

# Rozšíření dispečinkového systému o modul zpoždění

Bc. Dalibor Dobeš

---

Diplomová práce  
2018

 Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Dalibor Dobeš**  
Osobní číslo: **A16145**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**  
Forma studia: **kombinovaná**

Téma práce: **Rozšíření dispečinkového systému o modul zpoždění**  
Téma anglicky: **Expanding a Public Transport Dispatching System by a Delay Module**

Zásady pro vypracování:

1. Analyzujte požadavky dispečinkového systému na modul zpoždění.
2. Navrhněte technické řešení pro výpočet zpoždění.
3. Implementujte ukládání naměřených hodnot do databáze.
4. Realizujte vizualizaci spoje na trase.
5. Vytvořte webové služby pro přístup třetích stran k údajům o zpoždění.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **URMA, Raoul-Gabriel, Mario FUSCO a Alan MYCROFT. Java 8 in action: lambdas, streams, and functional-style programming. Shelter Island: Manning, 2015. ISBN 1617291994.**
2. **WALLS, Craig. Spring in action. Fourth Edition. Shelter Island, NY: Manning, 2015. ISBN 161729120x.**
3. **Maven: the definitive guide. Sebastopol: O'Reilly, c2008. ISBN 0596517335.**
4. **OBE, Regina O. a Leo S. HSU. PostGIS in action. Second edition. Shelter Island, NY: Manning, 2015. ISBN 1617291390.**
5. **PRICE, Maribeth Hughett. Mastering ArcGIS. Seventh edition. New York, NY: McGraw-Hill Education, 2016. ISBN 978-0078095146.**

Vedoucí diplomové práce:

**Ing. Tomáš Dulík, Ph.D.**

Ústav informatiky a umělé inteligence

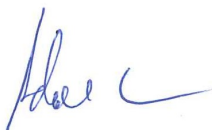
Datum zadání diplomové práce:

**1. prosince 2017**

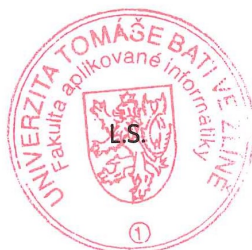
Termín odevzdání diplomové práce:

**16. května 2018**

Ve Zlíně dne 11. prosince 2017



doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



prof. Mgr. Roman Jašek, Ph.D.  
*garant oboru*

## Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

## Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 16.5.2013



.....  
podpis autora

## **ABSTRAKT**

Cílem diplomové práce je analyzovat, doplnit a rozšířit stávající dispečinkový systém o nové zdroje dat zpoždění a polohy spojů. Práce zahrnuje připojení na SOAP [7] API společnosti SŽDC, odkud se načítá aktuální zpoždění a další doplňkové informace. Účelem návrhu výpočtu zpoždění a polohy vozů je připojení se k RESTovému API Polohavozu.cz a následné načtení struktury vlakového spoje pomocí JavaMail API, s následujícím uložením do databáze stávajícího dispečinkového systému.

Klíčová slova: Java, REST, SOAP, API, EXCEL, CSV, WSDL, JSON

## **ABSTRACT**

The aim of this diploma thesis is to analyze, add and expand the existing dispatching system by new sources of data of delay and position of trains. It will include a connection of company SŽDC to the SOAP [7] API, from where current delays will be loaded and other additional informations. Connection to REST API polohavozu.cz will be needed for the purpose of calculating delay and position of vehicles. The JavaMail will be needed to load train connections data which will be saved to the database of an existing dispatching system.

Keywords: Java, REST, SOAP, API, EXCEL, CSV, WSDL, JSON

Na tomto místě bych rád poděkoval především svému vedoucímu diplomové práce Ing. Tomášovi Dulíkovi, Ph.D. za odborné vedení, poskytnutí cenných rad a osobních konzultací, za věnovaný čas a jeho trpělivost. Dále bych rád poděkoval společnosti STUDENT AGENCY k.s. za poskytnutí zázemí a podkladů pro tvorbu této práce a za ochotu pomoci při řešení problémů.

# OBSAH

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>9</b>
<b>1 VÝVOJOVÉ NÁSTROJE, PROTOKOLY A APLIKACE</b> .....	<b>11</b>
1.1 SPRING .....	11
1.2 HIBERNATE .....	12
1.3 MAVEN .....	12
1.4 SOAP .....	14
1.4.1 WSDL .....	14
1.4.2 SoapUI .....	14
1.5 REST .....	15
1.5.1 JSON .....	15
1.5.2 Postman .....	16
1.6 JAVAMAIL .....	17
1.7 CSV .....	17
1.8 POSTGIS .....	17
<b>II ANALYTICKÁ ČÁST</b> .....	<b>17</b>
<b>2 ANALÝZA OBECNĚ</b> .....	<b>19</b>
<b>3 DIFERENČNÍ ANALÝZA</b> .....	<b>20</b>
3.1 POPIS STÁVAJÍCÍHO STAVU .....	20
3.1.1 L1 pohled .....	20
3.2 POPIS PLÁNOVANÉHO STAVU .....	20
3.3 URČENÍ ROZDÍLU (MEZERY) MEZI STÁVAJÍCÍM A CÍLOVÝM STAVEM.....	21
3.4 NÁVRH VARIANT DOSAŽENÍ CÍLOVÉHO STAVU (ALTERNATIVNÍ STRATEGIE) .....	21
3.5 POPIS STAVU PO ZMĚNÁCH .....	22
<b>4 NÁVRH TECHNICKÉHO ŘEŠENÍ PRO VÝPOČET ZPOŽDĚNÍ</b> .....	<b>23</b>
4.1 TRASA SPOJE .....	23
4.2 DEFINICE (ZÁKLADNÍ) TRASY SPOJE POMOCÍ GPS SOUŘADNIC .....	23
4.3 VÝPOČET ZPOŽDĚNÍ .....	23
<b>III PROJEKTOVÁ ČÁST</b> .....	<b>24</b>
<b>5 PŘIPOJENÍ NOVÝCH ZDROJŮ DAT</b> .....	<b>26</b>
5.1 PŘIPOJENÍ SŽDC API .....	26
5.1.1 Datové rozhraní GRAPP .....	26
5.1.2 Datové rozhraní „Informační tabule“ .....	34
5.2 PŘIPOJENÍ POLOHAVOZU.CZ .....	34
5.3 NAČÍTÁNÍ STRUKTURY VLAKU DO DISPEČERSKÉHO SYSTÉMU .....	36
<b>6 DISTRIBUCE ZPOŽDĚNÍ</b> .....	<b>43</b>

<b>ZÁVĚR</b> .....	<b>44</b>
<b>SEZNAM POUŽITÉ LITERATURY</b> .....	<b>45</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK</b> .....	<b>49</b>
<b>SEZNAM OBRÁZKŮ</b> .....	<b>50</b>
<b>SEZNAM ZDROJOVÉHO KÓDU</b> .....	<b>51</b>
<b>SEZNAM PŘÍLOH</b> .....	<b>52</b>

---

## ÚVOD

Tato diplomová práce je zaměřena na požadavky dispečinkového systému. Z větší části je zaměřená na výpočet zpoždění, respektive na modul výpočtu zpoždění.

Teoretická část se věnuje prostředkům, nástrojům a frameworkům použitým při realizaci této práce.

Krátký pohled je také věnován analýze systému pomocí diferenční analýzy. V následujících kapitolách je rozveden jak současný stav, tak možnosti na vylepšení v novém modulu zpoždění. Jsou zde také představeny návrhy variant cílového stavu. Technické řešení následuje ve čtvrté kapitole.

Závěr práce je věnován projektové části a distribuci zpoždění. Je zde zdrojovými kódy prokládán postup a popis během realizace prací nejen na datovém rozhraní.

