VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ POROVNÁVÁNÍ VLAKOVÝCH SPOJŮ

INTERACTIVE COMPARISON OF TRAIN CONNECTIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN UHLÍŘ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/17511/2016/xuhlir16

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Uhlíř Jan**
Obor: Informační technologie
Téma: **Interaktivní porovnávání vlakových spojů**
Interactive Comparison of Train Connections
Kategorie: Uživatelská rozhraní

Pokyny:

1. Proveďte průzkum dostupných aplikací a služeb určených pro vyhledávání vlakových spojů. Analyzujte jejich výhody a nedostatky. Prozkoumejte princip vlakových grafikonů.
2. Seznamte se s principem tvorby webových aplikací (se zaměřením na možnost tvorby vlastních typů diagramů).
3. Na základě pokynů vedoucího a poznatků získaných z bodů 1 a 2 navrhněte aplikaci určenou pro interaktivní porovnávání vlakových spojů prostřednictvím vlakového grafikonu.
4. Navrženou aplikaci implementujte.
5. Otestujte přívětivost implementovaného uživatelského rozhraní a navrhněte možná rozšíření.

Literatura:

- Tufte, E. R.: *The Visual Display of Quantitative Information*. 2nd edition. Cheshire: Graphics Press, 2007, ISBN 978-0961392109.
- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Burlington: Morgan Kaufmann Publishers/Elsevier, 2010, ISBN 978-0-12-375030-3.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hynek Jiří, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá problematikou interaktivního porovnávání vlakových spojů pomocí grafikonu vlakové dopravy. Nejprve byla provedena analýza existujících aplikací. Následně proběhla analýza dostupných technologií pro tvorbu vlastních diagramů a databázových systémů. Výsledkem práce je aplikace určená pro webovou platformu, jejímž úkolem je zobrazit provoz na dané trati ve formě interaktivního grafického elementu. Na základě vlastní konfigurace uživatele lze filtrovat zobrazovaná data.

Abstract

This bachelor's thesis is concerned by question of interactive comparison of train connections with graphic timetable. The first step was to analyze existing applications. Subsequently, an analysis of the available technologies for creating custom diagrams and database systems was performed. The product of this thesis is a web application which purpose is a graphic representation of traffic on the track in the form of an interactive graphic element. The displayed data can be filtered according to user's configuration.

Klíčová slova

grafikon vlakové dopravy, GVD, NJŘ, jízdní řád, GUI, diagram, vizualizace dat, SPA, JavaScript, Plotly, jQuery, PHP, SQL

Keywords

graphic timetable, rail timetable, GUI, diagram, data visualization, SPA, JavaScript, Plotly, jQuery, PHP, SQL

Citace

UHLÍŘ, Jan. *Interaktivní porovnávání vlakových spojů*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek

Interaktivní porovnávání vlakových spojů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Uhlíř
16. května 2017

Poděkování

Touto cestou bych chtěl poděkoval vedoucímu mé bakalářské práce Ing. Jiřímu Hynkovi, za jeho rady, ochotu, motivaci a čas věnovaný konzultacím a průběžnému testování.

Obsah

Kapitola 1

Úvod

Hlavním cílem této práce je návrh a následná implementace webové aplikace pro interaktivní porovnávání vlakových spojů prostřednictvím vlakového grafikonu. Ta bude sloužit pro snadné a intuitivní plánování cest vlakem, s ohledem na přívětivé uživatelské rozhraní. Veškeré spoje na dané trati mezi uživatelem zadanými dvěma uzly budou zobrazeny v přehledném grafu. Zároveň bude mít uživatel možnost zobrazit si podrobnější informace o konkrétním spoji, jako je například čas odjezdu, příjezdu, číslo vlaku, jeho typ a jméno.

Tvorba této webové aplikace je motivována především snahou o rozdílný přístup ke zvolené tématice, na rozdíl od současně dostupných aplikací zabývajících se vlakovými jízdními řády. Ty často prezentují uživateli informace o spojích především v textové podobě, což může být vhodné pro kratší tratě, bez navazujících spojů. Pokud ale uživatel plánuje delší trasu, navíc s různými přestupy, bylo by pro něj mnohem pohodlnější mít přehled o celkovém provozu na trati. Je pak schopen daleko lépe a efektivněji najít pro něj nejvhodnější spoj, čehož by při neustálém zadávání času odjezdu a příjezdu z jednotlivých nástupišť dosahoval velice obtížně.

Nejprve bude popsána problematika vlakového grafikonu a jeho využití v uživatelském rozhraní aplikace. Následně proběhne analýza současných aplikací zabývajících se vlakovou dopravou, spolu s rozborem jejich výhod a nedostatků. Následně budou stanoveny jednotlivé požadavky na navrhovanou aplikaci. Poté budou analyzovány dostupné technologie pro tvorbu vlastních diagramů spolu s dostupnými databázovými systémy. Zmíněn bude také způsob uložení a získání dat nutných pro funkci aplikace. Dále bude podrobně popsán návrh jednotlivých částí aplikace, společně s jejich implementací. Na závěr bude popsáno, jak probíhalo testování implementované aplikace a na základě zpětné vazby bude navrženo možné budoucí vylepšení.

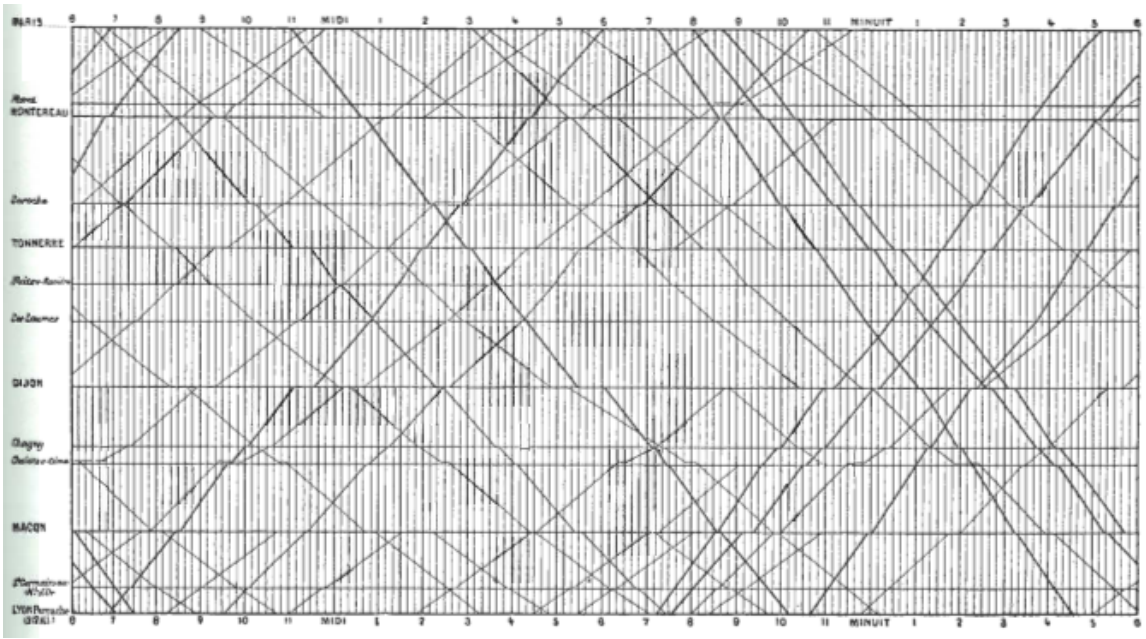
Kapitola 2

Grafikon vlakové dopravy

V této kapitole se zaměříme na stěžejní prvek celé práce, a to na *grafikon vlakové dopravy*. Postupně se zaměříme na jeho historii, současné využití a odůvodníme jeho použití v aplikaci.

2.1 Historie

Historicky první zmínka o vlakovém grafikonu pochází z Paříže roku 1880. Jeho autorem byl E.J. Marey. Jednalo se o grafickou podobu jízdního řádu na trati z Paříže do Lyonu. Roku 1881 byl díky jeho publikaci grafikonu zaveden nový expresní spoj, který zvládl cestu z Paříže do Lyonu pod hranicí 3 hodin oproti dřívějším 9 hodinám. Tento grafikon je znázorněný na obrázku 2.1.



Obrázek 2.1: E.J. Marey, La Méthode Graphique (Paříž, 1885) [8]

2.2 Popis

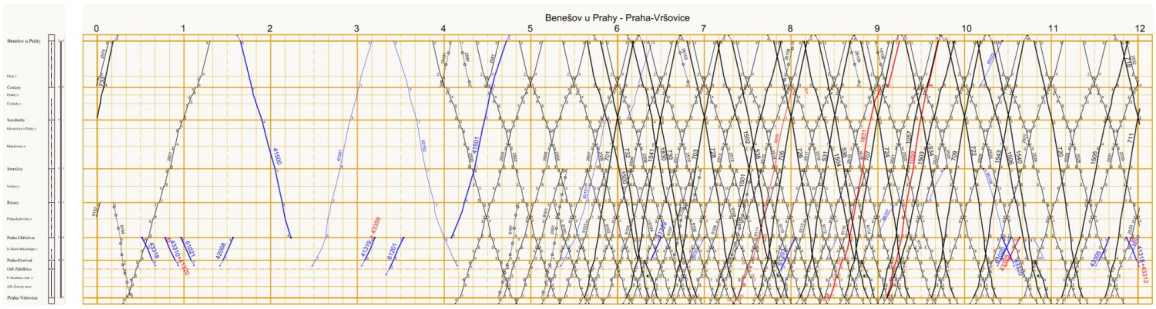
Grafikon vlakové dopravy používaný v dnešní době [12] je velice podobný jeho první verzi z roku 1880. Jeho základem jsou 2 osy v kartézské soustavě souřadnic. Na vertikální ose jsou znázorněny jednotlivé zastávky na trati v závislosti na vzdálenosti mezi nimi. Horizontální osa je časová osa, na které je znázorněn celý provozní den na trati.

Jednotlivé vlaky poté představují šikmé přímky, jejichž směr a rychlost jsou dány jejich sklonem. Čím strmější přímka, tím je daný spoj rychlejší. Odstávka spoje na trati je znázorněna vodorovnou čarou.

2.3 Současné využití

Grafikon vlakové dopravy je běžně používán dopravními podniky při plánování drážní dopravy, jejím řízení, kontrole a tvorbě jízdních řádů. Lze jej ovšem stejně dobře využít i při plánování cesty po trase. Přesto jej kromě dopravních podniků využívají výhradně strojvedoucí, dispečeři, nebo výpravčí. Neslouží tedy jako běžný prostředek používaný širokou veřejností.

V České republice se pojem *grafikon vlakové dopravy* často nahrazuje výrazem *nákresný jízdní řád*, neboli *NJŘ*, případně *list GVD*. Existuje služba gvd.cz[1], kde jsou široké veřejnosti poskytnuty grafikony vlakové dopravy pro jednotlivé tratě v České a Slovenské republice. Ukázka nákresného jízdního řádu je znázorněno na obrázku 2.2.



Obrázek 2.2: Nákresný jízdní řád, zdroj: www.gvd.cz

2.4 Využití v aplikaci

Jak bylo řečeno v kapitole 2.2, grafikon vlakové dopravy je klasický graf v kartézské soustavě souřadnic. Z toho vyplývá fakt, že jej lze implementovat jakýmkoliv nástrojem, který dokáže vytvořit lineární graf. Při vhodně zvolené implementační platformě tedy lze získat širokou škálů nástrojů pro tvorbu grafů, ze kterých vybereme ten nejvhodnější.

[1] http://www.gvd.cz

Kapitola 3

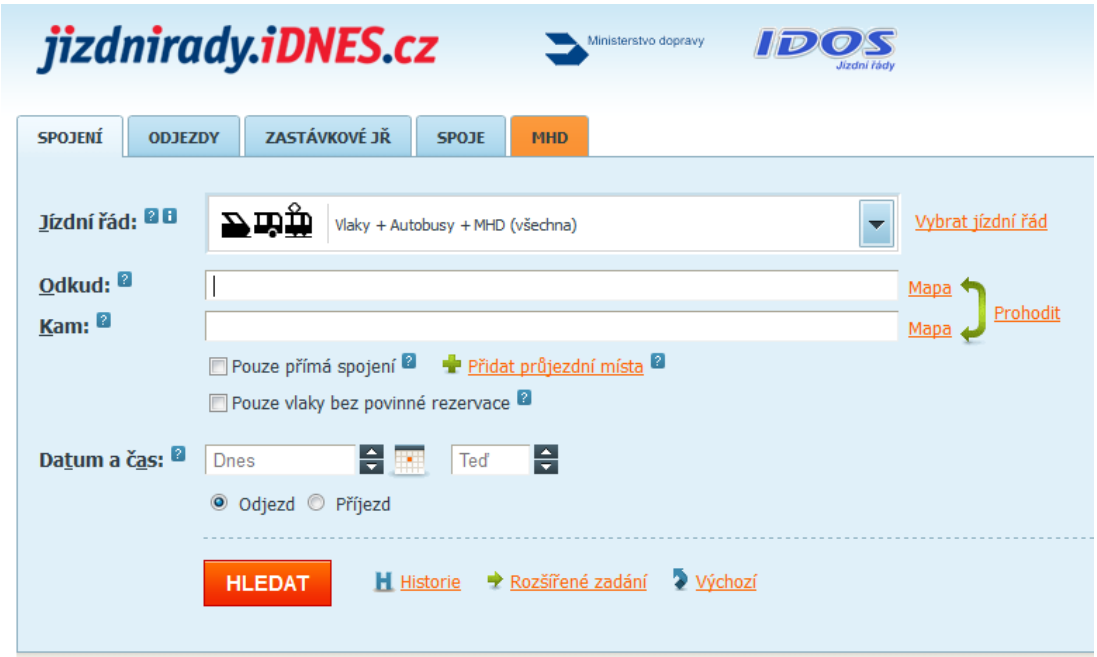
Analýza existujících aplikací

Tato kapitola se zaměřuje na již existující aplikace zabývající se vlakovou dopravou napříč různými platformami. Postupně zde budou rozebrány jak jejich výhody, tak i nevýhody. U obojího bude vysvětlené, jak se s danou oblastí příslušné aplikace vypořádávají.

