato funkce asynchronně načte veškerá potřebná data pomocí PHP skriptů z tabulku 8Tabulka 8.

Tabulka 8 Soubory pro získání dat

PHP soubor	Získávaná data
get_machines.php	Všechna data o strojí z *machines*
get_rails.php	Všech data o kolejích z *rails*
get_spurs.php	Všechna data o vlečkách ze *spurs*
get_directions.php	Všechna data o směrech z *directions*
get_actions.php	Vybraná data o akcích z *actions*
get_arrivals.php	Vybraná data o příjezdech z *arrivals*

Zdroj: autor

Soubor `get_machines.php` získává data o strojích (*machines*) uložená v databázi a kombinuje je s daty z *machine_categories* pomocí MySQL příkazu `INNER JOIN` kdy jsou tyto tabulky spojeny přes identifikátor *ID_machine_category*. Výsledná data poté obsahují kompletní údaje o strojích jako je jejich délka, název, typ, označení a další. Soubor `get_rails.php` slouží k načítání informací o kolejích a také se pomocí `INNER JOIN` kombinuje s daty ze *spurs* přes identifikátor *ID_spur* pro získání kompletních dat o kolejích, tj. souřadnicích koleje, čísla koleje a označení vlečky na které se kolej nachází. Soubor `get_spurs.php` slouží k získávání informací o vlečkách, aby bylo možné vizualizovat vlečky v aplikační části programu. Soubor `get_directions.php` získává data o možných otočení strojů a slouží jako zdroj dat pro tabulku *machines*. Největším souborem získávajícím data je `get_actions.php`, který poskytuje data o akcích a propojuje je s *machines*, *rails*, *directions* a *spurs* pro zajištění všech potřebných informací pro každou akci. Posledním souborem získávajícím data je soubor `get_arrivals.php`, který poskytuje data o budoucích odstaveních či odjezdech.

Vzhledem k tomu, že počet akcí a příjezdů bude v průběhu užívání programu neustále růst, je nutné filtrovat data rozsah získávaných dat již na straně serveru. K tomu slouží příkaz `WHERE`, po kterém následuje seznam podmínek, jež musí být splněny, aby byly informace vstup

přidány do odpovědi požadavku. Limitujícími podmínkami pro `get_actions.php` jsou data příjezdu a odjezdu strojů a jejich pozice. Je zbytečné získávat data akce pro stroje, který ještě nepřijel nebo již odjel nebo nemá pozici na koleji. Celý příkaz pro získání dat akcí je zobrazen na obrázku 5Obrázek 5.

organizace-vlecky - get_actions.php

```
13  $sql = "SELECT actions.*, machines.*, rails.*, directions.*, spurs.*, machine_categories.*
    FROM actions
14          INNER JOIN machines ON actions.ID_machine = machines.ID_machine
15          INNER JOIN rails ON actions.ID_rail = rails.ID_rail
16          INNER JOIN directions ON actions.ID_direction = directions.ID_direction
17          INNER JOIN spurs ON rails.ID_spur = spurs.ID_spur
18          INNER JOIN machine_categories ON machines.ID_machine_category = machine_categories.
    ID_machine_category
19          WHERE actions.arrival_date < NOW() && (actions.departure_date > NOW() || actions.de
    parture_date IS NULL) && actions.machine_position IS NOT NULL
20          ORDER BY actions.ID_rail, actions.machine_position";
```

Obrázek 5 SQL příkaz pro získání dat akcí (autor)

Všechny soubory pro získání dat jsou volány pomocí `fetch` a jejich odpověď, ve formátu JSON, je uložena do globálních proměnných pro pozdější využití, jak je tomu na obrázku 6Obrázek 6.

organizace-vlecky - fetch_data.js

```
23  // Fetch all data concurrently
24  const [machinesResponse, actionsResponse, railsResponse, spursResponse,
    directionResponse, arrivalsResponse] = await Promise.all([
25      fetch('get_machines.php'),
26      fetch('get_actions.php'),
27      fetch('get_rails.php'),
28      fetch('get_spurs.php'),
29      fetch('get_directions.php'),
30      fetch('get_arrivals.php'),
31  ]);
32
33  // Parse JSON responses
34  machinesData = await machinesResponse.json();
35  actionsData = await actionsResponse.json();
36  railsData = await railsResponse.json();
37  spursData = await spursResponse.json();
38  directionsData = await directionResponse.json();
39  allArrivals = await arrivalsResponse.json();
```

Obrázek 6 Získání dat (autor)

Aplikace dále obsahuje dva soubory pro aktualizaci dat, které jsou vypsány v tabulce 9Tabulka 9.

Tabulka 9 Soubory pro aktualizaci dat

PHP soubor	Aktualizované tabulky
update_actions.php	*actions*
update_arrivals.php	*arrivals*

Zdroj: autor

Tyto soubory pro aktualizaci dat jsou volány v různých místech programu, podle potřeby jejich využití. Oba soubory pro aktualizaci dat jsou volány funkcí `fetch()` s metodou typu `POST`, jelikož je bezpečnější z pohledu citlivosti dat (např. přístupové údaje nebo ID strojů nejsou viditelně zobrazeny v URL).

3.5.1 Příklad získání dat o strojích

Získávání dat zahrnuje odeslání požadavku z klientské části aplikace na server (obrázek 6Obrázek 6), zpracování požadavku na straně serveru pomocí PHP skriptu (obrázek 7), získání požadovaných dat z databáze a následné odeslání dat klientovi. Tento proces probíhá následovně:

1. Odeslání požadavku na server pomocí `fetch('get_machines.php')`
2. Připojení k databázi pomocí `connect.php`
3. Provedení SQL dotazu na tabulku *machines*
4. Zpracování dotazu do pole *$machine* a pomocí funkce `json_encode()` převedení na JSON formát
5. Odeslání dat zpět klientovi
6. Přijetí dat a zpracování JSON

organizace-vlecky - get_machines.php

```php
<?php
header('Content-Type: application/json');
require_once 'connect.php';

$result = mysqli_query($conn, "SELECT machines.*, machine_categories.*
FROM machines
    INNER JOIN machine_categories ON machines.ID_machine_category = ma
chine_categories.ID_machine_category");
$machines = [];

if ($result) {
    while ($row = mysqli_fetch_assoc($result)) {
        $machines[] = $row;
    }
} else {
    echo json_encode(['error' => mysqli_error($conn)]);
}

echo json_encode($machines);
?>
```

Obrázek 7 Příklad získání dat (autor)

3.5.2 Příklad aktualizace dat akcí

Aktualizace dat zahrnuje odeslání požadavku z klientské části aplikace na server, zpracování požadavku na straně serveru pomocí PHP skriptu, provedení změn v databázi a následné vrácení odpovědi na klienta. Tento proces probíhá následovně:

1. Odeslání požadavku na server (obrázek 8Obrázek 8)
2. Přijetí požadavku a dat ve formátu JSON
3. Připojení k databázi pomocí `connect.php`
4. Prověření platnosti přijatých dat
5. Rozhodnutí o typu aktualizovaných dat
6. Načtení proměnných
7. Provedení aktualizace dat
8. Vrácení odpovědi zpět klientovi

```javascript
// Send data to server
fetch('update_action.php', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
```

Obrázek 8 Požadavek pro aktualizaci dat (autor)

PHP skripty jsou navrženy tak, aby byly co nejefektivnější. Každý skript je optimalizován pro konkrétní úkol, například skript `get_machines.php` načítá pouze potřebná data o strojích a neprovádí nadbytečné dotazy na databázi. Výsledný formát JSON je kompaktní, což minimalizuje zatížení sítě a zvyšuje rychlost načítání dat na straně klienta. Tento přístup umožňuje efektivní správu dat v aplikaci.

3.6 Vizualizace kolejí a strojů

Pro vizualizaci kolejí a strojů do interaktivní mapy byla zvolena knihovna Leaflet.js**.** Celý systém je navržen tak, aby umožňoval detailní vizualizaci kolejí, přesné umisťování strojů na koleje a interaktivní práci s mapou, včetně výpočtu vzdáleností, přidávání popupů s informacemi a reakce na požadavky uživatele.

Celý proces začíná inicializací mapy ve funkci `drawMap(spurId)`. Tato funkce přijímá jako vstupní parametr *spurId* a na základě tohoto ID načte odpovídající souřadnice z objektu *spursData*. Souřadnice jsou uloženy jako JSON řetězec, který je dekódován pomocí `JSON.parse()`. Následně se vytvoří nová mapa pomocí `L.map()` z knihovny Leaflet [19]. Mapa je konfigurována tak, aby podporovala plynulé přibližování a oddalování díky parametrům *zoomDelta* a *zoomSnap*, které umožňují jemné kroky při změně měřítka. Souřadnice jsou poté nastaveny pomocí `setView(view)`, což umístí mapu na zadanou pozici. Po inicializaci je přidána vrstva OpenStreetMap [20] pomocí `L.tileLayer`, která načítá dlaždice mapy ze serverů OpenStreetMap s nastavením minimálního a maximálního přiblížení. Toto zajišťuje, že uživatel nemůže mapu příliš oddálit nebo přiblížit mimo stanovený rozsah.

Pro zajištění lepší správy vykreslených vrstev na mapě slouží funkce `setupCustomPanes()`. Tato funkce vytváří dvě samostatné vrstvy (tzv. "panes"), jednu pro koleje (*railsPane*) a druhou pro stroje (*machinesPane*). Vrstva pro koleje je pomocí *z-index* 400 umístěna níže než vrstva pro stroje s *z-index* 450, což znamená, že stroje budou vždy vykreslovány nad kolejemi, což eliminuje problém s překrýváním vykreslených prvků.

Samotné vykreslování kolejí je implementováno ve funkci `drawRails(rail)`. Nejprve jsou souřadnice kolejí zpracovány pomocí funkce `parseCoordinates`, která převede řetězec se souřadnicemi na pole. Pokud je vstupní formát chybný, nahradí nechtěné znaky pomocí `replace`. Jakmile jsou souřadnice načteny, kód zvolí barvu koleje podle ID koleje (*rail.ID_rail*) z předdefinovaného pole barev *railColors*. Pokud počet kolejí překročí počet definovaných barev, barvy se začnou opakovat, což slouží pouze ke zlehčení vizuálního rozlišení kolejí. Kolej je následně vykreslena pomocí metody `L.polyline`, která vytvoří křivku mezi jednotlivými body souřadnic. Na kolej je poté přidána událost `click`, která otevře popup s

informacemi o čísle koleje a tlačítkem pro otevření řazení na koleji. Kromě toho jsou implementovány i události `mouseover` a `mouseout`, které mění barvu a tloušťku koleje při najetí myší, což poskytuje vizuální zpětnou vazbu uživateli.

Výpočet vzdálenosti mezi dvěma body na mapě je implementován pomocí funkce `calculateDistance(point1, point2)`, která používá Haversinovu formuli [24] (obrázek 9). Tato metoda počítá vzdálenost mezi dvěma body na povrchu Země na základě jejich zeměpisných souřadnic.

organizace-vlecky - map.js

```
224  function calculateDistance(point1, point2) {
225      const R = 6378000; // Earth's radius in meters
226      const lat1 = point1[0] * Math.PI / 180;
227      const lat2 = point2[0] * Math.PI / 180;
228      const dLat = (point2[0] - point1[0]) * Math.PI / 180;
229      const dLon = (point2[1] - point1[1]) * Math.PI / 180;
230
231      const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
232                Math.cos(lat1) * Math.cos(lat2) *
233                Math.sin(dLon/2) * Math.sin(dLon/2);
234      const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
235      return R * c;
236  }
```

Obrázek 9 Výpočet vzdáleností mezi dvěma body (autor)

Pro výpočet je použita konstantní hodnota poloměru Země R = 6378 km. Po výpočtu jednotlivých rozdílů zeměpisné šířky a délky jsou rozdíly převedeny na radiány a následně je aplikována Haversinova rovnice k určení vzdálenosti. Tato funkce je klíčová pro přesné umisťování strojů na koleje a výpočet vzdáleností mezi jednotlivými body na koleji.