# I. TEORETICKÁ ČÁST

## 1 Vývojové nástroje, protokoly a aplikace

Pro vývoj v této diplomové práci bude použito mnoho různých frameworků, komunikačních protokolů a jejich znalost je klíčová k úspěšnému dokončení projektu.

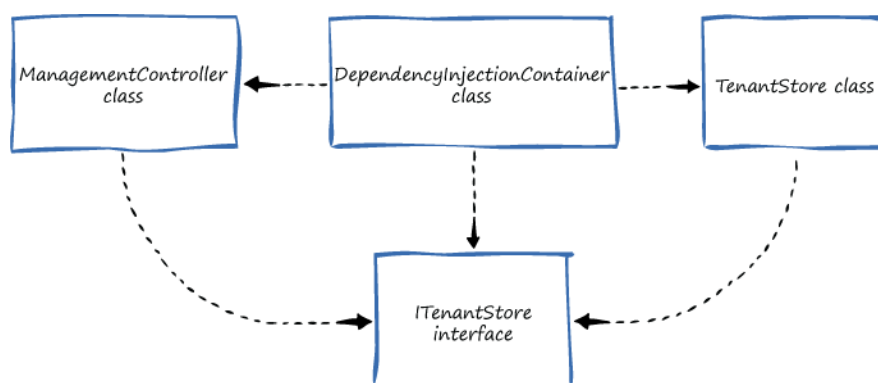
- Spring Framework
- Hibernate framework
- Apache Maven
- SOAP
- WSDL
- SoapUI
- REST
- JSON
- Postman
- JavaMail
- CSV
- PostGIS

### 1.1 Spring

Spring Framework [8] (dále jen spring) je populární open-source [14] pro vývoj J2EE [9] aplikací. Jádro springu [2] využívá návrhový vzor IoC (Inversion of Control [10]) a je označován jako IoC kontejner. Tento návrhový vzor funguje na principu přesunutí zodpovědnosti za vytvoření a provázání objektů z aplikace na framework.

Objekty lze získat prostřednictvím Dependency Injection [11] neboli vsazování závislostí (Obr. 1.1). Jedná se o speciální případ IoC. Dependency Injection lze realizovat třemi způsoby

- Setter Injection,
- Constructor Injection,
- Interface Injection.



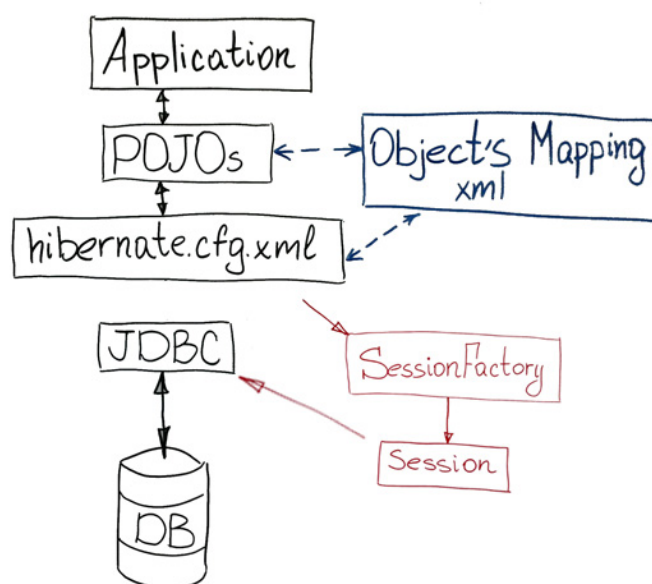
Obr. 1.1 Náhled možností dependency injection

Vytvářené objekty se nazývají JavaBeans [12]. Ty jsou vytvářeny typicky na základě načtení konfiguračního souboru ve formátu XML [13], který obsahuje definice těchto Beans. Spring se nezabývá řešením již vyřešených problémů. Místo toho využívá prověřených a dobře fungujících existujících open-source nástrojů, které v sobě integruje. Tím se stává jejich použití často jednodušším. Spring je modulární framework. Umožňuje využít jen část, která se zrovna hodí k řešení daného problému. Účelem springu je:

- zjednodušení návrhu J2EE aplikací se zaměřením na architekturu aplikace (místo na technologie),
- jednoduchá testovatelnost,
- neinvazivní rozvoj a modularita.

## 1.2 Hibernate

Hibernate framework [15] je napsaný v jazyce Java [16], který umožňuje tzv. objektově-relační mapování (ORM [17]). Uspadňuje řešení otázky zachování dat objektů i po ukončení běhu aplikace. Hibernate poskytuje způsob, pomocí něhož je možné zachovat stav objektů mezi dvěma spuštěnými aplikacemi. Říkáme tedy, že udržuje data persistentní. Dosahuje toho pomocí ORM, což znamená, že mapuje Java objekty na entity v relační databázi. K tomu používá tzv. mapovací soubory, ve kterých je popsáno, jakým způsobem se mají data z objektu transformovat do databáze a naopak, a jakým způsobem se z databázových tabulek mají vytvořit objekty (Obr. 1.2).

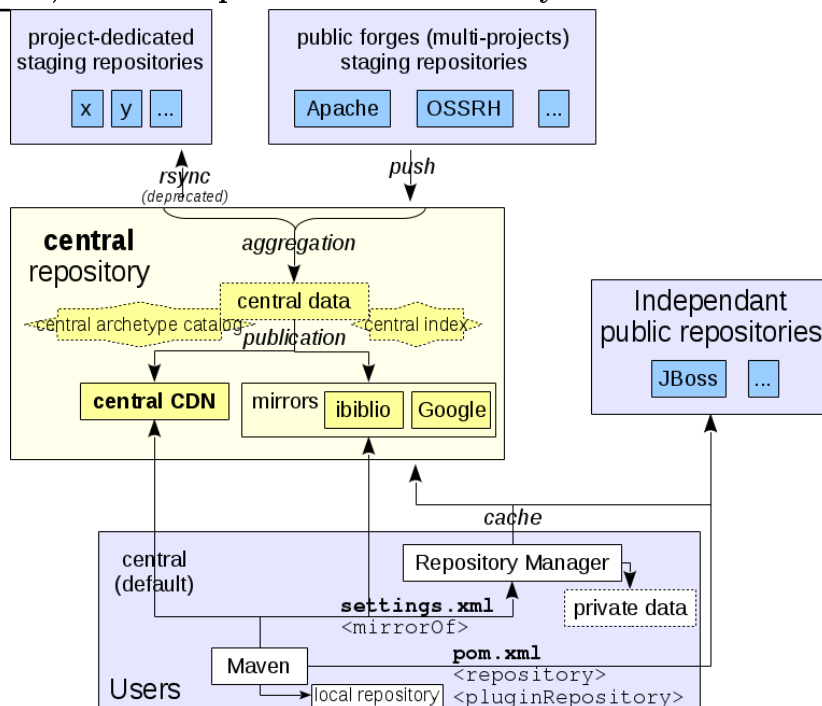


Obr. 1.2 Příklad integrace Hibernate do projektu [33]

Druhý způsob mapování objektů je použití anotace místo mapovacích souborů. V Hibernate se tedy pracuje s běžnými business objekty, přičemž mohou být sloupce tabulky spojeny přímo s atributy objektu, nebo mohou být připojeny skrze metody get/set a metody hashCode() a equals(). Nutno podotknout, že nelze použít EJB [18] (viz JavaBean), ale pouze klasické objekty - tzv. POJO (Plain Old Java Object) [19]. Poté, co jsou objekty uloženy v databázi, se na ně lze dotazovat jazykem HQL (Hibernate Query Language), který je odvozen z SQL [20] a je mu tedy velice podobný.

## 1.3 Maven

Apache Maven [21] je nástroj pro správu, řízení a automatizaci sestavení aplikace. Ačkoliv je možné použít tento nástroj pro projekty psané v různých programovacích jazycích, podporován je především jazyk Java. Základním principem fungování Mavenu je popsání projektu pomocí POM.