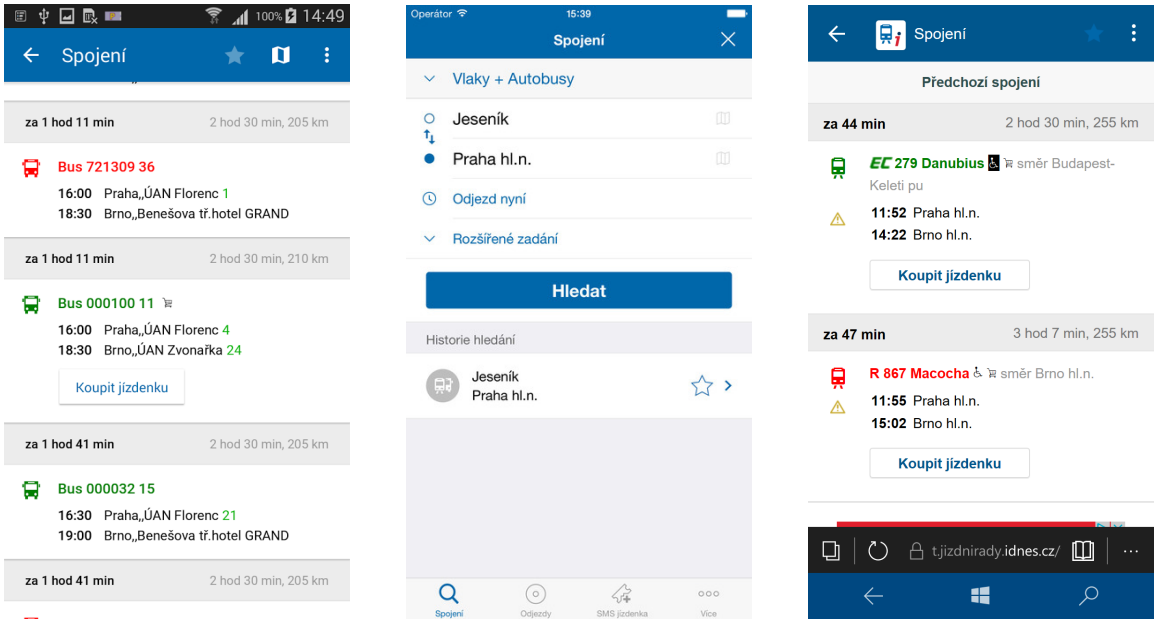
3.1 Jízdní řády IDOS

První aplikací, kterou si zde rozebereme, bude ta nejznámější na našem trhu. Jedná se o celostátní informační systém o jízdních řádech, jehož realizace vznikla na pokyn Ministerstva dopravy České republiky a Ministerstva informatiky České republiky (zrušeno r.2007). Zahrnuje nejen vlakovou, ale také autobusovou, leteckou, či městskou hromadnou dopravu. V rámci této práce se ale pro naše účely zaměříme pouze na první zmiňovanou.

Data jsou aplikaci poskytována firmou CHAPS spol. s.r.o. Jedná se o společnost zabývající se vývojem IT aplikací a systémů, jejich údržbou a provozem. Zaměřují se zejména na oblast osobní dopravy. Na obrázku 3.1 vidíme webovou podobu aplikace.



Obrázek 3.1: Webový prohlížeč, zdroj: www.jizdnirady.idnes.cz



Obrázek 3.2: Android, iOS, Web (zleva)

3.1.1 Výhody

Mezi hlavní výhody této aplikace patří především její rozsáhlost při výběru typů spojů. Jak již bylo zmíněno v kapitole 3.1, nezabývá se pouze striktně vlakovou dopravou. V porovnání s vlakovým grafikonem je tedy daleko flexibilnější. Na druhou stranu, cílem naší aplikace je pouze vlaková doprava, proto je tedy tato výhoda v našem případě nepodstatná a aplikace jí nehodlá v tomto ohledu konkurovat.

Jako další výhodu lze zmínit její multiplatformnost. Aplikace začínala jako čistě webová služba, ale dnes už má své vlastní nativní aplikace pro dva majoritní operační systémy na světovém trhu, a to operační systém Android od společnosti Google Inc., spolu s operačním systémem iOS od firmy Apple Inc. Verzi pro první ze dvou zmíněných systémů lze spatřit na obrázku 3.2, kde jsou patrné prvky moderního material designu. Verze pro iOS je také znázorněna na obrázku 3.2. Zároveň je neustále udržována její webová verze, kde je kladen důraz na její responzivní vzhled. V internetovém prohlížeči na libovolném chytrém telefonu vypadá prakticky stejně jako nativní aplikace. Webová podoba aplikace byla pořízena na mobilním telefonu s operačním systémem Windows 10.

Mobilní verze aplikace nabízí přívětivé uživatelské rozhraní, které je přehledné a jednoduché. Aplikace pracuje rychle, obsahuje našeptávač nejpoužívanějších spojů a je schopná zadat jako výchozí místo zastávku nejblíže poloze uživatele.

Nespornou výhodou je také její přístup k aktuálním datům. Firma CHAPS spol. s.r.o. spravující tuto databázi totiž nabízí pouze placené služby. Není tedy možné implementovat vlastní aplikace pro jízdní řády bez počáteční investice. Jediným řešením je vytvoření vlastní databáze spojů.

3.1.2 Nevýhody

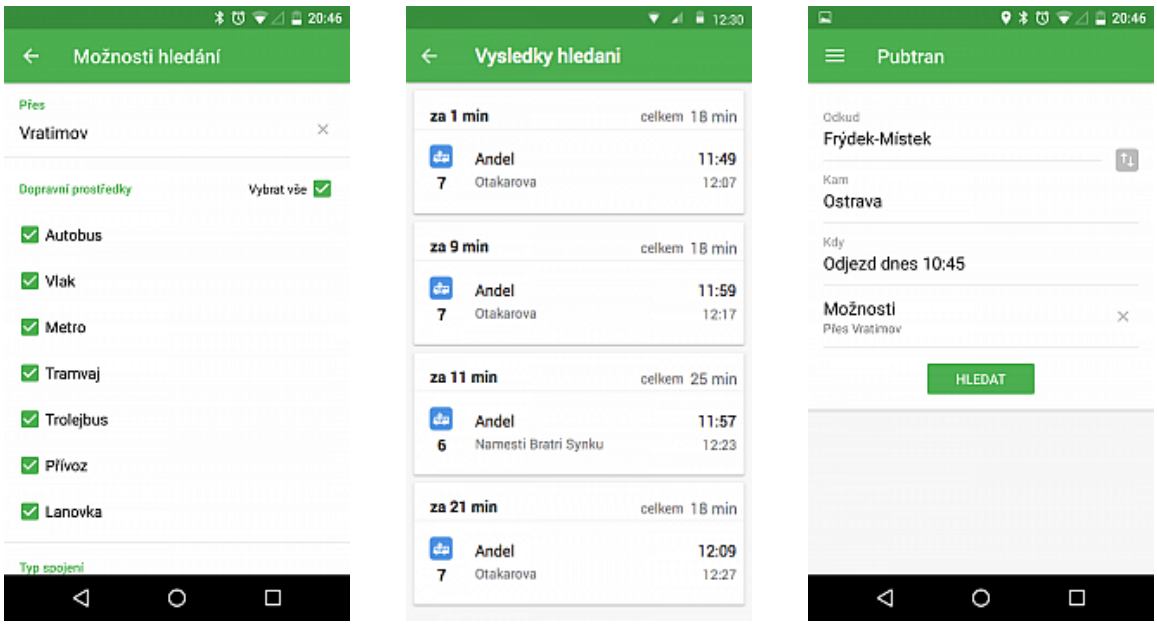
Za hlavní nevýhodu této aplikace považuji její webový vzhled pro desktopová zařízení. Na rozdíl od jejího responsivního vzhledu, který jsem chválil v přechozí kapitole, je její webová podoba poněkud zastaralá. Uživatelské rozhraní působí nepřehledně.

Další nevýhodou, na kterou se zaměřuje právě naše aplikace, je interpretace dat o spojích, které uživatel hledá. Hlavní myšlenkou naší aplikace je snaha zobrazit tyto informace graficky za pomocí vlakového grafikonu, aby měl uživatel lepší přehled o spoji, především při delších tratích a při přestupech. IDOS poskytne informace uživateli v textové podobě, kde uživatel primárně vidí čas odjezdu ze stanice A spolu s časem příjezdu do stanice B. Pokud chce zjistit, jak dlouho který spoj jede, případně kde musí přestupovat, musí otevřít nové okno pro detailnější informace. Zároveň nemá možnost porovnat jednotlivé odjezdy ze stanice, kdy následující spoj například může být rychlejší a čekal by kratší dobu na přestup.

Poslední nevýhodou je závislost aplikace na internetovém připojení, protože nemá svoji vlastní databázi, ale při každém požadavku se dotazuje vzdálené databáze. Je tedy závislá na jejím provozu. Na druhou stranu s vlastní databází by se výrazně zvýšila velikost aplikace. Navíc firma CHAPS spol. s.r.o. je velká zavedená firma, takže se zpravidla nečeká, že by došlo k nějakému výpadku a databáze by nebyla přístupná.

3.2 Pubtran

Další velmi známou a oblíbenou aplikací u nás je Pubtran. Aplikace je momentálně ve vlastnictví firmy Seznam.cz, a.s. Jedná se o velice častou alternativu k výše zmíněné aplikaci IDOS. Na rozdíl od ní nečerpá data od firmy CHAPS spol. s.r.o., ale jsou poskytována přímo firmou Seznam.cz. V současné době je dostupná pouze pro platformu Android.



Obrázek 3.3: Android

3.2.1 Výhody

Stejně jako u aplikace IDOS, tak i Pubtran má velmi přívětivé uživatelské prostředí, které je znázorněno na obrázku 3.3. Plně těží z výhod své současné mateřské firmy Seznam.cz, především z jejich mapových podkladů. Umožňuje tedy uživateli najít zastávku nejen podle aktuální polohy, ale také podle adresy, nebo firmy.

Velmi užitečnou funkcí je upozornění na odjezd určitého spoje, kdy je uživatel upozorněn se zvoleným předstihem formou notifikace. Stejnou funkci nabízí aplikace i pro přestupy. Má také rozsáhlejší repertoár způsobu dopravy. Kromě tradičního vlaku a autobusu dovoluje uživateli více specifikovat druh městské dopravy, konkrétně trolejbus nebo tramvaj. Nabízí také možnost dopravy metrem. Zajímavou možností je potom možnost zvolit dopravu přívozem či lanovkou.

Nabízí také možnost detailu spojení, včetně zobrazení mapy spojení, právě díky mapovým podkladům od Seznamu, kde mohou být zahrnuty i pěší přesuny. Lze prohledávat i offline historii, čímž odpadá nutnost internetového připojení. Uživatel si také může uložit své oblíbené spoje.

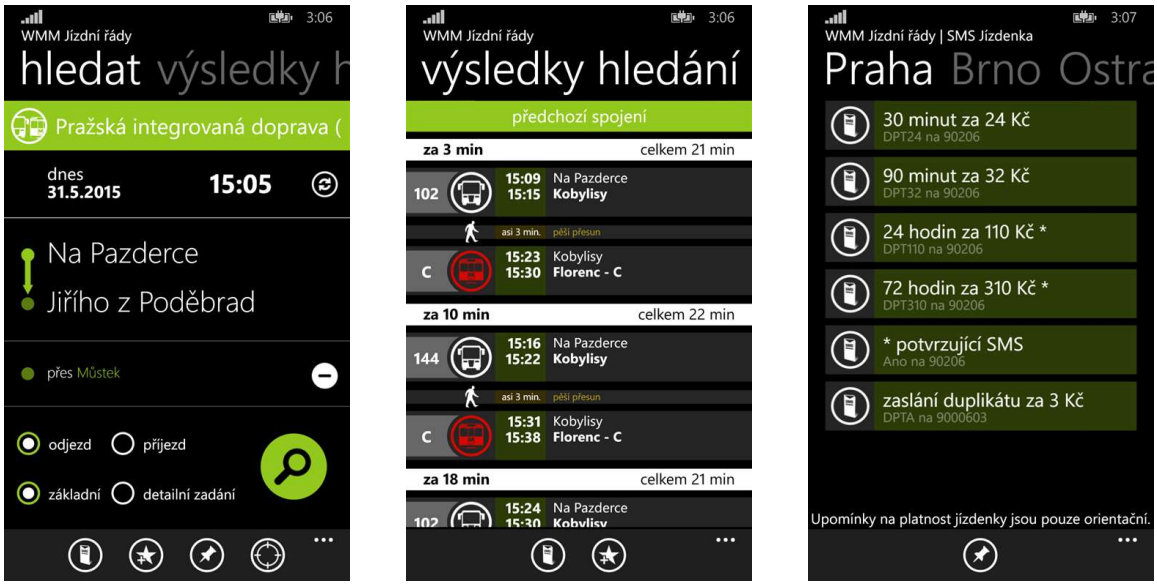
3.2.2 Nevýhody

Aplikace je momentálně dostupná pouze pro zařízení Android, čímž je její používání omezené pouze na jednu konkrétní platformu, i když Seznam do budoucna oznámil její rozšíření na platformy iOS a Windows [9]. Stejně jako IDOS, tak i Pubtran podává data uživateli převážně v textové formě. Toto klasické řešení je opět vhodné pouze pro kratší spoje bez složitějších přestupů.

I tato aplikace je závislá na internetovém připojení při vyhledávání spojů, i když zde existuje možnost procházet svoji offline historii spojů, která tuto závislost částečně eliminuje.

3.3 WMM Jízdní řády

Aplikace je určená výhradně pro mobilní platformu Windows, dostupná pro Windows 7.8, Windows 8, Windows 8.1, i nejnovější Windows 10. Jedná se v podstatě o jedinou nativní aplikaci sloužící k vyhledávání dopravních spojů pro tento systém, kterou mají uživatelé v České i Slovenské republice k dispozi. Stejně jako aplikace Jízdní řády IDOS, i WMM Jízdní řády získává data od firmy CHAPS spol. s.r.o.



Obrázek 3.4: Windows 10 mobile, zdroj: www.microsoft.com

3.3.1 Výhody

Hlavní nespornou výhodou je aktuálnost dat poskytovaných firmou CHAPS spol. s.r.o., stejně jako je tomu u Jízdních řádů IDOS. Díky tomu se uživatel může vždy spolehnout na aktuální informace o jednotlivých spojích. Zároveň má k dispozici podrobnější informace jako zpoždění spojů nebo výluky tratí.

Aplikace nabízí velice přehledné a přívětivé uživatelské rozhraní. I když podporuje nejnovější verzi Windows 10, plně těží z uživatelského rozhraní zavedeného ve Windows 8, jak můžeme vidět na obrázku 3.4. Je přehledně členěná na jednotlivé karty s rozšířenou nabídkou ve formě spodního panelu.

Disponuje různými funkcemi, jako například *moje poloha*, kdy aplikace vyhledá zastávky v okolí pomocí GPS, a následně zobrazí nejbližší odjezdy z těchto zastávek. Je možné uložit si své vlastní oblíbené trasy pro rychlé vyhledávání. To je zjednodušeno také díky našeptávači zastávek, nebo možností vyhledání zastávky pomocí GPS signálu. Lze také nastavit upozornění na konkrétní spoje. Další užitečnou funkcí aplikace je možnost zakoupit si SMS jízdenku. Aplikace dovoluje kombinovat různé druhy dopravy, ať už klasickou hromadnou, vlakovou, autobusovou, či například leteckou.