Pro přesné umisťování strojů po délce koleje, což je jedním z klíčových prvků tohoto systému, slouží funkce `findPointAtDistance(coordArray, targetDistance)` (obrázek 10Obrázek 10), která umožňuje najít bod na koleji ve specifické vzdálenosti od počátku. Funkce iteruje přes jednotlivé segmenty koleje a akumuluje vzdálenost mezi sousedními body. Jakmile je dosaženo požadované vzdálenosti, funkce vypočítá polohu bodu pomocí lineární interpolace mezi dvěma sousedními body.

organizace-vlecky - map.js

```
239 function findPointAtDistance(coordArray, targetDistance, startPoint = null) {
240     let accumulatedDistance = 0;
241     let startIndex = 0;
242 
243     // If startPoint is provided, find the closest point on the rail
244     if (startPoint) {
245         let minDistance = Infinity;
246         coordArray.forEach((coord, index) => {
247             const distance = calculateDistance(startPoint, coord);
248             if (distance < minDistance) {
249                 minDistance = distance;
250                 startIndex = index;
251             }
252         });
253     }
254     // Calculate from the starting point
255     for (let i = startIndex; i < coordArray.length - 1; i++) {
256         const segmentLength = calculateDistance(coordArray[i], coordArray[i + 1]);
257 
258         if (accumulatedDistance + segmentLength >= targetDistance+5) {
259             // Calculate the remaining distance needed on this segment
260             const remainingDistance = targetDistance - accumulatedDistance;
261             // Calculate ratio of remaining distance to segment length
262             const ratio = remainingDistance / segmentLength;
263 
264             // Interpolate the point
265             const lat = coordArray[i][0] + (coordArray[i + 1][0] - coordArray[i][0]) * ratio;
266             const lon = coordArray[i][1] + (coordArray[i + 1][1] - coordArray[i][1]) * ratio;
267 
268             return [lat, lon];
269         }
270         accumulatedDistance += segmentLength;
271     }
272     // If target distance is beyond rail length, return last point
273     return coordArray[coordArray.length - 1];
274 }
```

Obrázek 10 Funkce pro nalezení bodu v dané vzdálenosti (autor)

Pro umístění strojů na koleje slouží funkce `placeParkedMachines(parkedMachines)`. Tato funkce iteruje přes seznam strojů získaných z databáze a umisťuje je na příslušné koleje. Funkce pracuje s kumulovanou vzdáleností, která jí umožňuje umisťovat stroje na přesné místo koleje podle jejich řazení a bezpečnostních mezer. Pokud funkce zjistí, že další stroj je již na jiné koleji, funkce nuluje kumulovanou vzdálenost a začne znovu od počátku koleje. Vykreslování strojů zajišťuje funkce `drawMachineRectangle`, která generuje obdélníky reprezentující stroje v odpovídajících rozměrech vzhledem k jejich délce a otočení. Obdélník je vykreslen pomocí `L.polygon` a jeho vzhled je řízen parametry jako barva, tloušťka a průhlednost. Obdélník obsahuje také popup s detaily o stroji, jako je jeho délka, otočení a bezpečnostní mezera před strojem. Funkce rovněž mění styl obdélníku při najetí myší, tedy zvětšuje se tloušťka obrysu a mění se průhlednost.

3.7 Ruční správa vlečky

Správa a rozmístění železničních strojů na kolejích v rámci vlečky je klíčovým prvkem efektivního provozu drážní dopravy. Soubor `normal_planing.js` představuje jeden z klíčových modulů webové aplikace *Vlečkostroj*. Tento skript zajišťuje veškerou logiku spojenou s ručním přiřazováním strojů ke konkrétním kolejím, a to jak z hlediska uživatelského rozhraní, tak i technického zpracování dat.

Základní funkcionalitu představuje metoda `addMachineToRail`, která se aktivuje při výběru konkrétní koleje. Tato funkce dynamicky zobrazí informace o dané koleji, jako je její délka, aktuální obsazenost a dostupná délka. Zároveň zpřístupní interaktivní prostředí pro výběr strojů, které lze na kolej umístit. Uživatel má k dispozici dva seznamy: seznam všech dostupných strojů a seznam strojů již odstavených nebo vybraných pro danou kolej. Stroje lze mezi těmito seznamy přesouvat kliknutím na tlačítko pro přesunutí a v seznamu strojů odstavených na koleji je možné pomocí funkce `dragDrop` měnit pořadí strojů jejich přetažením. Kromě toho je umožněno nastavit pro každý stroj jeho otočení a případnou mezeru před ním, čímž je dosaženo vysoké míry přizpůsobení.

Další funkcí je `saveMachinesOnRail`, která zpracovává a odesílá rozmístění strojů zpět na server. Tato funkce vytvoří strukturovaný datový objekt obsahující informace o pozici každého stroje, jeho identifikátoru, otočení a nastavené mezeře. V případě, že byl některý stroj z koleje odebrán, označí se jako „odjíždějící“. Po sestavení celkového datového souboru se uživateli zobrazí potvrzovací dialog s výčtem změn. Pokud uživatel změny potvrdí, data se odešlou na server pomocí skriptu `update_action.php`, který je dále zpracuje a uloží do databáze.

Součástí skriptu pro správu řazení na koleji je také vizualizační komponent, který vytváří zmenšený model koleje a rozmístění strojů na základě jejich délky a pořadí. Tato vizualizace pomáhá uživateli lépe pochopit fyzické rozložení strojů na trati a zároveň identifikovat případné problémy s nedostatečnou délkou koleje.

Z uživatelského pohledu je skript navržen tak, aby byl intuitivní, vizuálně přehledný a zároveň dostatečně robustní – detekuje například překročení dostupné délky koleje a varuje uživatele před uložením nevalidního stavu. Díky těmto vlastnostem se `normal_planing.js` stává klíčovým prvkem systému pro efektivní řízení železniční logistiky ve vlečkovém provozu.

3.8 Doporučení koleje pro odstavení

Klíčovou součástí aplikace *Vlečkostroj* je inteligentní plánování příjezdů a odjezdů strojů na konkrétní kolej ve vlečkovém areálu, o což se stará skript `smart_planning.js`. Aplikace poskytuje uživateli prostředí, ve kterém si může zvolit stroje, čas jejich příjezdu a odjezdu, možnost odstavení pouze na volné koleje či na halu nebo preferované vlečky a následně mu aplikace nabídne několik možností kolejí vhodných k odstavení vybraných strojů.

Toto chytré plánování je iniciováno funkcí `machineArrival()`, která otevře modální okno s formulářem, ve kterém uživatel:

- zadá časy příjezdu a odjezdu,
- zvolí, zda chce pouze prázdné koleje,
- určí, zda lze použít i speciální koleje (např. v hale),
- vybere stroje, jejich otočení a případně bezpečnostní mezeru.

Po potvrzení těchto vstupních dat se spustí funkce `parkMachines()`, která navrhne na základě zvolených parametrů vhodné koleje.

Ve funkci `parkMachines()` se z nasbíraných dat sestaví vlaková souprava a aplikace vyfiltruje pouze ty koleje, které splňují podmínky: mají dostatek místa, patří do vybraných vleček a podle preferencí uživatele jsou prázdné nebo „speciální“. Pro tyto vhodné koleje se aplikují tři optimalizační strategie:

- Dijkstrův algoritmus pro nalezení nejbližší možné kolej z pohledu vzdálenosti od vstupního bodu výhybky ke konci koleje.
- TOPSIS, který zohlední více parametrů současně a vybere kolej, která má nejlepší kompromis mezi vzdáleností, obsazeností a „hodnotou“ koleje.
- výběr koleje s nejmenším dostupnou délkou pro odstavení, která pojme soupravu.

3.8.1 Filtrování kolejí

Před samotným výpočtem nejlepší koleje pro odstavení vlakové soupravy je klíčovým krokem filtrování dostupných kolejí, které mají potenciál splnit základní technické a provozní požadavky. Tento proces je nezbytný pro to, aby byly do následné analýzy zahrnuty pouze ty koleje, které skutečně připadají v úvahu a zároveň, aby se výpočetní náročnost algoritmů, jako je Dijkstra nebo TOPSIS, omezila na relevantní podmnožinu kolejiště.

Filtrování kolejí probíhá v několika logických krocích a kombinuje požadavky zadané uživatelem s provozními charakteristikami jednotlivých kolejí.

Vstupní data jsou filtrovány podle čtyř oblastí:

1. **Vybrané vlečky** – Uživatel si v rozhraní aplikace zvolí jednu nebo více vleček, ve kterých chce stroje odstavit. To výrazně zužuje počet kolejí, které je potřeba posuzovat, a zároveň odpovídá reálnému provoznímu postupu, kdy je předem známa cílová oblast příjezdu.
2. **Požadavek na prázdné koleje** – Pokud uživatel zaškrtne možnost „Pouze prázdné koleje“, systém vyřadí všechny koleje, které jsou aktuálně alespoň částečně obsazené. Tento filtr je užitečný zejména tehdy, pokud dispečer nechce (nebo nemůže) provádět další posunování již zaparkovaných strojů.
3. **Povolení speciálních kolejí** – Některé koleje mohou být označeny jako „speciální“, například koleje vedoucí do haly nebo s provozním omezením. Pokud uživatel nezvolí možnost „Povolit speciální koleje“, tyto koleje jsou ze seznamu automaticky vyloučeny. V kódu je tento požadavek ošetřen pomocí atributu *allowed*.
4. **Dostupná délka koleje vs. délka soupravy** – Jednou z nejzásadnějších podmínek je fyzická vhodnost koleje, kdy kolej musí mít dostatek volné délky pro celou vlakovou soupravu, včetně mezer mezi stroji. Pokud kolejiště nedokáže požadovanou délku pojmout, daná kolej se automaticky vyřazuje z dalšího posuzování.

Filtrování je implementováno pomocí funkce `filter()` (obrázek 11), která prochází celou datovou sadu *railsData* (obsahující informace o všech kolejích v systému) a aplikuje na každou kolej zvolené podmínky.

organizace-vlecky - smart_planning.js

```
189  let filteredRailsData = railsData;
190  filteredRailsData = railsData.filter(rail =>
191      (emptyRailsOnly ? rail.occupied_length == 0 : true) &&
192      (specialRailAllowed ? rail.allowed != 0 : rail.allowed == 1) &&
193      selectedSpursId.some(id => id == rail.ID_spur) &&
194      rail.remaining_length >= lengthOfTrainSet);
```

Obrázek 11 Filtrování kolejí (autor)

Výsledkem je pole *filteredRailsData*, které obsahuje všechny technicky přijatelné a provozně vhodné koleje, jež budou dále hodnoceny různými metodami pro doporučení optimální volby.

Filtrování představuje kritický předstupeň optimalizačních algoritmů, protože výrazně zmenšuje počet variant, které je potřeba detailně analyzovat, zajišťuje, že každý další výpočet (např. nejkratší vzdálenost nebo TOPSIS) bude aplikován pouze na skutečně možné alternativy,

eliminuje riziko technicky neproveditelných rozhodnutí (např. přiřazení příliš krátké koleje) a umožňuje uživateli zachovat kontrolu nad výběrem podle jeho aktuální potřeby (např. z hlediska provozních omezení).

3.8.2 Dijkstrův algoritmus

Dijkstrův algoritmus představuje jeden ze základních a zároveň nejefektivnějších přístupů ke hledání nejkratší cesty v ohodnoceném grafu. V aplikaci *Vlečkostroj* je jeho úloha klíčová pro inteligentní výběr vhodné koleje pro příjezd vlakové soupravy. Konkrétně umožňuje určit, která z vhodných kolejí je fyzicky nejblíže vstupnímu bodu vlečky.

Dijkstrův algoritmus je algoritmus typu greedy [25], který iterativně rozšiřuje množinu navštívených uzlů tím, že vždy vybírá ten s nejmenší aktuální známou vzdáleností od počátečního uzlu. Výsledkem je nalezení optimální trasy (nejkratší cesty) z jednoho výchozího bodu do všech ostatních uzlů grafu.

V aplikaci je každý bod koleje reprezentován jako uzel v grafu a každý úsek mezi dvěma sousedními body jako hrana ohodnocená délkou úseku.

Nejprve se ze souřadnic všech kolejí vytvoří ohodnocený neorientovaný graf (obrázek 12), kde každá dvojice sousedních bodů na koleji tvoří hranu a váha každé hrany je rovna geometrické vzdálenosti dvou bodů.

organizace-vlecky - smart_planing.js

```
439 function buildGraph(rails) {
440     const graph = {};
441     rails.forEach(rail => {
442         const coords = parseCoordinates(rail.rail_coordinates);
443         for (let i = 0; i < coords.length - 1; i++) {
444             const a = JSON.stringify(coords[i]);
445             const b = JSON.stringify(coords[i + 1]);
446             const distance = calculateDistance(coords[i], coords[i + 1]);
447 
448             if (!graph[a]) graph[a] = {};
449             if (!graph[b]) graph[b] = {};
450 
451             graph[a][b] = distance;
452             graph[b][a] = distance; // bidirectional
453         }
454     });
455     return graph;
456 }
```

Obrázek 12 Vytvoření grafu (autor)

Poté je pro každou potenciálně vhodnou kolej určen počáteční bod (souřadnice vstupu vlečky) a cílový bod (konec koleje). Následně proběhne Dijkstrův algoritmus [26], který vypočítá nejkratší cestu a její délku (obrázek 13).

organizace-vlecky - smart_planing.js

```
459 function dijkstra(graph, startCoord, endCoord) {
460     const start = JSON.stringify(startCoord);
461     const end = JSON.stringify(endCoord);
462 
463     const distances = {};
464     const previous = {};
465     const visited = new Set();
466     const pq = new Map();
467 
468     Object.keys(graph).forEach(node => {
469         distances[node] = Infinity;
470         pq.set(node, Infinity);
471     });
472 
473     distances[start] = 0;
474     pq.set(start, 0);
475 
476     while (pq.size > 0) {
477         // Get the node with the smallest distance
478         const current = [...pq.entries()].reduce((a, b) => a[1] < b[1] ? a : b)[0];
479         pq.delete(current);
480 
481         if (current === end) break; // Found the shortest path
482         if (!graph[current]) continue;
483 
484         visited.add(current);
485 
486         for (const neighbor in graph[current]) {
487             if (visited.has(neighbor)) continue;
488 
489             const alt = distances[current] + graph[current][neighbor];
490             if (alt < distances[neighbor]) {
491                 distances[neighbor] = alt;
492                 previous[neighbor] = current;
493                 pq.set(neighbor, alt);
494             }
495         }
496     }
497     // Build path
498     let path = [];
499     let node = end;
500     while (node) {
501         path.unshift(JSON.parse(node));
502         node = previous[node];
503     }
504 
505     return {distance: distances[end], path};
506 }
```

Obrázek 13 Dijkstrův algoritmus (autor)

Po nalezení cesty do cílového uzlu je možné zrekonstruovat samotnou trasu (*path*) pomocí pomocného pole *previous*, ale pro účely *Vlečkostroje* je relevantní především celková vzdálenost, kterou algoritmus spočítá.