Obr. 1.3 Příklad integrace Maven repository [34]

Tento model popisuje softwarový projekt nejen z pohledu jeho zdrojového kódu, ale včetně závislostí na externích knihovnách, popisu procesu sestavení projektu a různých funkcí s tím spojených (jako je spuštění testů, sbírání informací o zdrojových kódech a podobně). Maven [3] sám je postaven na modulární architektuře a funguje na principu volání jednotlivých pluginů. Maven pouze obstarává dodání a spuštění nedefinovaných pluginů. Maven nemá žádné vlastní grafické uživatelské rozhraní a běží pouze na příkazové řádce a pluginy tak mohou využívat všechny nástroje, které dokáží komunikovat pomocí standardních vstupů.

POM soubor poskytuje všechny konfigurace pro jeden projekt. Za tímto účelem je definovaná jednoduchá XML struktura, která popisuje jednotlivé části projektu a jeho závislosti na externích knihovnách a nástrojích. Současně je možné definovat konstanty, které pak mohou využít jednotlivé pluginy. Tento XML dokument se nachází v kořenovém adresáři projektu a je pojmenován pom.xml. Pokud je projekt složen z více dílčích projektů nebo modulů, každý z nich má pak svůj vlastní pom.xml soubor, který dědí vlastnosti od nadřazeného souboru a může přidávat další položky. Díky této struktuře je pak možné sestavit celý projekt jediným příkazem. V pom.xml je možné u každého projektu nedefinovat jeho závislosti na externích knihovnách.

Jednotlivé prvky Artifacts jsou jednoznačně definovány podle atributů viz. (Kod. 1.1). Maven pak automaticky vyhledá a nainstaluje potřebné knihovny. Samotné vyhledávání probíhá v definovaných úložištích (repository). Kromě globální maven repository, která je veřejně přístupná, je možné založit i další soukromá nebo firemní úložiště.

Kod. 1.1 pomExample.xml

```

1 <project>
2   <!-- model version is always 4.0.0 for Maven 2.x POMs -->
3   <modelVersion>4.0.0</modelVersion>
4
5   <!-- project coordinates, i.e. a group of values which
6       uniquely identify this project -->
7
8   <groupId>com.mycompany.app</groupId>
9   <artifactId>my-app</artifactId>
10  <version>1.0</version>

```

```
11 <!-- library dependencies -->
12
13 <dependencies>
14 <dependency>
15
16 <!-- coordinates of the required library -->
17
18 <groupId>junit</groupId>
19 <artifactId>junit</artifactId>
20 <version>3.8.1</version>
21
22 <!-- this dependency is only used for running and compiling tests -->
23
24 <scope>test</scope>
25
26 </dependency>
27 </dependencies>
28 </project>
29
```

## 1.4 SOAP

SOAP [7] je protokolem pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP [32]. Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace. Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Nejznámější z nich je RPC šablona, kde jeden z účastníků komunikace je klient a na druhé straně je server. Server ihned odpovídá na požadavky klienta.

### 1.4.1 WSDL

WSDL [23] je jazykem pro popis funkcí, jež nabízí tzv. webová služba, a dále pro popis vstupů a výstupů těchto funkcí (jinými slovy, co webová služba poskytuje a jak si o to říci). Jelikož webová služba v principu komunikuje protokolem SOAP, WSDL zpravidla popisuje SOAP komunikaci. WSDL vychází z formátu XML. Podporované operace a zprávy jsou popsány abstraktně, a potom se omezují na konkrétní síťový protokol a formát zprávy. Z toho plyne, že WSDL popisuje veřejné rozhraní webové služby.

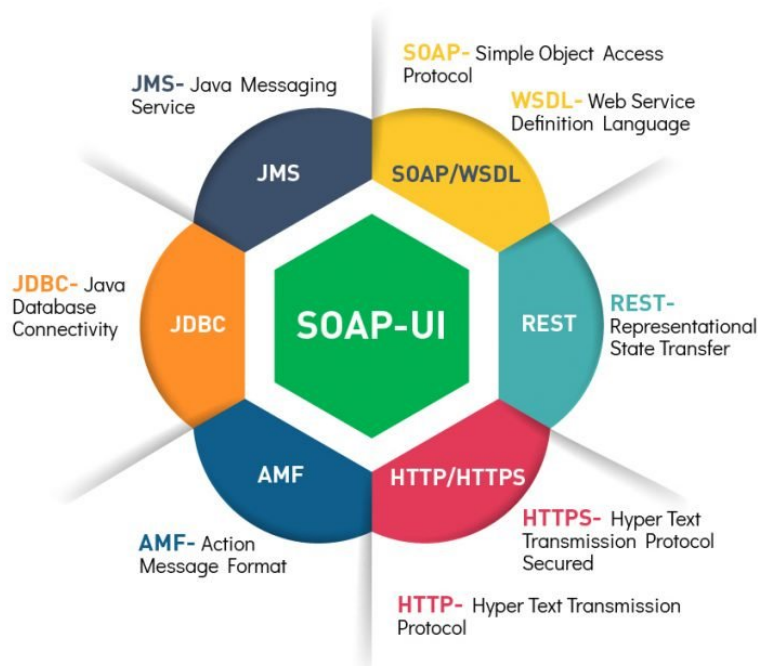
### 1.4.2 SoapUI

SoapUI [25] je open source aplikace pro testování webových služeb pro architektury orientované na služby (SOA [26]) a REST. Její funkce zahrnují kontrolu webových služeb od provolání přes testování funkčnosti až po testování výsledků dotazů na shodu s očekávaným výsledkem.

Komerční verze SoapUI Pro, která se zaměřuje hlavně na funkce určené ke zvýšení produktivity, byla také vyvinuta firmou Eviware. V roce 2011 společnost SmartBear Software firmu Eviware koupila. Projekt SoapUI byl zpočátku (v září 2005) převeden do společnosti SourceForge. Je to svobodný software, licencovaný na základě podmínek veřejné licence Evropské unie.

Je postaven výhradně na platformě Java a používá rozhraní Swing pro uživatelské rozhraní. To znamená, že SoapUI je multiplatformní. Dnes SoapUI podporují všechny hlavní vývojová prostředí (Obr. 1.4) (IDE [24]), jako je např. IDEA, Eclipse nebo NetBeans a spousta dalších. SoapUI může testovat webové služby SOAP a REST, JMS [28], AMF [27], stejně jako volání HTTP(S) a JDBC [29].

- Výhody
  - je jednoduše čitelnější pro člověka.
- Nevýhody



Obr. 1.4 Podporované protokoly SOAP-UI [30]

- komplexní zápis komunikace,
- složitost,
- pomalé zpracování jednotlivými systémy (složitě na parsování a validaci).

## 1.5 REST

REST [31] je architektura rozhraní, navržená pro distribuovaná prostředí. REST navrhl a popsal v roce 2000 Roy Fielding (jeden ze spoluautorů protokolu HTTP [32]) v rámci disertační práce *Architectural Styles and the Design of Network-based Software Architectures* [6]. Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). REST je tedy na rozdíl od známějších XML-RPC či SOAP, orientován datově, nikoli procedurálně. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim, které jsou známé pod označením CRUD:

- **Create** - vytvoření dat,
- **Retrieve** - získání požadovaných dat,
- **Update** - změnu,
- **Delete** - smazání.

Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.

### 1.5.1 JSON

JSON [35] je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat, který může být organizován v polích nebo agregován v objektech. Vstupem je libovolná datová struktura (číslo, řetězec, boolean, objekt nebo z nich složené pole), výstupem je vždy řetězec. Složitost hierarchie vstupní proměnné není teoreticky nijak omezena.