3.3.2 Nevýhody

Za nevýhodu může být považováno její omezení pouze pro mobilní zařízení s operačním systémem Windows. Po vzoru Jízdních řádu IDOS, i tato aplikace zobrazje informace o spojích

v textové formě. Je zde ovšem možnost zobrazení dráhy spoje na mapě, což ale slouží pouze k orientačním účelům.

Aplikace je zcela závislá na internetovém připojení při vyhledávání spojů a nedisponuje žádnou možností uložit si některé spoje pro pozdější použití offline. Za nevýhodu lze považovat i závislost na firmě CHAPS spol. s.r.o., kdy je pro provoz aplikace nutný chod jejich infrastruktury. Jedná se ovšem o zanedbatelné riziko.

Kapitola 4

Analýza požadavků

V této kapitole budou postupně shrnuty požadavky na výslednou aplikaci. Zároveň zde bude vysvětlena hlavní motivace pro tvorbu aplikace, myšlenky použité při jejím návrhu a její přístup k dané problematice.

4.1 Popis problému

Hlavním problémem, na který cílí tato aplikace, je grafické uživatelské rozhraní využívané v aplikacích pro zobrazení vlakových spojů. To má tradičně jednoduchou textovou podobu, jako je tomu u aplikací popisovaných v předchozí kapitole. Uživatel má tedy tradičně zobrazeny jednotlivé vlakové spoje z bodu A do bodu B, společně s časy odjezdu a příjezdu.

Cílem této práce je návrh aplikace, která bude k tomuto problému přistupovat poněkud odlišně. Konkrétně se pro zobrazení dat o jednotlivých spojích snaží využít *grafikon vlakové dopravy*. V současné době neexistuje nástroj, který by informace takto poskytoval široké veřejnosti.

4.2 Požadavky na aplikaci

Aplikace bude určena pro širokou veřejnost. Neexistuje tedy konkrétní skupina uživatelů a její používání by mělo být pro naprostou většinu lidí bezproblémové, přehledné a co možná nejjednodušší. S ohledem na dostupnost byla jako cílová platforma zvolena právě webová aplikace. Předpokládá se, že většina uživatelů má v dnešní době prakticky neomezený přístup k internetu, ať už prostřednictvím chytrých telefonů, tabletů, nebo klasických stolních počítačů. Požadavky na aplikaci lze shrnout do následujících bodů:

- jednoduché a přehledné ovládání,
- efektivní využití grafikonu vlakové dopravy,
- vizuální rozlišení zobrazovaných dat,
- možnost filtrace zobrazovaných dat,
- uživatelsky přívětivý vzhled aplikace,
- návrh a implementace vlastní databáze.

4.2.1 Uživatelské rozhraní

Důraz bude kladen na jednoduché ovládání. Aplikace by měla sloužit především k vyhledávání a porovnávání jednotlivých vlakových spojů, a neměla by obsahovat žádné rušivé či matoucí prvky. K tomuto účelu bude použit grafikon vlakové dopravy implementovaný formou interaktivního prvku, tzv. *widget*[1]. Samotná problematika vlakového grafikonu byla podrobněji popsána v kapitole 2. Aplikace by měla mít jednoduchý a příjemný vzhled, aby se uživatel při práci s ní cítil co možná nejlépe a měl důvod ji využívat opakovaně.

Bude nutné implementovat základní prvky pro interakci s uživatelem. Sloužit budou především pro zvolení tratě, případně výchozí a cílové stanice na dané trati. Kromě těchto prvků sloužících pro získání vstupů od uživatele bude implementován widget pro zobrazení odpovídajících výstupů.

Widget

Stěžejní částí celé aplikace bude interaktivní widget, který bude plnit roli vlakového grafikonu a bude graficky zobrazovat výsledné spoje. Tyto spoje bude nutné vhodně vizuálně odlišit, například barevně nebo silou čáry. Důležitá je také možnost filtrace zobrazovaných dat. Widget bude schopen filtrovat mezi jednotlivými typy vlaků, jako jsou:

- Os - osobní vlak,
- Sp - spěšné vlak,
- Ex - expres,
- Rx - ryhclík vyšší kvality,
- RJ - RegioJet,
- EC - EuroCity,
- EN - EuroNight.

Typy vlaků na jednotlivých tratích jsou závislé na dopravci [2]. Zároveň bude možné volit mezi zobrazením vlaků v jednotlivých směrech: tam a zpět. Ve výsledku tedy bude možné zobrazit si oba dva směry zároveň, nebo vždy jen jeden konkrétní směr.

Zobrazovat se budou jen spoje, které zastavují v každé ze zvolených stanic. Pokud tedy na trati existují spoje, které například vyjíždějí o zastávku později, než je ta výchozí, tento spoj se ve výsledném grafu nezobrazí. To i za předpokladu, že by zastavoval ve zvolené cílové stanici. Ve výchozím stavu, kdy uživatel zvolí pouze trať, budou za výchozí a cílovou stanici nastaveny první a poslední stanice na této trati.

Pro lepší orientaci mezi zobrazenými spoji bude možné v rámci grafu přibližovat pomocí nástroje zoom. Při přiblížení bude možné v nastavní zafixovat velikost osy Y, čímž bude docházet pouze k zoomu na časové ose. Toto chování je vhodné především při porovnávání rychlejších spojů, které staví jen v některých zastávkách. U osobních vlaků, které zastavují v téměř každé stanici na trase, je ale vhodné toto chování eliminovat a porovnávat je mezi jednotlivými stanicemi.

[1] Grafický element sloužící k interakci uživatele s programem.

4.2.2 Databáze

Pro funkci aplikace je nutné mít určitou formou uložená data o jednotlivých spojích. Oficiálním poskytovatelem těchto dat v České republice je firma CHAPS spol. s.r.o. Tato data jsou zpoplatněná a poskytuje je pouze registrovaným klientům. Samotná data ovšem jsou volně dostupná skrz aplikace, které je využívají. Jediné, co dostupné není, je aplikační rozhraní k těmto datům a není tedy možný jejich přímý zisk za pomoci programu. K vytvoření databáze pro aplikaci je tedy nutné zvolit vhodnou platformu a následně ručně nebo za pomocí vlastního skriptu opsat potřebná data pro demonstraci funkčnosti aplikace.

Kapitola 5

Analýza dostupných technologií

V této kapitole budou popsány některé dostupné knihovny pro webovou platformu s možností tvorby vlastních diagramů. Bude zde vysvětleno, v čem vyhovují našim požadavkům, nebo naopak z jakého důvodu nejsou vhodné pro naše řešení.

Vzhledem ke zvolené webové platformě a architektuře *klient-server* budeme pro vykreslení grafu využívat výpočetní zdroje na straně klienta. To znamená využití jazyka JavaScript a některé z jeho knihoven.

5.1 Knihovny jazyka JavaScript pro tvorbu grafů

5.1.1 CanvasJS

Jde o knihovnu jazyka JavaScript sloužící pro práci s *HTML5 canvas* [10]. Hlavní výhodou této konkrétní knihovny je její účelnost a přehlednost při tvorbě vlastních diagramů[1]. Oproti ostatním knihovnám pracujícím s *HTML5 canvas* není zbytečně obsáhlá a zaměřuje se čistě jen na problematiku spojenou s tvorbou grafů, což je v rámci této aplikace klíčové. Nabízí snadnou implementaci různých užitečných prvků, jako je například zoom v rámci grafu, nebo dialogové okno s dodatečnými informacemi k danému bodu. Vše je navíc v poměrně moderní a jednoduché grafice, která je také nedílnou součástí uživatelsky přívětivého GUI. Jelikož se jedná o velice propracovanou knihovnu často využívanou velkými firmami jako Microsoft, Apple, BMW nebo Samsung, je pro komerční účely dostupná pouze pod placenou licencí.

5.1.2 D3

Knihovna D3 (známá také jako *Data-Driven Documents*) je *open-source* knihovna jazyka JavaScript sloužící k interaktivní a dynamické vizualizaci dat, pro kterou využívá prvky HTML, CSS a SVG grafiky. Jedná se o velice výkonnou knihovnu, která ovšem nenabízí žádné předpřipravené základní grafy, jako většina ostatnich knihoven[2]. Nabízí ovšem sadu šablon pro určité typy grafů, podle kterých se lze při tvorbě řídit[3]. Poskytuje prakticky neomezenou kontrolu nad vzhledem a vlastnostmi jednotlivých částí výsledného grafu. Umožňuje navázat libovolná data na objektový model dokumentu (*DOM*) a následně je trasnformovat nebo s nimi různě manipulovat.

[1] http://canvasjs.com
[2] https://d3js.org
[3] https://github.com/d3/d3/wiki/Gallery

Její hlavní nespornou výhodou je právě zmíněná naprostá kontrola nad vzhledem a chováním grafu, která s sebou ovšem přináší i nevýhodu ve formě neexistujících předpřirpavených grafů. Konkrétně v našem případě požadujeme klasický lineární graf s nástroji jako například zoom, který tato knihovna implicitně neobsahuje.

5.1.3 Plotly

Původně placená knihovna, která je od 17. 11. 2015 zpřístupněna pod *open-source* licencí [6]. Jedná se o první vědeckou knihovnu pro tvorbu grafů určenou pro webovou platformu, která na pozadí využívá právě knihovnu D3. Každý graf vytvořený pomocí knihovny Plotly lze tedy poměrně jednoduše přepsat pro knihovnu D3. Oproti ní je však daleko více uživatelsky přívětivější. Podporuje více jak 20 typů grafů. V rámci *open-source* licence obsahuje rozhraní pro programování aplikací v jazycích Matlab, Python, JavaScript a jazyku R[4]. Jedná se o velice propracovanou knihovnu, která poskytuje v rámci každého grafu předdefinovaný ovládací panel s nástroji jako zoom, posunutí, výběr oblasti, nebo uložení grafu ve formátu PNG.

Jedná se o uživatelsky přívětivou knihovnu, s moderním vzhledem. Ovládání je díky panelu s nástroji intuitivní. Velice dobře se vyrovnává s vykreslením velkého množství zobrazovaných dat a při následném přiblížení/oddálení nebo posunu v oblasti grafu nedochází k žádnému viditelnému zpomalení či trhání. Tato skutečnost může být výhodou především u dlouhých tratí s velkou hustotou spojů a zastávek.

5.1.4 Chartist

Knihovna Chartist klade důraz především na jednoduchost a minimalismus. Jedná se o volně dostupnou knihovnu, šířenou pod licencí *open-source*, implicitně určenou pro tvorbu různých grafů a diagramů. Její diagramy jsou označovány jako *DPI independent*, což znamená, že jsou zobrazovány ve stejném rozlišení napříč různými zařízeními, jako mobil, tablet, nebo klasický počítač[5]. Jsou tedy plně responsivní. Je kompatibilní s naprostou většinou internetových prohlížečů. Knihovna také nabízí jednoduchý a moderní vzhled, stejně jako různé animace při vykreslování grafů.

I vzhledem k jejím výhodám jsem ale velice postrádal možnost zobrazení podrobnějších informací o uzlu v plovoucím okně při najetí kurzorem nad tento uzel, které by sloužily jako informace o konkrétním spoji. Postrádá také možnost implementace nástroje zoom. Je dostupná také pro offline použití bez nutnosti připojení k serveru.

5.1.5 Chart

Tato knihovna je, stejně jako Chartist, navržena v duchu jednoduchosti, minimalismu a tzv. *flat design*. Je také plně responsivní a šířená pod licencí *open-source*. Je vhodná především pro malé projekty[6]. Nabízí základní typy grafů, jako lineární, sloupcový, kruhový, atp. Využívá *HTML5 canvas* pro vykreslování. Její zajímavostí je možnost mixování více typů grafů do jednoho výsledného grafu.

Oproti výše zmíněné knihovně Chartist nabízí postrádanou možnost zobrazení informací u jednotlivých uzlů. I grafické zpracování a použité barvy jsou dle mého názoru lepší, než u knihovny Chartist. Bohužel i zde chybí jakákoliv implementace nástroje zoom.

[4] https://plot.ly/api/
[5] https://gionkunz.github.io/chartist-js/
[6] http://www.chartjs.org

5.1.6 Google chart

Knihovna sloužící k tvorbě grafů od vývojářů firmy Google. Je poskytována zdarma, ovšem zdrojové kódy nejsou dostupné a po poslední změně uživatelských práv je pro její využití vyžadováno připojení k serveru firmy Google. K dispozici je rozsáhlá a přehledná online dokumentace[7]. Díky své mateřské firmě má knihovna širokou škálu uživatelů a nabízí dobrou podporu prostřednictvím aktivních diskuzních fór.

Knihovna poskytuje implementaci prvků jako dodatečné modifikovatelné informace k jednotlivým bodům grafu. Poskytuje také experimentální formu nástroje zoom, která ovšem prozatím bohužel nefunguje pro všechny typy grafů a trpí nedostatky jako invertovaný směr přiblížení a oddálení kolečkem myši. Jedná se ovšem o dobře známou chybu a lze čekat její napravení v dalších verzích. Jak se také ukázalo, knihovna se při použití přiblížení a následném pohybu s oblastí grafu hůře vypořádává s překreslením většího množství dat, což podstatně ovlivňuje uživatelský zážitek. Poměrně dlouho ji trvá také samotné vykreslení statického grafu. Výkonnostně je na tom podstatně hůře než ostatní knihovny.

5.2 Dostupné databázové systémy

Tato sekce pojednává o některých dostupných databázových systémech, které lze využít při implementaci aplikace. Budou zde popsané jak tradiční standartizované relační databáze, tak poměrně nové, tzv. nerelační (*noSQL*) databáze, využívané v některý moderních webových aplikacích.

5.2.1 Relační databáze

Relační databáze jsou vhodné pro řízení hromadného přístupu k velkému množství dat. Každá taková databáze zajišťuje integritu uchovávaných dat skrz integritní omezení. Její hlavní výhodou je její standartizace, zahrnující například i jednotný dotazovací jazyk SQL do všech relačních databází [1]. Zároveň je tato technologie dostupná prakticky na každém webhostingu. Za nevýhodu by se dala označit poněkud omezená manipulace s daty oproti jiným typům databází, která je ovšem pro tuto aplikaci plně postačují.

MySQL

Jedná se o systém řízení báze dat (*SŘBD*) pro relační databázový model. MySQL byl původně vytvořen firmou MySQL AB, nyní je vlastněn firmou Oracle Corporation, dceřinou společností firmy Sun Microsystems. Je dostupná jak pod bezplatnou licencí GPL (*General Public License*), tak pod komerční placenou licencí[8].

MySQL je multiplatformní databáze využívající dotazovací jazyk SQL. V současné době zabírá vysoký podíl mezi používanými databázemi, především pro její snadnou implementaci a vysoký výkon. Na tomto faktu se podílí i skutečnost, že se jedná o volně šířený software. Její výhodou je možnost použití bez internetového připojení. Poskytuje server jako samostatný program pro síťové prostředí typu klient-server. Obsahuje komponenty jako úložiště dat InnoDB pro zpracování transakcí, které je velmi blízké principům ACID.