Všechny koleje, které projdou předchozími filtry (dostupnost, délka, typ atd.), jsou vyhodnoceny pomocí Dijkstry. Výsledky jsou pak uloženy do pole výsledků (*results*). Toto pole je následně seřazeno podle vzrůstající vzdálenosti a kolej s nejnižší hodnotou je zvolena jako první doporučená kolej a uložena do pole nejlepších kolejí (*bestRails[0]*). Celý proces je zobrazen na obrázku 14.

organizace-vlecky - smart_planning.js

```
196     // Option 1 - Use nearest rail that will fit the train set
197 let start = null;
198 let end = null;
199 const result = [];
200 // Prepare graph for Dijkstra's algorithm
201 const graph = buildGraph(railsData);
202 filteredRailsData.forEach(rail => {
203     // Parse coordinates to correct format without spaces
204     start = parseCoordinates(spursData[rail.ID_spur-1].entry_node)[0];
205     end = parseCoordinates(rail.rail_coordinates)[0];
206     // Calculate distance using Dijkstra's algorithm
207     const pathResult = dijkstra(graph, start, end);
208     // Push result and rail to array
209     result.push({
210         rail: rail,
211         distance: pathResult.distance,
212         path: pathResult.path,
213     });
214 });
215 // Sort results by distance
216 result.sort((a, b) => a.distance - b.distance);
217 // Get the best rail from the result
218 bestRails[0] = result[0].rail;
219 showOption(bestRails[0], data, 1);
```

Obrázek 14 Nalezení nejbližší vhodné koleje (autor)

Implementace Dijkstrova algoritmu v systému *Vlečkostroj* představuje nástroj pro zajištění efektivního a racionálního využívání železniční infrastruktury v rámci vlečkového areálu. Namísto spoléhání na ruční rozhodování nebo statické preference umožňuje tento algoritmus dynamicky analyzovat prostorové uspořádání kolejiště a určit nejvhodnější kolej čistě na základě skutečné vzdálenosti mezi vstupem a cílovým bodem. Tím přináší vyšší provozní efektivitu, snižuje potřebu nadbytečného posunování strojů a zároveň podporuje rozhodování založené na datech.

3.8.3 TOPSIS – Vícekriteriální rozhodování

TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) je jednou z nejrozšířenějších metod vícekriteriálního rozhodování [27]. Její klíčová výhoda spočívá ve schopnosti zohlednit několik vzájemně si konkurujících kritérií a vybrat tu alternativu (v tomto případě kolej), která má nejlepší kompromisní charakteristiky ve vztahu k ideálnímu a nejméně výhodnému řešení. V rámci systému *Vlečkostroj* slouží metoda TOPSIS k inteligentnímu doporučení koleje pro odstavení vlakové soupravy na základě vícekriteriální analýzy.

Zatímco Dijkstrův algoritmus optimalizuje volbu koleje čistě z pohledu geografické vzdálenosti, realita provozu často vyžaduje komplexnější rozhodování. Například může být kolej sice blízko, ale částečně obsazená nebo má nízkou prioritu. Navíc dlouhé koleje mohou být nevhodné pro krátké soupravy kvůli neefektivnímu využití prostoru. TOPSIS tak umožňuje zohlednit i technické a provozní aspekty, které Dijkstrův algoritmus nebere v úvahu.

V systému *Vlečkostroj* jsou při výběru koleje zohledňována tři hlavní kritéria:

1. Vzdálenost podle Dijkstry – čím kratší, tím lepší (negativní kritérium).
2. Efektivita využití koleje – vyjádřená jako poměr zbývající délky po odstavení soupravy k celkové délce koleje (nižší hodnota značí efektivnější využití).
3. Hodnocení koleje (*rail_weight*) – interní priorita nebo důležitost (čím vyšší, tím lepší).

Každé kritérium má vlastní váhu, přičemž součet vah je roven hodnotě 1. Tyto váhy vyjadřují relativní důležitost jednotlivých kritérií. Po odborném odhadu byly váhy kritérií stanoveny na hodnoty 0.4 pro vzdálenost podle Dijkstry, 0.3 pro efektivitu využití koleje a 0.3 pro ohodnocení koleje.

Pro každou kolej, která splňuje základní technické podmínky se vytvoří vektor tří hodnot, který tvoří jeden řádek v rozhodovací matici (obrázek 15Obrázek 15).

organizace-vlecky - smart_planing.js

```
238 filteredRailsData.forEach(rail => {
239     const dijkstraDistance = result.find(r => r.rail.ID_rail == rail.ID_rail);
240     if (dijkstraDistance) { // Skip if rail not in result
241         decisionMatrix.push([
242             dijkstraDistance.distance,
243             (rail.remaining_length - lengthOfTrainSet) / rail.rail_length,
244             rail.rail_weight
245         ]);
246         tempOrder.push(rail);
247     }
248 });
```

Obrázek 15 Rozhodovací matice (autor)

Každá hodnota v matici se přepočítá pomocí Eukleidovské normalizace dle vzorce (1) tak, aby byl každý sloupec (kritérium) v jednotném měřítku (obrázek 16).

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{1}$$

kde:

r_{ij}…normalizovaná hodnota kritéria

x_{ij}…hodnota kritéria

organizace-vlecky - smart_planing.js

```
384 // Step 1: Normalize the decision matrix
385 const normalizedMatrix = Array.from({ length: numAlternatives }, () => new Array(numCriteria));
386 for (let j = 0; j < numCriteria; j++) {
387     let sumSquares = 0;
388     for (let i = 0; i < numAlternatives; i++) {
389     sumSquares += decisionMatrix[i][j] ** 2;
390     }
391     const norm = Math.sqrt(sumSquares);
392     for (let i = 0; i < numAlternatives; i++) {
393     normalizedMatrix[i][j] = decisionMatrix[i][j] / norm;
394     }
395 }
```

Obrázek 16 Normalizace kritérií (autor)

Každý normalizovaný prvek je vynásoben váhou příslušného kritéria (obrázek 17).

organizace-vlecky - smart_planing.js

```
397 // Step 2: Multiply the normalized matrix by weights
398 const weightedMatrix = normalizedMatrix.map(row =>
399     row.map((value, j) => value * weights[j])
400 );
```

Obrázek 17 Násobení normalizované rozhodovací matice maticí kritérií (autor)

Pro každé kritérium se určí nejlepší (ideální) hodnota (minimum pro negativní kritéria, maximum pro pozitivní) a nejhorší hodnota (maximum pro negativní kritéria, minimum pro pozitivní) viz obrázek 18.

organizace-vlecky - smart_planing.js

```
402  // Step 3: Determine the ideal best and ideal worst for each criterion
403  const idealBest = new Array(numCriteria);
404  const idealWorst = new Array(numCriteria);
405  for (let j = 0; j < numCriteria; j++) {
406      const columnValues = weightedMatrix.map(row => row[j]);
407      if (isBenefit[j]) {
408      idealBest[j] = Math.max(...columnValues);
409      idealWorst[j] = Math.min(...columnValues);
410      } else {
411      idealBest[j] = Math.min(...columnValues);
412      idealWorst[j] = Math.max(...columnValues);
413      }
414  }
```

Obrázek 18 Výpočet nejlepší a nejhorší varianty (autor)

Následně se pro každou alternativu vypočítá vzdálenost k nejlepší S_i^+ a nejhorší S_i^- hodnotě podle vzorců (2) a (3) (obrázek 19).

$$S_i^+ = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^+\right)^2} \tag{2}$$

$$S_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2} \tag{3}$$

kde:

S_i^+…vzdálenost k nejlepší variantě

S_i^-… vzdálenost k nejhorší variantě

v_{ij}…alternativa *ij*

v_j^+…nejlepší varianta

v_j^-…nejhorší varianta

organizace-vlecky - smart_planing.js

```
416  // Step 4: Calculate the Euclidean distance from each alternative to the ideal best and worst
417  const distancesToBest = new Array(numAlternatives).fill(0);
418  const distancesToWorst = new Array(numAlternatives).fill(0);
419  for (let i = 0; i < numAlternatives; i++) {
420      let sumBest = 0;
421      let sumWorst = 0;
422      for (let j = 0; j < numCriteria; j++) {
423      sumBest += (weightedMatrix[i][j] - idealBest[j]) ** 2;
424      sumWorst += (weightedMatrix[i][j] - idealWorst[j]) ** 2;
425      }
426      distancesToBest[i] = Math.sqrt(sumBest);
427      distancesToWorst[i] = Math.sqrt(sumWorst);
428  }
```

Obrázek 19 Výpočet nejlepší a nejhorší hodnoty (autor)

Nakonec se pro každou variantu spočítá skóre podle vzorce (4) (obrázek 20).

$$C_i^* = \frac{S_i^-}{S_i^- + S_i^+} \tag{4}$$

kde:

C_i^*…hodnocení varianty

S_i^+…vzdálenost k nejlepší variantě

S_i^-… vzdálenost k nejhorší variantě

organizace-vlecky - smart_planning.js

```
435  // Step 5: Calculate the relative closeness to the ideal solution
436  const scores = distancesToBest.map((distBest, i) => {
437      return {score: distancesToWorst[i] / (distBest + distancesToWorst[i]),
     rail: tempOrder[i]};
438  });
```

Obrázek 20 Hodnocení variant (autor)

Hodnota C_i^* může nabývat hodnot od 0 do 1. Čím je hodnota C_i^* blíže 1, tím je kolej „ideálnější“.

Po výpočtu skóre pro všechny koleje je pole *scores* setříděno sestupně podle hodnoty C_i^* a kolej s nejvyšším skóre je pak nabídnuta jako optimální varianta podle TOPSIS.

Zavedení metody TOPSIS do rozhodovací logiky aplikace *Vlečkostroj* umožňuje přehledné a systémové hodnocení možných variant kolejí z více hledisek najednou. Díky tomu dochází k objektivnímu vyhodnocení alternativ, které nejsou založeny pouze na jedné dominantní charakteristice (například vzdálenosti), ale berou v potaz i efektivitu využití a strategickou důležitost jednotlivých kolejí. Tato metoda by umožňovala dispečerovi rychle se zorientovat ve složité situaci a přijmout rozhodnutí, které je nejen prakticky proveditelné, ale také optimalizované z hlediska provozního řízení. TOPSIS tak v kontextu železničního plánování přináší důležitý prvek prediktivní inteligence a adaptivního řízení.

3.8.4 Nejkratší vhodná kolej

Pro rozšíření možností doporučování kolejí je v aplikaci *Vlečkostroj* navržena metoda pro nalezení vhodné koleje o nejkratší volné délce, která pojme soupravu. Výběr probíhá mezi kolejemi, které spují základní technická kritéria – především musí být dostatečně dlouhé pro celou soupravu a musí náležet k uživatelem vybraným vlečkám. Pokud uživatel aktivuje volbu „pouze prázdné koleje“, jsou navíc zohledňovány jen ty, které nejsou obsazené jinými stroji.

Podobně volba „povolit stání na speciálních kolejích“ určuje, zda se mají do výběru zahrnout i technicky omezené nebo jinak specifické koleje (např. uvnitř haly).

Jakmile je aplikován tento filtr, metoda vybere z výsledné množiny nejkratší kolej, která ještě soupravu pojme. Cílem je zabránit neefektivnímu využití prostoru, tj. krátké soupravy by neměly zbytečně blokovat dlouhé koleje, které by později mohly být potřeba pro větší vlaky. Z tohoto pohledu je tato metoda pragmatickým řešením, které zohledňuje základní provozní logiku železničního dispečinku.