Kolekce párů „název/hodnota“ bývá v rozličných jazycích realizována jako objekt, záznam (record), struktura (struct), slovník (dictionary), hash tabulka, klíčový seznam (keyed list) nebo asociativní pole.

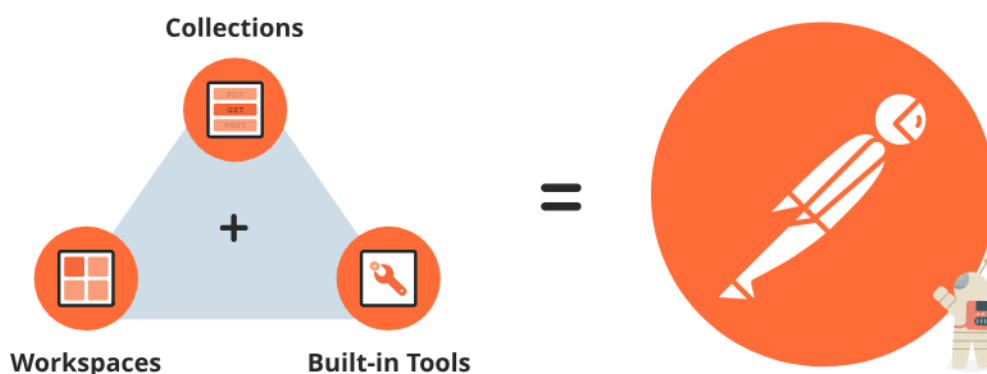
Seřazený seznam hodnot je ve většině jazyků realizován jako pole, vektor, seznam (list) nebo posloupnost (sequence).

Jedná se o univerzální datové struktury a v podstatě všechny moderní programovací jazyky je v nějaké formě podporují. Je tedy logické, aby na nich byl založen i na jazyce nezávislý výměnný formát.

- Object (Objekt) je uvozen znakem „{“ (levá složená závorka) a zakončen znakem „}“ (pravá složená závorka). Každý název je následován znakem „:“ (dvojtečka) a páry „název/hodnota“ jsou pak odděleny znakem „，“ (čárka).
- Array (Pole) je seřazenou kolekcí hodnot. Začíná znakem „[“ (levá hranatá závorka) a končí znakem „]“ (pravá hranatá závorka). Hodnoty jsou odděleny znakem „，“ (čárka).
- Value (Hodnotou) rozumíme řetězec uzavřený do dvojitých uvozovek, číslo, true, false, null, objekt nebo pole. Tyto struktury mohou být vnořovány.
- String (Řetězcem) je nula nebo více znaků kódování Unicode, uzavřených do dvojitých uvozovek a využívající únikových sekvencí (escape sequence) s použitím zpětného lomítka. Znak je reprezentován jako řetězec s jediným znakem. Řetězec je velmi podobný řetězcům z jazyků C nebo Java.
- Number (Číslo) je podobné číslům z jazyků C a Java. Jedinou výjimkou je, že není používán oktalový ani hexadecimální zápis.

### 1.5.2 Postman

Postman [36] (Obr. 1.5) je open source aplikace pro testování webových služeb pro architektury orientované na REST.



Obr. 1.5 Schéma užití Postman [36]

---

## 1.6 JavaMail

Knihovna pro přijímání a odesílání mailů. Javamail [37] je rozhraní jazyka Java, které pracuje na základě protokolů SMTP, POP3 a IMAP. JavaMail je integrován do platformy Java EE, ale nabízí i volitelný balíček pro použití v Java SE.

## 1.7 CSV

CSV [38] je jednoduchý souborový formát pro výměnu tabulkových dat. Soubor ve formátu CSV sestává z řádků, ve kterých jsou jednotlivé položky odděleny znakem čárka „,“. Hodnoty položek mohou být uzavřeny do uvozovek „““, což umožňuje, aby text položky obsahoval čárku. Pokud text položky obsahuje uvozovky, jsou pak zdvojeny.

## 1.8 PostGIS

PostGIS [4] je open source software. Jedná se o nadstavbu pro objektově-relační databázový systém PostgreSQL, která přidává podporu pro geografické objekty [5] (tzv. geoprvky). PostGIS [39] implementuje specifikaci „Simple Features for SQL“ konsorcia Open Geospatial Consortium.

## II. ANALYTICKÁ ČÁST

---

## 2 Analýza obecně

Analýza je vědecká metoda založená na dekompozici celku na elementární části, je to metoda zkoumání složitějších skutečností rozkladem (dissolution) na jednodušší. Cílem analýzy je tedy identifikovat podstatné a nutné vlastnosti elementárních částí celku, poznat jejich podstatu a zákonitosti. Analýza je také způsob výkladu, odděluje jednotlivé jevy a zkoumá je izolovaně. Opačný postup k analýze se nazývá syntéza. Cílem analýzy u informačních systémů není pouze zdokumentování stávajícího stavu, ale zejména pochopení logiky fungování systému a tím pádem získání výchozích předpokladů pro možnou optimalizaci. Cílem analýzy tedy je:

- získat znalosti o systému,
- zjistit nedostatky a slabá místa,
- uvědomit si potřebné změny.

Analýza by měla postupovat od globálního pohledu k potřebným detailům.

### 3 Diferenční analýza

#### 3.1 Popis stávajícího stavu

Stávající systém je založený na komunikaci pomocí zasílání SMS mezi obsluhou vozu (řidič nebo steward) a dispečerem (systém). SMS mají pevnou základní strukturu a do systému jsou napojeny pomocí SMS brány (Gateway). Dále je možné zpoždění přímo zadat do systému (Obr. 3.1).

Obr. 3.1 Vkládání zpoždění v administrátorském systému

Všechny tyto možnosti ukládání dat o zpoždění jsou auditovány a je možné zobrazovat jejich historii (Obr. 3.2).

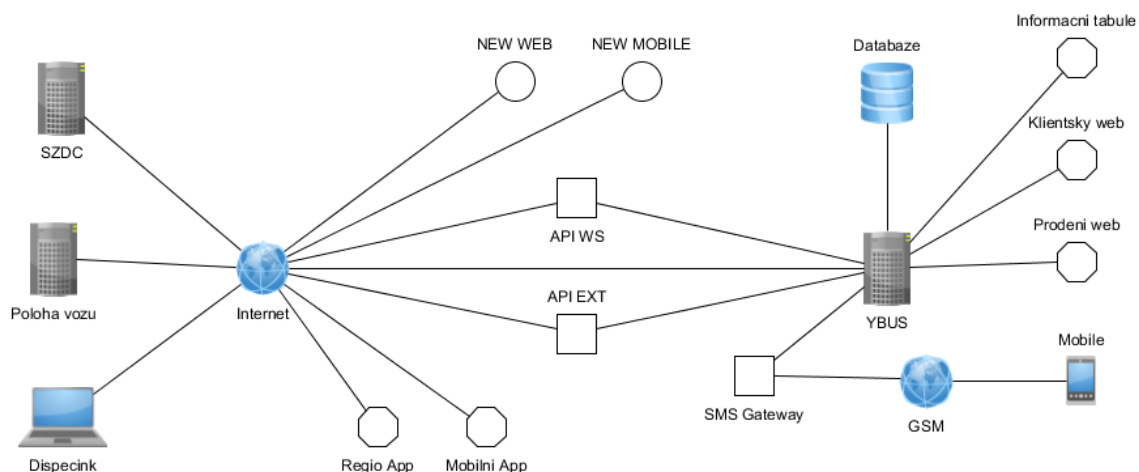
Historie zpoždění			
8.9.17 13:35	sms	55	MIN zo ZAH, V KOLINE sme menili HDV, vyluky
8.9.17 13:14	adela.rajcokova	57	MIN z CET, V KOLINE sme menili HDV, v Albrechticiach vyluka
8.9.17 12:40	sms	54	MIN z CET, V KOLINE sme menili HDV, v Albrechticiach vyluka
8.9.17 11:35	sms	37	MIN z OVS, V KOLINE sme menili HDV
8.9.17 10:16	sms	40	MIN zo ZABREHU na M, V KOLINE sme menili HDV
8.9.17 9:17	sms	43	MIN V KOLINE sme menili HDV
8.9.17 9:00	terezia.kopalova	35	MIN V KOLINE menime HDV
8.9.17 8:45	sms	25	MIN V KOLINE menime HDV
8.9.17 8:36	sms	15	MIN V KOLINE, BUDU SA menit HDV