[7] https://developers.google.com/chart/
[8] https://www.mysql.com

PostgreSQL

Dalším systémem řízení báze dat je PostgreSQL, někdy zjednodušeně označovaný jako *Postgres*. Je šířen pod licencí *MIT* (*Massachusetts Institute of Technology*), takže se také jedná o volně šířený svobodný software. Na rozdíl od MySQL není vlastněn žádnou firmou, ale je spravován a vyvíjen komunitou[9].

Primárně je vyvíjen pro unixové systémy, existují ale i balíčky pro platformu Windows. Jeho výhodou je možnost vytváření vlastních datových typů a dotazovacích metod. Umožňuje spouštět procedury typu *Trigger* a také uložené procedury v různých programovacích jazycích, jako například Java, Perl, Python, Ruby atp. Databáze efektivně splňuje principy ACID

5.2.2 Nerelační databáze

Hlavním problémem, kterému čelí nerelační databáze, je absence jakékoliv její standartizace. Jedná se o poměrně novou technologii, která si získává oblibu především u webových aplikací. Její výkon je ovšem silně závislý na struktuře používaných dokumentů [7]. I když vhodně zvolená implementace může vést v některých případech k lepším výsledkům, než u relační databáze, mohou nastat případy, kdy i poměrně elementární dotazy do databáze zaberou daleko více času. Z tohoto důvodu se jeví jako lepší možnost zvolit dlouhodobě osvědčenou relační databázi.

MongoDB

Jedná se o multiplatformní dokumentovou databázi. Místo relací využívaných v relačních databázích používá dokumenty podobné formátu JSON (konkrétně BSON) společně s dynamickým databázovým schématem[10]. To vše přispívá k rychlejší a jednodušší integraci dat pro aplikace. Je volně šířena jako Open-source software pod GNU licencí. Jedná se o nejpopulárnější nerelační databázový systém. Mezi hlavní funkce patří také vykonávání JavaScriptu na straně serveru, což znamená, že JavaScript může být použit v dotazech, agregačních funkcích a následně být přímo odeslán do databáze, aby se provedl.

CouchDB

Další dostupnou dokumentově orientovanou databází je CouchDB. Pro uložení dat v databázi využívá formát JSON[11]. Je označována jako *ACID compliant* (kompatibilní s modelem ACID). Stejně jako MongoDB může být používána k ukládání dat pro webové stránky. Poskytuje velice přívětivou webovou administrační konzoli pro správu databáze. Jedná se o velice dobré řešení pro webové aplikace. Je šířena jako svobodný *open-source* systém.

[9] https://www.postgresql.org
[10] https://www.mongodb.com
[11] http://couchdb.apache.org

Kapitola 6

Návrh aplikace

Tato kapitola se zabývá návrhem grafického uživatelského rozhraní a aplikační logiky výsledné aplikace.

6.1 Struktura aplikace

Při návrhu a následné implementaci aplikace je nutné rozdělit aplikaci na 2 části:

1. grafické uživatelské rozhraní (*GUI*),
2. aplikační logika.

Grafické uživatelské rozhraní se bude starat o celkovou interakci s uživatelem i o následnou vizualizaci dat v podobě interaktivního widgetu vlakového grafikonu. Základními prvky grafického uživatelského rozhraní bude několik polí pro vstupní text od uživatele, pomocí kterých uživatel zvolí danou trať, případně výchozí a cílový bod vlakového spoje na této trati spolu s konkrétním datem provozního dne. Dále budou přítomny prvky pro filtraci zobrazovaných dat. Uživatel bude mít možnost filtrovat spoje jednak pomocí jejich typu, ale také směru jízdy. Následně se na základě vstupních dat vše zobrazí uživateli v přehledném vlakovém grafikonu.

Aplikační logika bude poskytovat základní funkcionalitu, především tedy komunikaci klienta se serverem a s databází. Na základě dat získaných od databáze následně vykreslí příslušný grafikon vlakové dopravy.

6.2 Použité technologie

Na základě provedených analýz v kapitolách 5 a 5.2 budou pro jednotlivé části aplikace použity tyto technologie.

6.2.1 Výběr knihovny pro tvorbu grafikonu

Pro tento účel poslouží knihovna Plotly [5]. Její hlavní výhody jsou kombinace velké míry přizpůsobení jednotlivých částí grafu vycházející z knihovny D3.js, kterou využívá na pozadí, s předpřipravenými strukturami pro jednotlivé grafy. Oproti ostatním knihovnám tedy kombinuje vysoký výkon se snadnou použitelností. Knihovna podporuje také možnost zoomu v rámci grafu, vzhledem k pozici kurzoru myši. Jelikož se jedná o knihovnu určenou

primárně pro výzkumné účely, nedělá jí problém vykreslení většího množství dat. Splňuje tedy všechny požadavky pro naši aplikaci.

6.2.2 Výběr databázového systému

V kapitole 5.2.2 byly zmíněny hlavní nevýhody nerelačních databází. Z tohoto důvodu pro naší aplikaci použijeme klasickou relační databázi. Konkrétně MySQL, která je jednou z nejpoužívanějších relačních databází. Navíc je přítomná ve většině webových hostingů. Relační databáze nám zajistí bezproblémový hromadný přístup k datům, což je vzhledem k předpokládanému využití aplikace jednou z klíčových vlastností. Další výhodou relačních databází je zajištění integrity dat.

6.3 Návrh grafického uživatelského rozhraní

Aplikace bude implementovaná jako *SPA (single-page application)* [3]. To znamená, že se bude jednat o webovou aplikaci využívající pouze jednu webovou stránku s cílem poskytnout uživateli stejné zkušenosti, jako při práci s klasickou desktopovou aplikací. Jedná se o webovou stránku s dynamicky se měnícím obsahem na základě interakce s uživatelem. Využívá technologie jako JavaScript, HTML a CSS. Dynamičnost stránky bude zajištěna pomocí technologie AJAX.

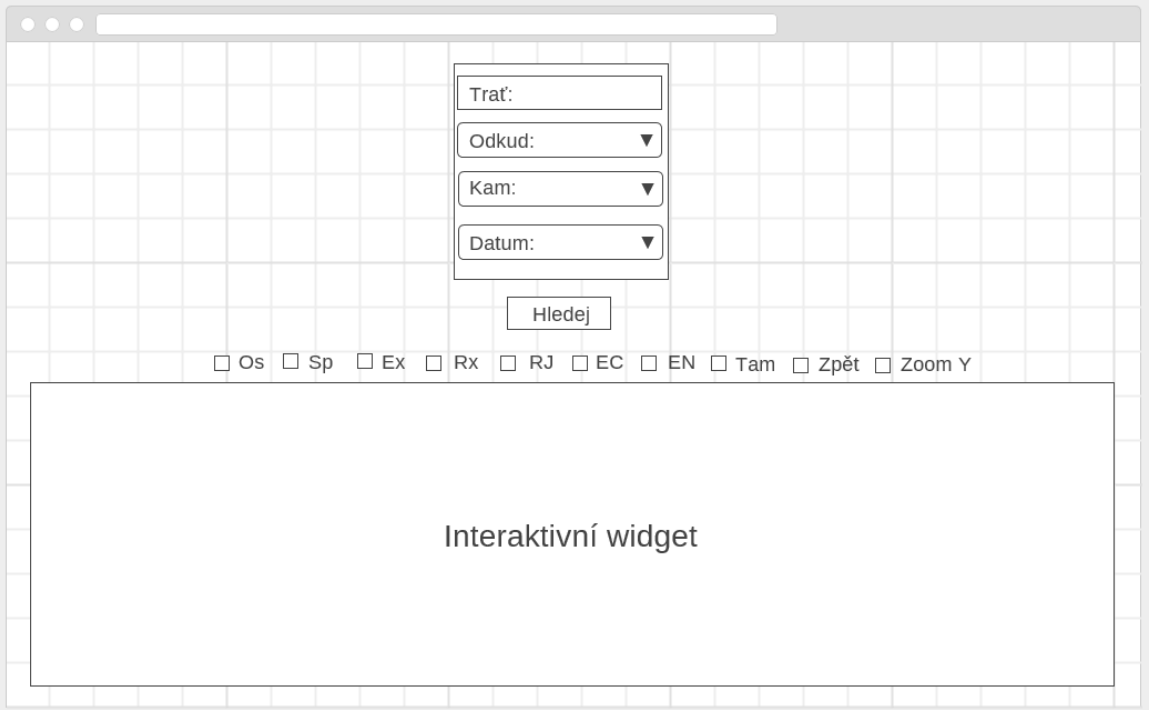
Při návrhu jednotlivých prvků uživatelského rozhraní bychom měli dodržovat některá základní pravidla popsána v knize [4]:

- Logicky rozdělit prvky aplikace.
- Dodržet účelnost jednotlivých prvků.
- Stanovit vizuální hierarchii aplikace.
- Vzhled jednotlivých prvků by měl odpovídat jeho funkci.
- Zvolit vhodné barevné kombinace prvků.
- Zvolit vhodné barevné rozlišení spojů (viz *daltonismus*).

6.3.1 Prvky pro zpracování vstupů od uživatele

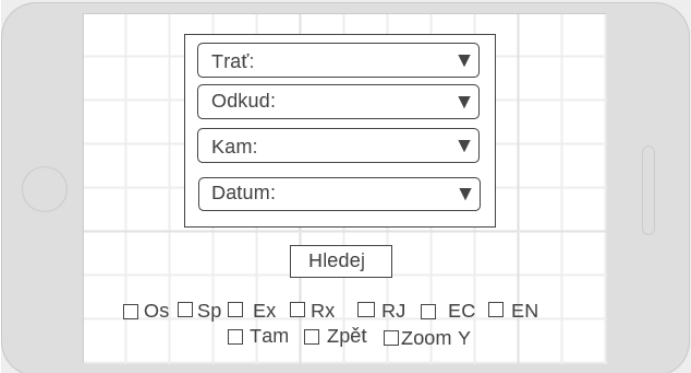
Aplikace bude pro komunikaci s databází potřebovat základní informace, jako je číslo trati, výchozí a cílový bod. Pro tyto účely bude aplikace obsahovat předpřipravený seznam jednotlivých tratí označených svým oficiálním číslem. Jelikož čísla jednotlivých tratí nemusí být známá široké veřejnosti, budou v názvu doplněna o výchozí a cílovou stanici, čímž se uživateli usnadní výběr. Následně se na základě zvolené tratě vytvoří seznamy jednotlivých stanic, ze kterých si bude uživatel moci zvolit. Bude zde implementováno tlačítko pro odeslání dotazu do databáze.

Dále budou přitomny prvky pro filtraci zobrazovaných dat. Provoz na některých tratích s velkou hustotou spojů může při grafickém zobrazení působit nepřehledně. Proto je důležité, aby uživatel mohl svůj výběr specifikovat například omezením typů vlaků, nebo směrem provozu na trati. Takto specifikovat požadované spoje může uživatel buď před jejich vykreslením, nebo dynamicky po vykreslení. Vzhled aplikace v desktopovém prostředí prohlížeče s jednotlivými ovládacími prvky je ilustrován na obrázku 6.1. Dále na obrázcích 6.2 - 6.4 je znázorněna podoba aplikace na mobilních zařízeních.



Obrázek 6.1: Drátěný model aplikace v desktopovém prostředí

Obrázek 6.2: Drátěný model aplikace na mobilním zařízení



Obrázek 6.3: Drátěný model aplikace na mobilním zařízení na šířku - konfigurace

Obrázek 6.4: Drátěný model aplikace na mobilním zařízení na šířku - widget

Z obrázků 6.2 - 6.4 je patrné, že aplikace bude podporovat responsivní vzhled. Bude tedy pro uživatele dostupná napříč různými zařízeními, i když primárně je zamýšlena pro desktop. Jak lze spatřit na obrázku 6.1, podstatnou část obrazovky zabírá právě interaktivní widget vlakového grafikonu, jelikož je stěžejní částí aplikace. Ten se přizpůsobí maximální možné velikosti podle velikosti obrazovky. Hned nad ním budou umístěné jednotlivé ovládací prvky, aby byly uživateli snadno přístupné při následné filtraci zobrazovaných dat.

Z mobilního vzhledu na obrázku 6.2 vyplývá, že prohlížení widgetu na mobilním zařízení v režimu na výšku není zvlášť uživatelsky přívětivé. Problém lze vyřešit například tím, že oblast pro vykreslení grafu bude fixně nastavena na větší rozměr, než reálně vykreslovaná oblast. Následně se bude uživatel moci v oblasti widgetu horizontálně posouvat. Mnohem lépe využívání aplikace vyhovuje režim na šířku, kdy je poté daleko více místa na zobrazení samotného widgetu. V tomto režimu nejdříve uživatel zvolí danou konfiguraci widgetu a následně se po jeho vykreslnení posune na obrazovce směrem dolů.

6.3.2 Interaktivní widget

Datové výstupy reprezentující vlakové spoje budou uživateli zobrazeny prostřednictvím interaktivního widgetu. Tento widget bude vytvořen pomocí knihovny Plotly jazyka JavaScript. Konkrétně bude využit *lineární graf (Line chart)*, kobinující dvě osy:

1. číselnou osu Y,
2. časovou osou X.

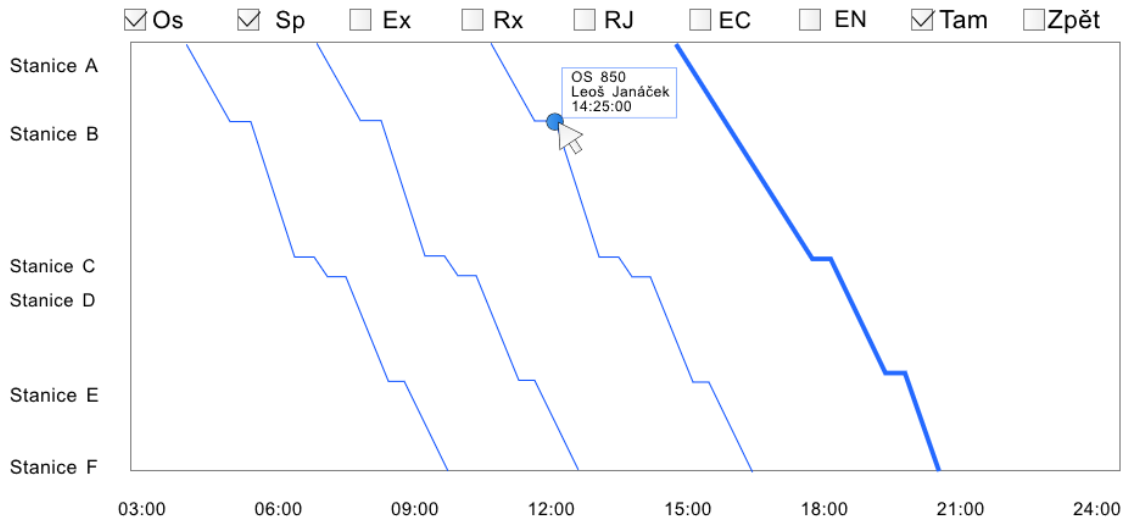
Stanice na ose Y budou zadány pomocí svých číselných hodnot reprezentujících vzdálenost na trase. Tyto číselné hodnoty bude nutné později nahradit jejich názvy. Dále bude třeba ponechat zobrazené pouze hodnoty příslušící některé ze stanic. Ostatní hodnoty, které knihovna automaticky doplňuje, bude zapotřebí skrýt. Je to důležitá úprava vzhledu, aby výsledný graf přesně odpovídal vlakovému grafikonu.

