Vysoká škola logistiky o.p.s.

Využití automatické identifikace k analýze a vizualizaci provozu na železnici

(Diplomová práce)

Přerov 2021 Bc. Tomáš Fejfar, DiS.

Vysoká škola logistiky o.p.s.

Zadání diplomové práce

student	**Bc. Tomáš Fejfar, DiS.**
studijní program	Logistika
obor	Logistika

Vedoucí Katedry magisterského studia Vám ve smyslu čl. 22 Studijního a zkušebního řádu Vysoké školy logistiky o.p.s. pro studium v navazujícím magisterském studijním programu určuje tuto diplomovou práci:

Název tématu: **Využití automatické identifikace k analýze a vizualizaci provozu na železnici**

Cíl práce:

Využít data automatické identifikace na modelové železnici k analýze provozu a následné vizualizaci pomocí webového rozhraní. Na základě sběru a vyhodnocení konkrétních dat implementovat uživatelské funkce pro pokročilejší analýzu dat. Data interpretovat pomocí webového rozhraní.

Zásady pro vypracování:

Využijte teoretických východisek oboru logistika. Čerpejte z literatury doporučené vedoucím práce a při zpracování práce postupujte v souladu s pokyny VŠLG a doporučeními vedoucího práce. Části práce využívající neveřejné informace uveďte v samostatné příloze.

Diplomovou práci zpracujte v těchto bodech:

Úvod
1. Automatická identifikace v logistice
2. Implementace uživatelských funkcí
3. Tvorba aplikace a vizualizace dat pomocí www
4. Pilotní ověření a vyhodnocení aplikace

Závěr

Rozsah práce: 55 – 70 normostran textu

Seznam odborné literatury:

GROS, Ivan a kol. Velká kniha logistiky. Praha: Vysoká škola chemicko-technologická v Praze, 2016. ISBN 978-80-7080-952-5.

KOFLER, Michael a Bernd ÖGGL. PHP 5 a MySQL 5: průvodce webového programátora. Brno: Computer Press, 2007. ISBN 978-80-251-1813-9.

FEJFAR, Tomáš. Vizualizace provozu železnice pomoci auto ID. Přerov, 2018. BP. Vysoká škola logistiky o.p.s. Vedoucí práce Libor Kavka.

Vedoucí diplomové práce:	Ing. Libor Kavka, Ph.D.
Datum zadání diplomové práce:	30. 10. 2020
Datum odevzdání diplomové práce:	13. 5. 2021

Přerov 30. 10. 2020

Ing. Blanka Kalupová, Ph.D.
vedoucí katedry

prof. Ing. Václav Cempírek, Ph.D.
rektor

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a že jsem ji vypracoval samostatně. Prohlašuji, že citace použitých pramenů je úplná a že jsem v práci neporušil autorská práva ve smyslu zákona č. 121/2000 Sb., o autorském právu, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

Prohlašuji, že jsem byl také seznámen s tím, že se na mou diplomovou práci plně vztahuje zákon č. 121/2000 Sb., o právu autorském, právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména § 60 – školní dílo. Beru na vědomí, že Vysoká škola logistiky o.p.s. nezasahuje do mých autorských práv užitím mé diplomové práce pro pedagogické, vědecké a prezentační účely školy. Užiji-li svou diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Vysokou školu logistiky o.p.s.

Prohlašuji, že jsem byl poučen o tom, že diplomová práce je veřejná ve smyslu zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, zejména § 47b. Taktéž dávám souhlas Vysoké škole logistiky o.p.s. ke zpřístupnění mnou zpracované diplomové práce v její tištěné i elektronické verzi. Tímto prohlášením souhlasím s případným použitím této práce Vysokou školou logistiky o.p.s. pro pedagogické, vědecké a prezentační účely.

Prohlašuji, že odevzdaná tištěná verze diplomové práce, elektronická verze na odevzdaném optickém médiu a verze nahraná do informačního systému jsou totožné.

V Přerově, dne 16. 08. 2021 .....

podpis

Poděkování

Děkuji tímto panu Ing. Liboru Kavkovi, Ph.D. za rady a nápady při zpracování mé diplomové práce.

Anotace

Z důvodu dřívější tvorby bakalářské práce na téma Vizualizace provozu železnice pomocí auto ID, přichází myšlenka na její kompletní úpravu a přepracování do lépe fungující a více funkcemi opatřené webové aplikace. Na přípravu webové aplikace je nutné získání informací z příslušené databáze modelové železnice, která je umístěna v prostorách Vysoké školy logistiky v Přerově. Pomocí této modelové železnice je možné zkoušet fungování RFID čteček. V první části diplomové práce jsou obsáhlé informace o automatické identifikaci v logistice. Ve zbylé části je kompletní tvorba webové aplikaci od první myšlenky po finální odzkoušení webové aplikace.

Klíčová slova

RFID, automatická identifikace, vizualizace, databáze, MySQL, PHP, modelová železnice

Annotation

Due to the earlier creation of a bachelor's thesis on the topic of visualization of railway operation using auto ID, the idea comes to its complete modification and reworking into a better functioning and more functional web application. To prepare a web application, it is necessary to obtain information from the relevant model railway database, which is located on the premises of the University of Logistics in Přerov. Using this model railway, it is possible to test the operation of RFID readers. The first part of the thesis contains comprehensive information on automatic identification in logistics. The remaining part is the complete creation of a web application from the first idea to the final testing of the web application.

Keywords

RFID, automatic identification, visualization, database, MySQL, PHP, model railway

Obsah

Úvod

V prostorách Vysoké školy logistiky v Přerově je možné pracovat za dozoru vyučujících s modelovou železnicí. Je možné manipulovat s vagony a lokomotivami a spouštět provoz. Při spuštění se aktivuje systém na zapisovaní do databáze, která je právě potřebná pro tuto diplomovou práci.

Hlavním cílem této práce je vytvoření webové aplikace, který bude schopna přehledně a srozumitelně zobrazovat vlaky a informace o nich, když projíždějí okolo RFID čteček, které zapisují data a automaticky identifikují vlaky.

Dalším cílem této práce je teoretické vysvětlení automatické identifikace v logistice od možnosti čárových kódů a použití radiofrekvenční identifikace, která je využita u modelové železnice pro komunikaci.

Mezi poslední dva cíle patří implementace uživatelských funkcí, správné propojení a nasazení do aplikace. A tím posledním je kompletní odzkoušení aplikace a zhodnocení její funkčnosti.

1 Automatická identifikace v logistice

Samostatná identifikace je potřebná, jak již název napovídá, na identifikaci konkrétního zboží, výrobku, materiálu nebo nákladu. Identifikace obsahuje všechny potřebné informace, například to může být hmotnost, typ zboží u nákladu lokalita a mnoho dalších možností. Všechny informace jsou uloženy v počítačovém systému a je možné je ze systému načíst, uložit nebo změnit.

V logistice se nejčastěji použije identifikace pro získání informací o nákladu, který je převážen. Je možné zjistit lokalitu, odkud a kam je náklad převážen.

Automatická identifikace je použita pro zkrácení času mezi zjišťováním informací o nákladu nebo zboží. Pro nejrychlejší získání informací je vše zautomatizované, tzn. že stačí načíst štítek pomocí optického skeneru nebo načíst kód pomocí radiofrekvenčního signálu a získat informace z počítačového systému.

Hlavními prvky automatické identifikace jsou [1]:

- Nosič kódu,
- Snímací zařízení,
- Programová jednotka,
- Vyhodnocovací jednotka.

Nosič kódu

Zařízení, na kterém jsou uloženy data o objektu. Je většinou uloženo na povrchu objektu.

Snímací zařízení

Pomocí toho zařízení je možné načíst data uložená na nosiči kódu.

Programová jednotka

Zajišťuje zabezpečení identifikačního kódu na nosič dat.

Vyhodnocovací jednotka

Využívána pro úpravu a zpracování dat do tvaru, kdy je možné určitá data zobrazit a přečíst.

1.1 Optická identifikace

Mezi zařízení optické identifikace patří čárové kódy, systém OCR a vizuální technologie. Tato technologie využívá snímání odráženého světla z určitého vzoru tmavých čar a světlých mezer.

„Hlavní oblastí využití této technologie je však při sledování toků zboží v celém dodavatelském systému. Pro identifikaci výrobků a manipulačních jednotek jsou využívány štítky, na kterých jsou vhodnou kombinací svislých čas a mezer zakódovány číselné údaje.“ [2, s. 410]

1.1.1 Čárové kódy

Čárový kód je nejpoužívanější metoda automatické identifikace. V oblasti logistiky je možné najít čárový kód na objektu, který je převážen nebo uskladněn. V obchodní oblasti je každé zboží vybaveno štítkem s čárovým kódem nebo je přímo vytištěn na obalu výrobku.

Výhodami použití čárového kódu je jeho [3]:

- Přesnost,
- Rychlost,
- Flexibilita,
- Produktivita,
- Efektivnost,
- Dosledovatelnost,
- Cena.

Z výhod tedy vyplývá, že používání čárových kódů je velice aktuální a jednoduché. Jak již bylo uvedeno výše, tak čárové kódy patří do oblasti optické identifikace, která pracuje na principu odrážení světla od tmavých čar a světlých mezer. Tmavé čáry a světlé mezery mají vždy jinou šíři, aby bylo možné poznat o jaký znak se jedná. U čárových kódů je vše v tomto principu kódováno a každý čárový kód má svoje logické pořadí.

Na skenování čárových kódů se používají speciální skenery, které jsou schopné vyslat světelné paprsky, jež přečtou daný čárový kód. U tmavých čar je světlo pohlceno a u světlých odraženo. Pomocí tohoto principu je skener schopný poznat, o jakou kódovanou informaci se jedná.

Mezi základní a nejpoužívanější typy čárových kódů patří [4]:

- Code 39 a Code 39 Mod 43,
- U.P.C. A,
- EAN 13 a EAN 8,
- Code 93,
- Interleaved 2/5 a Interleaved 2/5 Mod 10,
- Code 128,
- Codabar,
- MSI,
- PDF417,
- DataMatrix,
- QR Code,
- Code16k

Code 39 a Code 39 Mod 43

Code 39 se skládá ze 43 znaků. Jsou to znaky abecedy od A do Z, čísla od 0 do 9 a speciální znaky (-, ., $, /, +, %, a mezera). Code 39 obsahuje ještě počáteční a ukončovací znak, je to znak hvězdičky (*). Vždy se skládá z 5 tmavých čas a 4 mezer, kdy je to vždy 3 široké a 6 úzkých ploch ze všech 9 možných prvků.

Code 39 Mod 43 je nadstavba pro Code 39, která obsahuje kontrolní znak vypočítaný ze součtu hodnot všech znaků řetězce.

Obr. 1.1 Zobrazení Code 39

Zdroj: vlastní zpracování

U.P.C. A

U.P.C. A se skládá z celkem 12 znaků, kdy první znak je určující, jaký je použit systém číslování, následujících 5 znaků je určením výrobce, dalších 5 znaků je určením o jaký výrobek se jedná a poslední znak je kontrolní. Celý tento čárový kód se se skládá pouze z číslic od 0 do 9.

Obr. 1.2 Zobrazení U.P.C. A

Zdroj: vlastní zpracování

EAN 13 a EAN 8

EAN je pokračováním kódu U.P.C. Kód EAN je možné zapisovat v několika variantách 8,12,13 a 14 znaků. Všechno musí být číselné znaky. Nejčastěji se využívají EAN 13 a EAN 8 a dle názvu je patrné, že se jedná o 13 znaků a o 8 znaků. EAN štítky se využívají hlavně na identifikaci zboží v obchodech.

Obr. 1.3 Zobrazení EAN 13

Zdroj: vlastní zpracování

Obr. 1.4 Zobrazení EAN 8

Zdroj: vlastní zpracování

EAN 13 se skládá ze dvou skupin 6 znaků, které jsou kódovány do skupin po 2 tmavých a 2 světlých pruzích. První 3 znaky vždy ukazují, v jaké státu je zaregistrovaný výrobní závod pro dané zboží. Pro Českou republiku to je číslo 859. Čísla jednotlivých státu přiřazuje společnost GS1, která sídlí v Bruselu.

Code 93

Code 93 funguje v základu na stejném principu jako Code 39. Je možné využívat všech 128 ASCII znaků. Code 93 je složen ze 3 tmavých a 3 světlých pruhů, kdy pruh má vždy 4 možnosti šířky. Typickou strukturou u Code 93 je nejdříve * jako startovací znak, následuje zakódovaná zpráva, dále 2 kontrolní znaky a nakonec * jako konečný znak. Zakódovaná zpráva a kontrolní znaky je možné rozluštit na obr. 1.5.

Obr. 1.5 Zobrazení Code 93

Zdroj: vlastní zpracování

Interleaved 2/5 a Interleaved 2/5 Mod 10

Jak název napovídá, tak tzv. prokládaný kód 2 z 5 je složen vždy ze dvou širokých pruhů z 5 ať už tmavých nebo 5 světlých. U toho kódu je nejdříve nutné přečíst 5 tmavých pruhů

pro první znak a následně 5 světlých pruhu pro druhý znak a takto se to střídá, až se přečte celý čárový kód. Tento čárový kód smí obsahovat pouze číselné hodnoty.

Obr. 1.6 Zobrazení Interleaved 2/5

Zdroj: vlastní zpracování

Interleaved 2/5 Mod 10 je rozšířen o 1 znak na konci, který obsahuje kontrolní znak. Tento kontrolní znak je vypočítán součtem všech hodnot, a který je vydělen číslem 10. Následný zbytek po dělení je kontrolním znakem.

Tento čárový kód je vždy složen ze sudého počtu znaků. Jestli se kóduje pouze lichý počet znaků, tak se vždy na začátek kódu připíše 0.

Code 128

Code 128 je schopný zakódovat až 128 znaků z ASCII tabulky. Je schopný rozlišovat velká a malá písmena. Znak u Code 128 je složený ze 3 tmavých a 3 světlých pruhů a je možné rozlišovat až 4 šířky pruhu. Code 128 je využíván převážně v logistice.

Code 128 má 3 různý typy kódování:

- 128A – obsahuje ASCII znaky od 00 do 95,
- 128B – obsahuje ASCII znaky od 32 do 127,
- 128C – obsahuje ASCII znaky od 00 do 99.

Obr. 1.7 Zobrazení Code 128

Zdroj: vlastní zpracování

Codabar

Pomocí Codabar je možné kódovat číselné hodnoty a speciální znaky. Codabar obsahuje i písmena A, B, C a D, která jsou počátečními a konečnými znaky a nezobrazují se v čárovém kódu. Znak čárového kódu Codabar se skládá vždy ze 4 tmavých a 3 světlých pruhů.

Obr. 1.8 Zobrazení Codabar

Zdroj: vlastní zpracování

MSI

MSI se skládá pouze z číselných znaků a nemá pevnou délku řetězce. Obsahuje 1 nebo 2 kontrolní znaky, které se získávají zbytkem po dělení číslem 10 nebo 11, případně jejich kombinací.

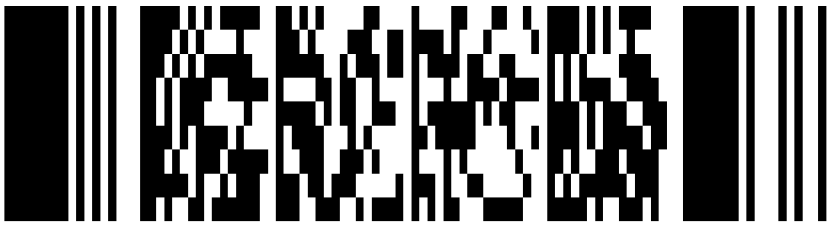
Obr. 1.9 Zobrazení MSI

Zdroj: vlastní zpracování

PDF417

PDF 417 je jedním z dvourozměrných čárových kódů. Jelikož se jedná o dvourozměrný čárový kód, tak se čte vždy po řádcích. Je možné zakódovat kompletní znaky ASCII tabulky. PDF 417 se většinou využívá pro získání dat z externí databáze, ale je možné mít zapsané jen přímo informace. Znak se vždy skládá ze 4 tmavých a 4 světlých pruhů a čárový kód je sám o sobě rozdělen do sekcí:

- Počáteční sekce,
- Levé řádkování,
- Kódovaná data,
- Pravé řádkování,
- Konečná sekce.



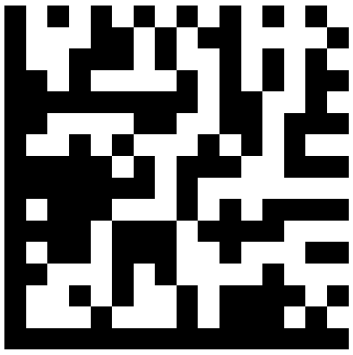
Obr. 1.10 Zobrazení PDF417

Zdroj: vlastní zpracování

PDF417 má i své limity, maximálně je možné, aby bylo v řádku 30 sekcí, které se dají přečíst, a 90 řádků. Velikost datového soubor může být až 1,1kB, tzn. že může obsahovat veliké množství informací.