Obr. 3.2 Audit zpoždění

##### 3.1.1 L1 pohled

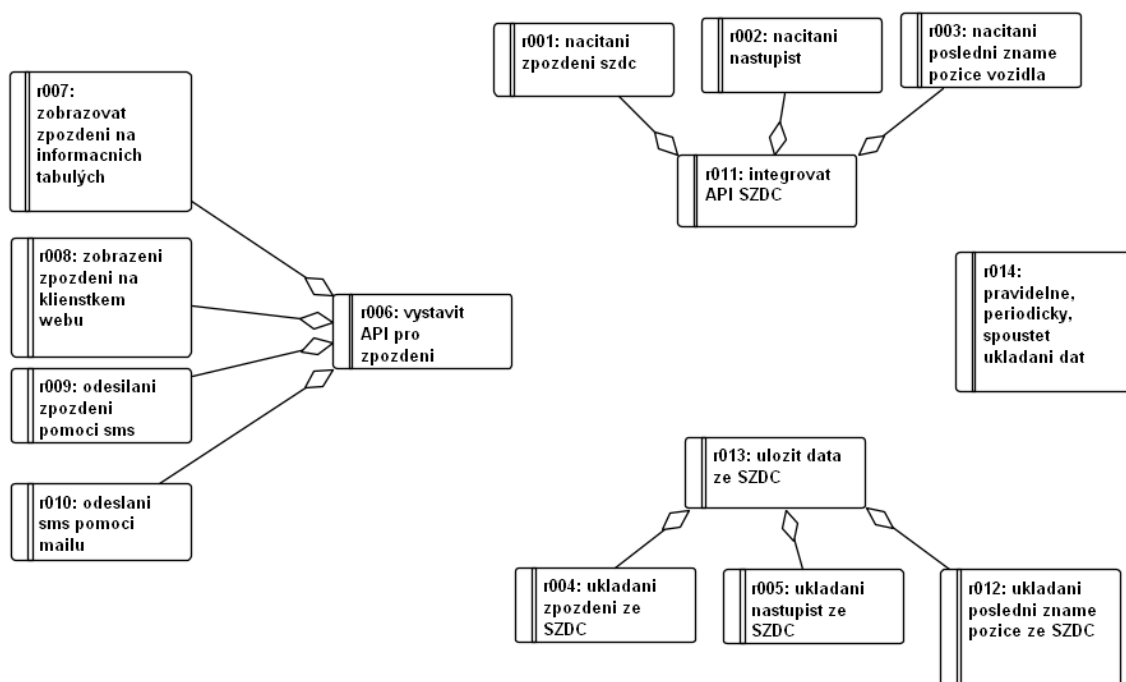
#### 3.2 Popis plánovaného stavu

Autobusy a vlaky (vagony) budou vybaveny zařízením na monitoring polohy (GPS). Souřadnice budou odeslány a zpracovány na sběrném serveru. Tyto pak budou vystaveny třetím stranám. Bude provedeno napojení nového zdroje informací o zpoždění



Obr. 3.3 Pohled L1 na současnou HW architekturu

(SŽDC) a napojení nového zdroje informací o zastávkách a stanicích (SŽDC). Bude potřeba vytvořit webové rozhraní pro distribuci informací o zpoždění.



Obr. 3.4 Funkční požadavky

### 3.3 Určení rozdílu (mezery) mezi stávajícím a cílovým stavem

Ve stávajícím stavu chybí napojení na rozhraní SŽDC (GRAPP a Informační Tabule). Chybí také napojení na server Polohavoju.cz k načítání GPS souřadnic, načítání struktury vlakových spojení. Je potřeba doplnit nové funkce pro distribuci zpoždění.

### 3.4 Návrh variant dosažení cílového stavu (alternativní strategie)

Spoje je nutné osadit technikou pro možnost sběru informací o poloze. Spojení je dále nutné napojit na servery agregující data o poloze. Agregovaná data musí být dále vyhodnocena pro potřeby výpočtu zpoždění.