Implementačně je tato metoda velmi přímočará. Ze všech vhodných kolejí se jednoduše vybere ta, jejíž dostupná délka (tedy délka zbývající po odstavení již přítomných strojů) je co nejmenší, ale stále dostačující. V kódu aplikace je tato logika vyjádřena jednoduchým porovnáním hodnot na obrázku 21.

organizace-vlecky - smart_planning.js

```
259     // Option 3 - Use shortest rail that will fit the train set
260 filteredRailsData.forEach(rail => {
261     // Select only rails from selected spurs
262     if (bestRails[2] === null || bestRails[2].remaining_length >
    rail.remaining_length) {
263         bestRails[2] = rail;
264     }
265 });
```

Obrázek 21 Výběr nejkratší přípustné koleje (autor)

Tato přístup je především nástrojem pro rychlé rozhodování v situacích, kdy není potřeba složité porovnávání více hledisek, nebo když dispečer požaduje srozumitelné a okamžité doporučení.

3.8.5 Vlastní výběr koleje

Pro doplnění automatických metod výběru koleje v případě, že uživateli ani jedna metoda nevyhovuje, je ve skriptu `smart_planing.js` představena čtvrtá metoda. Tato metoda výběru nejlepší koleje umožňuje uživateli ručně vybrat libovolnou vhodnou kolej ze všech dostupných možností.

Po výběru této možnosti se nejprve vygeneruje uživatelské rozhraní s příslušnými informacemi o délce soupravy. Zároveň se zobrazí tlačítko „Vybrat ručně“, které po kliknutí spustí asynchronní funkci `pickRail(lengthOfTrainSet)`. Tato funkce otevře nové modální okno s mapou, na které jsou vykresleny všechny koleje dostupné dle zvolené vlečky. V případě výběru více vleček je uživatel upozorněn, aby vybral pouze jednu vlečku. Každá kolej je

barevně zvýrazněna podle toho, zda má dostatečnou zbývající délku pro plánovanou soupravu. Uživatel může libovolně kliknout na koleje, čímž si jednu z nich vybere. Po potvrzení výběru se ID této koleje vrátí zpět volající funkci.

Jakmile uživatel vybere kolej, je tato kolej zobrazena jako nová volba v rozhraní spolu s dalšími dostupnými možnostmi. Následně se pomocí funkce `showOption()` vizualizuje konkrétní rozložení strojů na mapě, včetně strojů již umístěných na koleji i těch, které mají nově přijet. Součástí této vizualizace je i interaktivní tlačítko „Vybrat“, kterým může uživatel výběr potvrdit. Po potvrzení se vytvoří strukturovaný datový objekt s informacemi o ID koleje, ID strojů, mezerách před stroji, orientaci, čase příjezdu a případném čase odjezdu. Tyto údaje jsou následně odeslány na server prostřednictvím `update_arrival.php`.

Tato čtvrtá metoda je důležitá zejména z hlediska flexibility plánování, protože umožňuje lidský zásah do jinak automatizovaného rozhodovacího procesu. Zatímco ostatní metody se snaží optimalizovat výběr koleje na základě algoritmických kritérií, manuální výběr nabízí uživateli plnou kontrolu a vizuální oporu v mapovém zobrazení kolejí a jejich aktuální obsazenosti. Díky tomu je metoda vhodná v případech, kdy automatické algoritmy neposkytují optimální výsledek nebo je třeba zohlednit specifické provozní požadavky, které nelze snadno algoritmizovat.

3.8.6 Vizualizace doporučených kolejí

Jakmile systém *Vlečkostroj* na základě zadaných parametrů a vstupních údajů dokončí výpočet vhodných kolejí, následuje fáze vizualizace výsledků. Cílem této části je intuitivně a přehledně zobrazit uživateli jednotlivé doporučené možnosti, včetně všech důležitých informací, které mu pomohou učinit informované rozhodnutí o výběru konkrétní koleje.

Celý proces začíná v rámci funkce `parkMachines()`, která po určení tzv. *bestRails* – tedy kolejí vybraných různými metodami (Dijkstra, TOPSIS, nejkratší vhodná kolej, případně uživatelem vybraná kolej) volá funkci `showOption()` pro každou z těchto kolejí. Tato funkce je zodpovědná za kompletní zpracování zobrazení jedné konkrétní varianty.

Funkce `showOption()` využívá komponenty knihovny Leaflet.js pro zobrazení mapy a její interaktivní manipulaci. Nejprve je připraven datový model, který kombinuje stávající stroje, které už na koleji stojí, nově plánované stroje (včetně jejich délky, pozice a orientace) a ostatní stroje na jiných kolejích v rámci vlečky.

Z těchto dat je sestaven ucelený seznam objektů, které mají být na mapě zobrazeny. Pro každou doporučenou kolej se vytvoří nová mapa (s unikátním ID), do které jsou následně

vykresleny samotné koleje (barevně odlišeny, červeně zvýrazněna doporučená kolej) a pozice jednotlivých strojů.

Mapový výřez je automaticky nastaven tak, aby uživatel viděl celé kolejiště vlečky, ve kterém se daná kolej nachází. Díky tomu lze na první pohled posoudit doporučené rozmístění strojů. Všechny výřezy jsou následně vykresleny v modálním okně, kde má uživatel přehled všech nabízených možností. Vzor zobrazení je na obrázku 25, kde jsou možnosti odstavení stroje ASP 08-275/3S UNIMAT.

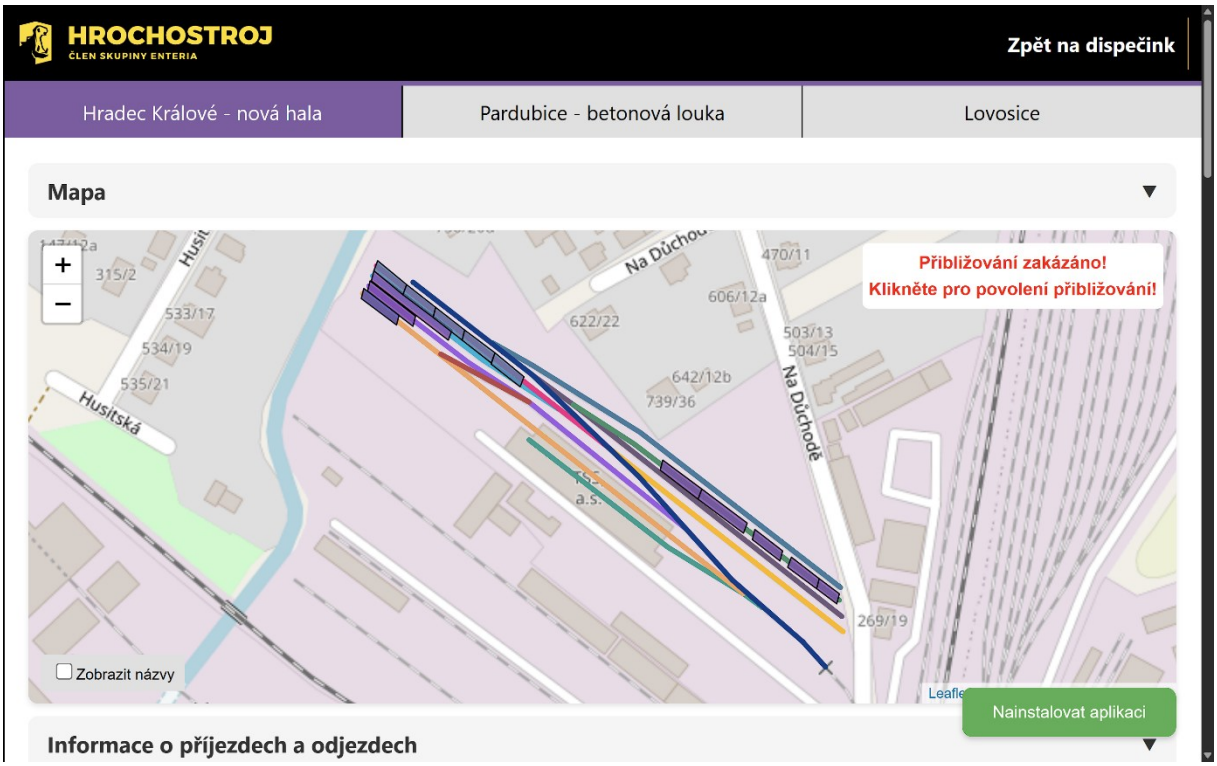
Každá vizualizovaná možnost je vybavena tlačítkem „Vybrat“, kterým uživatel potvrzuje svůj výběr. Po jeho stisknutí se připraví strukturovaná datová struktura, obsahující potřebná data (ID vybrané koleje, seznam ID všech strojů, bezpečnostní mezery, orientace strojů a časy příjezdu a odjezdu). Tato data jsou následně odeslána na server prostřednictvím volání `fetch(update_arrival.php)`. Pokud operace proběhne úspěšně, zobrazí se potvrzení a stránka se obnoví s novým stavem kolejiště.

3.9 Hlavní stránka Vlečkostroj

Hlavní stránka aplikace *Vlečkostroj* představuje komplexní rozhraní pro správu železničních kolejí a manipulaci se stroji ve vlečkovém provozu.

Stránka je definována v souboru `index.html` a její jádro je postaveno na JavaScriptu spolu s Leafletem pro mapové zobrazení. Při načtení stránky se spustí soubor `fetch_data.js`, který získá data z databáze a připraví je pro vykreslení na stránce a načte data z *local storage*, kde jsou uložené uživatelem nastavené vlastnosti pro přizpůsobení stránky, které může mít každý uživatel specificky nastavené.

Hlavní stránka aplikace *Vlečkostroj* představuje uživatelské rozhraní pro plánování, správu a vizualizaci kolejového provozu v rámci vlečkového systému. Po otevření stránky je zahájeno načítání všech potřebných dat ze serveru, konkrétně seznamů strojů, kolejí, vleček, jejich otočení a plánovaných příjezdů či odjezdů. Tato data jsou zpracována a následně použita k vykreslení mapy, naplnění seznamů a dynamickému generování interaktivního obsahu. Přiblížené zobrazení hlavní stránky je zobrazené na obrázku 22 a standartní zobrazení je v příloze B.

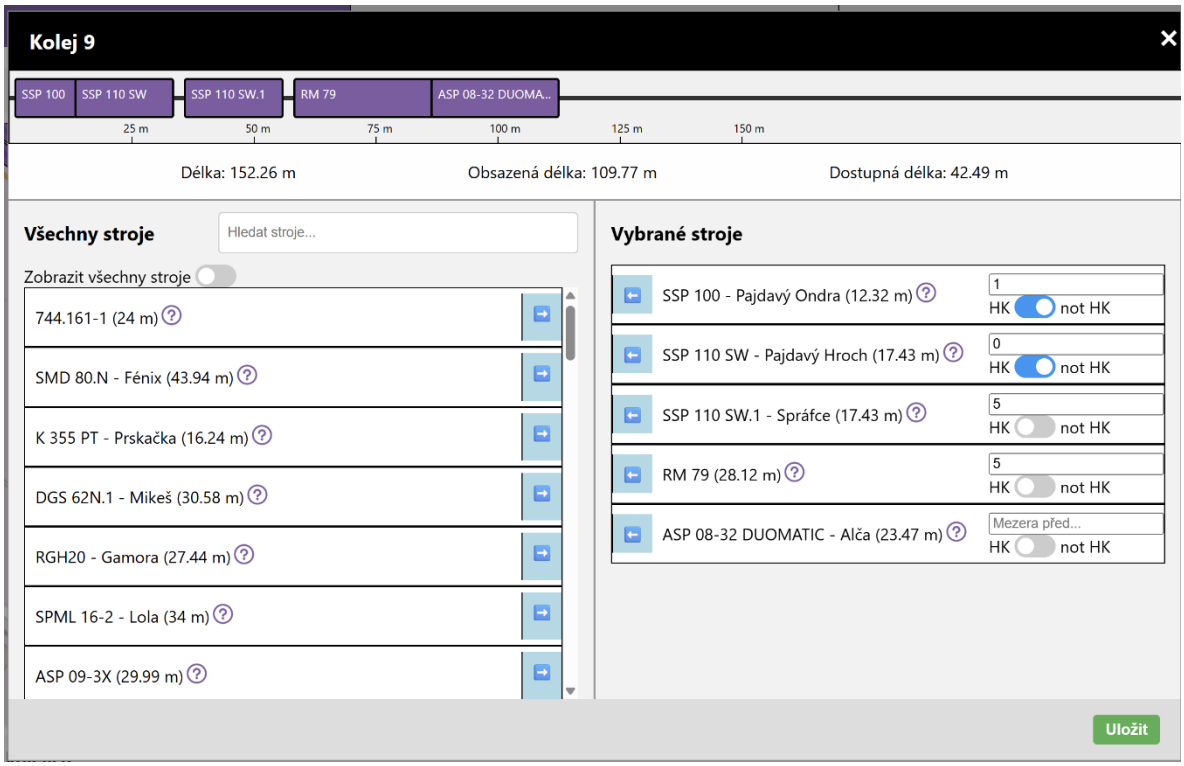


Obrázek 22 Hlavní stránka (autor)

Základní část rozhraní tvoří interaktivní mapa, která využívá knihovnu Leaflet. Na ní jsou znázorněny jednotlivé koleje jako barevné křivky s popisky a stroje. Uživateli je umožněno přiblížení, zobrazení detailních údajů o strojích a interakce s kolejemi, například pro otevření rozhraní pro manipulaci se stroji.