DataMatrix

DataMatrix je dalším dvourozměrným čárovým kódem. Čárový kód DataMatrix se již neskládá z pruhů, ale skládá se z tmavých a světlých čtverců. Maximálně může obsahovat 2335 znaků při kombinaci písmen a čísel nebo 3116 znaků při číselné kombinaci. Na skenování se již nepoužívá klasický laserový skener a využívá se již CCD kamera. Je tedy možné snímat čárové kódy i ze vzdálenosti 1 metru i pod různým úhlem. Jelikož se kód DataMatrix skládá pouze ze čtverců, tak jeho velikost je většinou čtvercová. Z toho vyplývá, že minimální velikost kódu je 10x10 čtverečků a maximální velikost kódu je 144x144 čtverečků. Tento čárový kód je možné spatřit na elektronických součástkách například na základní desce od počítače. Je také možné kód vyrýt nebo vypálit do železné konstrukce. DataMatrix je možné přečíst i když je z 30% poničen.



Obr. 1.11 Zobrazení DataMatrix

Zdroj: vlastní zpracování

QR Code

V dnešní době se QR Code může využívat na získání různých informací. V kódu mohou být uchovány vizitky, URL adresy, doplňující informace z magazínů nebo platební údaje u bankovních společností. QR Code může obsahovat maximálně 7089 pouze číselných hodnot nebo 4296 znaků při kombinaci písmen a čísel. QR Code je možné přečíst i při 30% poškození nebo špatné kvalitě tisku. Stejně jako u čárového kódu DataMatrix je čtvercový a má minimální a maximální velikost. Minimální velikost u QR Code je 21x21 a maximální velikost je 177x177.

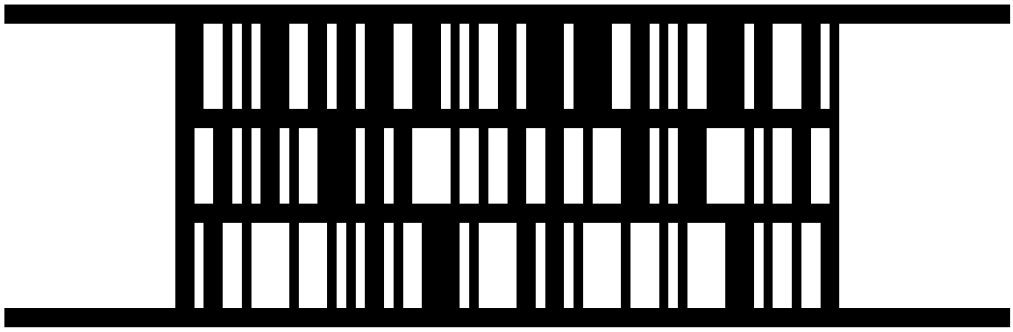
Obr. 1.12 Zobrazení QR Code

Zdroj: vlastní zpracování

QR Code obsahuje ve 3 rozích poziční značky, jak je vidět na obr.1.12. Jsou důležité pro rychlou lokalizaci kódu v obraze. Z toho důvodu může být kód otočený pod různým úhlem.

Code16k

Struktura Code16k vychází z čárového kódu Code 128, ale je rozšířený na více řádků. Každý znak v Code16k je obsažen ve 2 až 16 řádcích. A je možné dát dohromady až 107 znaků na 16 řádků. Maximálně je možné pomocí Code16k zakódovat až 8025 ASCII znaků nebo 16050 číselných hodnot. Code16k je možné využívat pro svoji rozmanitost a škálu použitelnosti a je možné jej tisknout na jakémkoliv zařízení.



Obr. 1.13 Zobrazení Code16k

Zdroj: vlastní zpracování

1.1.2 Systém OCR

Systém OCR je použit pro digitalizaci vytištěného či ručně napsaného textu a je možné s ním pracovat v počítači pomocí textového editoru. Skenování se provádí pomocí speciálního skeneru. Systém OCR je možné využít například pomocí multifunkční tiskárny, která obsahuje klasický skener, jen výsledek nebude dokonalý. Vždy je důležité zajistit, aby text byl dostatečně čitelný a aby se text nepřekrýval. Jakmile bude text špatně čitelný, tak se bude prodlužovat doba potřeba pro správné naskenování nebo se může stát, že bude text oskenován špatně. Skenovaný text může mít problémy s podobnými znaky jako je například písmeno **O** nebo číslice **0**, případně se velikostí písmen u malého **z** nebo velkého **Z**. Ve většině případě je ale stále OCR rychlejší možností než přepisování celého textu samostatně na počítači.

1.1.3 Vizuální technologie

Funguje na podobném principu jako systém OCR, jen nepřevádí vytištěný či ručně napsaný text. Ale je schopný převést různé obrázky do digitální podoby, aby s nimi bylo možné pracovat v počítači.

1.2 Magnetická identifikace

Magnetická identifikace využívá principu magnetů. Je to jedna z hodně rozšířených metod identifikace, jelikož je využívána hlavně na platebních kartách v bankovnictví. Využívají se pro identifikaci účtu a majitele účtu v bankomatu nebo při platbách u obchodníka. Magnetická identifikace je využívána také u zabezpečovacích karet nebo u tzv. členských karet u obchodníků.

1.3 Biometrická identifikace

Biometrická identifikace se nejvíc využívá pro identifikaci osob a ověření identity. Pomocí biometriky je možné identifikovat osobu dle různých části těla. Je možné identifikovat otisk prstu, oční sítnici, hlas, písmo, podpis, obličej nebo DNA. Všechny tyto možnosti má každý člověk jedinečné, díky tomuto není možné je hned zfalšovat a následně se dostat například do systému za jinou osobu. Zpracování těchto údajů při skenování je v rámci sekund a jedná se tak o rychlou možnost identifikace.

Všechny výše uvedené údaje je důležité nejdřív uložit do připraveného databázového systému, který je pro to určen. Pomocí toho systému už stačí jen umístit potřebný skener nebo kameru na správné místo a udělit přístup pomocí porovnání dat s databází.

Jak již bylo uvedeno výše, tak biometrická identifikace slouží pro kontrolu, zda se nevydává někdo za cizí osobu. Je tedy možné při správné identifikaci například vpustit osobu do uzamčené místnosti nebo je možné podle podpisu uzavřít smlouvu.

V dnešní době je možné si nechat vytvořit občanský průkaz, který bude obsahovat biometrické údaje. Jsou jimi 2 otisky prstů a zobrazení obličeje držitele občanského průkazu. Pomocí této možnosti bude občanský průkaz lépe chráněn a nebude možné jej tak snadno falšovat. Všechny uložené údaje budou uloženy na zabezpečeném čipu na bezkontaktním čipu v občanském průkazu.

1.4 Induktivní identifikace

Funguje na podobném principu jako radiofrekvenční identifikace, jen s tím rozdílem, že využívá elektromagnetickou indukci. Tato identifikace se nejvíce využívá v logistice pro identifikaci palet a kontejnerů a jejich obsahu. Snímané objekty obsahují indukční

štítky, které obsahují patřičné informace a je možné je pomocí systému upravit podle aktuální potřeby.

1.5 Radiofrekvenční identifikace

Radiofrekvenční identifikace nebo RFID je nástupcem čárových kódů. Jedná se o bezdrátovou technologie, která funguje na principu elektromagnetických vln na rádiové frekvenci. Díky této technologii je možné snímat potřebné RFID Tagy i bez přímé viditelnosti, a hlavně na větší vzdálenost. RFID mají výhodu oproti čárovým kódům ve velikosti obsažených dat. Je možné obsáhnout informace o velikosti až 64kb. Další výhodou je, že jsou opravdu rychlé, rychlost získání informací se pohybuje v řádech milisekund až desítek milisekund. V jednu chvíli je možné snímat i více čipů najednou.

Výhody RFID:

- Informace v reálném čase,
- Zlepšení kvality výroby,
- Více informací,
- Umožňuje čtení na delší vzdálenosti.

Nevýhody RFID:

- Ze začátku to jsou vysoké náklady na pořízení,
- Každý produkt je vystopovatelný,
- Obrovské množství dat.

Systém RFID se skládá z několika hlavních prvků:

- RFID Tag,
- RFID čtečka,
- RFID kontroler.

1.5.1 RFID Tag

Jedná se o nosič informací, která jsou zakódovaná a umístěna na daném objektu. Slouží tedy pro identifikaci přečtením daných informací nebo je možné i dané informace nechat zapsat do systému, z toho vyplývá, že data budou přečtena a následně uložena.

RFID obsahují čip a tyto čipy dělíme na 3 typy podle způsobu přenosu a napájení [2]:

- Pasivní čipy,
- Aktivní čipy,
- Polopasivní čipy.

Pasivní čipy

Pasivní čipy neobsahují zdroj napájení, takže mají jen určitý dosah v řádu několika metrů. Dosah je odvozen od síly a velikosti RFID čtečky. Nejvíce se tyto typy čipů dají nalézt na oblečení v obchodech a je tím zabezpečeno proti krádeži. U pasivních čipů jsou všechny údaje stále zapsány do paměti čipu uloženém na integrovaném obvodu. Výhodou těchto čipů je jejich životnost, která se uvádí v řádech desítek let.

Pasivní čipy můžeme rozdělit na:

- Pouze pro čtení,
- Stálá možnost čtení, ale jen jeden zápis,
- Čtení a zápis.

Aktivní čipy

Aktivní čipy obsahuji vlastní zdroj napájení, takže se jim mnohonásobně zvyšuje dosah na rozdíl od pasivních čipů. Tento dosah může být až 100 m. Jelikož je pasivní čip napájen, tak je možná uložená data aktualizovat a tím opět použít RFID Tag u jiného objektu.

Polopasivní čipy

Polopasivní čipy obsahují vlastní zdroj napájení. Tento zdroj však není neustále využíván a je možné šetřit energii v době nenačítání. Jakmile je vyslán na polopasivní čip radiový signál, tak se RFID Tag spustí a odešle všechny patřičné informace o objektu. Polopasivní čipy jsou schopné také samy sbírat údaje pro pozdější použití. Pomocí RFID čtečky se nové údaje přečtou a je možné zjistit všechny informace, které byly sbírány.

RFID Tagy jsou schopny pracovat na třech různých frekvenčních oblastech. První je **nízká frekvence** a to 125kHz, další je **vysoká frekvence** a to 13,56MHz a poslední jsou

ultra krátké vlny, které se pohybují v rozsahu 860 až 950MHz. Ultra krátké vlny jsou nejlepší možností použití, jelikož dosahují nejvyšších rychlostí a dosahů.

1.5.2 RFID čtečka

Jedná se o zařízení, které se využívá na přečtení RFID Tagu a kterým lze získat všechny informace obsažené v tomto tagu. Využívá se pro komunikaci s RFID kontrolerem. RFID čtečka je schopna přečíst všechny informace, ale je schopna i nové informace na RFID tag uložit a aktualizovat patřičné údaje. RFID čtečka neustále vysílá radiový signál a čeká, kdy se objeví v dosahuji RFID tag, aby přečetla všechny jeho informace a uložila je do svojí interní paměti nebo do externí paměti RFID kontroleru, případně jiného zařízení, které spravuje RFID čtečku.

RFID čtečky je možné rozdělit do dvou hlavních kategorií [5]:

- Stacionární RFID čtečky,
- Mobilní RFID čtečky.

Stacionární RFID čtečky

Je možné je využívat v logistických závodech a mohou vypadat jako průchozí brány. Vykazují vysokou odolnost v náročných podmínkách. Pro jejich fungování a načtení požadovaných objektů stačí projet skrz stacionární čtečku a všechny údaje se vypíšou na ovládací systémy a nejčastěji se uloží do databáze.

Mobilní RFID čtečky

Jsou to přenosná zařízení, kam se ukládají načtená data do interní paměti zařízení. Je tedy nutné tato zařízení připojit k externímu přístroji, aby bylo možné uložená data převést do systému.

1.5.3 RFID kontroler

Jedná se o zařízení, které má na starosti ovládání RFID čteček. Přes kontroler prochází data, vypisují se a ukládají se. Ve většině případů se jedná o počítač, který obsahuje kompletní databázi všech možných skenovaných objektů a je schopný je podle určitých

přečtených informací uložit. Jednoduše je to tak, že dekóduje uložené informace na RFID Tagu.

1.6 Technologie NFC

Jedná se o bezdrátovou technologii založenou stejně jako u RFID na bázi radiofrekvenčních vln. NFC má dosah pouze do 4 cm a z toho důvodu zatím nemá moc využití v delším rozsahu. Na krátké vzdálenosti je možné použít NFC na obousměrnou komunikaci pro neustálé zjišťování potřebných informací. NFC je možné aktuálně využívat nejvíce v mobilních telefonech na obousměrnou komunikaci při odesílání souborů nebo jiném sdílení nebo je možné pomocí NFC platit u obchodníka přiložením mobilního telefonu k platebnímu terminálu.

2 Implementace uživatelských funkcí

Implementace uživatelských funkcí znamená vymyšlení a přípravu na fungování určitého systému. Před implementací je důležité zajistit analýzu kompletního fungování systému, který bude využíván. Modelová železnice je již kompletně připravena a jsou z ní brána všechny data. Tato data je schopna modelová železnice uložit do databáze.

2.1 Funkce

Pro kompletní analýzu dat z modelové železnice je nutné připravit funkce, které budou schopny vyhledávat většinu nejdůležitějších informací. Tyto informace musí mít patřičnou důležitost pro uživatele, který systém využívá.

2.1.1 Funkce vyhledávání podle čidla

První funkcí, která má informační charakter pro uživatele, je funkce vyhledávání vlaků dle určitého čísla čidla.

2.1.2 Funkce vyhledávání dle určitého data

Druhá funkce pro uživatele je možnost vyhledávání vlaku podle vyplněného data, kdy měl daný vlak nebo vlaky projet čidlem.

2.1.3 Funkce vyhledávání dle určitého data a času

Třetí funkcí je možnost vyhledat vlak podle data a přesného času. Jakmile je tato informace zadána, tak se zobrazí jen přesný vlak, který byl v té době uložen do databáze.

2.1.4 Funkce vyhledávání dle určitého rozmezí dat

Čtvrtou funkcí je poslední možnost vyhledávání, kdy je zadáno počáteční datum a konečné datum rozmezí, pro které je potřeba vypsat požadované vlaky.

2.1.5 Funkce zobrazení informací o vlaku

Pátou funkcí je možnost zobrazit při jakémkoliv vyhledávání vlaků jeho informace a seřazení vagonu, které má připojené.

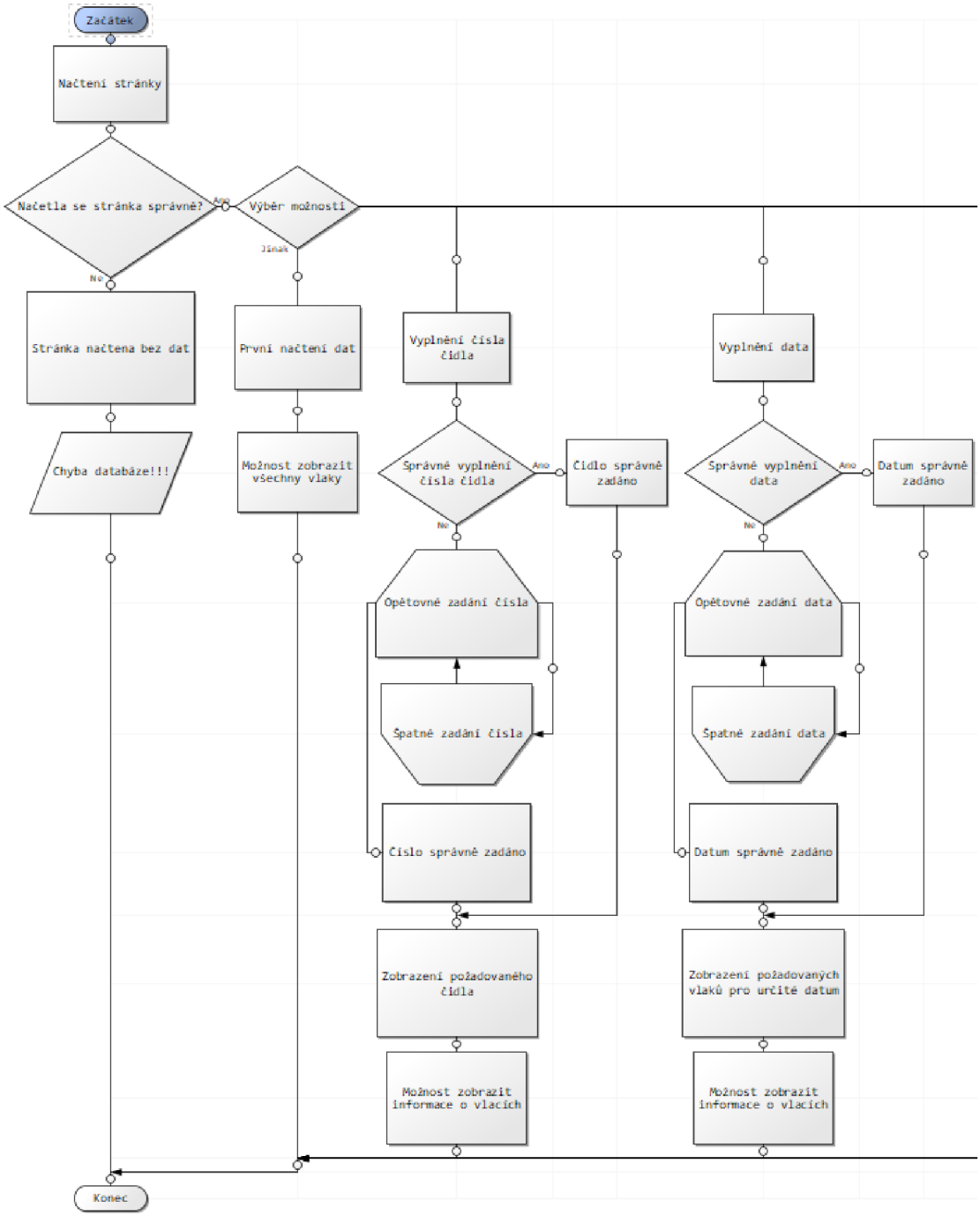
2.1.6 Funkce zobrazení všech dostupných vagonu a lokomotiv

Poslední funkcí je možnost zobrazit kompletní tabulku obsahující důležité informace o všech dostupných lokomotivách a vagonech.