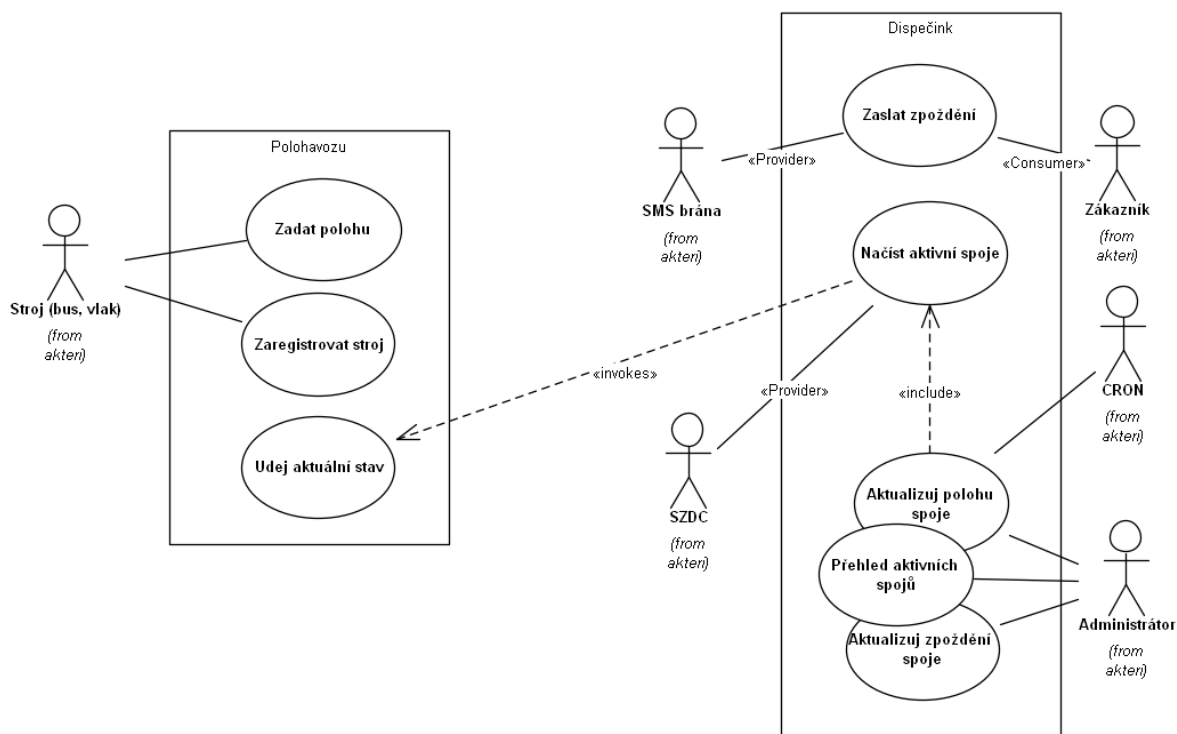
r022:  
Programovací  
jazyk JAVA

r023: Provoz 24  
hodin/ 7 dní v  
týdnu

r024:  
Dostupnost  
systému 99.9%

r025:  
Komunikační  
rozhraní v REST  
architektuře

Obr. 3.5 Nefunkční požadavky



Obr. 3.6 Případy užití

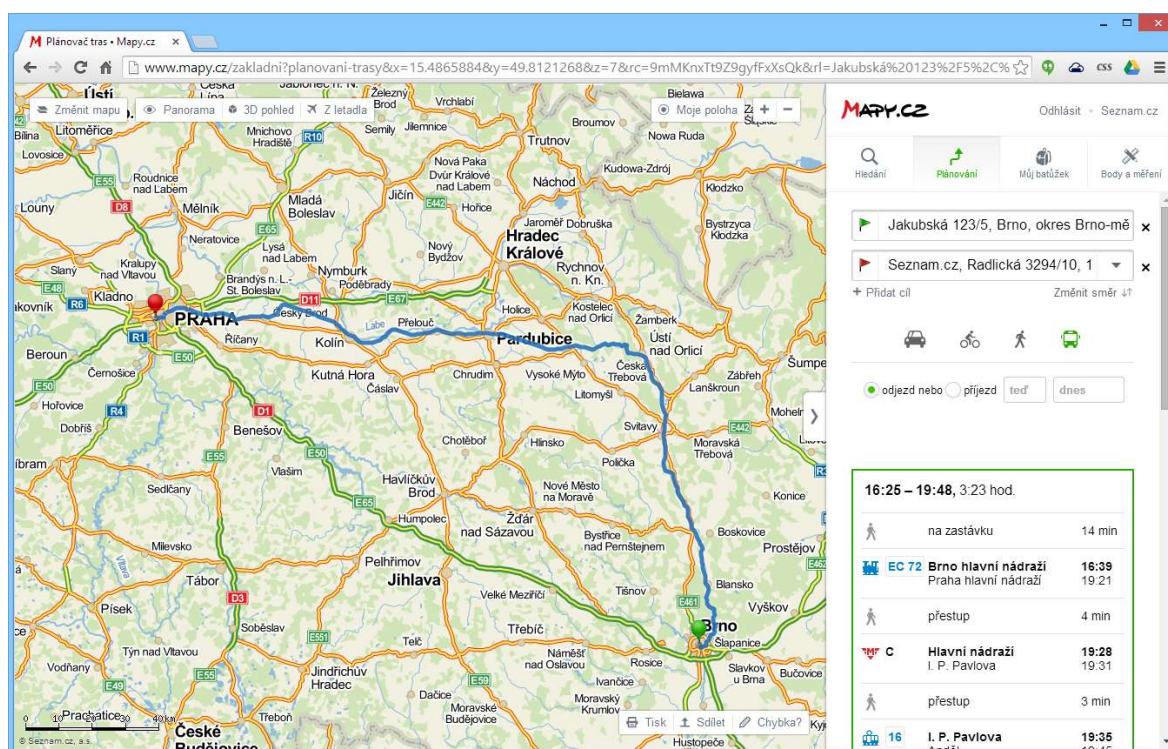
### 3.5 Popis stavu po změnách

I nadále je využíván původní systém, založený na odesílání SMS se zadaným textem a hodnotou zpoždění. Nově jsou pak přidány zdroje dat o poloze a zpoždění spojů.

## 4 Návrh technického řešení pro výpočet zpoždění

### 4.1 Trasa spoje

Trasu spoje lze popsat pomocí definovaného datového formátu KML (Obr. 4.1). Zjednodušeně řečeno, jde o řadu GPS souřadnic, která popisuje konkrétní trasu spoje veřejné dopravy. Takto uchovaná data o trase spoje jsou výhodná jak pro budoucí distribuci mapovým systémům třetích stran, tak jsou plně dostačující pro interní potřeby správy a vizualizace na mapovém podkladu.



Obr. 4.1 Náhled spoje trasovaného pomocí KML na mapy.cz

### 4.2 Definice (základní) trasy spoje pomocí GPS souřadnic.

Spoj se může za mimořádných okolností vychýlit z plánované trasy. Pro tyto účely je nezbytné mít nadefinovanou plánovanou trasu, popsanou pomocí GPS souřadnic s dostatečnou hustotou pro následné vyhodnocení odchylky (jak v prostoru, tak v čase).

Pro snazší správu a údržbu (nejen tras) je více než vhodné trasu rozdělit na dílčí celky. Nabízí se rozdělit ji na úseky mezi zastávkami (aby nebyl příliš velký počet souřadnic).

Realizace trasování spojů proběhne pomocí serveru polohavozu.cz, který agreguje data o poloze spojů. Tyto agregovaná data vystavuje pomocí REST API s dostatečnou propustností na dotaz o poloze spoje jednou za sekundu. Spolu s GPS bude uložen i čas pořízení GPS, který odpovídá referenčnímu času vůči uražené vzdálenosti na trase.

### 4.3 Výpočet zpoždění

Na základě referenčních hodnot a aktuálních dat o spoji z provozu na právě sledované trase již lze snadno získat zpoždění daného spoje. Při realizaci je nutné řešit problematiku jednak prostorového uspořádání GIS obecně, ale také nepřesnost užitých zařízení.

---

Naměřená data je nutné nejprve normalizovat a až následně použít jako vstup do mapových funkcí pro výpočet vzdálenosti od referenčního bodu. Využití PostGIS k vracení nejbližší souřadnice tento výpočet velmi usnadňuje. Zároveň také poslouží pro samotné uložení do DB.

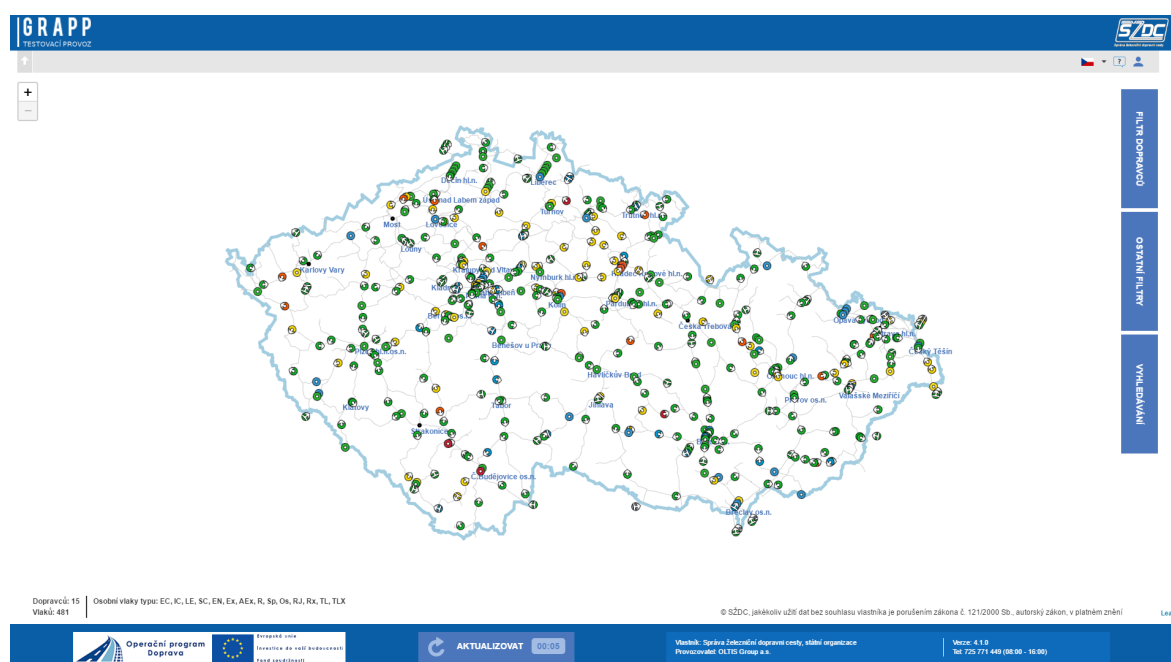
## III. PROJEKTOVÁ ČÁST

## 5 Připojení nových zdrojů dat

Pro zlepšení přesnosti stávající situace informací o zpoždění bylo rozhodnuto o napojení se na nové zdroje informací, které nám pomohou zpřesnit a hlavně automatizovat načítání, zpracování a zobrazení informací o zpoždění spojů.