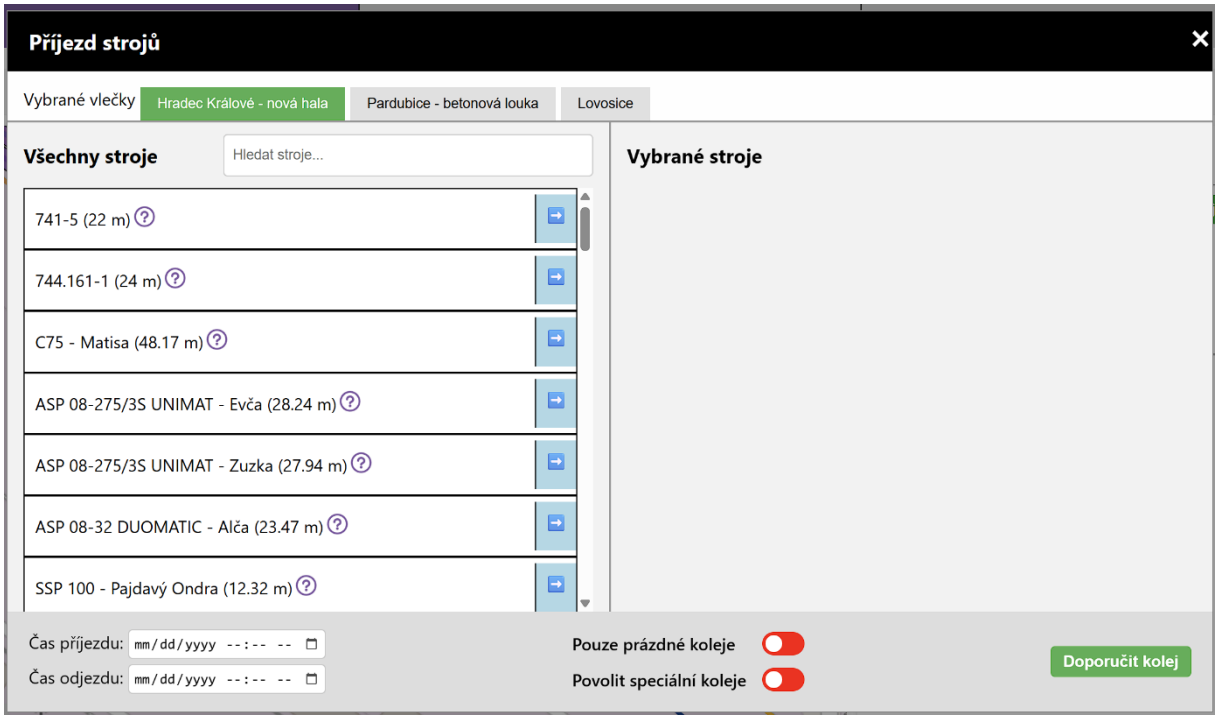
Vedle mapy se nachází sekce s informacemi o plánovaných příjezdech a odjezdech strojů. Každý záznam obsahuje seznam strojů, cílovou kolej, čas příjezdu a plánovaný odjezd. Uživatel má možnost aktivovat akci příjezdu či odjezdu, případně danou akci zrušit. Tato interakce automaticky aktualizuje stav v databázi prostřednictvím dalších skriptů v souboru `smart_planning.js`.

Další klíčovou sekcí je přehled kolejí. U každé koleje je uvedena její délka, obsazená a dostupná kapacita a seznam strojů, které se na ní aktuálně nachází. Kliknutím na tlačítko u konkrétní koleje se otevře modální okno, zobrazené na obrázku 23, kde uživatel může interaktivně přiřazovat stroje na kolej. Výběr se provádí zvolením stojů v seznamu dostupných strojů, přičemž je možné nastavovat směr, mezery před stroji a pořadí. Okno obsahuje také vizualizaci koleje a přehledné měřítko. Rozložení celé stránky při zobrazení řazení na koleji je v příloze C.



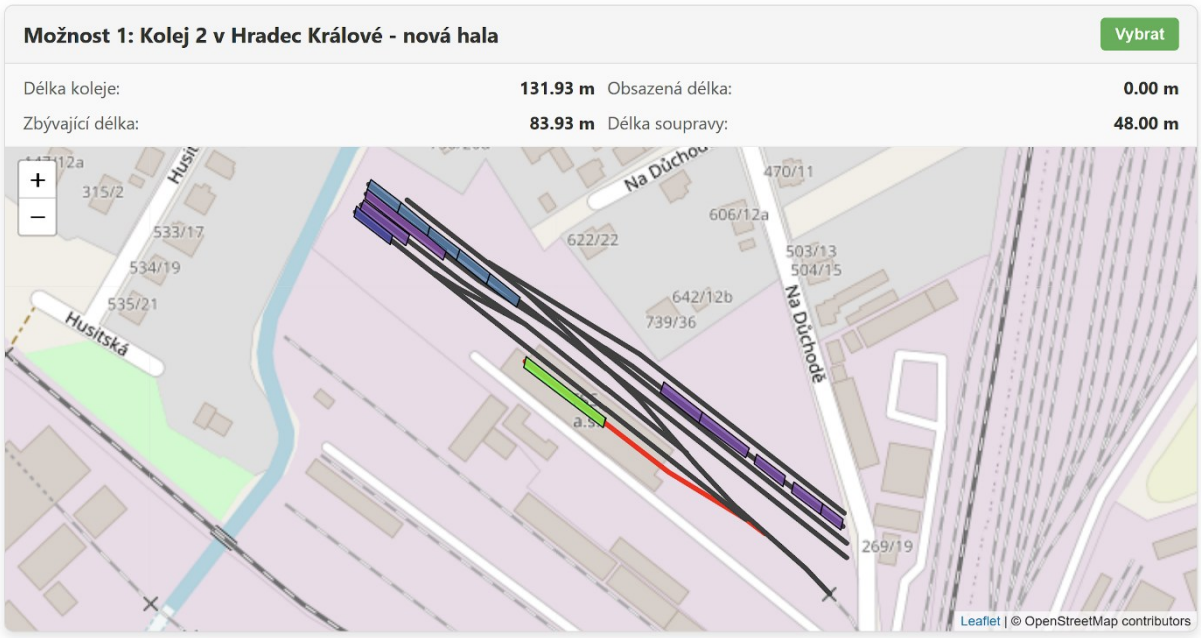
Obrázek 23 Modální okno pro řazení na koleji (autor)

Velmi významným prvkem je funkcionalita inteligentního plánování příjezdu. Po kliknutí na tlačítko „Příjezd strojů“ se otevře modální okno, zobrazené na obrázku 24, kde uživatel může vybrat stroje a stanovit jejich parametry s daty odjezdu a příjezdu. Celé rozložení je v příloze D.



Obrázek 24 Modální okno pro plánování příjezdů a odjezdů (autor)

Po kliknutí na tlačítko „Doporučit kolej“ se otevře doporučení vhodné koleje na základě různých optimalizačních strategií. Jedno z možných zobrazení je na obrázku 25 a celé rozložení stránky a všech možností je v příloze E. Tyto strategie jsou popsány v kapitole 3.8.



Obrázek 25 Možnosti odstavení strojů (autor)

Aplikace je navíc postavena jako PWA, což znamená, že podporuje offline režim. Pomocí Service Workeru je možné uchovávat často používané soubory a data v cache. V případě výpadku internetu aplikace nadále funguje a informuje uživatele o používání uložených dat, která nemusí být aktuální.

Celé uživatelské rozhraní je responsivní a poskytuje velmi přehledné prostředí pro dispečery a správce vlečkových areálů. Spojuje mapovou vizualizaci, plánování příjezdů a efektivní správu kolejí a vozidel do jediné centrální aplikace, která výrazně zjednodušuje každodenní operativu v rámci železniční dopravy.

4 ZHODNOCENÍ APLIKACE VLEČKOSTROJ

Zavedení aplikace *Vlečkostroj* představuje posun v oblasti správy železničních vleček. Tato kapitola se zabývá vyhodnocením přínosů z hlediska snížení personální náročnosti, efektivity provozu a možností dalšího rozvoje s komerčním potenciálem.

4.1 Úspora lidských zdrojů

V současnosti je organizace odstavování železničních strojů na vlečce společnosti Hrochostroj a.s. řízena dispečerem a zaměstnanci pověřenými správou vleček. Tito pracovníci jsou zodpovědní za plánování příjezdů a odjezdů, výběr vhodných kolejí, evidenci strojů a řešení provozních kolizí. Tyto činnosti mohou být časově náročné a vyžadují nutnost plánování a komunikace mezi zaměstnanci.

Aplikace *Vlečkostroj* tyto procesy výrazně automatizuje. Využívá algoritmy pro výběr koleje (Dijkstrův algoritmus, TOPSIS), vizualizuje obsazenost kolejí a umožňuje přímé přidělení kolejového prostoru jednotlivým strojům. Dispečer se tak může více soustředit na strategické plánování a operativní rozhodování, zatímco běžná evidence a rozřazování probíhá automaticky.

To se projeví v úspoře pracovního času, snížení rizika chyb způsobených lidským faktorem a menší potřebě zapojení více pracovníků do každodenního provozu. V podmínkách firmy, kde vozový park čítá omezený počet strojů, může aplikace zcela nahradit manuální evidenci a výrazně zjednodušit pracovní postupy.

4.2 Zvýšení provozní efektivity

Jedním z hlavních přínosů systému je optimalizace využití kolejové infrastruktury na vlečkách. Automatické výpočty potřebné délky soupravy, volného prostoru a vzdálenosti kolejí od vstupního bodu umožňují přesné a racionální plánování. Systém zamezuje odstavení strojů na nevhodné nebo příliš dlouhé koleje, čímž šetří prostor pro jiné vlaky a zvyšuje kapacitní propustnost vlečky.

Navíc umožňuje plánování příjezdů a odjezdů v předstihu včetně jejich grafického znázornění, čímž minimalizuje riziko kolizí a neefektivního odstavování strojů. Funkce vizualizace vlakové soupravy na miniatuře koleje poskytuje uživatelům okamžitý přehled o využití délky a orientaci strojů.

4.3 Možnost komerčního využití

Vzhledem k tomu, že celková kapacita železničních vleček provozovaných společností Hrochostroj a.s. přesahuje aktuální interní potřebu, nabízí se prostor pro komerční využití volných kolejí. Aplikace umožňuje správu více vleček současně a je tedy připravena i na provozování vleček pro komerční účely.

Volné koleje mohou být nabídnuty jiným železničním dopravcům, například pro:

- krátkodobé odstavení cizích strojů (při čekání mezi stavbami nebo zakázkami),
- dlouhodobé parkování (sezónní mimo provoz),
- technické odstavení (před nebo po servisu).

Díky možnosti plánovat příjezdy cizích strojů a jejich správu v databázi lze systém využít jako komerční nástroj pro správu odstavení. Výhodou je plná kontrola nad kapacitou, transparentní evidence a možnost jednoduchého reportingu.

Tím vzniká nový podnikatelský model „odstavovací služba“ která může být poskytována na základě smlouvy a cenového sazebníku. Systém v aktuální podobě není připraven pro tuto možnost, ale vzhledem k požadavkům vedení společnosti Hrochostroj a.s., je plánované v budoucnu tuto funkcionalitu přidat.

4.4 Návratnost investice a dlouhodobé přínosy

V krátkodobém horizontu přináší aplikace úsporu na personálním zajištění provozu vleček a snížení provozních nákladů spojených s chybným plánováním, nadbytečnými posuny nebo nedostatečným využitím kapacity. Dlouhodobě však může přinést dodatečné příjmy z komerčního využití a vytvořit datovou základnu pro další rozvoj logistiky.

Mezi další dlouhodobé přínosy patří lepší využití infrastruktury a jednodušší školení nových dispečerů díky standardizovanému rozhraní. Vzhledem k tomu, že aplikace běží v prostředí webového prohlížeče, je přístupná odkudkoliv a podporuje i offline režim. To zajišťuje vysokou dostupnost a flexibilitu provozu i v terénních podmínkách.