2.2 Tvorba vývojového diagramu

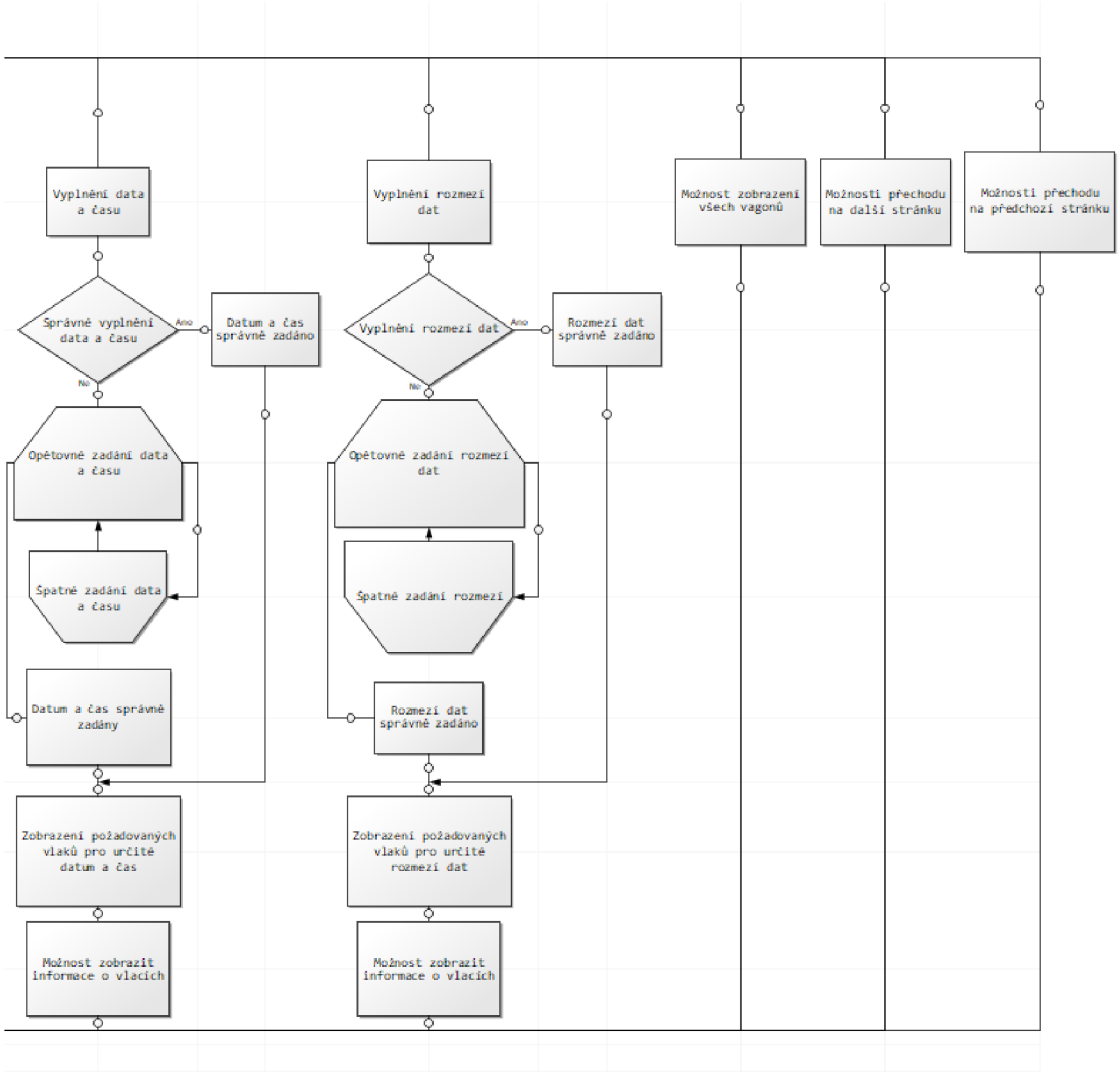
Vývojový diagram je možné připravit pro snadnější vytvoření aplikace a jednodušší pochopení jejímu fungování.

Důležité je si rozmyslet fungování www stránky a následně ho zobrazit v následujícím vývojovém diagramu na obr.2.1 a obr.2.2.



Obr. 2.1 Znázornění fungování www stránky část 1

Zdroj: vlastní zpracování



Obr. 2.2 Znázornění fungování www stránky část 2

Zdroj: vlastní zpracování

Na výše uvedených diagramech je patrné, co se může stát při správném chování uživatele a správném chování www stránky. V diagramu není uvedena vývojová část aplikace, jelikož tato funkce není uživateli k dispozici. Uživatel není schopen, jakkoliv upravovat uložené informace v databázi modelové železnice.

3 Tvorba aplikace a vizualizace dat pomocí www

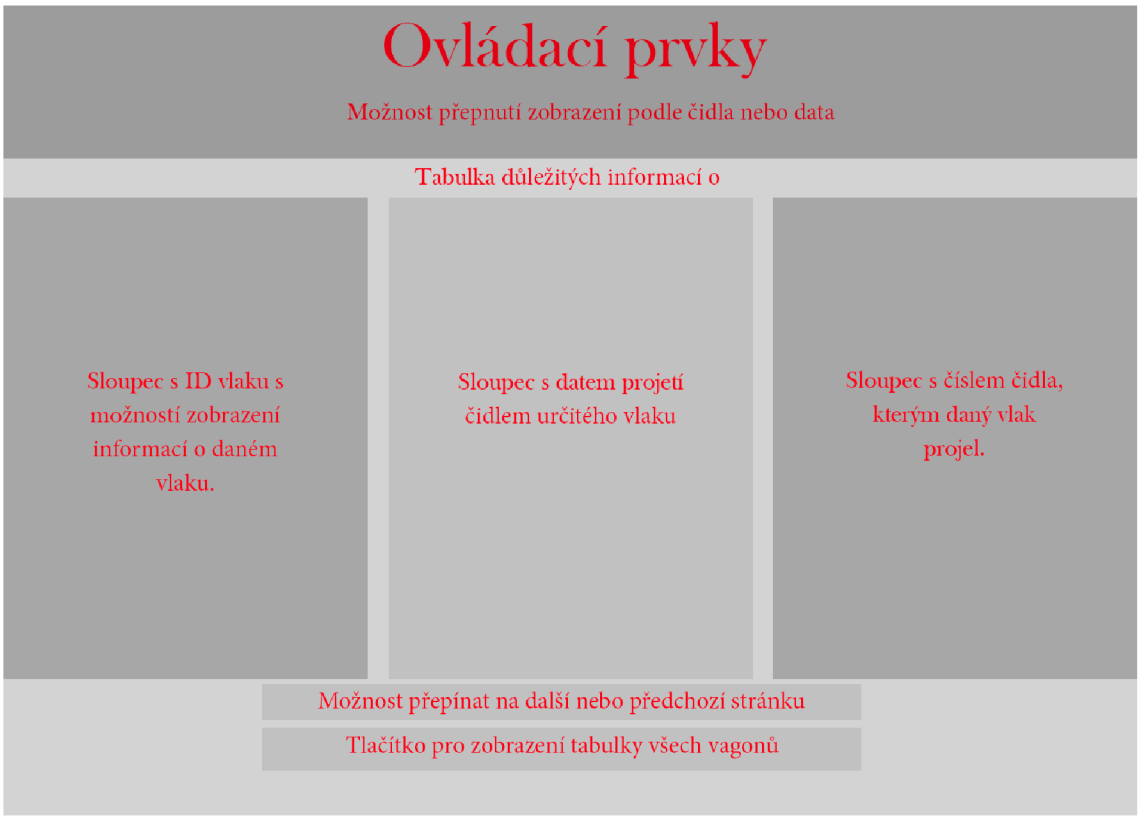
Kompletní tvorba aplikace pro uživatele se řídí vývojovým diagramem uvedeným v předchozí kapitole. V této kapitole je kompletně popsána tvorba aplikace Využití automatické identifikace k analýze a vizualizaci provozu na železnici.

3.1 První myšlenka o aplikaci

Prvním návrhem aplikace je snaha zjednodušit zobrazení informací získaných pomocí modelové železnice. Modelová železnice využívá technologii RFID pro získávání informací o vagonech projíždějících senzory. Všechny potřebné údaje jsou uloženy v databázi a z ní je potřebné všechna data získat. Uživatel má možnosti si vybrat údaje, které chce získat z databáze a zobrazit si důležité informace. Při první myšlence je důležité rozhodnout, v jakém programovacím jazyku bude aplikace vytvořena.

3.2 Grafický design www stránky

Grafický design www stránky se provádí z důvodu názorné ukázky, jak bude v budoucnu stránka vypadat. Po grafické stránce musí být www stránka přehledná a nesmí obsahovat zbytečné prvky, a hlavně svítivé barvy, které by mohli uživatele spíše odlákat z dané stránky. Stránka musí být pro uživatelé provedena tak, aby ji byl uživatel schopný ovládat bez nějakých problémů. Písmo musí mít dostatečnou velikost, aby bylo čitelné.



Obr. 3.1 Grafické zobrazení hlavní www stránky

Zdroj: vlastní zpracování

Sekce ovládací prvky bude obsahovat hlavní ovládání, jako je možnost vyplnění čísla čidla nebo možnost přepnout na vyplnění data a času, a následně vyfiltrovat přesné a požadované informace.

Číslo čidla bude možné vybrat přesně v hodnotách od 1 do 30, jelikož více čidel databáze neobsahuje. Datum bude možné vybrat pomocí kalendáře nebo napsat ve správném formátu.

Vyplnění možnosti hodin, minut a sekund bude záležet čistě na uživateli. Je možné vyplnit pouze hodinu a vlaky se zobrazí přesně pro určité hodiny. Jakmile budou vyplňovány detailnější hodnoty, tak se zobrazí přesnější výstupy v tabulce.

Tabulka důležitých informacích obsahuje tři sloupce, které obsahují důležité informace.

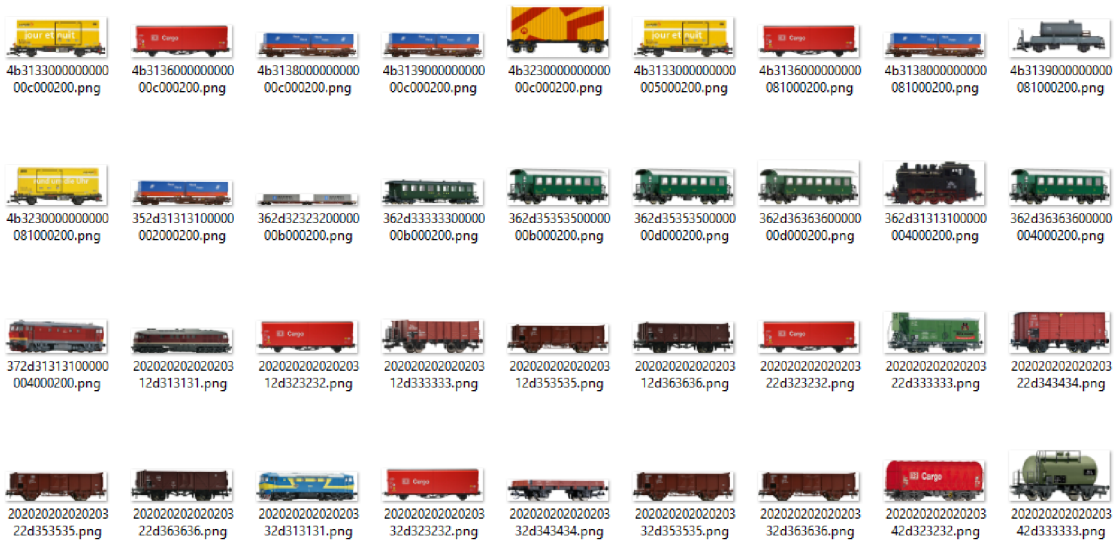
Sloupec ID vlaku zobrazuje přesné ID, pod kterým je daný vlak uložen v databázi. V tomto sloupci je možné kliknout na odkaz a zobrazit podrobnější informace o určitém vlaku.

Sloupec s datem zobrazuje přesné datum a čas, kdy daný vlak projel čidlem a byl zapsán do databáze.

Sloupec s číslem čidla je důležitý pro případné zobrazení vlaků pro určité čidlo. Tento sloupec je informativní a přehledný. Jakmile se vyplní číslo čidla v ovládací sekci, tak budou v tomto sloupci zobrazena pouze daná čidla a k nim příslušné vlaky.

Možnost přepínání stránek je zobrazena tehdy, pokud vyfiltrované vlaky obsahují více než 10 vlaků. Jakmile tato podmínka je splněna, tak se zobrazí možnost další stránky nebo případně předchozí stránky.

Tlačítko zobrazení všech vagonů po stisknutí zobrazí novou stránku, která obsahuje kompletní tabulku všech dostupných vagonů a lokomotiv a jejich příslušných informací.



Obr. 3.2 Zobrazení všech vagonů a lokomotiv

Zdroj: vlastní zpracování podle [6]

Na obr.3.2 je vidět vzhled vagonů, které mohou být zobrazeny pomocí aplikace. Všechny názvy obsahují potřebnou informací, pomocí které je jednoduché v databázi nalézt správný obrázek pro určitý vagon. Název obsahuje RFID kód, pod kterým je vagon nebo lokomotiva uložen v databázi u daného čidla.

3.3 Úprava a struktura databáze

Základní databáze modelové železnici využívá databázový systém MySQL Workbench. Proto bylo nutné zkopírovat všechna data a vložit je do databáze přes systém

PHPMyAdmin, který spravuje MySQL databáze a je s ním pracováno v této diplomové práci. Bylo nutné upravit základní vlastnosti PHPMyadmin, jako dílku trvání příkazu a množství dat v MB, aby byly spuštěny SQL příkazy, které obsahují a zároveň vytvářejí potřebné tabulky do připravené databáze. Tímto byla databáze připravena na základní používání.

V databázi bylo potřeba provést jednu úpravu. Tato úprava je důležitá pro zrychlení načítání vyhledávaných vlaků. V bakalářské práci nebyla tato úpravy provedena a načítání tak trvalo někdy i více než 10 sekund, když se měly načíst všechny vlaky. Změna spočívá v tom, že je potřebné přidat k určitým sloupcům v databázi indexy, které dokážou zrychlit vyhledávání v těchto určitých sloupcích. Indexy byly nastaveny celkem u třech sloupců, jak je vidět na obr.3.3.

#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Komentáře	Další	Operace
1	id	int(11)			Ne	Žádná		AUTO_INCREMENT	Změnit Odstranit Více
2	timestamp	timestamp			Ano	current_timestamp()			Změnit Odstranit Více
3	timestamp_processed	timestamp			Ano	NULL			Změnit Odstranit Více
4	type	int(11)			Ano	NULL			Změnit Odstranit Více
5	data	varchar(64)	utf8_general_ci		Ano	NULL			Změnit Odstranit Více
6	board	int(11)			Ano	NULL			Změnit Odstranit Více
7	dio_nr	int(11)			Ano	NULL			Změnit Odstranit Více
8	operator	int(11)			Ano	NULL			Změnit Odstranit Více
9	extint	int(11)			Ano	NULL			Změnit Odstranit Více
10	extstr	varchar(40)	utf8_general_ci		Ano	NULL			Změnit Odstranit Více
11	status	int(11)			Ano	NULL			Změnit Odstranit Více

Obr. 3.3 Úprava databáze a struktura tabulky events

Zdroj: vlastní zpracování

Přidání indexu je označené ikonou šedivého klíče, z toho vyplývá, že sloupce *timestamp*, *data* a *dio_nr* obsahují indexy a je tedy možné rychleji vyhledávat data v těchto sloupcích. Jak je vidět na obr.3.3, tak tabulka events obsahuje 11 různých sloupců, kdy každý obsahuje určité informace.