Časová osa X bude zdánlivě nekonečná, aby uživatel mohl grafem libovlně posouvat zleva doprava a nebyl omezený vykresleným prostorem, přestože zobrazená data budou příslušet vždy jednomu konkrétnímu dni. U nočních vlaků, které mají odjezd těsně před koncem dne, musíme zajistit jejich vykreslení i v následujícím dni.

Vykreslovat se budou do grafu jednotlivé série dat pro každý spoj, tvořené dvojicemi hodnot určující bod – číselným údajem vzdálenosti pro osu Y a časovým údajem pro osu X. Ke každému takovému bodu bude možné dopsat vlastní informace, které se zobrazí po přejetí myší nad tímto bodem, jako je znázorněno na obrázku 6.5. Tyto informace budou tvořené:

- typem vlaku,
- číslem vlaku,
- jménem vlaku,
- časem příjezdu/odjezdu do/ze stanice.

Tyto body se následně spojí a vznikne posloupnost přímek pro každý spoj. U každé přímky budou na základě typu vlaku specifikované další vlastnosti, jako je barva nebo síla přímky. Tyto vlastnosti jsou důležité pro snadnou a rychlou orientaci v grafu.



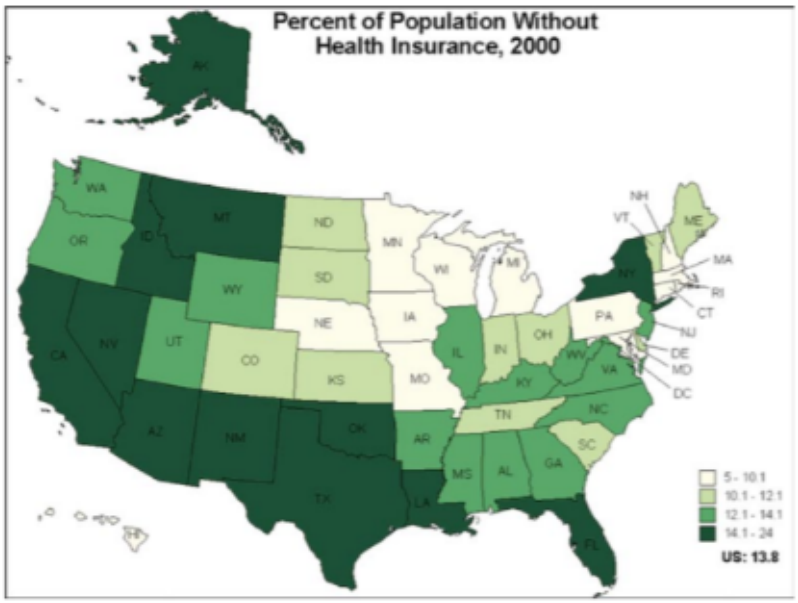
Obrázek 6.5: Drátěný model interaktivního widgetu

Rozlišení spojů

Pro barevné rozlišení spojů je vhodné použít takové barvy, na které je lidské oko nejvíce citlivé. Obecně se využívá vysokého kontrastu mezi jednolivými barvami. Lidské oko využívá pro jejich vnímání 3 kanály, každý pro jednu z následujících barev: červenou, zelenou a modrou. Neboli barevný *RGB* model (red, green, blue) [11]. Nejlépe lidské oko rozlišuje takové barvy, které vyvolají co nejsilnější signál na jednom z těchto kanálů a velice slabý nebo žádný signál na ostatních kanálech. Mezi tyto barvy patří černá, bílá, červená, zelená, žlutá a modrá (znázorněno na obrázku 6.6). Zároveň ale není vhodné používat příliš intenzivní barvy. Takové barvy většinou slouží především pro upoutání pozornosti a jejich použití by mělo být opodstatněné a spíše vyjímečné.

Obrázek 6.6: Nejlépe rozlišitelné barvy [4]

Lidé s poruchou barvocitu ovšem mohou mít problémy s vnímáním některé z těchto barev. Jako nejlepší přístup se poté jeví použití pouze jedné barvy, přičemž se k rozlišení použijí její jednotlivé odstíny, spolu s kombinací různé síly jednotlivých čar. V takovém grafu poté bude rozdíl patrný pro jakéhokoliv uživatle, ať s poruchou nebo bez poruchy barvocitu. Tohoto principu, který je také znázorněný na obrázku 6.5, využívá graf na obrázku 6.7.



Obrázek 6.7: Využití odstínů jedné barvy pro rozlišení dat [4]

6.4 Návrh aplikační logiky

Základními prvky budou políčka pro selekci tratě, výchozí a cílové stanice a data, která budou tvořit formulář. Po výběru určité tratě z připraveného seznamu se následně doplní seznamy stanic pro příslušné vstupy. Za pomoci zadaných dat se po potvrzení formuláře získá požadovaná posloupnost dat pro jednotlivé spoje. Tato data se použijí pro vykreslení interaktivního widgetu. Samotná URL adresa se v jednotlivých krocích nebude měnit a zároveň nebude docházet k opětovnému načítání stránky. Uživateli se tedy vše bude jevit plynulé a intuitivní. Sdílení aktuální konfigurace grafu bude realizováno pomocí vygenerovaného odkazu. Jednoduše postačí zkopírovat tento odkaz a přeposlat ho jinému uživateli.

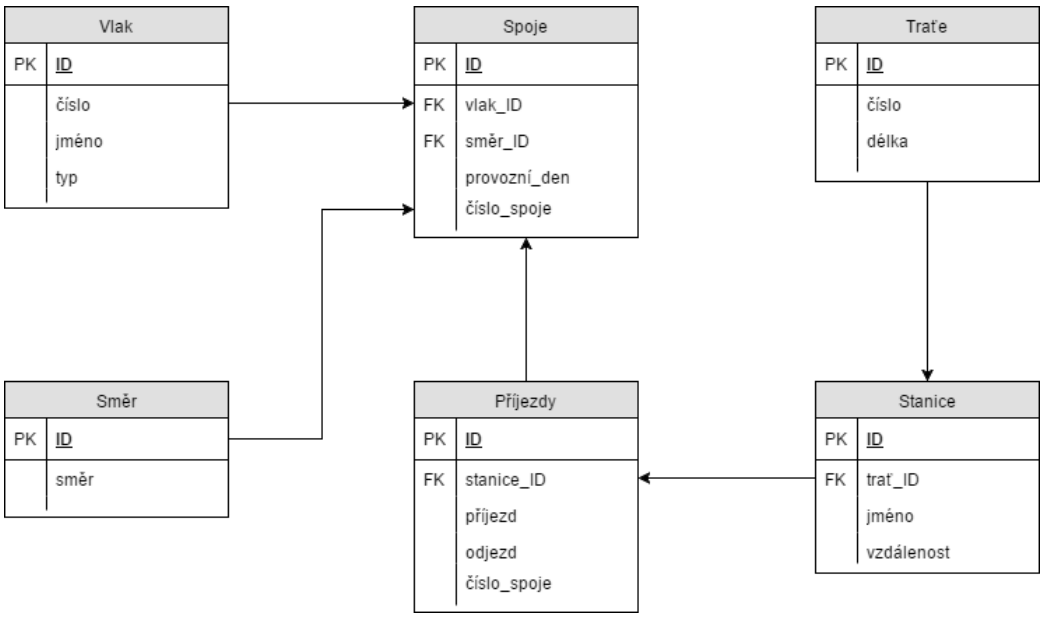
U prvků sloužících k nastavení vlastností grafu, konkrétně filtrace zobrazovaných dat, bude nutné rozlišovat, zda byly změněny před a nebo po vykreslení grafu. Pokud se bude jednat o konfiguraci před vykreslením, dojde pouze k nastavení příslušných proměnných a bude se kontrolovat potvrzení formuláře, případně stisknutí klávesy *Enter*, indikující vykreslení grafu. Pokud je graf již vykreslen, jakákoliv změna těchto prvků se okamžitě projeví dynamickým překreslením grafu.

6.5 Návrh databáze

Pro funkčnost aplikace je nutné implementovat vlastní databázi, která bude obsahovat potřebné informace o jednotlivých spojích. Tato databáze musí obsahovat následující informace:

- tratě,
- stanice,
- vlaky,
- příjezdy/odjezdy do/ze stanic,
- spoje.

6.5.1 Struktura databáze



Obrázek 6.8: Návrh struktury databáze

Obrázek 6.8 znázorňuje návrh databáze. O každém vlaku budou uchovány informace o jeho typu, čísle a jméně. Ke každému vlaku následně přísluší neomezený počet spojů v konkrétní den a v daném směru. Tento spoj obsahuje určitý počet příjezdů do jednotlivých stanic s časy odjezdu a příjezdu. Jsou zde uloženy také informace o jednotlivých tratích, jako je jejich oficiální číslo a celková délka trati. Ke každé trati přísluší stanice, které obsahují své jméno a vzdálenost na trati.

Tato databáze počítá pouze se zobrazením dat provozu na jedné konkrétni trati. Není zde řešen problém, kdy uživatel zadá cestu mezi dvěma stanicemi nacházejícími se na různých tratích. Pro tyto účely je zapotřebí vymyslet sofistikovanější řešení. Dále je možné databázi rozšířit o tabulku provozních dní. Ta by uchovávala informace o jednotlivých dnech v roce,

přičemž by mohla obsahovat další informace, jako například zda je svátek, víkend, nebo pracovní den. V tabulce Spoje by se poté používalo ID tohoto dne jako cizí klíč namísto konkrétního data.

6.5.2 Získávání dat

Pro oficiální zdroj dat k vlakovým spojům neexistuje žádné volně dostupné API. Z tohoto důvodu bude nutné data získat jiným způsobem, například prostřednictvím volně dostupných jízdních řádů, nebo některé z existujících aplikací. Protože se ovšem jedná o poměrně velké množství dat, jejichž ruční zapisování do databáze by bylo poněkud zdlouhavé, a bylo by jednoduché udělat při tomto zápisu chybu, bylo by vhodné implementovat jednoduchý skript, který by tato opsaná data automaticky zpracoval, a následně jimi naplnil databázi, nebo je alespoň převedl na posloupnost SQL příkazů.

Kapitola 7

Implementace

V této kapitole bude vysvětleno, jak jednotlivé části aplikace spolupracují na vykreslení interaktivního widgetu, jak jsme získali data o vlakových spojích a také zde popíšeme databázi, ve které jsou tato data uložena.

7.1 Databáze

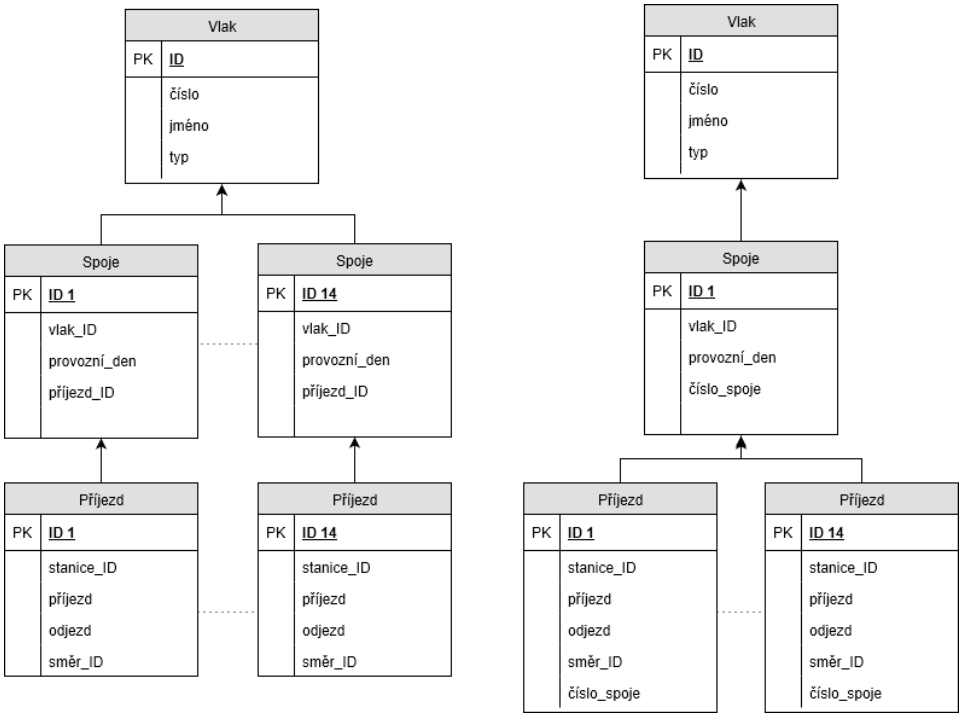
Začneme samotnou databází, bez které by byla tato aplikace prakticky nepoužitelná. Jedná se o klasickou relační databázi, konkrétně MySQL, která byla součástí webového hostingu, na kterém aplikace běží. Schéma této databáze bylo ilustrováno v kapitole 6.5.1 na obrázku 6.8. Toto schéma plně pokrývá naše požadavky na data, která potřebujeme pro chod aplikace. Při návrhu databáze bylo potřeba dbát na tyto věci:

1. Více spojů může patřit pod jeden vlak.
2. Jeden vlak může jet vícekrát během dne.
3. Jednotlivé spoje bývají neměnné ve všední dny / o víkendu.
4. U jednotlivých spojů je potřeba rozlišit jejich směr.
5. Noční spoje dojíždí do cílové stanice následující den.
6. Některé vlaky nezastavují ve všech zastávkách.

První dva body mají jednotné řešení, a to sice shromažďovat více příjezdů pod jedním identifikátorem značícím konkrétní spoj. Ke každému vlaku se poté budou vázat jednotlivé identifikátory příjezdů, podle jejichž hodnot se následně dohledají požadované příjezdy do stanic.