### 5.1 Připojení SŽDC API

Prvním z nich je datové rozhraní GRAPP (Obr. 5.1), ze kterého se bude načítat zpoždění a poslední potvrzené průjezdové místo vozu pomocí kontrolního bodu. Druhým je datové rozhraní „Informační tabule“. Zde se budou načítat aktuální názvy nástupišť v dané zastávce pro daný spoj. Tato datová rozhraní spravuje společnost OLTIS group a.s., která jej provozuje pro společnost SŽDC.



Obr. 5.1 GRAPP [40]

#### 5.1.1 Datové rozhraní GRAPP

*Informační systém GRAPP* společnosti SŽDC poskytuje datové rozhraní formou webové služby pro další systémy (poskytování dat z informačního systému GRAPP třetím stranám). Aplikace třetích stran se mohou dotazovat do informačního systému GRAPP na aktuální data týkající se polohy vlaků veřejné osobní dopravy, respektive jejich poslední potvrzené polohy, na dopravní síti SŽDC včetně doplňujících údajů.

*Úprava pom.xml* je prvním úkolem při implementaci propojení s IS GRAPP (Kod. 5.1). Pom.xml je soubor, do kterého je třeba doplnit novou část kódu v sekci *plugin*. Pomocí tohoto přidaného kódu pak při opětovném sestavení projektu (s použitím Maven) dojde k autogenerování tříd datového rozhraní GRAPP, které pak využijeme pro načítání dat.

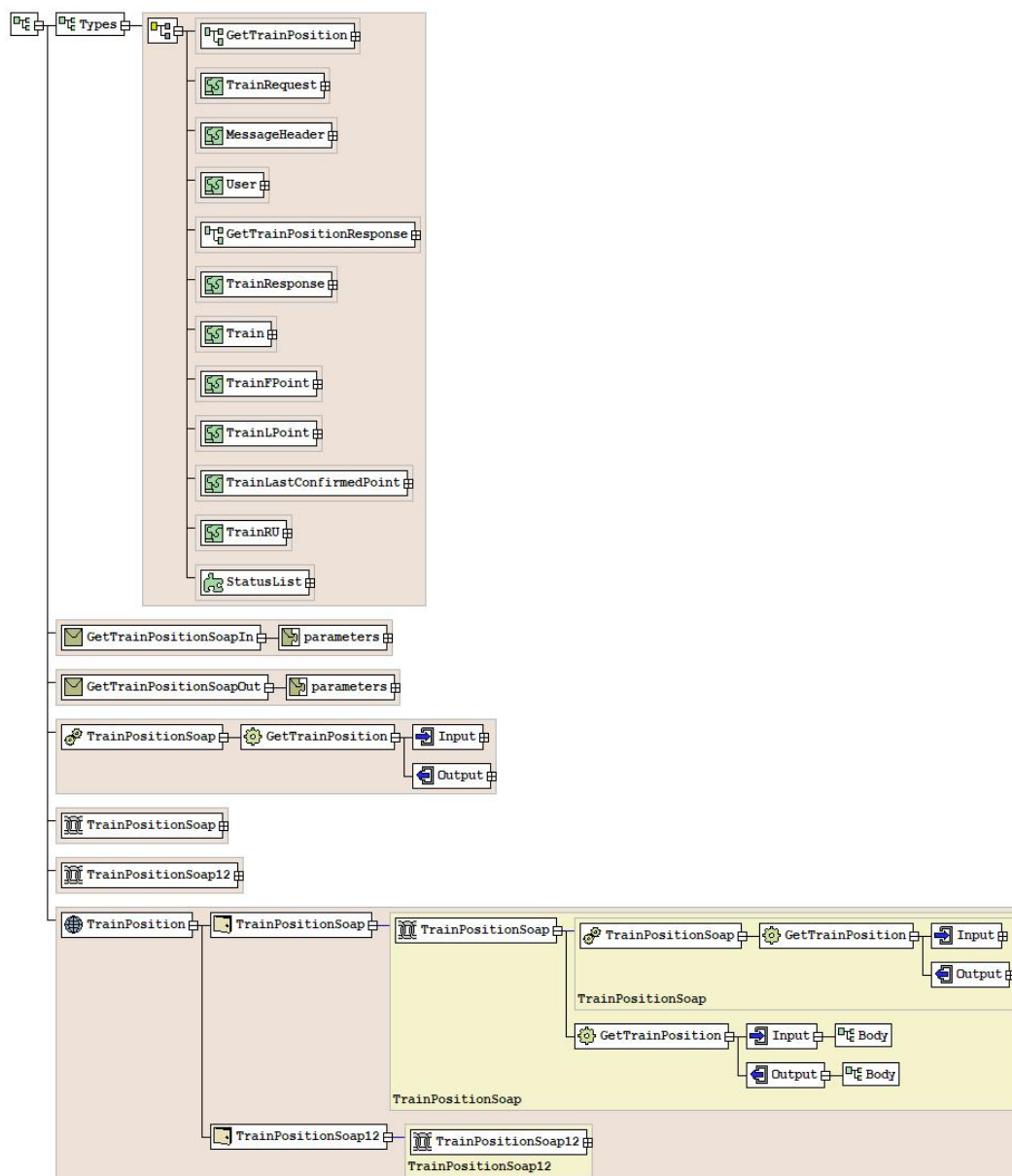
## Kod. 5.1 uprava-pom-szdc-position.xml

```

1 <execution>
2 <id>SzdcPositionWS</id>
3 <goals>
4 <goal>wsimport</goal>
5 </goals>
6 <configuration>
7 <wsdlDirectory>${project.basedir}/src/main/resources/cz/sa/szdcPosition/api</wsdlDirectory>
8 <wsdlUrls>
9 <wsdlUrl>http://provoz.szdc.cz/grappnv/trainposition.asm?wsdl</wsdlUrl>
10 </wsdlUrls>
11 <wsdlLocation>SZDCPositionApiWebService.wsdl</wsdlLocation>
12 <destDir></destDir>
13 <sourceDestDir>${project.build.directory}/generated-sources/webservices</sourceDestDir>
14 <packageName>cz.sa.szdcPosition.api</packageName>
15 <keep>true</keep>
16 <verbose>false</verbose>
17 <extension>true</extension>
18 <xnocompile>true</xnocompile>
19 </configuration>
20 </execution>

```

*Interface providera* se tvoří jako další krok (Kod. 5.2) a dále komunikační přepravky (Obr. 5.2).



Obr. 5.2 Vizualizace WSDL pro SŽDC (getTrain position)

## Kod. 5.2 SzdcProvider.java

```

1 package cz.sa.ybus.server.infrastructure.provider.szdc;
2
3 import java.util.List;
4
5 import cz.sa.ybus.core.szdc.service.dos.StationD0;
6 import cz.sa.ybus.core.szdc.service.dos.TrainPositionD0;
7 import cz.sa.ybus.core.szdc.service.dos.ViewD0;

```

```

8 import cz.sa.ybus.core.szdc.service.enums.DirectionEnum;
9
10 public interface SzdcProvider {
11     List<StationDO> getStations();
12     List<TrainPositionDO> getTrainPositions();
13     List<ViewDO> getViews(List<Integer> sr70numbers, DirectionEnum departure);
14 }
15
16
17
18

```

*Tvorba služby* je dalším krokem (Kod. 5.3). Přes tuto službu bude dotazováno datové rozhraní GRAPP.