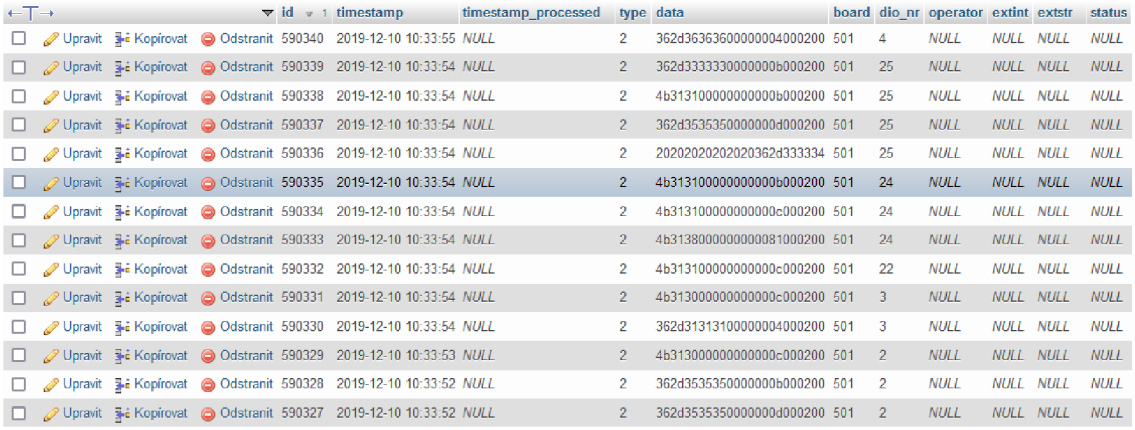
Položka *Id* je automatická, tzn. že údaje do tohoto sloupce se ukládají automaticky. A zároveň se vždy přičítá číslo 1 jakmile je přidán nový záznam do tabulky *events*.

Sloupec *timestamp* ukládá hodnotu data a času v určité chvíli, kdy je do tabulky zapsána nová hodnota.

Sloupec *data* obsahuje údaje získané z RFID Tagu, je to identifikátor vagonu případně lokomotivy.

Sloupec *dio_nr* obsahuje důležitou informaci o čísle čidla, z kterého byly všechny údaje uloženy.

Na obr.3.4 je patrné zapsání hodnot při průjezdu okolo RFID čtečky. Každý řádek obsahuje jeden záznam a jeden průjezd. V řádku je hodnota id, která se automaticky zvětšuje o číslo 1 při každém dalším uložení záznamu. Hodnota v sloupci data obsahuje změť číslic a písmen. Přesně jsou tyto údaje uloženy na RFID tagu na každém vagonu nebo lokomotivě.



	id	timestamp	timestamp_processed	type	data	board	dio_nr	operator	extint	extstr	status
Upravit Kopírovat Odstranit	590340	2019-12-10 10:33:55	NULL	2	362d3636360000000004000200	501	4	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590339	2019-12-10 10:33:54	NULL	2	362d3333330000000b000200	501	25	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590338	2019-12-10 10:33:54	NULL	2	4b3131000000000000b000200	501	25	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590337	2019-12-10 10:33:54	NULL	2	362d3535350000000d000200	501	25	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590336	2019-12-10 10:33:54	NULL	2	20202020202020362d333334	501	25	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590335	2019-12-10 10:33:54	NULL	2	4b3131000000000000b000200	501	24	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590334	2019-12-10 10:33:54	NULL	2	4b3131000000000000c000200	501	24	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590333	2019-12-10 10:33:54	NULL	2	4b3138000000000081000200	501	24	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590332	2019-12-10 10:33:54	NULL	2	4b3131000000000000c000200	501	22	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590331	2019-12-10 10:33:54	NULL	2	4b3130000000000000c000200	501	3	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590330	2019-12-10 10:33:54	NULL	2	362d3131310000000004000200	501	3	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590329	2019-12-10 10:33:53	NULL	2	4b3130000000000000c000200	501	2	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590328	2019-12-10 10:33:52	NULL	2	362d3535350000000b000200	501	2	NULL	NULL	NULL	NULL
Upravit Kopírovat Odstranit	590327	2019-12-10 10:33:52	NULL	2	362d3535350000000d000200	501	2	NULL	NULL	NULL	NULL

Obr. 3.4 Uložené záznamy v tabulce events

Zdroj: vlastní zpracování

3.4 Použité technologie na programování

Při programování aplikace je využit programovací jazyk PHP, databázový systém MySQL, skriptovací jazyk JavaScript, základní značkovací jazyk HTML a kaskádové styly CSS.

3.4.1 Programovací jazyk PHP

„PHP je programovací jazyk, který se používá převážně pro budování webů. Program PHP obvykle neběží na desktopovém počítači, který používá jen jedna osoba, ale typicky běží na nějakém webovém serveru a prostřednictvím nějakého webového prohlížeče k němu přistupuje spousta lidí.“ [7, s.15]

PHP je neustále vyvíjeno a aktuálně nese verze číslo 8.0.

3.4.2 Databázový systém MySQL [8]

Jedná se systém, který dokáže uschovávat ohromné množství dat a zůstat stále přehledný. Komunikace v tomto systému probíhá pomocí jazyk SQL, pomocí kterého je možné vypsat, upravit, uložit nebo smazat data, která potřebujeme. Systém obsahuje různé databáze, které se skládají z tabulek. Tyto tabulky obsahují data. Data jsou zarovnána do sloupců a řádků, kdy řádek většinou ukazuje jeden důležitý záznam. Každý sloupec v tabulce má své specifikum, je možné, aby se do toho sloupce ukládaly aktuální informace o datu a času, nebo nějaké forma textového řetězce, čísla a mnoho dalšího. Důležité je, že tento systém spolupracuje právě s programovacím jazykem PHP, pomocí kterého je napsána tato aplikace.

3.4.3 Skriptovací jazyk JS

Jazyk JavaScript je využíván pro programování různých funkcí u webových stránek. JavaScript nefunguje na stejném principu, jako je tomu u PHP, ale funguje na straně uživatele přímo ve webovém prohlížeči. Hodně se využívá například pro kontrolu formulářových polí, jestli obsahují správné informace. Tento formulář je JavaScript schopný vyhodnotit ještě před odesláním na server. Jakmile je ve formuláři nějaká chyba, tak není formulář neodeslán. V dnešní době se již dynamické webové stránky neobejdou bez fungujícího kód JavaScript.

3.4.4 Značkovací jazyk HTML

HTML je základní kámen všech webových stránek. Každý internetový prohlížeč zobrazuje právě webovou stránku v HTML, kdy tuto část kódu obdrží od nějakého webového serveru.

„Každá webová stránka se zakládá alespoň na minimálním množství kódu jazyka HTML; jinak by to nebyla webová stránka.“ [9, s.15]

Z uvedených informací vyplývá, že uživatel přijde do styku s HTML pouze v jeho konečné podobě na webové stránce. Důležité ale je, že samotné HTML nestačí, pokud tedy nemá webová stránka vypadat jen jako změť textu. Pro kompletní úpravu vzhledu webové stránky je ještě nutné přidat Kaskádové styly CSS, které dokážou upravit vše tak, jak je plánované.

3.4.5 Kaskádové styly CSS

CSS se využívá pro základní podobu webové stránky. V dnešní době se žádná webová stránka bez tohoto neobejde. Je možné pomocí CSS upravit velikost textu, zarovnání textu, pozadí celé webové stránky a mnoho dalšího. Možností je nepřeberné množství. CSS se vždy vkládá do webové stránky. Je možné psát styly přímo v daném HTML souboru nebo lepším způsobem je možnost připojit do HTML souboru externí soubor s kompletními styly pro danou webovou stránku. Těchto souborů může být ohromné množství a každý může upravovat jinou část webové stránky. Vždy je ale důležité dodržovat určitou hierarchii ve stylech. Vždy se kaskádové styly čtou od vrchu směrem dolu. To znamená, že jakmile bude například pro určitý text nastavena pevná velikost textu 15px a následně bude v daném souboru pro stejný text také nastavena pevná velikost textu 20px, tak webový prohlížeč vše vyhodnotí a nastaví pro daný textu ve finálním zobrazení velikost písma 20px. Toto samé platí i pro vkládání externích souborů do HTML, vždy je nutné dodržovat hierarchii čtení textu.

Kaskádové styly prochází neustálým vývojem. Aktuálně se nejvíc využívá verze CSS3, která přidává nepřeberné množství nových funkcí a možností pro programátory. Vše, co bylo v předchozích verzích CSS, je obsaženo v CSS3, takže se nemůže stát, že by v nové verzi nefungoval nějaký styl webové stránky.

CSS1

Jedná se úplně o první verzi kaskádových stylů. V prvotní verzi obsahuje pouze možnosti úpravy vlastností písma, barvy textu, vlastnosti blokových elementů. Je možné již využívat základní jednotky pro délku, ať už *procento*, *pixel* nebo tzv. *em*, což je základní výška písma, která se u každého prohlížeče liší.

CSS2

CSS2 již rozšiřuje možnosti kaskádových styly o mnoho nových funkcí. Například je možné nastavovat pozici, kdy je na výběr možnost relativní nebo absolutní, kdy absolutní pozice je vždy vázána na předchozí relativní pozici a dle ní se dokáže zarovnat. Samozřejmě je k tomu nutné použít i přímo vzdálenost o horního, pravého, spodního

nebo levého okraje. Jakmile budu vše takto splněno, tak je možné mít umístěn objekt, dle požadavku.

Dalšími možnostmi je přidání maximální a minimální šířky a výšky ať už webové stránky, tak i daného prvku.

Důležitým parametrem je *overflow*, který hlídá tzv přetékání textu. Dále parametr *visibility* pomocí kterého je možné nastavit viditelnost daného prvku.

CSS3

CSS3 je aktuálně poslední verzí kaskádových stylů. Je propojené s nejnovějším HTML5. CSS3 přidává další veliké množství nových možností, například animování, ať už textu nebo prvků, flexibilní blokové elementy, kdy je již jednodušší připravit zarovnání pro různý počet objektů. Obsahuje další vlastnosti písma nebo zajímavou funkci *drag`n`drop*, kdy se jedná o vlastnost tzv. možnosti přesunu daného prvku po webové stránce.

Výhody CSS

Mezi hlavní výhody CSS patří možnost upravit si vzhled přesně podle požadavků. Další nespornou výhodou je možnost propojit CSS a jazykem JavaScript, kdy je možné upravovat CSS pomocí přímých odkazů v JavaScriptu.

CSS je možné i tzv. cachovat, tzn. že je možné do webových prohlížečů ukládat určité vlastnosti prvků, které se při dalším načtení stránky již nemusí znovu stahovat z internetu a je tam možné šetřit objem stahovaných dat a také rychlost načítání webové stránky.

Pro koncové uživatele je zde i možnost upravit si vlastní CSS dle svých možností. Většina webových prohlížečů již tuto funkci podporuje.

V dnešní době je již možné pomocí jednoho souboru CSS připravit webovou stránku pro všechna dostupná zařízení, ať už se jedná o počítač, tablet nebo mobilní zařízení. Webové stránky se přizpůsobují velikosti daného zařízení. U počítačů se řeší rozlišení obrazovky monitoru, kdy aktuálně je nejčastější FullHD, tzn. rozlišení 1980x1080px, je také možné jej nazvat 1080p. U tabletů a mobilních zařízení je to s rozlišením

problémové, jelikož každé zařízení má své rozlišení dle výrobce. Tak se u CSS nastavují rozmezí šířek, pro která se má jaká vlastnost použít.

Nevýhody CSS

Mezi hlavní nevýhodu patří špatná podpora nejnovějšího CSS3 ve starších prohlížečích. Například ve webovém prohlížeči Internet Explorer 8 není možné používat CSS3, jelikož neumí tyto vlastnosti správně zobrazit. Pro tyto webové prohlížeče naštěstí končí kompletní podpora ze strany vývojářů, což platí i pro poslední verzi Internet Explorer 11, kdy se přestanou všechny tyto prohlížeče instalovat do zařízení. Jak tomu již bývá, většinou má člověk možnost volby, jaký prohlížeč bude používat, ať už se jedná o prohlížeč Mozilla Firefox, Google Chrome, Opera nebo Safari. Všechny tyto prohlížeče se automaticky aktualizují, díky tomu je vždy zaručena kompatibilita s CSS3 a jinými novými jazyky.

Ukázky CSS a vysvětlení

Zde je možné nalézt příklady a vysvětlení určitých vlastností CSS.

```
body {text-align: center; background-color: #ffffff; padding: 0; margin: 0; color: #1e1e1e;}
input[type="text"], input[type="date"], input[type="number"] {font-size: 17px;
-webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px; background-color: rgb(255,255,255);
 margin: 0 5px; padding: 7px 5px; width: 200px;}
input[type="number"]{width: 80px;}
#prepinac-cidlo-vlak {position: relative; margin: 30px auto; display: block; width: 80px;}
.moznosti {display: block; position: relative; margin: 0 auto; width: 370px;}
```

Obr. 3.5 Příklad CSS

Zdroj: vlastní zpracování

Na výše uvedeném obr.3.5 je patrné, jak vypadá kód v souboru CSS.

Text **body** značí celé tělo dokumentu HTML. Tzn. že celý dokument bude mít vlastnosti:

- **Text-align: center** – vycentrované zarovnání textu,
- **Background-color: #ffffff** – barva pozadí bude nastavena na bílou, barvy je možné psát v šestnáctkové soustavě nebo slovně v anglickém jazyku,
- **Padding: 0** – určuje vnitřní odstoupení od okraje prvku, v tomto případě se jedná o 0px, je možné zadávat jednotky délky,

- **Margin: 0** – funguje stejně jako padding, jen s tím rozdílem, že se jedná o vnější vzdálenost od okraje prvku,
- **Color: #1e1e1e** – barva textu, který je napsaný v HTML.

Text **input[type="text"], input[type="date"], input[type="number"]** značí tzv. inputy. Vždy se jedná buď o tlačítka, checkboxy nebo kolonky, do kterých se vyplňuje text. V tomto případě se jedná o tři kolonky s možností psát text s tím rozdílem, že každá má trochu jiné vlastnosti. Možnost **type** vždy určuje, o jakou vlastnost se bude jednat. **Type="text"** určuje jednoduchou kolonku na vyplňování jakéhokoliv textového řetězce.

Type="date" určuje současně s HTML5, že se jedná o kolonku s možností vyplnit datum. Po kliknutí na kolonku se otevře kalendář, ve kterém je možné vyhledat konkrétní datum. Samozřejmostí je i možnost napsat určité datum přímo do kolonky.

Type="number" určuje současně s HTML5, že se jedná o kolonku s možností vyplnit pouze čísla. Jakékoliv znaky jsou v této kolonce zakázány. Je možné nastavit i rozmezí čísel od minimální hodnoty po maximální. Vyšší nebo nižší číslo nepovolí nastavení zapsat. Případně je možnost naklikat pomocí přidaných šipek, které mají vlastnost přidávat nebo odebírat hodnotu čísla o 1.

U těchto inputů jsou vidět tyto vlastnosti:

- **Font-size: 17px** – to znamená, že text obsažený v inputu bude mít velikost 17px. Optimální velikost klasické textu na webové stránce je přibližně 15px, ale samozřejmě záleží na rozlišení obrazovky monitoru. Jakmile bude 23palcový monitor s rozlišením UltraHD (rozlišení 3840x2160px), tak je velice pravděpodobné, že text o velikosti 15px bude nedostačují. Ale jakmile bude 23palcový monitor s rozlišení FullHD, tak je tato velikost optimální a text je čitelný,
- **Border-radius: 5px** – to znamená jednoduché zaoblení rohů u prvku. Rohy budou zakulaceny v délce 5px. Vlastnost border-radius je nutné psát ještě s dvěma různými prefixy, tzn. v tomto případě musí border-radius začínat spojením -moz- nebo -webkit-. Oba tyto prefixy jsou potřebné jako informace pro určité webové prohlížeče, aby správně použili danou vlastnost. Tyto prefixy je nutné používat pouze u určitých vlastností, ne u všech,

- **Background-color: rgb(255,255,255)** – další možností zápisu jakékoliv barvy je tzv. určení přímo smícháním RGB barev. Číselnými hodnotami je možné určit přesné poměry barev červená (R – Red), zelené (G – Green) a modré (B – Blue). Smícháním všech tří barev v maximálním množství, hodnota 255, se dostane vždy bílá barva. Jakmile se smíchá tzv. minimální množství, hodnota 0, tak se dostane vždy černá barva. Takto se dají různé míchat barvy a dostat tak jakýkoliv odstín. Ještě je možné u rgb přidat jednu hodnotu, a to rgba, kdy to bude vypadat rgba(255,255,255,0.8). Hodnota **a** určuje neprůhlednost dané barvy. Určuje se v desetinných hodnotách, kdy číslo 1 je 100% neprůhlednost a například číslo 0.5 je 50% neprůhlednost. Minimální hodnota je číslo 0,
- **Width: 200px** – tato vlastnost určuje šířku prvku. Je možné ji uvádět v pixelech nebo v jiných délkových jednotkách.