Třetí bod souvisí se snahou eliminovat redundantní data v databázi. Je potřeba si uvědomit, že většina spojů ve všední dny a o víkendu (kromě svátků) jezdí ve stejné časy. To v praxi znamená, že není potřeba pro každý den zvlášť ukládat nové a nové příjezdy, pouze se vytvoří nový spoj s konkrétním datem a číslem spoje, kterému se následně přiřadí určitý vlak. Vazbu k jednotlivým příjezdům zařídí právě hodnota značící číslo spoje. Kdyby například na dané trati bylo až 14 zastávek, museli bychom v rámci pracovního týdne bez tohoto řešení vytvořit celkem 70 záznamů o příjezdech, a to pouze pro jeden vlak. Po krátké době bychom měli databázi plnou zbytečných dat. Zároveň toto řešení způsobí, že nám postačí jen jeden záznam o spoji pro všech 14 příjezdů, než by tomu bylo v případě,

kdy bychom místo čísla spoje použili identifikátor příjezdu. V tomto případě by na jeden záznam příjezdu připadl jeden záznam spoje. Budeme-li dále uvažovat 14 zastávek na trati, máme celkem 14 záznamů o spoji, které všechny patří v jeden den k jednomu vlaku. Graficky je tento problém znázorněný na obrázku 7.1:



Obrázek 7.1: Optimalizace databáze

Z obrázku 7.1 je patrné, že před optimalizací bychom měli pro jeden konrétní vlak v konrétní den celkem 28 záznamů, za předpokladu 14 zastávek na trati. Optimalizovaná verze obsahuje pouze 15 záznamů. V rámci pracovního týdne to celkově činí 140 záznamů v neoptimalizované databázi, oproti 19 záznamům po optimalizaci, což už je značná úspora.

Čtvrtý bod týkající se rozlišení směru jednolivých spojů souvisí s možností filtrace zobrazovaných dat v interaktivním widgetu. Jak bylo zmíněno v požadavcích na aplikaci v kapitole 4.2.1, je potřeba rozlišit, zda spoj jede směrem z počáteční do koncové stanice, nebo opačně. Z tohoto důvodu byla v databázi zavedena tabulka *Směr*, jejíž identifikátor je použit jako cizí klíč u každého záznamu o spoji a značí jeho směr.

Pátý bod uvažuje noční vlaky (např. EN - EuroNight), které vyjíždějí chvíli před půlnocí a do cílové stanice dorazí následující den. To je problém především kvůli skutečnosti, že uživatel během konfigurace grafu zadává konkrétní datum, pro které chce spoje vyhledat. Pro správné vykreslení grafu ale část příjezdů souvisejících s tímto spojem obsahuje jiné datum, než to, pro které je hledání realizováno. Je potřeba tyto spoje hlídat, a přizpůsobit jim dotazy do databáze, aby se vrátila ta správná data.

Šestý a poslední bod se týká situace, kdy některé vlaky, jako například spěšné vlaky, vlaky typu Expres, RailJet atp., nezastastavují ve všech zastávkách. Můj prvotní přístup k tomuto problému byl takový, že jsem takové záznamy příjezdů pro tyto vlaky nevytvářel. Ovšem z důvodu jednotného spracování a vytvoření dat ve formátu JSON pro všechny typy vlaků bylo zapotřebí nastavit pro tyto vlaky hodnoty časů příjezdů a odjezdů na NULL.

7.1.1 Data

Jak již bylo konstatováno v kapitole 6.5.2, jediný možný způsob, jak získat potřebná data, je za pomoci již existujícího jízdního řádu. Z důvodu již zmíněných nevýhod ručního opisování dat do databáze jsem se rozhodl pro implementaci skriptu v jazyce Python, který převede data z textového souboru na posloupnost SQL příkazů, které následně vytvoří a naplní databázi. Tento skript očekává data v následujícím formátu:

```
číslo_vlaku  typ_vlaku  jméno_vlaku
čas_odjezdu  čas_příjezdu
```

Obrázek 7.2: Formát dat pro skript

Jednotlivé údaje z obrázku 7.2 musí být odděleny přesně dvěmi mezerami. Časy odjezdu a příjezdu jsou uvažovány od počáteční stanice ke koncové, vždy jeden řádek pro jednu stanici. V případě, že vlak v této stanici nestaví, zapíše se místo času hodnota NULL. Při spuštění skriptu si uživatel nastaví vstupní a výstupní soubor, počet zastávek a počáteční číslo identifikátoru pro spoje a příjezdy.

7.2 Zadní část aplikace

Zadní část aplikace je tvořena skripty v jazyce PHP, které jsou volány asynchronně pomocí technologie AJAX. V této kapitole budou tyto skripty sloužící pro komunikaci s databází popsány, bude vysvětlen jejich účel a využití.

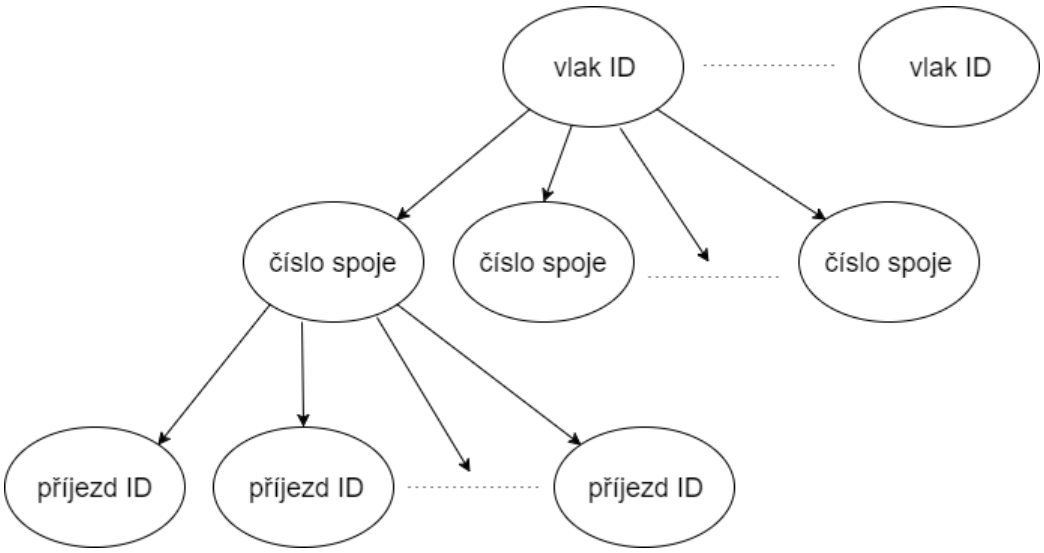
7.2.1 Serializace dat

Knihovna Plotly, která je použitá v aplikaci pro vykreslení interaktivního widgetu, požaduje vstupní data ve formátu JSON. Hlavní skript aplikace, který obstarává tuto funkci, je skript `getDataPlotly.php`. Tomuto skriptu je předán textový řetězec ve formě lokální proměnné, reprezentující aktuální konfiguraci grafu. Na začátku tohoto skriptu se převezmou jednotlivé hodnoty řetězce metodou POST a uloží se do jednotlivých lokálních proměnných skriptu. Následně skript podle obdržených dat vykoná tyto kroky:

1. Zjistí a uloží identifikátory počáteční a koncové stanice.
2. Porovná, zda jsou tyto stanice na stejné trati.
3. Zjistí veškeré stanice, které se mezi nimi nachází.
4. Pro tyto stanice zjistí veškeré spoje v daný den pro jednotlivé vlaky.
5. Uloží k vlakům dodatečné informace.
6. Vytvoří se pole s konečnými daty pro osu Y.
7. Pro jednotlivé vlaky k daným stanicím uloží časy odjezdu a příjezdu.
8. Pomocí této struktury vytvoří konečná data ve formátu JSON.

První a druhý krok slouží především pro kontrolu, zda má smysl zahájit vyhledávání. Aktuální verze aplikace nepočítá s možností, že by se widget vykresloval pro stanice nacházející se na různých tratích, ale počítá se se zobrazením provozu pro jednu konkrétní trať.

Ve třetím kroku se vytvoří celkově dvě pole typu *klíč : hodnota*, jedno s názvy stanic a druhé s jejich vzdálenostmi na trati. V obou případech jako klíče slouží identifikátory stanic. Stanice se řadí podle zvoleného směru na trati. Následně se ve čtvrtém kroku v cyklu přes všechny identifikátory stanic provede sada dotazů do databáze, jejichž výsledkem je vícerozměrné pole, kde jsou ke každému vlaku přiřazena jednotlivá čísla spojů a ke každému spoji sada identifikátorů jednotlivých příjezdů. Tato struktura je zobrazena na obrázku 7.3:



Obrázek 7.3: Datová struktura

Při dotazech se klade důraz na typy vlaků povolené v aktuální konfiguraci a na směr spojů. V případě, že uživatel požaduje zobrazení obou směrů zároveň, vytvoří se pomocné pole, do kterého se ukládají identifikátory vlaků se spoji v tomto směru. V pátém kroku se pomocí identifikátorů vlaků provedou další dotazy do databáze, které vlakům přidají další informace týkající se jejich čísla, jména a typu. Poté se vytvoří jednotlivá data pro osu Y, a to konkrétně jednotlivé vzdálenosti stanic na trati, které jsme si uložili ve třetím kroku. Každá tato vzdálenost musí být zapsána 2x, protože se k ní váží 2 hodnoty - čas odjezdu a čas příjezdu. Jedná se o vlastnost knihovny Plotly.

V dalším kroku se poté ke každému vlaku s jeho spoji a jednotlivými příjezdy uloží dané časy odjezdu a příjezdu, společně s datem. Datum se zapisuje z důvodu, který byl zmíněn v kapitole 7.1. Konkrétně se jedná o problém s nočními vlaky.

Posledním krokem je vytvoření struktury reprezentující konečná data ve formátu JSON pro knihovnu Plotly. Zde se pohybujeme přes vícerozměrné pole vytvořené předchozími kroky a postupně vytváříme datovou strukturu požadovaného formátu. Každý spoj je reprezentovaný vlastním JSON objektem s jednotlivými vlastnostmi. V tomto kroku lze na základě typu vlaku nastavit požadovaný vzhled přímky reprezentující spoj. Zároveň se zde vytváří pole s informacemi k jednotlivým bodům, které se zobrazí po překrytí bodu kurzorem myši. Tyto informace se skládají z čísla, typu a jména vlaku, společně s aktuálním

časem odjezdu nebo příjezdu v daném bodě. V případě, že uživatel zvolil zobrazení spojů v obou směrech, u vlaků uložených v pomocném poli pro opačný směr je potřeba zaměnit pořadí času odjezdu a příjezdu, z důvodu korektního vykreslení v grafu. Nakonec se výsledná struktura převede do formátu JSON.

7.2.2 Úprava osy Y

Samotné vytvoření dat pro vykreslení grafu nestačí. Graf totiž implicitně očekává na ose Y číselné hodnoty. Abychom docílili toho, že náš widget bude vypadat jako vlakový grafikon, je nutné tyto hodnoty nahradit názvy jednotlivých stanic. Za tímto účelem jsou v aplikaci implementovány skripty `getTicksY.php` a `getLablesY.php`. První ze zmíněných skriptů navrací objekt ve formátu JSON s jednotlivými vzdálenostmi stanic na trati. V nastavení grafu budou tyto údaje sloužit k určení, které hodnoty se na ose Y zobrazí. Ostaní budou skryty, jak bylo vysvětleno v kapitole 6.3.2. Druhý skript následně navrátí další objekt ve formátu JSON, tentokrát s jednotlivými názvy stanic, které budou sloužit jako zobrazované aliasy pro jednotlivé číselné hodnoty. Tímto docílíme stejného vzhledu widgetu, jako má grafikon vlakové dopravy.

7.2.3 Orientace osy Y

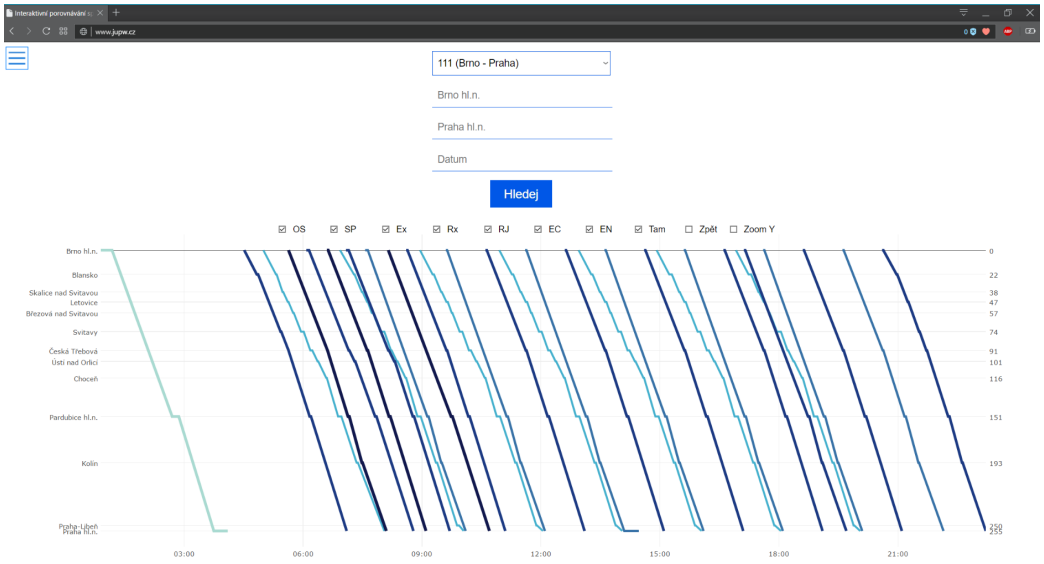
Počáteční stanice bývá tradičně v grafikonu vlakové dopravy zobrazena nahoře na ose Y, zatímco cílová stanice značí hodnotu 0 a je tedy na ose Y úplně dole. V kapitole 6.5 bylo ukázáno, že jednotlivé stanice mají uloženou informaci o své vzdálenosti na trati. Abychom v případě, kdy půjdeme směrem od menší k větší vzdálenosti nemuseli tyto hodnoty přepočítávat z důvodu správného vykreslení v grafu, je zde implementován skript `getDirection.php`, jehož úkolem je porovnat vzdálenosti jednotlivých stanic, a navrátit jednu z číselných hodnot -1/1, pokud má počáteční stanice menší, respektive větší vzdálenost vůči cílové. Na základě vykreslovaného směru se poté podle navrácené hodnoty v nastavení grafu rozhodne o invertaci osy Y.

7.2.4 Naplnění seznamu stanic

Skript `whisperer.php` navrací objekt ve formátu JSON obsahující jednotlivé názvy stanic na trati, podle převzaté hodnoty čísla tratě, která slouží pro dotaz do databáze. Tento objekt je poté využit ve frontendové části aplikace pro naplnění seznamu se jmény stanic.

7.3 Přední část aplikace

Přední část aplikace je tvořena sadou několika skriptů v jazyce JavaScript využívajících knihovnu jQuery, které mezi sebou navzájem komunikují a pomocí kterých aplikace asynchronně volá jednotlivé skripty v jazyce PHP. V následující kapitole budou tyto skripty popsány a bude zde vysvětlen jejich účel a použití. Na obrázku 7.4 je znázorněn výstup aplikace.