4.5 Zapojení dalších uživatelů

Aplikace *Vlečkostroj* je navržena nejen jako nástroj pro dispečerské řízení provozu, ale také jako informační a komunikační platforma pro další zaměstnance podílející se na provozu železniční vlečky. Díky své webové architektuře a přístupnosti z běžného internetového prohlížeče je vhodná i pro širší okruh uživatelů, včetně vedení společnosti, zaměstnanců správního oddělení nebo strojvedoucích.

Jedním z významných přínosů systému je možnost získávat informace o aktuálním rozmístění strojů na jednotlivých kolejích vlečky. Pracovníci, kteří nejsou přímo zapojeni do dispečerské činnosti, tak mohou snadno zjistit, které koleje jsou obsazeny a jakými stroji, jaké jsou plánované příjezdy a odjezdy, kde se nachází konkrétní stroj nebo jaká je aktuální dostupnost volných míst. Tato funkcionalita přispívá k vyšší transparentnosti provozu a usnadňuje každodenní rozhodování na provozní úrovni.

Pro strojvedoucí a obsluhu strojů může aplikace sloužit jako navigační nástroj, který jim sděluje, kam přesně mají daný stroj odstavit. To snižuje potřebu operativního spojení s dispečerem a urychluje se proces odstavování.

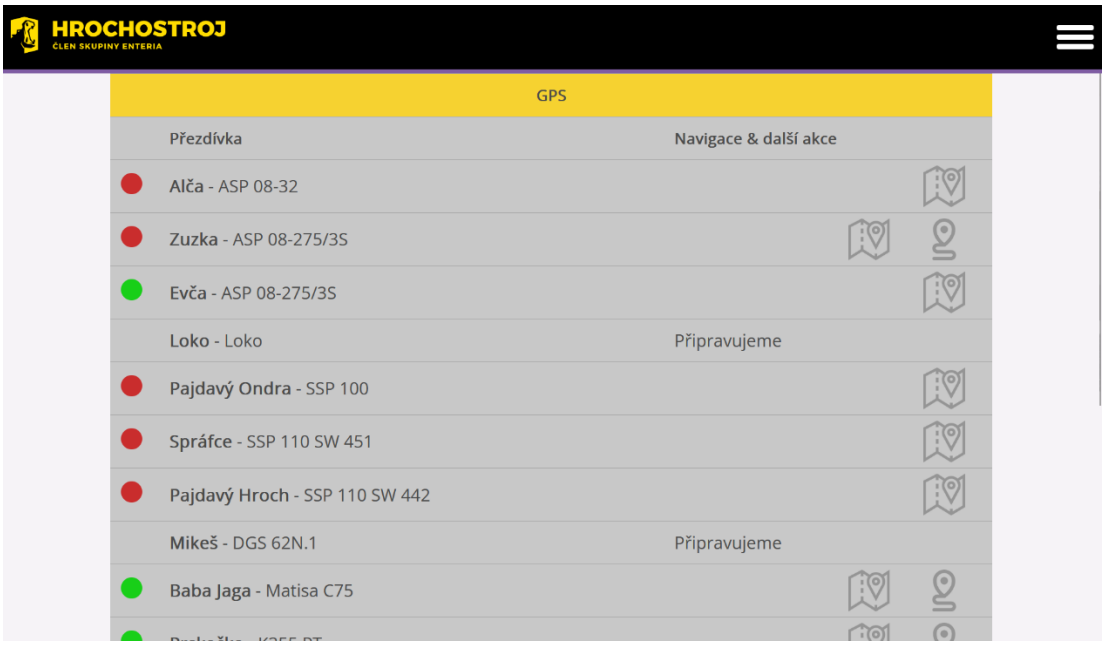
Dalším možností k zjednodušení procesu správy odstavování může být udělení oprávnění strojníkům a vlakvedoucím, kteří by po odstavení stroje v aplikaci potvrdili, že došlo k odstavení, případně zadali přesné parametry jako pozici, orientaci a mezery před vozem. Tím by se ještě více snížila operativní zátěž dispečera a celý proces by se stal samostatnějším a efektivnějším.

Tato možnost může být v budoucnu rozšířena i o autorizaci přístupů, tedy že různí uživatelé by měli rozdílná práva (např. pouze prohlížení, potvrzení odstavení nebo plnou editaci), čímž lze řízení provozu přizpůsobit organizační struktuře společnosti.

Rozšíření aplikace mezi více pracovníků podporuje nejen efektivitu, ale také zvyšuje informovanost mezi jednotlivými články provozu. Všichni uživatelé mají přístup ke stejným informacím, čímž se eliminuje riziko nedorozumění nebo zpoždění způsobené předáváním informací ústně či telefonicky.

4.6 Rozšíření aktuálního systému

Ve společnosti Hrochostroj a.s. je v provozu interní systém pro sledování polohy železničních strojů založený na GPS lokalizaci prostřednictvím lokalizačních boxů instalovaných přímo na jednotlivých strojích. Tento systém je dostupný přes vnitropodnikovou webovou aplikaci *HrochoTrek*, která uživatelům umožňuje zobrazit výčet všech aktivních strojů spolu s jejich aktuální geografickou polohou na mapovém podkladu. Přesnost lokalizace je přibližně 8 metrů, což je plně dostačující pro provoz strojů na výlukách v otevřeném terénu. Vzor vzhledu *HrochoTrek* je na obrázku 26 a plné zobrazení je v příloze F.



Obrázek 26 *HrochoTrek* (interní dokument)

Tato úroveň přesnosti však není dostačující pro detailní lokalizaci v prostředí železniční vlečky, kde často dochází k odstavení více strojů na koleje vzdálené jen několik metrů od sebe, jak je tomu na obrázku 27 a v příloze G. V těchto případech dochází k překrývání poloh na mapě, chybné interpretaci umístění stroje a nemožnosti přesného přiřazení ke konkrétní koleji. Zejména při manipulaci, plánování údržby nebo řízení provozu může taková nepřesnost představovat provozní komplikaci.

Obrázek 27 Umístění stroje v *HrochoTrek* (interní dokument)

Aplikace *Vlečkostroj* proto přichází jako nadstavba a doplněk ke stávajícímu systému. Namísto sledování stroje v otevřeném prostoru se zaměřuje na přesné řízení a evidenci umístění strojů na konkrétních kolejích vlečky. Zatímco GPS systém zajišťuje orientační přehled o stavu strojů v terénu, *Vlečkostroj* poskytuje operační přesnost na úrovni jednotlivých metrů a pořadí strojů na koleji, včetně bezpečnostních mezer, otočení a času odstavení.

Dalším rozdílem mezi oběma systémy je způsob práce s daty. Zatímco současný systém pasivně zobrazuje pozici bez možnosti zásahu, *Vlečkostroj* umožňuje aktivní řízení provozu plánováním příjezdů a odjezdů, přidělováním kolejí a interaktivní správu rozmístění strojů včetně vizualizace.

Výhodou tohoto dvouvrstvého systému je kombinace širokoplošného přehledu a lokální provozní přesnosti, která umožňuje řídit jak provoz v rámci železničního areálu, tak podává přehled o strojích v terénu, a tím zajišťuje vyšší úroveň provozní kontroly a bezpečnosti.

ZÁVĚR

Cílem diplomové práce bylo navrhnout a implementovat systém pro plánování a řízení stání a posuvných operací na budoucí vlečce společnosti Hrochostroj a.s. Výsledkem je moderní webová aplikace *Vlečkostroj*, která výrazně zvyšuje efektivitu provozu vlečky prostřednictvím kombinace interaktivní vizualizace, optimalizačních algoritmů a podpory offline režimu.

Z hlediska funkčního pokrytí aplikace lze konstatovat, že systém naplňuje všechny základní požadavky formulované v zadání. Implementovaná databáze postavená na MySQL nabízí rozšiřitelnou architekturu s jasně definovanými relacemi mezi tabulkami. PHP skripty zabezpečují rychlé a spolehlivé spojení mezi klientskou aplikací a databází.

Hlavním přínosem projektu je především zavedení rozhodování při výběru koleje pro odstavení strojů. Systém integruje Dijkstrův algoritmus pro určení nejbližší volné koleje, vícekriteriální rozhodování metodou TOPSIS, která zohledňuje více parametrů, a výběr nejkratší přípustné koleje pro rychlý výběr.

Díky těmto nástrojům se eliminuje plánování provozní konfliktů, zvyšuje se kapacitní využití infrastruktury a snižuje se potřeba ručních zásahů dispečera. Aplikace také umožňuje vlastní výběr koleje, čímž poskytuje flexibilitu pro mimořádné provozní situace.

Z pohledu uživatelského rozhraní je aplikace intuitivní a responzivní, a to i díky implementaci vizualizačních komponent s využitím knihovny Leaflet.js a interaktivních seznamů v JavaScriptu.

Aplikace je navíc navržena jako PWA, což umožňuje její použití i v podmínkách bez internetového připojení. Funkcionalita offline režimu, podpořená skriptem Service Worker, umožňuje zachování základních funkcí i při výpadku sítě, což je klíčové při práci v provozních podmínkách mimo kancelář.

Z ekonomického pohledu systém nabízí snížení nákladů na ruční plánování a dispečerské řízení, zkrácení času manipulací díky optimalizovaným rozhodnutím, eliminaci neefektivních odstavení strojů a potenciálně i snížení provozních rizik a nákladů spojených s kolizními situacemi.

Navržený systém představuje moderní nástroj, který může výrazně přispět k efektivitě a bezpečnosti provozu na železničních vlečkách. Díky otevřené architektuře je navíc připraven pro další rozvoj a přizpůsobení specifickým požadavkům provozovatele. Zdrojový kód aplikace *Vlečkostroj* s potřebnými daty je možné nalézt na https://github.com/Holubix/organizace-vlecky/releases/tag/v1.0.

POUŽITÁ LITERATURA

[1] *Zákon č. 266/1994 Sb., Zákon o dráhách*.

[2] *Vyhláška č. 177/1995 Sb., vyhláška, kterou se vydává stavební a technický řád drah*.

[3] *Zákon č. 367/2019 Sb., zákon, kterým se mění zákon č. 266/1994 Sb., o dráhách, ve znění pozdějších předpisů, a další související zákony*.

[4] HASTUS software - GIRO Inc. *Home - GIRO Inc.* [online]. [cit. 2025-05-01]. Dostupné z: https://www.giro.ca/en-ca/our-solutions/hastus-software/

[5] Fixed Route Scheduling - Trapeze Software. *Trapeze Software - Passenger Transportation Management Solutions* [online]. [cit. 2025-05-01]. Dostupné z: https://trapezegroup.com/fixed-route-scheduling/

[6] IVU | IVU.rail for rail transport | IVU Traffic Technologies. *IVU | Integrated IT systems for bus and rail | IVU Traffic Technologies* [online]. [cit. 2025-05-01]. Dostupné z: https://www.ivu.com/solutions/highlights/ivurail

[7] Release: RailSys® 2024 - RMCon International. *RMCon International - Rail Management Consulation International GmbH* [online]. [cit. 2025-05-01]. Dostupné z: https://rmcon-int.de/en/railsys-2024-release-en/

[8] *OpenTrack Railway Technolygy - Railway Simulation* [online]. [cit. 2025-05-01]. Dostupné z: https://www.opentrack.ch/opentrack/opentrack_e/opentrack_e.html

[9] Transport Planner - NavSuite powered by Betrian. - *NavSuite - A Comprehensive Ecosystem for Rail Transport* [online]. [cit. 2025-05-01]. Dostupné z: https://navsuite.cz/en/transport-planner-en/

[10] Controlguide OCS / TMS - Siemens Mobility Global. *Siemens* [online]. [cit. 2025-05-01]. Dostupné z: https://www.mobility.siemens.com/global/en/portfolio/rail-infrastructure/mass-transit/controlguide-ocs.html

[11] PROCHÁZKA, David. *PHP 6: začínáme programovat*. Praha: Grada, 2012. Průvodce. ISBN 978-80-247-3899-4.

[12] HTML: HyperTextMarkup Language | MDN. *MDN Web Docs* [online]. [cit. 2025-04-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML

[13] LAURENČÍK, Marek. *Tvorba www stránek v HTML a CSS*. Praha: Grada Publishing, 2019. Průvodce. ISBN 978-80-271-2241-7.