Text **#prepinac-cidlo-vlak** je další z možností, jak zapsat vlastnosti pro určitý prvek. V tomto případě se jedná o selektor *ID* s názvem *prepinac-cidlo-vlak*. Tento selektor je identifikátorem v HTML kódu dané webové stránky. Tento identifikátor je vždy na jedné stránce jedinečný. V HTML je možné jej najít zapsán jako *id="prepinac-cidlo-vlak"*.

- **Position: relative –** jedná o pozicování daného prvku. Relativní pozice určuje, že se jedná o prvek, který je na pevné pozici a je relativní pro dceřiné prvky, které obsahují a mají vlastnost **position: absolute**,
- **Margin:** 30px auto – tato vlastnost určuje vnější vzdálenost od okraje ve velikosti 30px. Jedná se ale jen o vzdálenost od horního a spodního okraje. Hodnota auto určuje automatické nastavení vzdálenost od levého a pravého okraje. Tato vlastnost se využívá často pro vycentrování prvků. Kompletní zápis vlastnosti margin je **margin: 0px 0px 0px 0px** s tím, že každá z těchto hodnot je pro jiný okraj prvku. Hodnoty se berou dle seřazení - horní okraj, pravý okraj, spodní okraj, levý okraj. Je možné tento zápis zkrátit, jak se vidět v předchozím zápisu. Ale je důležité, že jakmile chybí nějaká hodnota, tak se automaticky přebírá hodnota protilehlé strany. V případě zápisu pouze jedné hodnoty se tyto hodnoty použijí pro všechny okraje,
- **Display: block** – určuje důležité zobrazení, jestli bude prvek zarovnán do celého bloku nebo v případě použití hodnoty inline bude prvek zobrazen v jedné linii. Platí to pro více prvku zapsaných vedle sebe.

Text **.moznosti** je poslední základní možností, jak zapisovat vlastnosti pro určitý prvek. Tentokrát se jedná o tzv. třídu neboli *Class*. Selektor se u prvku v HTML zapisuje jako *class="moznosti"* a je možné jej použít v jedné stránce nekonečně mnohokrát, není totiž jedinečný. Z tohoto vyplývá, že pro jedinečné prvky v HTML je dobré využívat selektor ID, ale pro prvky, které se mají opakovat v HTML je dobré využívat selektor Class.

CSS obsahuje ohromné množství vlastností, které se dají využívat pro jakékoliv úpravy webových stránek. Další vlastnosti jsou zobrazeny v následujících kapitolách, které se zabývají kompletní tvorbou kódu pro aplikaci.

3.4.6 Modelová železnice

Modelová železnice je využita pro snímání informací o pohybu vlaku po železniční dráze. Jakmile vagon nebo lokomotiva projede okolo RFID čtečky, tak je načten RFID Tag a jeho hodnota je zapsána do databáze, ze které tato diplomová práce vychází. Kompletní popis a fungování modelové železnice je možné nalézt v bakalářské práci Aplikace standardů GS1 v modelu železniční dopravy od autorky Petry Setlíkové z roku 2017 [10].

3.5 Tvorba webové aplikace

Má-li uživatel hotový grafický návrh webové aplikace a dobré znalosti potřebných technologií, je možné zahájit kompletní tvorbu webové aplikace. Bez těchto znalostí není možné začít vytvářet jakoukoliv aplikaci.

Při čtení této kapitoly je důležité vzít v potaz, že autor není profesionálním programátorem, a tak je možné, že některé části kódu budou zmatené nebo zbytečně rozsáhlé. Důležité je, že vše funguje podle představ.

3.5.1 Tvorba hlavní stránky index.php

Hlavní stránka webové aplikace je vždy první stránka, která se uživateli objeví při prvotním načtení, pokud přistupuje na webovou stránku bez nějaké další cesty například https://www.vslg.cz.

Jak již název a koncovka napovídá, tak se jedná o hlavní stránku v programovacím jazyku PHP. Samozřejmě název hlavní stránky je možný jiný, například *default.php*, *homepage.php*. Vše záleží na nastavení softwarové balíčku, který má na starosti dekódování jazyka PHP.

```
try {
  $db = new PDO ('mysql:host=localhost;dbname=vslg_diplomka','vslg','vslgprerov');
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $db->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
```

Obr. 3.6 Připojení k databázi

Zdroj: vlastní zpracování

Na obr.3.6 je možné vidět pokus o připojení k databázi. Důležité při připojení je určit správně server, na které se databáze nachází. V tomto případě se jedná o server *localhost*, jelikož je server umístěn na stejném počítači jako je softwarový balíček s PHP. Je možné zadávat i IP adresy, které databázový server identifikují a připojí se k němu.

Dále je důležité specifikovat název databáze, ke které je potřeba se připojit. Databázi je potřeba mít připravenou již před prvotním spuštěním aplikace, aby se mohla aplikace k ní již připojit. Nemusí ještě obsahovat tabulky, ty se mohou vytvořit v průběhu tvorby aplikace. Nebo je může aplikace vytvořit sama přesně podle potřebných kritérií.

Dalším krokem v pokusu o připojení je nastavení uživatelského jména a hesla s přístupem k databázi. V tomto případě se jedná o uživatelské jméno *vslg* a heslo *vslgprerov*. Tyto údaje je potřeba mít řádně uschované, a hlavně je někomu nesdělovat. V rámci této diplomové práce je možné tento fakt pominout, jelikož se jedná o informace, které nepatří k žádnému internímu objektu. Tyto údaje jsou použity jen v rámci diplomové práce a je možné je kdykoliv změnit v prostřední MySQL.

Všechny údaje uvedené výše jsou uloženy do proměnné *$db*, které je potřeba ještě nastavit správné atributy, jako například zobrazení chyb.

Jestli nastane při připojení nějaká chyba, tak se provede kód na obr.3.7 a bude vypsána chyba, která to způsobila. Zapsání chyby je uloženo do proměnné $e a ta se následně získá příkazem *getMessage()* a bude vypsána za text „nelze se připojit k DB“.

```
} catch (PDOException $e) {
  print "nelze se připojit k DB: " . $e->getMessage();
  exit();
}
```

Obr. 3.7 Chyba připojení k databázi

Zdroj: vlastní zpracování

Pokud se podaří připojit správně k databázi, tak je možné pokračovat ve čtení PHP kódu.

Další částí kódu je potřebné pro správné fungování této aplikace zjistit, zda URL adresa obsahuje určité části textu. Jedná se o URL **http://localhost/dp/index.php?date1=2015-08-04&date2=2019-08-12&page=2**:

- **Localhost** – jedná se o doménu, na které je web uložen. V tomto případě se jedná o server na stejném počítači, jak je k němu přistupováno.,
- **/dp/** – jedná se o podložku, kde jsou uloženy všechny soubory pro aplikaci,
- **Index.php** – je to soubor s hlavní stránkou, která se načte při prvotním zadání webové stránky,
- **?date1=2015-08-04&date2=2019-08-12&page=2** – určuje důležité proměnné, do kterých jsou postupně ukládány hodnoty *2015-08-04*, *2019-08-12* a *2*. Tyto hodnoty jsou ukládány pro pozdější pracování s webovou stránkou,

S výše uvedenými údaji je potřebné prakticky pracovat, jak je tomu napsán kód uvedený na obr.3.8.

```php
$page = 1;
if (isset($_GET['page'])) {
  $page = filter_var($_GET['page'], FILTER_SANITIZE_NUMBER_INT);
}
if (!empty($_POST['time'])) {
$time = htmlspecialchars($_POST['time']);
}
if (empty($_POST['time'])) {
$time = "";
}
if (isset($_GET['time'])) {
  $time = filter_var($_GET['time'], FILTER_SANITIZE_STRING);
}
if (!empty($_POST['cidlo'])) {
$cidlo = htmlspecialchars($_POST['cidlo']);
}
if (empty($_POST['cidlo'])) {
$cidlo = "";
}
if (isset($_GET['cidlo'])) {
  $cidlo = filter_var($_GET['cidlo'], FILTER_SANITIZE_STRING);
}
if (!empty($_POST['date'])) {
$date = htmlspecialchars($_POST['date']);
}
if (empty($_POST['date'])) {
$date = "";
}
if (isset($_GET['date'])) {
  $date = filter_var($_GET['date'], FILTER_SANITIZE_STRING);
}
if (!empty($_POST['date1'])) {
$date1 = htmlspecialchars($_POST['date1']);
}
if (empty($_POST['date1'])) {
$date1 = "";
}
if (isset($_GET['date1'])) {
  $date1 = filter_var($_GET['date1'], FILTER_SANITIZE_STRING);
}
if (!empty($_POST['date2'])) {
$date2 = htmlspecialchars($_POST['date2']);
}
if (empty($_POST['date2'])) {
$date2 = "";
}
if (isset($_GET['date2'])) {
  $date2 = filter_var($_GET['date2'], FILTER_SANITIZE_STRING);
}
```

Obr. 3.8 Kompletní kontrola hodnot v URL adrese

Zdroj: vlastní zpracování

Všechny podmínky začínají textem *if*, kde v kulaté závorce je uvedena podmínka, s jakou má daná podmínka *if* pracovat, a kde ve složených závorkách je funkce nebo příkaz, který

se má provést, jakmile je daná podmínka splněna. Pokud není podmínka splněna, tak se daný obsah složené závorky neprovede.

První podmínka *if (isset($_GET[`page`]))* dokáže získat z URL adresy hodnotu uloženou pod proměnnou *page*. V tomto případě se jedná o hodnotu číslo *2*, která odkazuje při listování stránkami na stránku číslo *2*. Jestli tato podmínka obsahuje hodnotu, tak ji vložený příkaz tzv. vyfiltruje a příkazem *FILTER_SANITIZE_NUMBER_INT* převede na číselnou hodnotu, která je potřebná pro prohledávání databáze a další příkazy. Tato podmínka se v průběhu kódu opakuje, ale s různými proměnnými, aby bylo možné vždy získat potřebné informace pro fungování webové aplikace s tím rozdílem, že se hodnoty proměnných filtrují příkazem *FILTER_SANITIZE_STRING*. Do proměnné *time* je ukládán vyplněný čas, který uživatel hledal. Do proměnné *date*, *date1* a *date2* je ukládáno datum, pro které chce uživatel nalézt vlaky. A do proměnné *cidlo* je uloženo číslo čidla, pro které se mají zobrazit všechny vlaky, které jím projely.

Druhá podmínka *if (!empty($_POST[`time`]))* kontroluje vyplnění formulářového pole s název *time*. Příkaz *!empty* má za úkol zjistit, zda je prvek neprázdný, čili obsahuje nějakou hodnotu, jakmile je příkaz pravdivý, tak provede vložený kód. Do uvedené proměnné *$time* se uloží údaje z odeslaného formulářového pole a jsou zároveň převedeny všechny speciální znaky na normální text. A je tím zabráněno v odeslání speciálních znaků a poškození databáze.

Třetí podmínka *if (empty($_POST[`time`]))* kontroluje také vyplnění formulářového pole *time*, ale s tím rozdílem, že jakmile je prázdný, tak uloží do proměnné *$time* prázdný textový řetězec.

Druhá a třetí podmínka je prováděna i pro proměnné *cidlo*, *date*, *date1* a *date2*. Ve všech případech je postup stejný jako u proměnné *time*.

Následující kód v obr.3.9 obsahuje příkazy pro přípravu počtu záznamů dat v databázi.

```
$per_page = 10;
if ((!empty($date)) AND (empty($time))) {
$cas = $date . '%';
$sqlcount = "SELECT COUNT(*) as total_records from events WHERE type LIKE 2 and timestamp LIKE :cas and
(data LIKE '4b313000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')";
$stmt = $db->prepare($sqlcount);
$stmt->execute(['cas'=>$cas]);
}
elseif ((!empty($date)) AND (!empty($time))) {
$cas = $date . " " . $time . '%';
$sqlcount = "SELECT COUNT(*) as total_records from events WHERE type LIKE 2 and timestamp LIKE :cas and
(data LIKE '4b313000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')";
$stmt = $db->prepare($sqlcount);
$stmt->execute(['cas'=>$cas]);
}
elseif ((!empty($date1)) AND (!empty($date2))){
$cas1 = $date1 . ' 00:00:00';
$cas2 = $date2 . ' 23:59:59';
$sqlcount = "SELECT COUNT(*) as total_records from events WHERE type LIKE 2 and timestamp>=:casj AND timestamp<=:casd and
(data LIKE '4b313000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')";
$stmt = $db->prepare($sqlcount);
$stmt->execute(['casj'=>$cas1, 'casd'=>$cas2]);
}
elseif (!empty($cidlo)) {
$sqlcount = "SELECT COUNT(*) as total_records from events WHERE type LIKE 2 and dio_nr LIKE :cidlo and
(data LIKE '4b313000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')";
$stmt = $db->prepare($sqlcount);
$stmt->execute(['cidlo'=>$cidlo]);
}
else {
$sqlcount = "SELECT COUNT(*) as total_records from events WHERE type LIKE 2 and
(data LIKE '4b313000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')";
$stmt = $db->prepare($sqlcount);
$stmt->execute();
}
$row = $stmt->fetch();
$total_records = $row['total_records'];

$total_pages = ceil($total_records / $per_page);

$offset = ($page-1) * $per_page;
```

Obr. 3.9 Příprava dat z databáze a zjištění daného počtu záznamů

Zdroj: vlastní zpracování

Všechny uvedené podmínky mají za úkol kontrolovat vyplnění uvedených proměnných. Například *!empty($date)* kontroluje, zda je v proměnné *$date* uložená hodnota z předchozí kódu. A *empty($time)* kontroluje, zda je proměnná *$time* prázdná. Jestli jsou všechny tyto podmínky splněny, tak se provedou opět příkazy, které jsou v dané podmínce vloženy.

V první podmínce *if* je nutné uložit do proměnné *$cas* hodnotu *$date* a zároveň přidat možnost, aby tato hodnota nemusela řešit textový řetězec za proměnou *$date*. Hodnota *%* určuje jakýkoliv text. V tomto případě je v hodnotě *$cas* uložen datum, který byl získán po odeslání formuláře a který je použit pro výběr dat z databáze. Do proměnné *$sqlcount* se ukládá počet hodnot, které by mohly být vypsány z tabulky events a podmínkou *WHERE*, kde je právě porovnání časového údaje s údaji v databázi. Zároveň je ještě pohlídáno, aby byly hledáno pouze lokomotivy, které mají určitou hodnotu uloženou ve sloupci *data*.

V druhé podmínce *elseif* je nutné uložit do proměnné *$cas* hodnotu *$date* a *$time*. Je to z důvodu vyplnění dvou formulářových polí, které jsou požadovány pro filtrování vlaků. Jakmile je toto splněno, tak se spustí stejný příkaz, jako je tomu u první podmínky.

Ve třetí podmínce *elseif* je nutné uložit do proměnné *$cas1* a *$cas2* hodnoty *$date1* a *$date2*, kdy každá z nich určuje rozmezí dat, pro které se budou vlaky hledat. *$date1* je počáteční hodnota, a proto má přidán čas *00:00:00*. A *$date2* je konečná hodnota a která má přidán čas *23:59:59*. Tímto je zaručeno, že příkaz bude vycházet pouze z vlaků, které jsou od začátku počátečního dne a do konce konečného dne rozmezí. V proměnné *$sqlcount* je navíc přidána podmínka, která právě určuje počáteční čas a konečný čas. Jedná se o *timestamp>=:casj* a *timestamp<=casd*.

Ve čtvrté podmínce *elseif* se žádné proměnné pro čas neobjevují, jelikož v tomto případě se zobrazují pouze vlaky, které jsou uloženy pod číslem uvedeného čidla. V proměnné *$sqlcount* je přidána podmínka pro zjištění čísla vlaků *dio_nr LIKE :cidlo*. Tímto je zaručeno, že se budou brát v potaz pouze vlaky, které mají stejné číslo čidla.

V poslední části *else*, která bude provedena, pokud nebude vyhovovat žádná z podmínek, budou počítány všechny vlaky, které jsou uloženy v databázi v tabulce events a zároveň splňují vnitřní podmínku.