Obrázek 7.4: Snímek aplikace - výstup aplikace

7.3.1 Výběr tratě

Uživatel má na výběr předpřipravený seznam tratí, obsahující číslo tratě spolu s počáteční a cílovou stanicí. Poté, co si uživatel zvolí trať, je pomocí skriptu `auto.js` tato událost zachycena. Do řetězce simulujícího data pro metodu POST se uloží hodnota tratě, následně se zavolá skript `whisperer.php`, popsaný v kapitole 7.2.4, kterému se tento řetězec předá. Následně se s pomocí obdrženého JSON objektu dynamicky vytvoří HTML obsah stránky reprezentující seznam tratí.

7.3.2 Ošetření výběru data

Pro výběr data je v aplikaci použit element *input* jazyka HTML typu *date*, který ale není podporovaný v prohlížeči Mozilla Firefox. Za tímto účelem byl v apliakci implementovaný skript `datepicker.js`, který využívá knihovnu jQuery UI. Ta dovoluje lokalizovat tento prvek, a vytvořit pro něj vlastní nástavbu uživatelského rozhraní. Jelikož oficiálně nepodporuje český typ kalendáře, bylo nutné lokalizovat ovládací prvky do češtiny a specifikovat jednotlivé názvy měsíců, dnů, a jejich zkratky. Tato nástavba se spouští pouze, je-li identifikován prohlížeč FireFox, ostatní prohlížeče mají pro výběr data své vlastní nástavby.

7.3.3 Konfigurace grafu

O nastavení aktuální konfigurace se stará skript `form.js`. Ten má na starost obsluhu formuláře se stanicemi, ale i jednotlivé HTML elementy typu checkbox, znázorněné na obrázku 7.5. Pokud je vyžadován řetězec reprezentující konfiguraci grafu, tento skript převezme z těchto elementů jejich hodnoty, ze kterých následně datový řetězec vytvoří.

7.3.4 Získání dat ve formátu JSON

Aplikace obsahuje také skript `getJSON.js`, jehož jediným úkolem je asynchronně zavolat skritp `getDataPlotly.php`, popsaný v kapitole 7.2.1. Tomuto skriptu předá současnou konfiguraci, získanou od skriptu `form.js` z kapitoly 7.3.3, a následně od něj obdrží datovou strukturu ve formátu JSON, kterou předá dalšímu skriptu.

7.3.5 Nastavení vlastností grafu

Nedílnou součástí tvorby widgetu je skript `optionsPlotly.js`, který zajišťuje veškeré jeho nastavení. Nejdříve zavolá jednotlivé skripty jazyka PHP pro zjištění směru a úpravu dat na ose Y, které byly popsány v kapitolách 7.2.2 a 7.2.3. Následně se ze získaných dat vytvoří objekt ve formátu JSON reprezentující vzhled a vlastnosti grafu. Zde je možné upravit další vlastnosti týkající se vzhledu, jako odsazení grafu, velikost, barvu atd.

7.3.6 Vykreslení widgetu

Veškeré kroky popsané v přechozích kapitolách řídí skript `script.js`, který nejdříve převezme konfiguraci grafu, následně ji předá skriptu z kapitoly 7.3.4 a získá data pro vykreslení. Poté požádá skript z kapitoly 7.3.5 o objekt popisující nastavení grafu, lokalizuje element HTML stránky určený pro umístění widgetu a pomocí funkce knihovny Plotly, které předá jednotlivé datové objekty, vykreslí samotný graf.

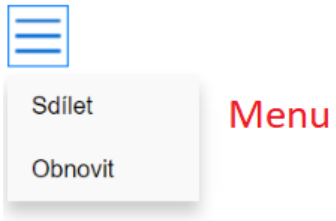
7.3.7 Obsluha vykreslení widgetu

Jako pokyn pro vykreslení interaktivního widgetu slouží tlačítko s nápisem *Hledej* znázorněné na obrázku 7.5. O obsluhu této události se stará skript `events.js`. Zároveň si ukládá do lokální proměnné informaci o tom, zda je widget již vykreslen, nebo se jedná o první vykreslení. Uložení tohoto stavu je klíčové, protože prvky pro konfiguraci se chovají jinak, pokud je widget vykreslen, oproti jeho prvotní konfiguraci. V tomto případě dojde okamžitě k projevení změny konfigurace, a to překreslením grafu. Uživateli se tak ihned projeví změny, které zamýšlel. Po zmáčknutí tlačítka pro vyhledávání dojde k zavolání skriptu z kapitoly 7.3.6. Je také odchytávána událost zmáčknutí klávesy *Enter*, která také způsobí vykreslení widgetu.

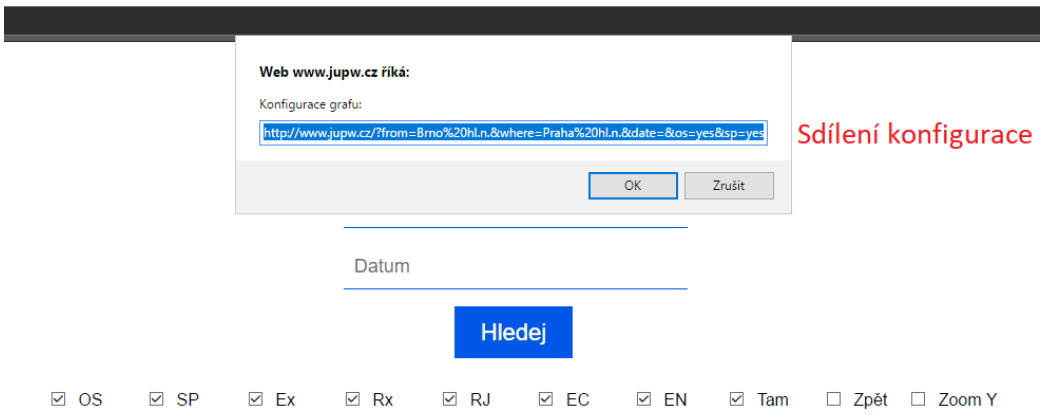
7.3.8 Sdílení a obnovení konfigurace grafu

Skript z kapitoly 7.3.7 zodpovědný za obsluhu vykreslení grafu má na starosti také obsluhu událostí jako jsou sdílení aktuální konfigurace grafu a její obnovení. Pro obě tyto události je v aplikaci implementováno menu, znázorněno na obrázku 7.6, umístěné v levém horním rohu obrazovky, které se rozbalí po překrytí kurzorem myši. Po kliknutí na možnost *Sdílet* je uživateli vygenerován odkaz pro sdílení. Tato událost je znázorněna na obrázku 7.7:

Obrázek 7.5: Snímek aplikace - jednotlivé prvky aplikace



Obrázek 7.6: Snímek aplikace - menu



Obrázek 7.7: Snímek aplikace - sdílení konfigurace grafu

Kapitola 8

Testování

Aplikaci jsem po dobu návrhu a implementace průběžně testoval za pomoci vedoucího mé bakalářské práce. V konečné fázi jsem aplikaci otestoval na běžných uživatelích se základní znalostí práce s počítačem. Na základě této zpětné vazby jsem následně upravoval různé aspekty aplikace, jako vzhled nebo rozložení jednotlivých prvků na stránce.

8.1 Prototyp aplikace

První prototyp aplikace využíval pro vykreslení interaktivního widgetu původně knihovnu Google Chart. Na základě negativní zpětné vazby, která se týkala především problému s výkonností knihovny, jsem následně přešel na knihovnu Plotly. Tento problém se projevoval trhavými pohyby a dlouhou dobou načítání, především při vykreslení většího množství spojů a následném pohybu v grafu. Knihovna Google Chart dále obsahovala nástroj zoom pro přiblížení, který byl stále ve vývoji. Tento nástroj sice fungoval společně s lineárním grafem, který jsem využíval pro tuto práci, ale jeho reakce přiblížení na posun kolečka myši byly opačné, než je zažitý standart u většiny aplikací. Knihovna tedy na posun kolečka směrem vzhůru reagovala oddálením namísto přiblížením, a opačně. Tento fakt spolu se zmíněnými výkonnostními problémy silně ovlivňovaly uživatelskou zkušenost při používání aplikace.

8.2 Závěrečné testování

Testování z pohledu programátora může být ovlivněno jeho stylem uvažování. Můj osobní názor na tuto aplikaci je zkreslen také tím, že jsem dopodrobna obeznámen s tím, jak aplikace funguje. Jelikož je aplikace určená především pro širokou skupinu uživatelů, vybral jsem pro její otestování skupinu pěti dobrovolníků, kteří mají jen základní zkušenosti práce s počítačem.

Těmto dobrovolníkům jsem postupně zadal sadu úkolů, které měli v aplikaci provést, a to bez jakéhokoliv obeznámení s jednotlivými ovládacími prvky. Jedinou informaci, kterou o aplikaci měli, byl její účel. Úkoly byly navrženy tak, aby prověřily veškeré funkce aplikace a ukázaly, jak si s nimi běžný uživatel poradí. Test obsahoval tyto kroky:

1. vykreslení grafu s výchozí konfigurací,
2. základní pohyb v interaktivním grafickém prvku,
3. filtrace zobrazovaných dat,

4. nalezení nejrychlejšího spoje z výchozí do cílové stanice,
5. zhodnotit vizuální rozlišení spojů,
6. sdílet aktuální konfiguraci grafu,
7. obnovit aplikaci do výchozího stavu.

8.2.1 Vykreslení grafu s výchozí konfigurací

První úkol sloužil především pro seznámení se základní funkcionalitou aplikace. Uživatel byl posazen před spuštěnou aplikaci a byl požádán, aby vyhledal provoz na trati z Brna do Prahy. V tomto případě stačí pouze zvolit danou trať, přičemž není nutné vyplnit počáteční a konečnou stanici. Všichni z pěti uživatelů tento úkol zvládli naprosto bez problémů. Všichni ovšem také vyplnili počáteční a cílovou stanici. Pro uživatele tedy není dostatečně zřejmé, že tento krok není nutné dělat. Na základě této zpětné vazby jsem provedl úpravu kódu. Nyní se po výběru tratě změní výchozí text políček pro zadávání stanic z řetězců *Odkud* a *Kam* na přislušné názvy stanic.

8.2.2 Základní pohyb v interaktivním grafickém prvku

Cílem druhého úkolu bylo prověřit, jak moc intuitivní je panel nástrojů grafu. Pro možnost posouvat se v grafu v různých směrech je nutno přepnout v tomto panelu na nástroj *Pan*. Výchozím nástrojem po vykreslení grafu je nástroj pro vyznačení oblasti grafu pro přiblížení. Jelikož všichni dobrovolníci jako první vyzkoušeli intuitivně posunout grafem pomocí kurzoru myši, a nebylo jim úplně jasné, co je potřeba přepnout, nastavil jsem nástroj *Pan* jako výchozí. Lze očekávat, že společně s přiblížením pomocí kolečka myši se bude z řad uživatelů jednat o základní očekávanou funkcionalitu.

8.2.3 Filtrace zobrazovaných dat

Třetí úkol byl zaměřený na práci s interaktivním widgetem. Každý uživatel byl požádán o zobrazení unikátní konfigurace grafu. Cílem tohoto testu bylo zjistit, zda je dostatečně zřejmé spojení mezi interaktivním widgetem a sadou elementů typu checkbox, umístěnými nad grafem. Kombinace v rámci konfigurace byly různé, bylo zapotřebí zatrhnout nové a zároveň i zrušit některé výchozí požadavky. I přesto, že požadované vlaky byly v rámci úkolu zadány celým názvem a nikoli jen zkratkou, opět tento úkol bez problému splnilo všech pět dobrovolníků. Výsledkem tohoto testu je důkaz, že ovládací prvky widgetu jsou dostatečně logicky umístěné, a také, že plně postačuje označení jednotlivých vlaků zkratkami.

8.2.4 Nalezení nejrychlejšího spoje z výchozí do cílové stanice

Čtvrtý úkol byl zaměřen na hlavní účel aplikace, a to sice porovnávání jednotlivých vlakových spojů. Uživatelé byli postupně požádáni, aby označili nejrychlejší spoj na trase Brno - Praha. Při tomto úkolu nebyla zobrazovaná data jakkoliv filtrována. Jednotlivé spoje jsou vizuálně odlišeny odstíny jedné barvy, přičemž čím tmavší daný spoj je, tím je také rychlejší. Rychlost spoje lze také na první pohled odhadnout podle toho, jak často je jeho přímka přerušovaná, což značí zastávku ve stanici. Ani s jednou z těchto informací nebyli dobrovolníci obeznámeni. Zde celkově čtyři z pěti dobrovolníků označili jako nejrychlejší spoj na trati vlak typu Expres, což je správná odpověď. Intuitivně přitom používali nástroj

zoom pro přiblížení. Pro kontrolu si zobrazovali jednotlivé časové údaje v dodatečných informacích, které se zobrazí při překrytí spoje kurzorem myši. Jeden z dobrovolníků označil jako nejrychlejší spoj na trati noční vlak EuroNight. Podle jeho vlastních slov se řídil právě počtem zastávek na trati spolu s odstínem barvy spoje. Tento vlak jsem měl při testování označený černou barvou, což tohoto uživatele zmátlo. EuroNight má, stejně jako Expres, na trati pouze jednu zastávku. Ta však trvá podstatně déle a Expres je tedy rychlejší. Výsledkem tohoto testu je úprava barevného rozlišení a zvýraznění síly čar jednotlivých spojů, aby byla délka odstávky ve stanici více zřetelná.

8.2.5 Zhodnocení vizuálního rozlišení spojů

V tomto testu mě zajímaly subjektivní pocity jednotlivých dobrovolníků. Jejich úkolem tentokrát bylo pouze pokusit se popsat, jak na ně působí barevné rozlišení spojů, které jsem navrhnul. Opět jsem tedy zobrazil provoz pro trať Brno - Praha, tentokrát jsem ovšem v konfiguraci grafu nastavil i zobrazení zpátečních spojů, ve snaze udělat graf co nejméně přehledný. Dobrovolníci si stěžovali především na příliš splývající odstíny u některých spojů. Na základě těchto připomínek jsem navrhnul nové barevné rozlišení s širší škálou odstínů.

8.2.6 Sdílení aktuální konfigurace grafu

Šestý úkol byl zaměřen na funkci sdílení konfigurace grafu s více uživateli. Dobrovolníkům jsem sdělil pouze informaci, že v aplikaci je možné sdílet odkaz na graf, který vytvořili. Následně jsem je požádal, aby vykreslili widget v libovolné konfiguraci a pokusili se mi na druhý počítač odeslat odkaz na tento graf. Po kratším zaváhání dvou z pěti dobrovolníků postupně celou skupinu napadlo zaměřit se na poněkud výrazný objekt v levém horním rohu obrazovky, pro který jsem použil dnes běžně užívaný znak pro rozbalovací menu. Po kliknutí na možnost *Sdílet* se všem dobrovolníkům podařilo odeslat mi požadovaný odkaz. Zároveň byla ověřena funkčnost tohoto sdílení vykreslením grafu na druhém počítači a jeho srovnáním s originálem.