[14] CSS: Cascading Style Sheets | MDN. *MDN Web Docs* [online]. [cit. 2025-04-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/CSS

[15] Javascript | MDN. *MDN Web Docs* [online]. [cit. 2025-04-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[16] MySQL :: MySQL 8.4 Reference Manual. *MySQL :: Developer Zone* [online]. [cit. 2025-04-16]. Dostupné z: https://dev.mysql.com/doc/refman/8.4/en/

[17] Progressive Web Apps (PWAs) documentation. *Microsoft Learn: Build skills that open doors in your career* [online]. [cit. 2025-04-19]. Dostupné z: https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/landing/

[18] Progressive web apps | MDN. *MDN Web Docs* [online]. 1998 [cit. 2025-04-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

[19] *Leaflet - a Javascript library for interactive maps* [online]. 2010 [cit. 2025-04-19]. Dostupné z: https://leafletjs.com/

[20] *OpenStreetMap* [online]. [cit. 2025-04-19]. Dostupné z: https://www.openstreetmap.org/

[21] JQuery API Documentation. *JQuery* [online]. [cit. 2025-04-22]. Dostupné z: https://api.jquery.com/

[22] *Home / PWABuilder* [online]. [cit. 2025-04-19]. Dostupné z: https://www.pwabuilder.com/

[23] PHP: PHP Manual - Manual. *PHP: Hypertext Preprocessor* [online]. [cit. 2025-04-16]. Dostupné z: https://www.php.net/manual/en/

[24] FREIBERGER, Marianne. Lost but lovely: The haversine | plus.maths.org. *Plus.maths.org* [online]. 4 July, 2014 [cit. 2025-04-24]. Dostupné z: https://plus.maths.org/content/lost-lovely-haversine

[25] What is Dijkstra's Algorithm? | Introduction to Dijkstra's Shortest Path Algorithm. *GeeksforGeeks | Your All-in-One Learning Portal* [online]. [cit. 2025-04-22]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/

[26] DSA Djikstra's Algorithm. *W3Schools Online Web Tutorials* [online]. [cit. 2025-04-22]. Dostupné z: https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php

[27] MADANCHIAN, Mitra a Hamed TAHERDOOST. A comprehensive guide to the TOPSIS method for multi-criteria decision making. *Sustainable Social Development*

[online]. 2023 [cit. 2025-04-22]. Dostupné z: https://aber.apacsci.com/index.php/SSD/article/viewFile/2220/2577

SEZNAM TABULEK

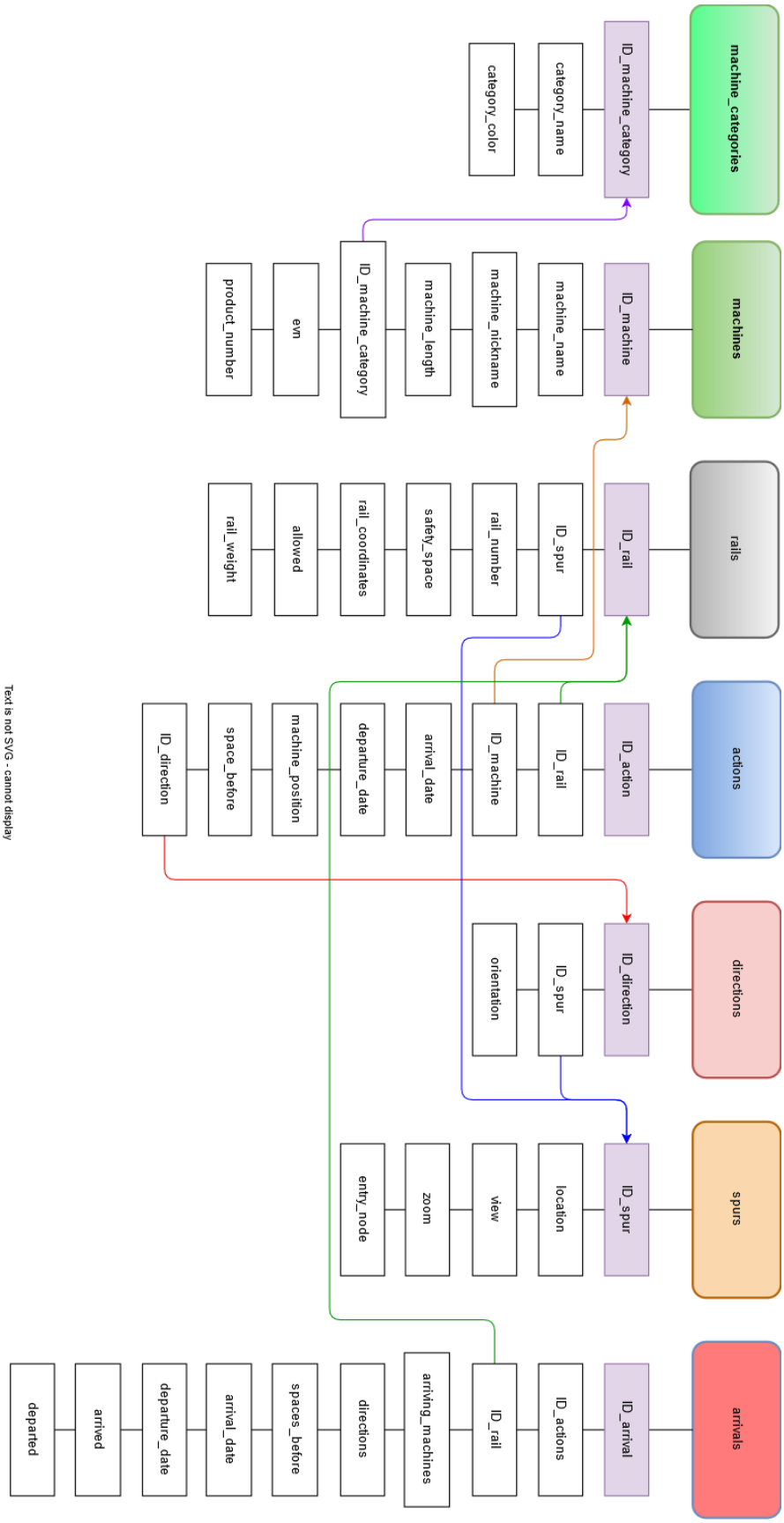
SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

AJAX	Asynchronous JavaScript and XML
	Asynchronní JavaScript a XML
API	Application Programming Interface
	Aplikační programové rozhraní
CSS	Cascading Style Sheets
	Kaskádové style
DOM	Document Object Model
	Objektový model dokumentu
ETCS	European Train Control System
GPS	Global Positioning Systém
HTML	Hypertext Markup Language
IoT	Internet of Things
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
PWA	Progressive Web Apps
	Progresivní webová aplikace
SQL	Structured Query Language
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
URL	Uniform Resource Locator