Jakmile je vybrána nějaká z výše uvedených podmínek, tak do proměnné *$row* je uložena hodnota počtu vlaků, která byla připravena v předchozích příkazech. Uložení a použití této hodnoty je potřebné pro stránkování tabulek s výpisy vlaků.

```
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="generator" content="PSPad editor, www.pspad.com">
  <title>Soupis vlaků</title>
  <link REL=STYLESHEET type="text/css" href="styles/styles.css">
  </head>
  <body>
  <div class="centered">
    <div class="item-1">
      <h1>Vítejte na stránce <br /> „<i><u>Využití automatické identifikace k analýze a vizualizaci provozu na železnici</u></i>“</h1>
      <div class="vyber-cidlo-vlak">
        <h2>Vyber jestli chceš vyhledávat podle čidla nebo data průjezdu vlaku!</h2>
        <div class="moznosti">
          <b class="prvni-moznost">Vyber podle čidla</b>
          <div id="prepinac-cidlo-vlak" class="<?php if(((!empty($date1)) AND (!empty($date2))) OR (!empty($date))){echo "active";}?>">
            <i class="indicator"></i>
          </div>
          <b class="druha-moznost">Vyber datum</b>
        </div>
      </div>
      <div id="vyber-cidlo" class="<?php if((empty($date1)) AND (empty($date2)) AND (empty($date))){echo "active";}?>">
        <div id="form-cidlo" class="formular">
          <form method="get" action="<?php echo $_SERVER["PHP_SELF"]?>">
            <label for="dat">Zadej číslo čidla 1-30:</label>
            <input type="number" id="dat" name="cidlo" min="1" max="30" value="<?php echo $cidlo; ?>" />
            <button type="submit">FILTROVAT</button>
            <input type="reset" name="reset" value="RESET" onclick="window.location.href = 'index.php'">
          </form>
        </div>
      </div>
```

Obr. 3.10 HTML kód pro formuláře

Zdroj: vlastní zpracování

Na obr.3.10 je patrné, že aktuálně se bude jednat o HTML. HTML je důležité v rámci programování webových stránek. Řádek *<link REL=STYLESHEET type="text/css" href="styles/styles.css">* připojuje externí CSS soubor, který obsahuje kompletní styly, aby webová stránka vypadala podle představ. Tato část textu se musí vložit mezi tagy *head* neboli hlavičky. Hlavička ještě obsahuje tag *title*, který informuje webový prohlížeč u názvu stránky, který je potřeba zobrazit v panelu webového prohlížeče.

Tag body je začátkem těla HTML webové stránky, tzv. tělo dokumentu. Toto tělo obsahuje kompletní HTML rozvržení stránky a může obsahovat i JavaScript případně i inline CSS, tzn. CSS přímo u prvku HTML.

Důležité bloky kódu jsou rozděleny do tagu *div*, který má vždy začátek a konec jako většina párových tagů v HTML. Je možné vkládat další div do divu a tvořit tak strukturu stránky. U každého tagu *div* je možné vidět vlastnost *class* případně vlastnost *id*. Jak již bylo vysvětleno v obr.3.5, jedná se o vlastnost, pomocí které se může identifikovat prvek v CSS případně v JavaScript. V případě CSS se tomuto prvku vždy nastaví nějaký vzhled. V případě JavaScript se tomuto prvku nastaví nějaké pokročilejší funkce, jako je například změna písma po kliknutí na určitý prvek, případně otevření popup okna a další.

Tento HTML kód obsahuje i prvky PHP, které mají vždy určitou roli. V prvním případě umístění kódu *<?php* je patrné, že obsahuje podmínku, která požaduje zjištění nulové hodnoty u třech různých proměnných. Jakmile je podmínka splněna, tak přidá text *active*

do vlastnosti *class*. Tímto je možné upravovat HTML, případně vzhled webové stránky po odeslání správných proměnných.

Text *<form method="get" action="<?php $_SERVER["PHP_SELF"]?>" />* určuje formulář, který je odesílán pomocí metody *GET*, která má za úkolu vložit patřičné údaje do URL adresy. Vlastnost *action* má za úkol provést všechny funkce, které se nacházejí v aktuální souboru *index.php*. Kdyby bylo potřebné, je možné do vlastnosti *action* napsat odkaz na nějaký soubor, který má za úkol provést patřičné příkazy.

V textu *<input type="reset" name="reset" value="RESET" onclick="window.location.href = `index.php`">* je nová vlastnost. Je to vlastnost *onclick*, do které je možné přidat JavaScript kód, který v tomto případě má za úkol znovu načtení stránky *index.php*. Je to z důvodu kompletního vyresetování filtru na webové stránce, které byly spuštěny při hledání určitých vlaků.

```
if ((!empty($date)) AND (empty($time))) {
  $sql = "SELECT id, timestamp, data, type, dio_nr from events WHERE type LIKE 2 and timestamp LIKE :cas and
  (data LIKE '4b3130000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')
  ORDER BY id DESC LIMIT :offset, :per_page";
  $stmt = $db->prepare($sql);
  $stmt->execute(['cas'=>$cas, 'offset'=>$offset, 'per_page'=>$per_page]);
}
elseif ((!empty($date)) AND (!empty($time))) {
  $sql = "SELECT id, timestamp, data, type, dio_nr from events WHERE type LIKE 2 and timestamp LIKE :cas and
  (data LIKE '4b3130000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')
  ORDER BY id DESC LIMIT :offset, :per_page";
  $stmt = $db->prepare($sql);
  $stmt->execute(['cas'=>$cas, 'offset'=>$offset, 'per_page'=>$per_page]);
}
elseif ((!empty($date1)) AND (!empty($date2))){
  $sql = "SELECT id, timestamp, data, type, dio_nr FROM events WHERE type LIKE 2 AND timestamp>=:casj AND timestamp<=:casd
  and (data LIKE '4b3130000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')
  ORDER BY id DESC LIMIT :offset, :per_page";
  $stmt = $db->prepare($sql);
  $stmt->execute(['casj'=>$cas1, 'casd'=>$cas2, 'offset'=>$offset, 'per_page'=>$per_page]);
}
elseif (!empty($cidlo)) {
  $sql = "SELECT id, timestamp, data, type, dio_nr FROM events WHERE type LIKE 2 AND dio_nr LIKE :cidlo and
  (data LIKE '4b3130000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')
  ORDER BY id DESC LIMIT :offset, :per_page";
  $stmt = $db->prepare($sql);
  $stmt->execute(['cidlo'=>$cidlo, 'offset'=>$offset, 'per_page'=>$per_page]);
  }
else {
  $sql = "SELECT id, timestamp, data, type, dio_nr FROM events WHERE type LIKE 2 and
  (data LIKE '4b3130000000000000c000200' or data LIKE '362d31313100000004000200' or data LIKE '372d31313100000004000200')
  ORDER BY id DESC LIMIT :offset, :per_page";
  $stmt = $db->prepare($sql);
  $stmt->execute(['offset'=>$offset, 'per_page'=>$per_page]);
}
```

Obr. 3.11 Struktura pro přípravu výpisu dat z databáze

Zdroj: vlastní zpracování

Všechny aktuální podmínky jsou velice podobné podmínkám, které měly za úkol počítat prvky, jež splňují podmínku v proměnné *$sqlcount*. Aktuálně je to ale příprava všech prvků, které splňují určitou podmínku v proměnné *$sql*. Prvky bude možné brzy vypsat

do tabulky. Příkazy *SELECT* zde neobsahují položku *Count*, která měla za úkol na obr.3.9 výpočet všech použitelných prvků.

```
echo "<table border='1' align='center'>";
echo "<tr><th>ID vlaku</th><th>Čas projetí čidlem</th><th>Číslo čidla</th></tr>";
while ( ($row = $stmt->fetch(PDO::FETCH_ASSOC) ) !== false) {
  echo "<tr>";
  echo "<td><a href='vlakinfo.php?id=".$row['id']."' onclick=\"window.open(this.href, 'mywin',
'left=100,top=100,width=1100,height=500,toolbar=1,resizable=0'); return false;\">klikni pro zobrazení <br />".$row['id']."</a></td>";
  echo "<td><strong>".$row['timestamp']."</strong></td>";
  echo "<td>".$row['dio_nr']."</td>";
  echo "</tr>";
}

echo "</table>";
```

Obr. 3.12 Vypsání prvků do tabulky

Zdroj: vlastní zpracování

V aktuální části kódu se provede kompletní vypsání všech prvků, které splnily všechny předchozí podmínky. Vždy se vytvoří tabulky pomocí příkazu *echo "<table border=`1` align=`center`>"*, který přidá tabulku do HTML dokumentu. Tag table je nutné ukončit, jelikož je to párový tag. A proto je zde příkaz *echo "</table>"*. Nejdůležitější částí v tomto kódu je cyklus *while*. Tento cyklus má za úkol neustále opakovat cykly, dokud podmínka platí. V tomto případě se jedná o podmínku, která postupně vypisuje prvky získané do proměnné *$sql* a následné úpravy do proměnné *$stmt*. Proměnná *$row* je zde použita pro snadnější získávání potřebných sloupců. Zde jsou získány sloupce *id*, *timestamp* a *dio_nr*. Jsou to sloupce obsahující jedinečné id číslo, čas zapsání vlaku a číslo čidla, které informaci o vlaku přijalo. U řádku, kdy je vypisováno id, je přidána funkce, která má za úkol zobrazení nového okna *vlakinfo.php*. Nové okno zobrazí detailnější informace o daném vlaku. Hlavní identifikátor toho zobrazení je sloupec *id*.

```
echo "<table class='padding'>";

echo "<tr>";

if (($page-1 >= 1) AND ($date != "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date=".($date)."&page=".($page - 1)."'>předchozí stránka</a></td>";
}
if (($page+1 <= $total_pages) AND ($date != "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date=".($date)."&page=".($page + 1)."'>další stránka</a></td>";
}

if (($page-1 >= 1) AND ($date != "") AND ($time != "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date=".($date)."&time=".($time)."&page=".($page - 1)."'>předchozí stránka</a></td>";
}
if (($page+1 <= $total_pages) AND ($date != "") AND ($time != "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date=".($date)."&time=".($time)."&page=".($page + 1)."'>další stránka</a></td>";
}

if (($page-1 >= 1) AND ($date == "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?page=".($page - 1)."'>předchozí stránka</a></td>";
}
if (($page+1 <= $total_pages) AND ($date == "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?page=".($page + 1)."'>další stránka</a></td>";
}

if (($page-1 >= 1) AND ($date == "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo != "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?cidlo=".($cidlo)."&page=".($page - 1)."'>předchozí stránka</a></td>";
}
if (($page+1 <= $total_pages) AND ($date == "") AND ($time == "") AND ($date1 == "") AND ($date2 == "") AND ($cidlo != "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?cidlo=".($cidlo)."&page=".($page + 1)."'>další stránka</a></td>";
}

if (($page-1 >= 1) AND ($date == "") AND ($time == "") AND ($date1 != "") AND ($date2 != "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date1=".($date1)."&date2=".($date2)."&page=".($page - 1)."'>předchozí stránka</a></td>";
}
if (($page+1 <= $total_pages) AND ($date == "") AND ($time == "") AND ($date1 != "") AND ($date2 != "") AND ($cidlo == "")) {
  echo "<td><a href='".$_SERVER['PHP_SELF']."?date1=".($date1)."&date2=".($date2)."&page=".($page + 1)."'>další stránka</a></td>";
}

echo "</tr>";

echo "</table>";
```

Obr. 3.13 Připravené stránkování pro zobrazení více prvků

Zdroj: vlastní zpracování

Kód pro stránkování je jednoduchý, jelikož obsahuje pouze samé podmínky, které musí kontrolovat, aby se správná tlačítka s odkazy zobrazila podle vyhledávaných vlaků. V případě vyhledávání podle čidla je potřebné, aby neustále zůstávala v URL adrese hodnota čidla, která bylo požadována. V případě vyhledávání podle data, času nebo rozmezí dat je nutné, aby vždy URL adresa obsahovala určité datum nebo čas. Všechny tyto hodnoty jsou kontrolovány pomocí podmínek v obr.3.8. Jakmile by nějaká hodnota chyběla, tak by se mohlo stát, že webová stránka špatně vyhodnotí údaje a zobrazí špatné výsledky filtrace. Celé toto stránkování je uzavřené v tabulce pro jednoduché zobrazení tlačítek *Další stránka* a *Předchozí stránka*.

```
?>
    <div class="zobrazvlaky">
      <span>Zde je možné si zobrazit všechny dostupné vagony</span>
      <a href="zobrazvagony.php" target="_blank">Zobrazit všechny vlaky</a>
    </div>
  </div>
  </body>
  <script>
    const prepnout_cidlo = document.getElementById("prepinac-cidlo-vlak");
    const vyber_vlak = document.getElementById("vyber-vlak");
    const vyber_cidlo = document.getElementById("vyber-cidlo");
    const form1 = document.getElementById("formular1");
    const form2 = document.getElementById("formular2");
    const prepnout = document.getElementById("prepinac-vlak");
    prepnout_cidlo.onclick = function() {
      prepnout_cidlo.classList.toggle("active");
      vyber_vlak.classList.toggle("active");
      vyber_cidlo.classList.toggle("active");
    };
    prepnout.onclick = function() {
      prepnout.classList.toggle("active");
      form1.classList.toggle("active");
      form2.classList.toggle("active");
    };
  </script>
</html>
```

Obr. 3.14 Možnost zobrazit všechny vlaky a JavaScript v HTML

Zdroj: vlastní zpracování

V bloku div je patrné, že obsahuje odkaz na soubor *zobrazvagony.php* s vlastností *target="_blank"*, která má za úkol zobrazení nového okna po kliknutí na odkaz. Pomocí tohoto odkazu se zobrazí nové okno, které obsahuje kompletní výpis všech dostupných vagonů a lokomotiv.

V části <*script*> je vložen JavaScript kód, který spouští funkci přepínacího tlačítka. Pomocí tohoto tlačítka je možné vybrat správný formulář pro filtrování vlaků, ať už podle čísla čidla, data a času nebo podle rozmezí dat. V kódu je patrné ukládání hodnot do konstant, kdy tato hodnota získává určitý prvek pomocí příkazu *document.getElementById*. Pomocí toho příkazu bude nalezen prvek, který obsahuje vlastnost *Id* se jménem uvedeným v závorce, například *prepinac-cidlo-vlak*. Tento kód obsahuje dvě funkce, kdy obě jsou vázány na kliknutí na patřičný prvek v HTML, a tím přepnutí vlastnosti *class* u správných prvků. Přepínání vlastnosti *class* funguje na principu, že jakmile je určitá *class* vypsána v HTML, tak ji odebere. Jakmile vypsána v HTML není, tak bude přidána. Toto vše se provádí u vybraných prvků.

Tímto kódem je ukončen soubor *index.php* a je tedy možné jej plně využívat a filtrovat vlaky dle libosti. Pro správné fungování celé aplikace je nutné ještě připravit dva soubory. Jsou to soubory *zobrazvagony.php* a *vlakinfo.php*.

3.5.2 Tvorba vedlejší stránky zobrazvagony.php

Vedlejší stránka *zobrazvagony.php* má za úkol zobrazovat všechny možné vagony, které je možné použít na modelové železnici a které jsou uloženy v databázi.

```
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="generator" content="PSPad editor, www.pspad.com">
  <title>Seznam dostupných vagonů</title>
  <link REL=STYLESHEET type="text/css" href="styles/styles.css">
  </head>
  <body>
  <div class="centered">
    <div class="item-1">
      <h1>Zde je seznam a přehled všech dostupných vagonů a lokomotiv.</h1>
<?php
try {
  $db = new PDO ('mysql:host=localhost;dbname=vslg_diplomka','vslg','vslgprerov');
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $db->exec("SET NAMES 'utf8';");
  $sqlvlak = "SELECT * FROM wagons WHERE rfid not like 'ice_l' and rfid not like 'ice_c'
  and rfid not like 'ice_r' and rfid not like 'rychlik_vagon' and rfid not like 'cistici' and rfid not like 'null'";
  $stmt = $db->query($sqlvlak);
  echo "<div class='tabulky'>";
  echo "<table border='1' align='center'>";
  echo "<tr><th>ID</th><th>Vlak</th><th>Obrázek</th><th>RFID</th></tr>";
  while ( ($row = $stmt->fetch(PDO::FETCH_ASSOC) ) !== false) {
  $rfid = "/dp/images/".$row["rfid"].".png";
  $rfidnahrada = "/dp/images/no-image.png";
    echo "<tr>";
    echo "<td>".$row['id']."</td>";
    echo "<td>".$row['description']."</td>";
    echo "<td>";
            if (file_exists($_SERVER['DOCUMENT_ROOT'].$rfid)) {
            echo "<img width='125' height='40' src='".$rfid."' title='".$row['description']."' >";
            }
            else {
            echo "<img width='125' height='40' src='".$rfidnahrada."' title='vlak bez obrázku' >";
            }
    echo "</td>";
    echo "<td>".$row['rfid']."</td>";
    echo "</tr>";
  }
  echo "</table>";
  echo "</div>";
} catch (PDOException $e) {
  print "nelze se připojit k DB: " . $e->getMessage();
  exit();
}
?>
    </div>
  </div>
  </body>

</html>
```

Obr. 3.15 Kód pro zobrazení vlaků

Zdroj: vlastní zpracování

V souboru *zobrazvagony.php* je patrné, že obsahuje opět HTML, kaskádové styly a PHP.

V prvním řádku PHP je snaha o připojení ke stejné databázi, jako je tomu u souboru *index.php*. V tomto případě je v proměnné *$db* přidána funkce, která správně nastaví kódování znaků, aby se všechny výstupní hodnoty z databáze správně zobrazovaly. Mohl by se stát, že diakritika bude zobrazena jako otazníky a nebude text čitelný. Tímto je tomu

zabráněno. Protože UTF8 umí pracovat s většinou světových jazyků, tak není problém zobrazit i diakritiku.