8.2.7 Obnovení aplikace do výchozího stavu

Jako poslední úkol jsem zvolil obnovení aplikace do jejího výchozího stavu. Jelikož se v předchozím úkolu dobrovolníci seznámili s prvkem menu, většina si všimla také druhé ze dvou jeho možností, kterou je možnost *Obnovit*. Dva dobrovolníci použili pro obnovení aplikace klasické obnovení celé stránky, což má stejný efekt. Tato možnost je tu pouze pro lepší uživatelský zážitek.

8.3 Možná rozšíření

Účelem této sekce je navrhnout na základě provedeného testování možné rozšíření aplikace pro její budoucí vývoj. Konkrétně se jedná o rozšíření v oblasti databáze, která byla implementována čistě pro účely této aplikace. Možné je také dále rozšiřovat samotný interaktivní widget.

8.3.1 Widget

Co se interaktivního widgetu týče, lze se více zaměřit na jeho mobilní verzi. Aplikace byla primárně navržena v desktopovém prostředí, kde byla také testovaná, a zobrazení na mobil-

ním zařízení není úplně ideální. Tento problém byl popsán v kapitole 6.3.1. Dále je možné navrhnout nové vizuální rozlišení jednotlivých spojů. Této problematice byl věnován test v kapitole 8.2.5. Také je možné přidávat nové filtry pro nové typy vlaků, v závislosti na nových tratích. Lze navrhnout novou šablonu CSS stylů pro jednotlivé prvky aplikace, jako jsou například HTML elementy typu checkbox.

8.3.2 Databáze

Databázi lze do budoucna vylepšit například vytvořením tabulky pro uchovávání jednotivých provozních dnů. V takovém případě by bylo datum v každém záznamu spojů nahrazeno jeho identifikátorem. Bylo by tak možné lépe dohledat dny v roce, kdy je potenciální změna v provozu oproti běžnému dni, jako jsou například státní svátky.

Dále je možné navrhnout takovou databázi, která bude počítat i se scénářem, kdy uživatel zadá jednotlivé stanice z různých tratí. V takovém případě je potřeba implementace algoritmu pro vyhledávání cesty v grafu tak, aby bylo možné nalézt nejkratší možnou cestu mezi těmito stanicemi.

Zaměřit se lze i na získání dat o vlakových spojích. Zde je možné například soustředit se na zdokonalení skriptu, který slouží pro převod jednoduchých textových dat na sadu SQL příkazů. Lze rozšířit jeho funkčnost, upravit jej pro jiný konkrétní datový typ, než je jednoduchý textový. Lze také navrhnout nový skript, který by získával data od jiné existující aplikace třetích stran.

Kapitola 9

Závěr

Mým úkolem bylo navrhnout, implementovat a následně otestovat webovou aplikaci sloužící pro interaktivní porovnávání vlakových spojů pomocí vlakového grafikonu. Za tímto účelem jsem nastudoval problematiku grafikonu vlakové dopravy a seznámil jsme se s nástroji pro tvorbu vlastních diagramů určenými pro webovou platformu. Zároveň jsem navrhnul a implementoval databázi spojů, ze které aplikace získává požadovaná data. Na závěr své práce jsem implementovanou aplikaci otestoval. Na základě zpětné vazby jsem aplikaci upravil a navrhnul možná rozšíření.

Výsledkem mé práce je volně dostupná webová interaktivní aplikace s vlastní databází spojů, jejíž hlavní výhodou oproti ostatním aplikacím, zabývající se problematikou vlakových spojů, je především přehledná grafická vizualizace dat. Tato aplikace umožňuje snadné a rychlé porovnávání jednotlivých vlakových spojů. Aplikace disponuje nástrojem zoom, určeným pro přibližování vykreslených dat. Součástí knihovny Plotly je také panel nástrojů, pomocí kterého je možné uložit si výsledný graf ve formátu PNG. Dále je možné zobrazit si podrobnější informace o daném spoji po překrytí jednotlivých bodů grafu kurzorem myši. Lze také provádět filtrování zobrazovaných dat, přičemž je možné tuto konfiguraci sdílet mezi uživateli. Při návrhu aplikace byl kladen důraz na jednoduché a intuitivní ovládání, dále pak na responsivní vzhled.

Do budoucna se lze více zaměřit na mobilní vzhled aplikace. Dále je možné navrhnout novou databázi spojů s možností nalezení nejkratší cesty pro zobrazení provozu mezi stanicemi nacházejícími se na různých tratích. V závislosti na jednotlivých dopravcích je možné aplikaci rozšířit o další typy vlaků. V příloze této bakalářské práce jsou vyobrazeny snímky aplikace demostrující výstupy s různou konfigurací grafu.

Literatura

[1] Date, C. J.: *An Introduction to Database Systems*. Addison-Wesley, 1995, ISBN 0-201-82458-2.

[2] České dráhy: Naše vlaky. [Online; navštíveno 28.11.2016].
URL https://www.cd.cz/nase-vlaky/

[3] Few, S.: *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly, 2006, ISBN 0596100167.

[4] Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Elsevier Inc., 2010, ISBN 978-0-12-375030-3.

[5] Plotly: *JavaScript Graphing Library*. [Online; navštíveno 6.12.2016].
URL https://plot.ly/javascript/

[6] Plotly: *Plotly.js Open-Source Announcement*. 2015, [Online; navštíveno 6.12.2016].
URL https://plot.ly/javascript/open-source-announcement/

[7] Stonebraker, M.: *SQL databases v. NoSQL databases. Magazine Communications of the ACM*, 2010: s. 10 – 11.

[8] Tufte, E. R.: *The Visual Display of Quantitative Information*. Graphic Press, 2001, ISBN 0961392142.

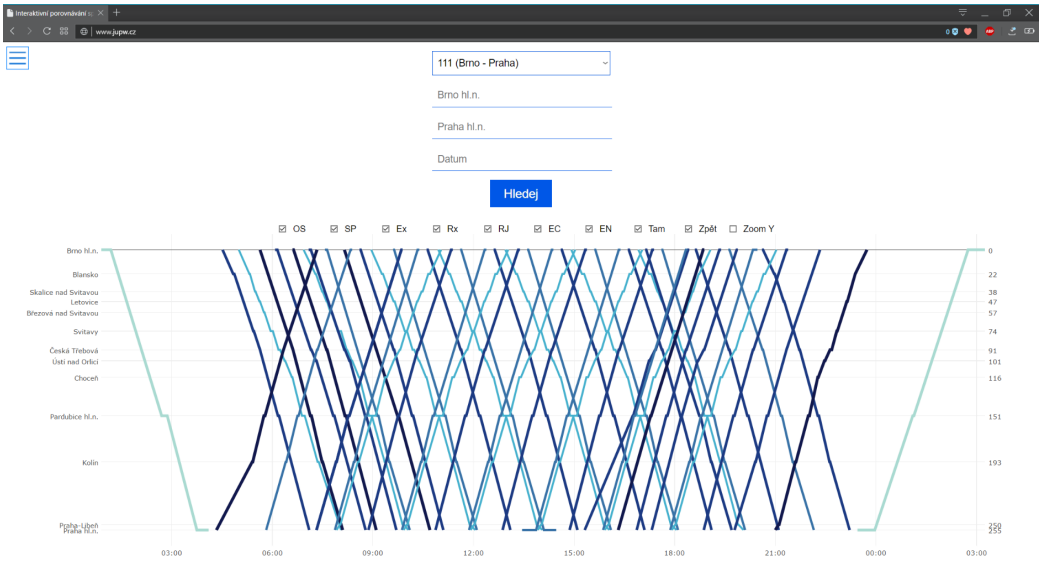
[9] Václavík, L.: *Seznam vytvoří vlastní jízdní řády, vylepší Pubtran a dostane jej na iOS a WP*. 2015, [Online; navštíveno 18.01.2017].
URL https://www.cnews.cz/seznam-vytvori-vlastni-jizdni-rady-vylepsi-pubtran-a-dostane-jej-na-ios-a-wp/

[10] W3Schools: *HTML5 canvas*. [Online; navštíveno 10.05.2017].
URL https://www.w3schools.com/html/html5_canvas.asp

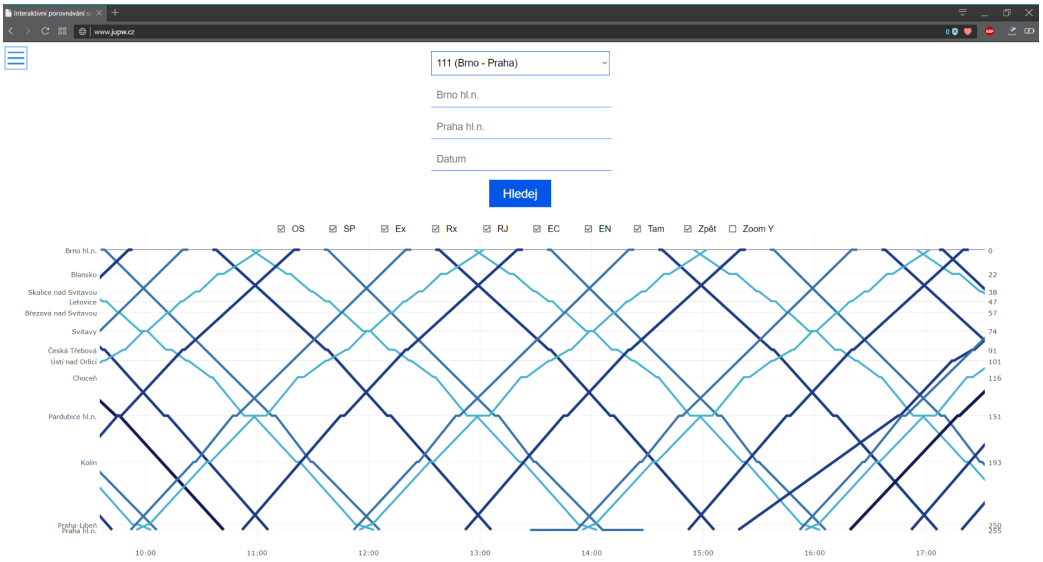
[11] Ware, C.: *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2004, ISBN 1-55860-819-2.

[12] Šotek, K.; Bachratý, H.; Greiner, K.; aj.: *Tvorba jízdního řádu pomocí výpočetní techniky na Českých drahách*. [Online; navštíveno 20.11.2016].
URL http://spz.logout.cz/zabezpec/sena/sena.html

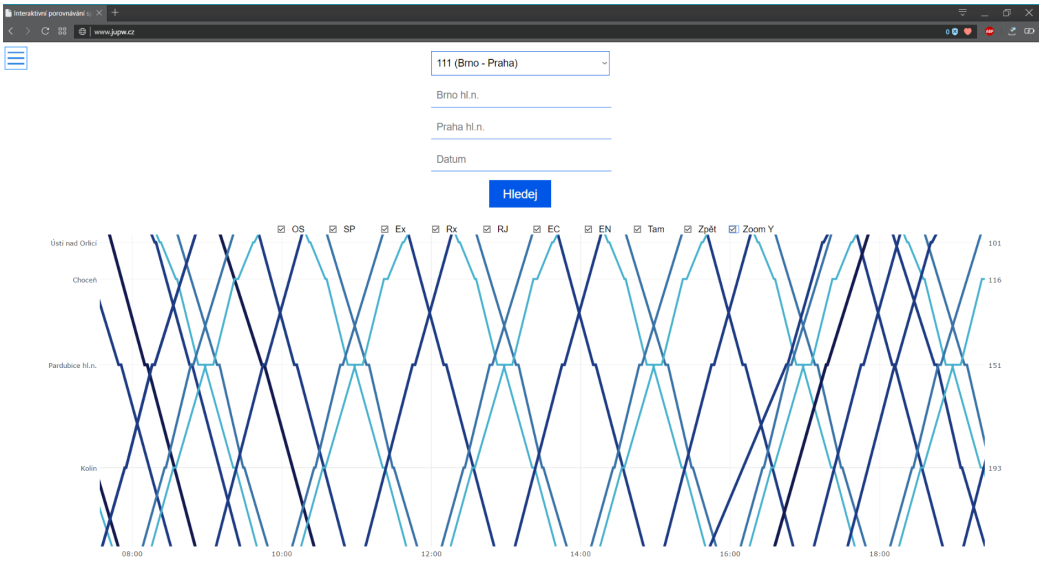
Přílohy



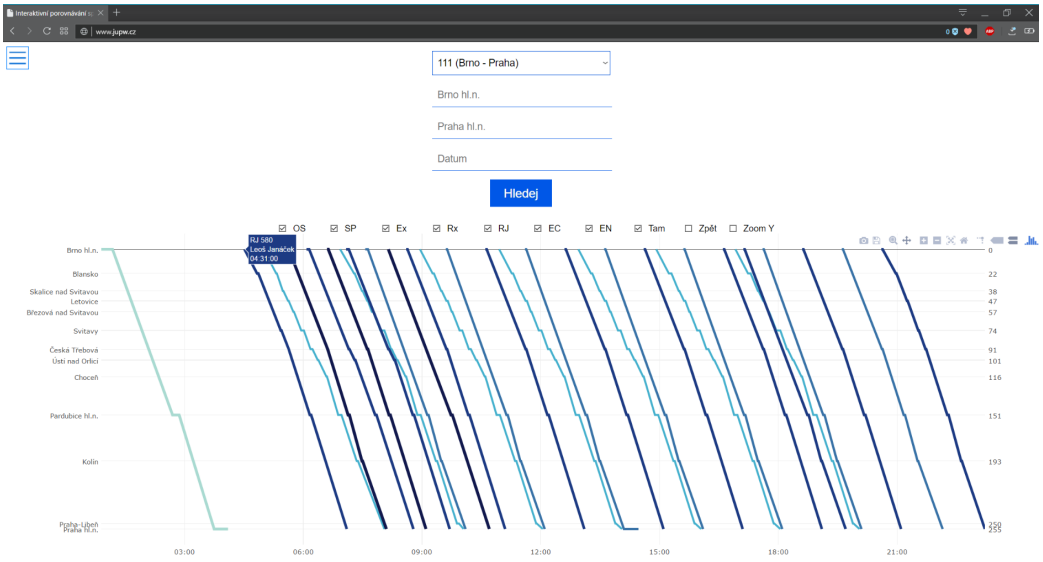
Obrázek 1: Snímek aplikace - obousměrný provoz



Obrázek 2: Snímek aplikace - přiblížení osy X



Obrázek 3: Snímek aplikace - přiblížení osy Y



Obrázek 4: Snímek aplikace - dodatečné informace