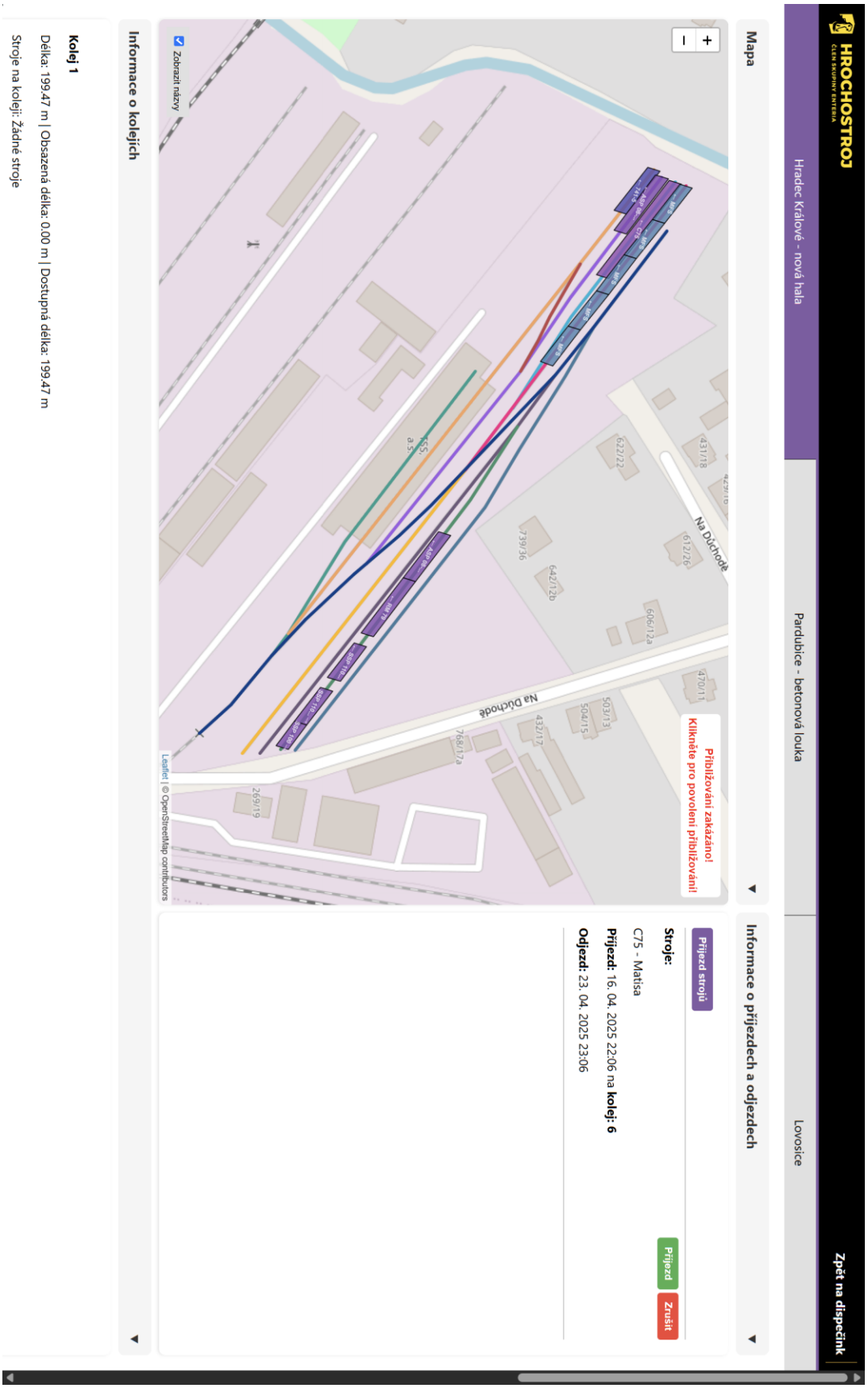
SEZNAM PŘÍLOH

Příloha A Schéma databáze



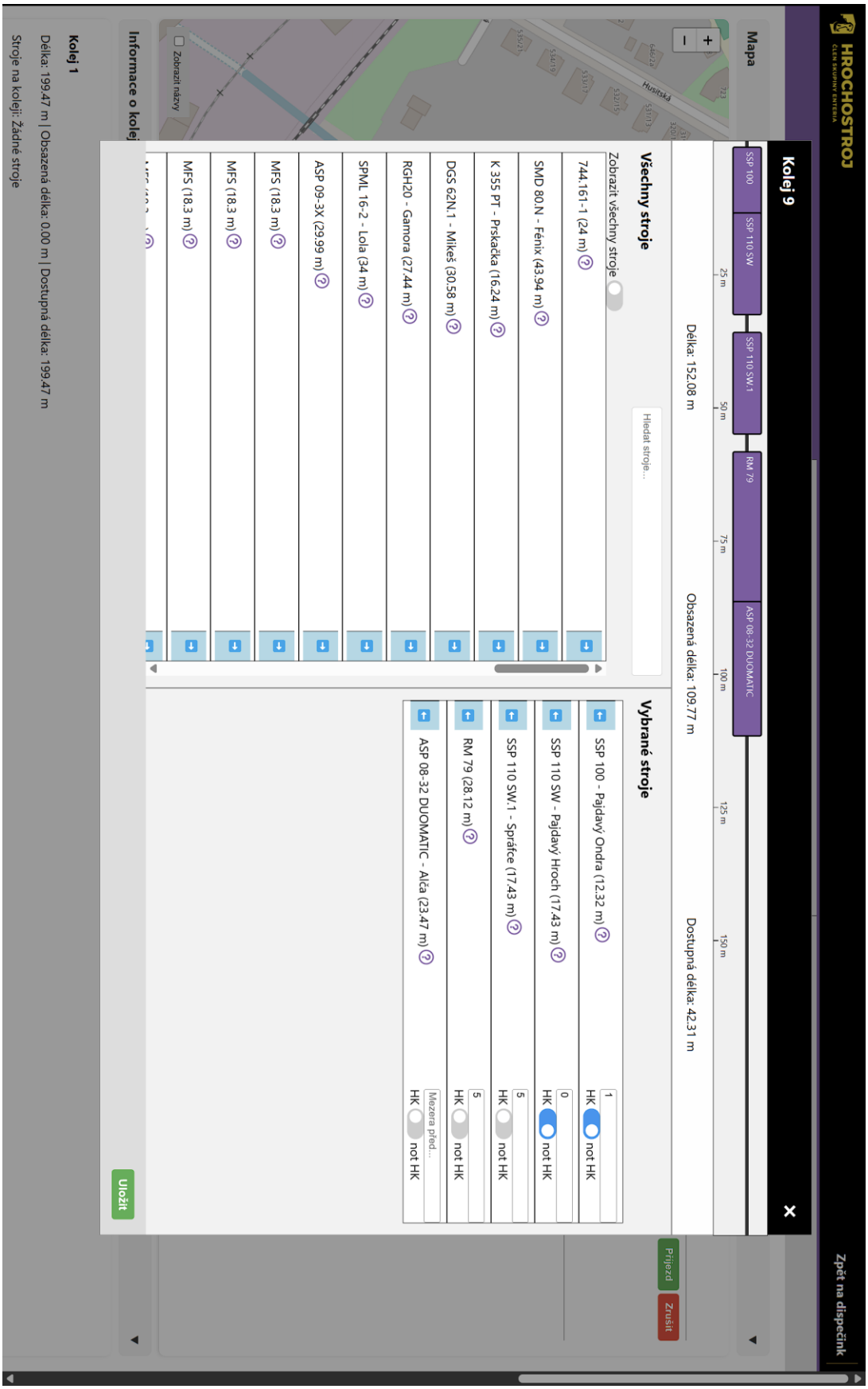
Zdroj: autor

Příloha B Hlavní stránka



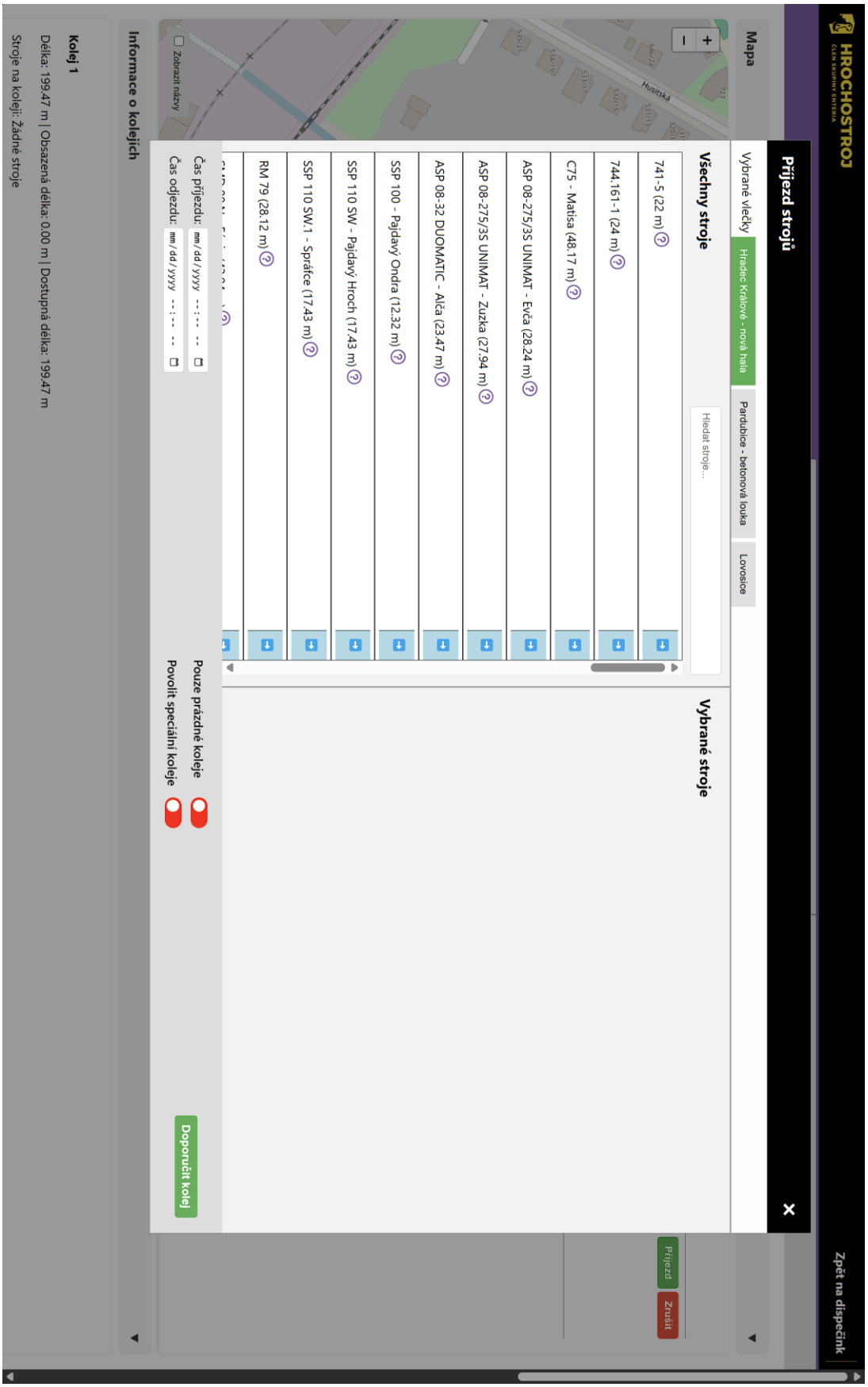
Zdroj: autor

Příloha C Řazení na koleji



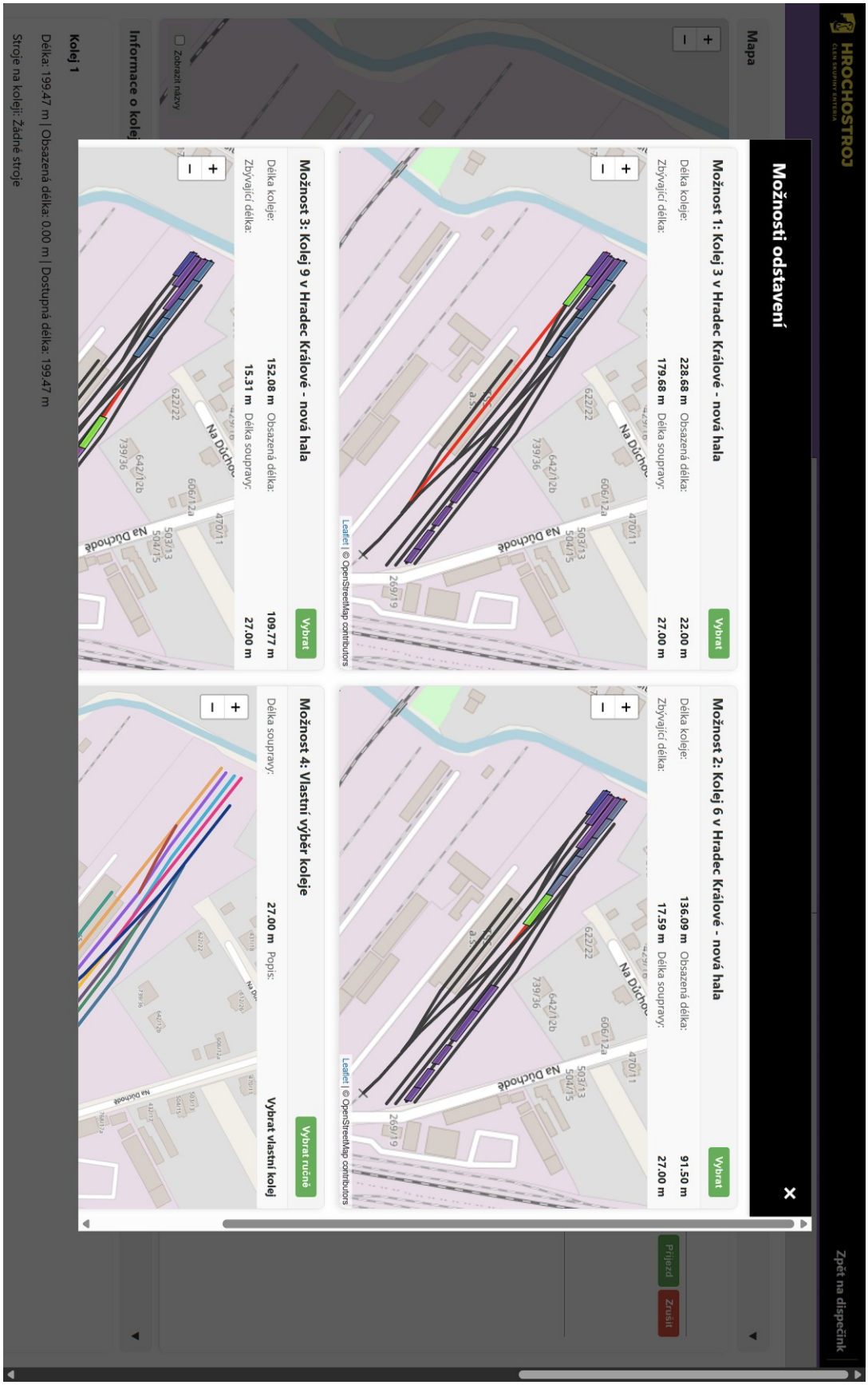
Zdroj: autor

Příloha D Plánování příjezdů a odjezdů



Zdroj: autor

Příloha E Možnosti odstavení strojů

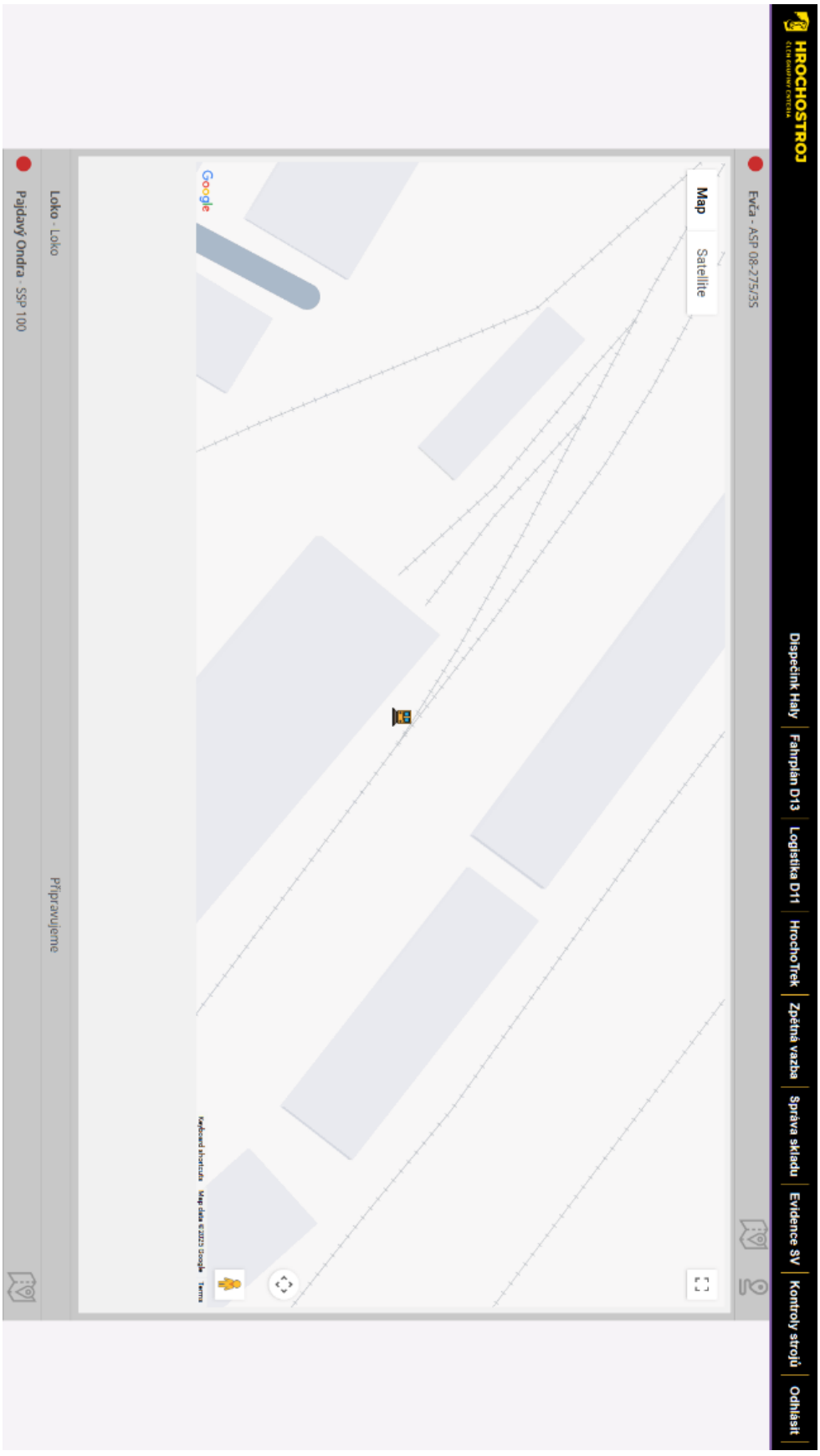


Zdroj: autor

Příloha F *HrochoTrek*

Zdroj: interní dokument

Příloha G Umístění stroje na mapě v *HrochoTrek*



Zdroj: interní dokument