V tomto kódu je změna ve výběru vagonů z databáze. Aktuálně se využívá tabulka *wagons*, kde je podmínka, že se nesmí vybrat vagony ve sloupci *rfid* s hodnotou *ice_l, ice_c, ice_r, rychlik_vagon, cistici* a pokud obsahují prázdný řetězec. Tato podmínka je z důvodu zbytečného vypisování vagonů, které aktuálně nejsou použity na modelové železnici.

Jelikož je tato tabulka malá (obsahuje malé množství záznamů), tak je možné využít příkaz *query*, který lze použít bez jakékoliv přípravy jako v souboru *index.php*.

Pro vypisování záznamů z tabulky je opět použít cyklus *while*, který bude probíhat do té doby, dokud bude platná podmínka. Jakmile podmínka dostane hodnotu *false*, tak se uzavře tabulka, ukončí se práce se souborem a budou vypsány všechny dostupné vagony.

V cyklu *while* je přidána funkce, která kontroluje, že je přítomen soubor s určitým názvem. Tento název je získáván ze sloupce *rfid* a je porovnáván s názvy souborů ve složce patřící k této webové stránce. Pokud uvedený soubor existuje, tak je zobrazen obrázek daného vagonu. Pokud soubor neexistuje, tak se zobrazí náhradní obrázek s informací, že obrázek vagonu neexistuje.

3.5.3 Tvorba vedlejší stránky vlakinfo.php

Vedlejší stránka *vlakinfo.php* je vytvořena pro detailnější informace o daném vlaku.

```php
<?php
try {
  $db = new PDO ('mysql:host=localhost;dbname=vslg_diplomka','vslg','vslgprerov');
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $db->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);

  $id = "";
  if (isset($_GET['id'])) {
    $id = filter_var($_GET['id'], FILTER_SANITIZE_NUMBER_INT);
  }
  if (!empty($id)) {
  $sqlvlak = "SELECT id, timestamp, data, type, dio_nr FROM events WHERE id LIKE :id";
  $stmt = $db->prepare($sqlvlak);
  $stmt->execute(['id'=>$id]);
  echo "<div class='informace'>";
  while ( ($row = $stmt->fetch(PDO::FETCH_ASSOC) ) !== false) {
  $casvlak = $row['timestamp'];
  $casvlakuminus = strtotime($casvlak) - "10";
  $casvlakuplus = strtotime($casvlak) + "8";
  $sqlvagony = "select id, timestamp, data, type, dio_nr from events WHERE
  timestamp >= '".date("Y-m-d H:i:s", $casvlakuminus)."' AND timestamp <= '".date(
  "Y-m-d H:i:s", $casvlakuplus)."' AND dio_nr LIKE '".$row["dio_nr"]."' and data NOT
   LIKE '4b313000000000000c000200' and data NOT LIKE '362d31313100000004000200' and data
    NOT LIKE '372d31313100000004000200'";
  $sqlvag = $db->query($sqlvagony);
  $rfid = "/dp/images/".$row["data"].".png";
  $rfidnahrada = "/dp/images/no-image.png";
    echo "<div class='id-cas'><div class='idcko'><span>ID vlaku</span>".$row['id']."</div>";
    echo "<div class='casek'><span>Datum a čas</span>".$row['timestamp']."</div></div>";
    echo "<div class='souhrn'><span>Přehled vlaku</span> <br />";
            if (file_exists($_SERVER['DOCUMENT_ROOT'].$rfid)) {
            echo "<img src='".$rfid."' title='".$row['data']."' >";
            }
            else {
            echo "<img src='".$rfidnahrada."' title='vlak bez obrázku' >";
            }
            while (($rowvagon = $sqlvag->fetch(PDO::FETCH_ASSOC)) !== false) {
              $rfidvagon = "/dp/images/".$rowvagon["data"].".png";
              if (file_exists($_SERVER['DOCUMENT_ROOT'].$rfidvagon)) {
              echo "<img src='".$rfidvagon."' title='".$rowvagon['data']."' >";
              }
              else {
              echo "<img src='".$rfidnahrada."' title='vagon bez obrázku' >";
              }
            }
    echo "</div>";
  }
  echo "</div>";
  }
  else {
  echo "BEZ ID NELZE ZOBRAZIT INFORMACE VLAKU!";
  }
} catch (PDOException $e) {
  print "nelze se připojit k DB: " . $e->getMessage();
  exit();
```

Obr. 3.16 Kód pro zobrazení informací o daném vlaku

Zdroj: vlastní zpracování

Tento kód se ze začátku opět připojuje k databázi. Bylo možné kompletní připojení do databáze vložit do jednoho externího souboru, ale tímto je patrné a řádně viditelné, že je neustále potřeba být připojen k databázi, aby se dalo pracovat s potřebnými daty.

Opět se v tomto případě používá funkce na zjištění určitých informací z URL adresy. Tentokrát je to hodnota *id*, která je jedinečná pro každý vagon a lokomotivu. Tuto hodnotu je možné získat z hlavní stránky *index.php*, kde jsou vyhledány všechny vlaky nebo vyfiltrované jen určité.

Druhá podmínka je v tomto případě velice důležitá, jelikož hlídá, aby *$id*, do které je ukládána hodnota z URL adresy, nebyla prázdná. Pokud se stane, že je proměnná prázdná, tak se celý kód neprovede a přeskočí se rovnou na druhou možnost za podmínkou. A touto možností je *else*, pomocí které se zobrazí informace, že bez *id* nelze zobrazit informace. Pokud ale proměnná *$id* obsahuje hodnotu, tak se provede kompletní obsah podmínky.

V první řadě je použit příkaz na získání všech hodnot z tabulky *events* s podmínkou, která vybírá sloupec *id* a získává pouze záznamy s *id* hodnotou.

Opět je zde cyklus, který bude vypisovat postupně všechny záznamy. Jen s tím rozdílem, že aktuálně je v tomto cyklus ještě jedna proměnná, která získává data z databáze. Tentokrát je to získávání záznamů dle časového rozmezí okolo vybraného vlaku, aby byla zajištěno zobrazení vagonů za danou lokomotivou. Pro časové rozmezí je nutné u počátečního času odečíst 10 vteřin a u konečného času zase přičíst 8 vteřin. Tímto je docíleno určitého časového rozmezí pro zobrazení informací. Tento postup je vybrán z důvodu ukládání dat do databáze z RFID čteček, jelikož se může stát, že nějaký vagon bude zaznamenán již dříve případně později. Při procházení tabulky events je patrné, že záznamy ukládané do databáze nejsou vždy seřazeny dle správného čísla čidla. Stává se, že se špatně uloží nějaký záznam, ať už při zastavení modelové železnice a jejím novém spuštění nebo při jejím neustálém fungování.

Za první částí s kontrolou existence souboru je vložen ještě jeden cyklus *while*, který načítá záznamy vagonů, které byly vybrány v časovém rozmezí. Tímto je soubor *vlakinfo.php* připraven na spuštění.

3.5.4 Tvorba kaskádových stylů

Kaskádové styly, jak již je napsáno pod obr.3.5, jsou v dnešní době základ, který je pro vzhled webových stránek nutný. Bez CSS by webová stránka vypadala pouze jako textový dokument.

```
.centered {width: 100%; max-width: 1400px; margin: 0 auto; position: relative;}
.centered-2 {width: 100%; max-width: 1000px; margin: 0 auto; position: relative;}
.item-1 {padding: 5px 0 10px; background: rgb(241,241,241);background: linear-gradient(90deg, rgba(241,241,241,1) 0%
box-shadow: 0px 10px 29px 0px rgba(117,117,117,1); box-shadow: 0px 10px 29px 0px rgba(117,117,117,1);}

#formular1.active {display: block;}
#formular2 {display: none;}
#formular1 {display: none;}
#formular2.active {display: block;}
#prepinac-cidlo-vlak {position: relative; margin: 30px auto; display: block; width: 80px; height: 40px; border-radiu
0.1); cursor: pointer;}
#prepinac-cidlo-vlak .indicator {position: absolute; top: 0; left: 0; width: 40px; height: 40px; background: #fff; b
255,0.2); transition: 0.5s;}
#prepinac-cidlo-vlak.active .indicator {left: 40px;}
#prepinac-vlak {position: relative; margin: 30px auto; display: block; width: 80px; height: 40px; border-radius: 40p
cursor: pointer;}
#prepinac-vlak .indicator {position: absolute; top: 0; left: 0; width: 40px; height: 40px; background: #fff; border-
2); transition: 0.5s;}
#prepinac-vlak.active .indicator {left: 40px;}
#vyber-cidlo {display: none;}
#vyber-cidlo.active {display: block;}
#vyber-vlak {display: none;}
#vyber-vlak.active {display: block;}

.moznosti {display: block; position: relative; margin: 0 auto; width: 370px;}
.prvni-moznost {position: absolute; top: 9px; left: 0;}
.druha-moznost {position: absolute; top: 9px; right: 29px;}
.moznosti.vlak {width: 475px;}
.moznosti.vlak .druha-moznost {right: 50px;}

.formular input {margin: 5px;}
```

Obr. 3.17 Část CSS

Zdroj: vlastní zpracování

Z uvedeného obrázku je patrné, že používat CSS je jednoduché. Určitě vlastnosti se neustále opakují, případně dědí pro další prvek. Možnost dědit ale mají jen prvky, který patří do jiného bloku, jsou to tzv. rodiče a děti.

Class centered neboli *.centered* je vytvořena z důvodu vycentrování celé stránky a uvedení maximální šířky *1400px*. Důležitá pro vycentrování je vlastnost *margin*, která určuje vzdálenost vnějších okrajů od okrajů celého okna automaticky. Bez této vlastnosti by byla celá webová stránka zarovnána doleva.

Pro příjemnější a přehlednější zobrazení pozadí jsou zde využity funkce *linear-gradient*, které postupně mění barvy podle nastavení v procentech. Pro další přehledné zobrazení mají důležité bloky nastaven stín, jsou tímto ohraničeny a lépe se v nich člověk orientuje, jelikož nepřekáží zbytečné veliké mezery mezi okraji celého okna webového prohlížeče.

#prepinac-cidlo-vlak a *#prepinac-vlak* jsou dva přepínače, které jsou připraveny na možnost kliknutí a zobrazení požadovaných formulářů. Vzhled je možné vidět na obr.3.18.

Obr. 3.18 Přepínač na výběr formuláře

Zdroj: vlastní zpracování

Přepínače obsahují nejdříve posuvník, po kterém se pohybuje přepínací tlačítko. Jakmile je na tlačítko kliknuto, tak se přesune doprava případně doleva. Po kliknutí se provede okamžitě funkce, která je připravena pomocí JavaScriptu na obr.3.14, která přidá nebo odebere *class active* předchozímu prvku.

Tlačítko v HTML má *class* s názvem *indicator*, tzn. že v CSS je možné jej nalézt jako *.indicator*. V CSS je vidět, že *.indicator* je součástí bloku *#prepinac-cidlo-vlak* a má vlastnosti, které nastavují absolutní pozici. Z toho vyplývá, že blok *#prepinac-cidlo-vlak* musí mít nastaveno pozici relativní, aby vždy fungovalo správně. Prvek *.indicator* má nastaveno přesné umístění a přesnou velikost. Při kliknutí se ale umístění změní, jelikož rodičovský prvek obdrží *class active*, a tlačítko se posune doprava a tím aktivuje formuláře.

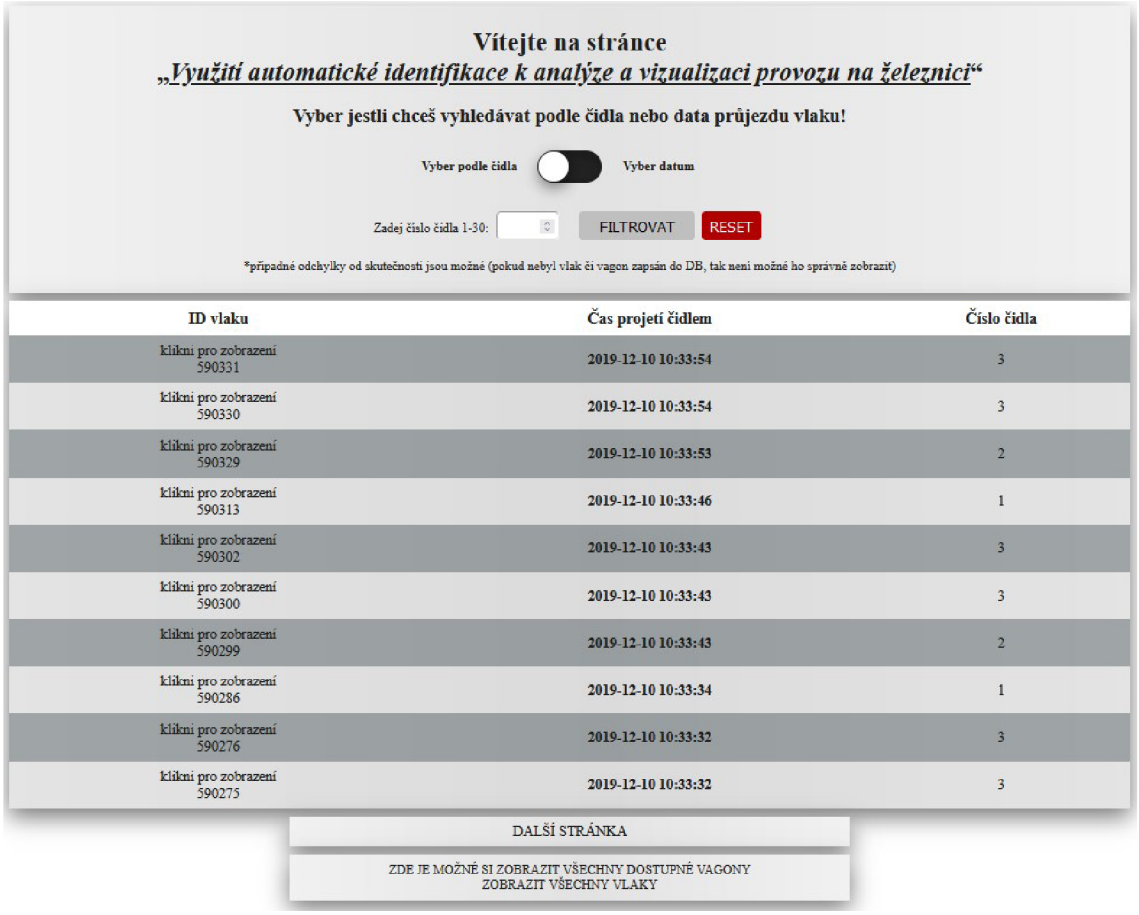
Kompletní zdrojový kód souboru styles.css je umístěn jako elektronická příloha na CD.

4 Pilotní ověření a vyhodnocení aplikace

Kompletní aplikace je vytvořená pomocí předchozích zdrojových kódů a výsledek je možné vidět následující kapitole.

4.1 Spuštění aplikace

Prvotní spuštění aplikace bude pro uživatele vypadat takto. Zobrazí se kompletní přehled všech vlaků seřazený od nejnovějšího po nejstarší.



Obr. 4.1 Finální podoba webové stránky

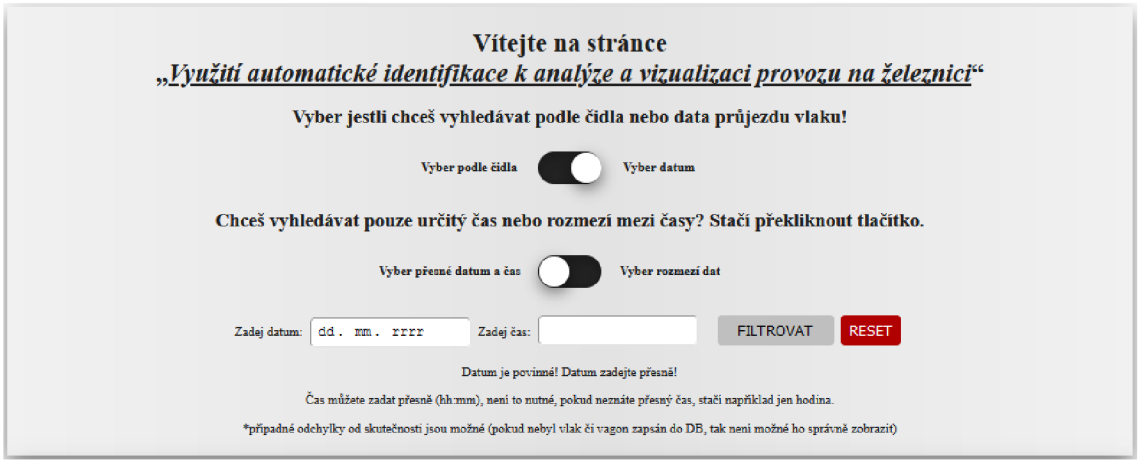
Zdroj: vlastní zpracování

V první části webové stránky, která slouží k ovládání, se nachází přepínací tlačítko. Je možné jen nechat takto a zadat příslušené číslo čidla, které je v rozmezí od 1 do 30. Vyšší nebo nižší není dovoleno, jelikož toto hlídá input sám. Následně stačí kliknout na tlačítko filtrovat a nechat si zobrazit požadované čidlo viz. obr.4.2.

Obr. 4.2 Vyfiltrované čidlo číslo 6

Zdroj: vlastní zpracování

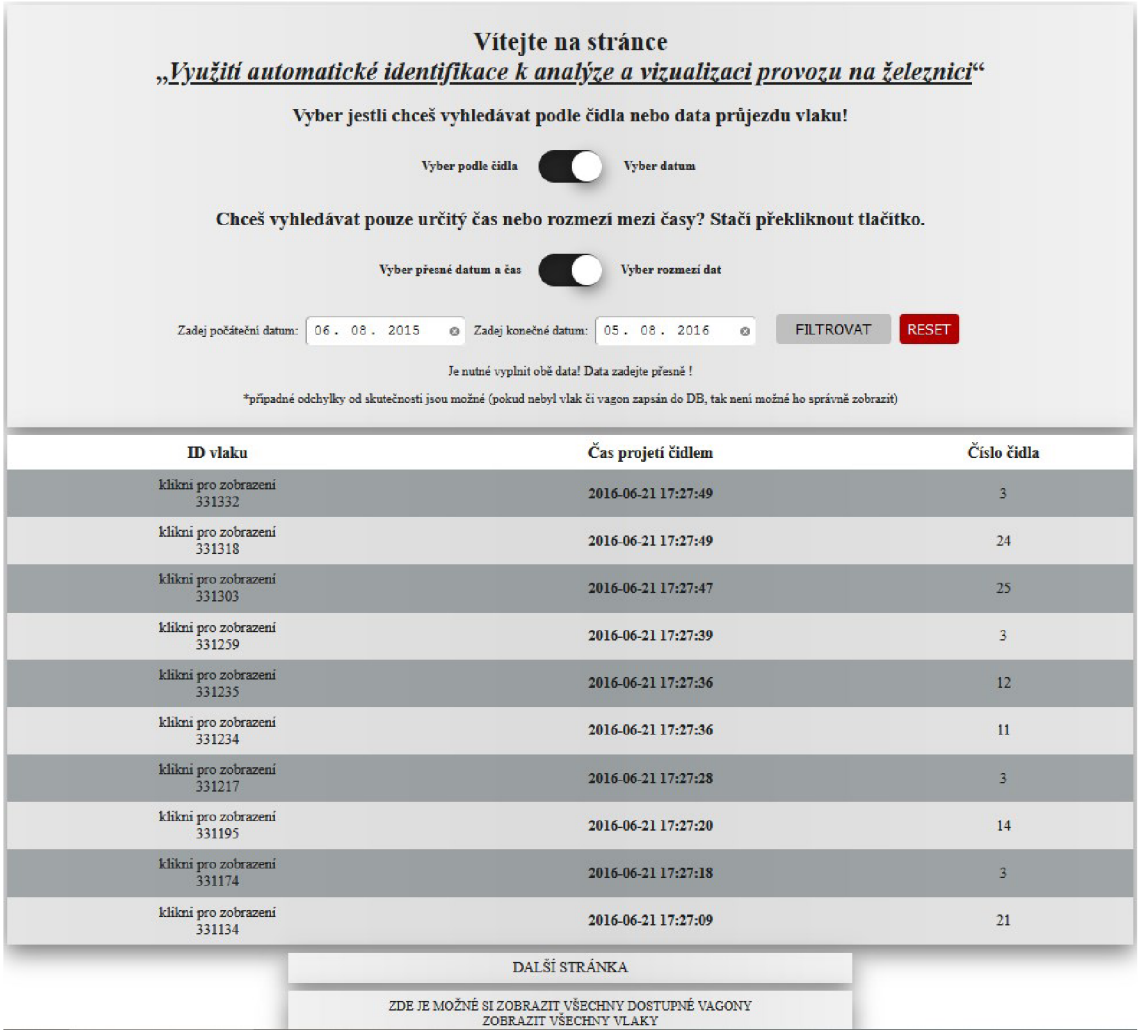
Po stisknutí přepínacího tlačítka je možné začít zadávat datum a čas. Je možné zadat pouze datum a čas není nutné. Také je možné zadat datum a čas jenom v hodinách, hodinách a minutách nebo v celém formátu hodiny, minuty a vteřiny. Při kompletním vyplnění to vyfiltruje přesný vlak, pokud v této době projel nějakým čidlem.



Obr. 4.3 Přepnutý přepínač na výběr data

Zdroj: vlastní zpracování

Při výběru možnosti rozmezí dat, je možné zadat počáteční a konečné datum, pro které je potřeba filtrovat. Tímto se zobrazí jen požadované vlaky jako na obr.4.4.

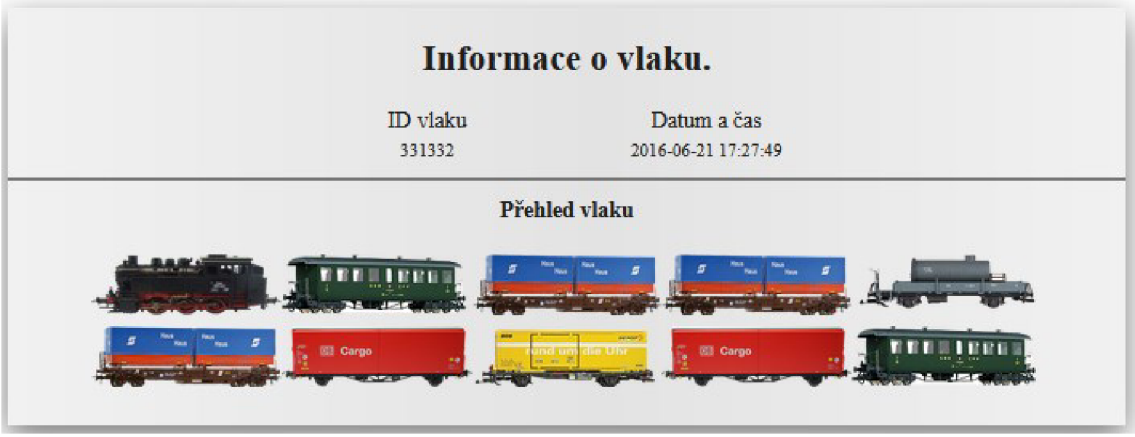


Obr. 4.4 Vyfiltrované rozmezí dat

Zdroj: vlastní zpracování

Při velikém množství zobrazených vlaků je možné použít stránkování a zobrazit další stránku, případně se vrátit na předchozí. Tímto se dají prozkoumat všechny vlaky, které jsou vyfiltrovány.

U všech možností filtrace je tlačítko na zobrazení detailů daného vlaku. Toto tlačítko nese název „klikni pro zobrazení“ a číslo daného vlaku. Po kliknutí na toto tlačítko se zobrazí nová stránka, která obsahuje důležité informace o požadovaném vlaku.



Obr. 4.5 Detail vlaku ID 331332

Zdroj: vlastní zpracování

Na uvedeném obr.4.5 je v první řadě nejdůležitější seřazení vagonů, které tento vlak převáží. Tímto bylo docíleno funkcemi v soubor *vlakinfo.php*. Zobrazení vagonů nemusí být 100% přesné, jelikož vždy záleží na zapsání daného vagonu do databáze a jeho času zapsání. Dále obsahuje stránka ID vlaku, pro který je hledáno seřazení vagonů, a přesný čas zapsání vlaku do databáze.

Takto je možné zobrazit informace o všech vlacích, které se vyfiltrují.

Na hlavní stránce je ještě jedno tlačítko, kterým je možnost zobrazit všechny vlaky. Po stisknutí se zobrazí kompletní seznam všech vagonů a lokomotiv i s obrázky.

Zde je seznam a přehled všech dostupných vagonů a lokomotiv.

ID	Vlak	Obrázek	RFID
1	Ragulin		4b31390000000000081000200
2	Vagon Skrinovy DB Cargo		20202020202020312d323232
3	Uhlak		20202020202020312d333333
4	Uhlak		e2002075691200912490 1bc0
5	Uhlak		20202020202020312d353535
6	Uhlak		20202020202020312d363636
7	Bardotka CD Cargo		e2002075691200912580159d
8	Vagon Skrinovy DB Cargo		20202020202020322d323232
9	Postovni zeleny		20202020202020322d333333
10	Vagon skrinovy hnedy		20202020202020322d343434
11	Uhlak		20202020202020322d353535
12	Uhlak		20202020202020322d363636
13	Bardotka Modro-zluta		20202020202020332d313131
14	Vagon Skrinovy DB Cargo		20202020202020332d323232
15	Nizky otevreny		e2002075691200902700 0d7d
16	Kontejner maly		20202020202020332d343434
17	Uhlak		20202020202020332d353535
18	Uhlak		20202020202020332d363636
19	Lokomotiva OBB		e2002075691200902580 1599
20	Vagon plachta cerveny		20202020202020342d323232

Obr. 4.6 Seznam možných vagonů a lokomotiv

Zdroj: vlastní zpracování

Aktuální obr.4.6 neobsahuje všechny vagony a lokomotivy. Všechny lokomotivy je možné zobrazit přímo na webové stránce nebo jsou součástí elektronické přílohy na CD ve složce *images*. Tato webová stránka obsahuje celý název vagonu a jeho příslušené rfid, které se ukládá do tabulky *events* do sloupce *data*.

4.2 Vyhodnocení aplikace

Tato webová aplikace je připravena pro používání uživatelem, kterého zajímají informace o vlaku. Obsahuje jen nejdůležitější informace, které jsou pro uživatele potřebné. Databáze obsahuje ještě mnoho informací, ale ty mají pro uživatele nulovou hodnotu, jelikož to jsou pouze interní údaje. Aplikace byla co nejvíce zjednodušena a po kliknutí na určitá tlačítka je možné zobrazit i detailní informace o vlaku. Všechno toto je

v grafické podobě, a tak není nutné číst zdlouhavé texty. Tento přístup zbytečně nezdržuje uživatele, jelikož si prohlédne seřazení vlaku a ví přesně, jak s tím zacházet.

V případě potřeba je možné zasáhnout do zdrojového kódu a nějaké funkce přidat nebo upravit zobrazovaná data, pokud by to uživatel vyžadoval.

Tato aplikace je potřeba pro přehledné řazení vagonů z logistických důvodů, aby byla zajištěna rychlejší manipulace s vagony. Seřazení vagonů může pomocí v mnoha dalších případech.

Závěr

Celá tvorba webové aplikace vycházela z již připravené webové aplikace v bakalářské práci, která byla vytvořena v roce 2018. Bohužel při podrobnějším zkoumání aplikace bylo zjištěno, že systém byl velice složitý a práce s aplikací byla pomalá. Dlouho se načítaly potřebné informace, aplikace byla nepřehledná a zdrojový kód byl zbytečně komplikovaný.

Webová aplikace byla kompletně přepracována, byla zjednodušena funkčnost a zlepšena přehlednost. Je možné si zobrazit přímo nějaké čidlo nebo vyhledat rozmezí dat, kdy měly být vlaky uloženy. Aktuálně je webová aplikace dostatečně rychlá při jakékoliv manipulaci a není nutné zdlouhavě čekat na jakékoliv načtení webové stránky nebo filtrovaných dat.

Webová aplikace má ještě jednu funkci. A to takovou, že jakmile jsou vyhledány nějaké vlaky nebo je zobrazena informace o vlaku, tak je možné tuto URL adresu sdílet s kýmkoliv. Díky tomuto je možné informovat i okolí o daném vlaku nebo vyhledávaných položkách.

Teoretický cíl s automatickou identifikací byl splněn a byl řádně popsán.

Uživatelské funkce byly implementovány a aplikace byla řádně odzkoušena na několika různých počítačích a několika různými uživateli a vždy byla shledána jako přehledná a fungující.

Seznam zdrojů

[1] *Systémy automatické identifikace*. Dopravní logistika profi [online]. Praha: Verlag Dashöfer, 2015, 1.8.2015 [cit. 2021-8-6]. Dostupné z: https://www.dlprofi.cz/33/systemy-automaticke-identifikace-sai-uniqueidmRRWSbk196FNf8-jVUh4EoSF6RcLfOnl-EzS1yVCetw/?query=syst%E9my%20automatick%E9%20identifikace&serp=1.

[2] GROS, Ivan a kol. *Velká kniha logistiky*. Praha: Vysoká škola chemicko-technologická v Praze, 2016. ISBN 978-80-7080-952-5.

[3] *Čárový kód - Základní prostředek automatické identifikace zboží*. Kodys [online]. Praha: KODYS, 2021 [cit. 2021-8-6]. Dostupné z: https://www.kodys.cz/technologie/carovy-kod.

[4] *Čárové kódy (teorie)*. Gaben [online]. Ostrava: GABEN, 2021 [cit. 2021-8-6]. Dostupné z: http://www.gaben.cz/cz/faq/carove-kody-teorie.

[5] HÁJEK, Tomáš. *Systémy automatické identifikace*. Praha, 2010. Bakalářská práce. Bankovní institut vysoká škola Praha. Vedoucí práce Ing. Vladimír Beneš, Ph.D.

[6] FEJFAR, Tomáš. *Vizualizace provozu železnice pomoci auto ID*. Přerov, 2018. Bakalářská práce. Vysoká škola logistiky o.p.s. Vedoucí práce Ing. Libor Kavka, Ph.D.

[7] SKLAR, David. *PHP 7: praktický průvodce nejrozšířenějším skriptovacím jazykem pro web*. Brno: Zoner Press, 2018. Encyklopedie Zoner Press. ISBN 978-807-4133-633.

[8] KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora*. Brno: Computer Press, 2007. ISBN 978-802-5118-139.

[9] CASTRO, Elizabeth a Bruce HYSLOP. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno: Computer Press, 2012. ISBN 978-802-5137-338.

[10] SETLÍKOVÁ, Petra. *Aplikace standardů GS1 v modelu železniční dopravy*. Přerov, 2017. Bakalářská práce. Vysoká škola logistiky o.p.s. Vedoucí práce doc. Dr. Ing. Oldřich Kodym.

Seznam grafických objektů

Seznam zkratek

ASCII	American Standard Cod efor Information Interchange, kódová tabulka definující všechny znaky světových jazyků
CD	Compact Disk – optický disk
CSS	Kaskádové styly – upravují vzhled webových stránek
DNA	Lidský genom obsahující genetické informace
EAN	European Article Number – jedná se o čárový kód spotřebních výrobků
Em	Jednotka velikosti písma
FullHD	Full High-Definition – jedná se o rozlišení obrazovky 1920x1080px
GS1	Systém umožňující jednoznačnou identifikaci položek
HTML	HyperText Markup Language – základní jazyk pro tvorbu webových stránek
ID	Identifikační číslo
IP	Internet Protocol – jedná se o adresu počítače nebo serveru v počítačové síti
JS	JavaScript – skriptovací jazyk
kHZ	KiloHertz – jednotka frekvence
MB	MegaByte – jednotka kapacity paměťového média
MHz	MegaHertz – jednotka frekvence
MSI	Numerická symbolika na čárovém kódu
MySQL	My Structured Query Language – jedná se o databázový systém pro databázový jazyk
NFC	Near Field Communication – technologie na komunikaci na krátkou vzdálenost
OCR	Optical Charakter Recognition – optické rozpoznávání znaků pomocí skeneru, používá se pro převod do počítače
PDF417	Portable Data File 417 – jedná se o čárový kód
PHP	Hypertext Preprocesor, dříve Personal Home Page – programovací jazyk
PHPMyAdmin	Nástroj napsaný v jazyku PHP pro práci s databázemi

Px	Pixel
QR Code	Quick Response Code – jedná se o dvoudimenzionální čárový kód
RFID	Radiofrekvenční identifikace
U.P.C.	Universal Product Code – je to 12-ti místný čárový kód
UltraHD	Ultra High-Definiton – jedná se o rozlišení obrazovky 3840x2160px
URL	Universal Resource Locator – webová adresa složená z textového řetězce
WWW	World Wide Web – celosvětová síť

Autor DP	**Bc. Fejfar Tomáš, DiS.**
Název DP	**Využití automatické identifikace k analýze a vizualizaci provozu na železnici**
Studijní obor	**LRDP**
Rok obhajoby DP	**2021**
Počet stran	56
Počet příloh	0
Vedoucí DP	**Ing. Libor Kavka, Ph.D.**
Anotace	Z důvodu dřívější tvorby bakalářské práce na téma Vizualizace provozu železnice pomocí auto ID, přichází myšlenka na její kompletní úpravu a přepracování do lépe fungující a více funkcemi opatřené webové aplikace. Na přípravu webové aplikace je nutné získání informací z příslušené databáze modelové železnice, která je umístěna v prostorách Vysoké školy logistiky v Přerově. Pomocí této modelové železnice je možné zkoušet fungování RFID čteček. V první části diplomové práce jsou obsáhlé informace o automatické identifikaci v logistice. Ve zbylé části je kompletní tvorba webové aplikaci od první myšlenky po finální odzkoušení webové aplikace.
Klíčová slova	RFID, automatická identifikace, vizualizace, databáze, MySQL, PHP, modelová železnice
Místo uložení	ITC (knihovna) Vysoké školy logistiky v Přerově
Signatura	