### Kod. 5.3 SzdcPositionWSApiService.java

```

1 package cz.sa.ybus.server.infrastructure.provider.szdc;
2
3 import java.net.URL;
4 import java.util.Map;
5
6 import javax.xml.namespace.QName;
7 import javax.xml.ws.BindingProvider;
8
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Service;
11
12 import cz.sa.szdcPosition.api.TrainPosition;
13 import cz.sa.szdcPosition.api.TrainPositionSoap;
14 import cz.sa.ybus.impl.common.general.Config;
15 import cz.sa.ybus.impl.common.general.ConfigKeys;
16 import cz.sa.ybus.server.infrastructure.util.webservice.transport.HandlerChainProvider;
17
18 @Service
19 public class SzdcPositionWSApiService{
20
21     @Autowired
22     private Config config;
23     private TrainPosition service;
24     private final String RESOURCE = "SZDCPositionApiWebService.wsdl";
25     private final String NAMESPACE = "http://provoz.szdc.cz/grappws/";
26     private final String LOCALPART = "TrainPosition";
27
28     public TrainPositionSoap getSoap() {
29         TrainPositionSoap port = getService().getTrainPositionSoap();
30         addPortProperties((BindingProvider) port);
31         return port;
32     }
33
34     private TrainPosition getService() {
35         if (service == null) {
36             URL resource = TrainPosition.class.getResource(RESOURCE);
37             service = new TrainPosition(resource, new QName(NAMESPACE, LOCALPART));
38             service.setHandlerResolver(new HandlerChainProvider());
39         }
40         return service;
41     }
42
43     private void addPortProperties(BindingProvider provider) {
44         final Map<String, Object> requestContext = provider.getRequestContext();
45         requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, config.getProperty(ConfigKeys.SZDC_POSITION_API_ENDPOINT));
46     }
47 }

```

*Implementace providera* a jeho metod (Kod. 5.4), které budou využity:

### Kod. 5.4 SzdcProviderImpl.java

```

1 package cz.sa.ybus.server.infrastructure.provider.szdc.impl;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.stream.Collectors;
6
7 import org.joda.time.DateTime;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.stereotype.Service;
12
13 import com.google.api.client.repackaged.com.google.common.base.Preconditions;
14
15 import cz.sa.szdcPosition.api.Train;
16 import cz.sa.szdcPosition.api.TrainRequest;
17 import cz.sa.szdcPosition.api.TrainResponse;
18 import cz.sa.szdcInformationBoard.api.DepartureArrival;
19 import cz.sa.szdcInformationBoard.api.Design;
20 import cz.sa.szdcInformationBoard.api.RequestInformationPanel;
21 import cz.sa.szdcInformationBoard.api.RequestStationsList;
22 import cz.sa.szdcInformationBoard.api.ResponseInformationPanel;
23 import cz.sa.szdcInformationBoard.api.ResponseStationsList;
24 import cz.sa.szdcInformationBoard.api.Station;
25 import cz.sa.szdcInformationBoard.api.StationInfoPanelIdent;
26 import cz.sa.szdcInformationBoard.api.StatusList;
27 import cz.sa.szdcInformationBoard.api.View;
28 import cz.sa.ybus.core.szdc.service.dos.DesignDO;
29 import cz.sa.ybus.core.szdc.service.dos.StationDO;
30 import cz.sa.ybus.core.szdc.service.dos.TrainInformationInStationDO;
31 import cz.sa.ybus.core.szdc.service.dos.TrainPositionDO;
32 import cz.sa.ybus.core.szdc.service.dos.ViewDO;
33 import cz.sa.ybus.core.szdc.service.enums.DirectionEnum;
34 import cz.sa.ybus.server.infrastructure.provider.szdc.SzdcPositionWSApiService;
35 import cz.sa.ybus.server.infrastructure.provider.szdc.SzdcProvider;
36 import cz.sa.ybus.server.infrastructure.provider.szdc.SzdcInformationBoardWSApiService;
37 import cz.sa.ybus.server.infrastructure.provider.szdc.factory.PositionRequestFactory;
38 import cz.sa.ybus.server.infrastructure.provider.szdc.factory.InformationBoardRequestFactory;
39
40 @Service
41 public class SzdcProviderImpl implements SzdcProvider {
42
43     @Autowired
44     private SzdcInformationBoardWSApiService informationBoardService;
45     @Autowired
46     private InformationBoardRequestFactory informationBoardRequest;
47
48     @Autowired

```

```

49 private SzdcPositionWSApiService positionService;
50 @Autowired
51 private PositionRequestFactory positionRequest;
52
53 private final Logger log = LoggerFactory.getLogger(getClass());
54
55 @Override
56 public List<StationD0> getStations() {
57     RequestStationsList request = informationBoardRequest.createRequestStationsList();
58     ResponseStationsList response = informationBoardService.getSoap().getStationList(request);
59     if (response.getStatus() == StatusList.OK) {
60         return response.getStation().stream().map(st -> convertStation(st)).collect(Collectors.toList());
61     }
62     log.error("StatusCode" + response.getStatus().value());
63     throw new IllegalStateException(response.getStatus().value());
64 }
65
66 @Override
67 public List<ViewD0> getViews(List<Integer> sr70numbers, DirectionEnum direction) {
68     RequestInformationPanel request = informationBoardRequest.createRequestInformationPanel();
69     for (Integer sr70number : sr70numbers) {
70         StationInfoPanelIdent ident = informationBoardRequest.createStationInfoPanelIdent();
71         ident.setSR70(sr70number);
72         ident.setDeparture(toDepartureArrival(direction));
73         request.getStationInfoPanelIdent().add(ident);
74     }
75
76     ResponseInformationPanel response = informationBoardService.getSoap().getInformationPanels(request);
77     if (response.getStatus() == StatusList.OK) {
78         return convertView(response);
79     }
80     log.error("StatusCode" + response.getStatus().value());
81     throw new IllegalStateException(response.getStatus().value());
82 }
83
84 private DepartureArrival toDepartureArrival(DirectionEnum direction) {
85     switch (direction) {
86         case ARRIVAL:
87             return DepartureArrival.ARRIVAL;
88         case DEPARTURE:
89             return DepartureArrival.DEPARTURE;
90         case BOTH:
91             return DepartureArrival.BOTH;
92         default:
93             throw new IllegalStateException("Unexpected direction: " + direction);
94     }
95 }
96
97 @Override
98 public List<TrainPositionD0> getTrainPositions() {
99     TrainRequest request = positionRequest.createTrainRequest();
100     TrainResponse response = positionService.getSoap().getTrainPosition(request);
101     if (response.getStatus() == cz.sa.szdcPosition.api.StatusList.OK) {
102         return response.getTrain().stream().map(train -> convertTrain(train)).collect(Collectors.toList());
103     }
104     log.error("StatusCode" + response.getStatus().value());
105     throw new IllegalStateException(response.getStatus().value());
106 }
107
108 /**
109  * Konvertor pro Station
110  * @param source
111  * @return StationD0
112  */
113 private StationD0 convertStation(Station source) {
114     return new StationD0(source.getStationName(), source.getSR70());
115 }
116
117 /**
118  * Konvertor pro Train
119  * @param sourceTrain
120  * @return TrainPositionD0
121  */
122 private TrainPositionD0 convertTrain(Train sourceTrain) {
123     TrainPositionD0 trainPosition = new TrainPositionD0();
124     trainPosition.setType(sourceTrain.getType());
125     trainPosition.setTrainNumber(sourceTrain.getNumber());
126     trainPosition.setTrainName(sourceTrain.getName());
127     trainPosition.setTrid(sourceTrain.getTRID());
128     trainPosition.setComapanyName(sourceTrain.getRU().getRUName());
129     trainPosition.setUicCode(Integer.parseInt(sourceTrain.getRU().getUICCode()));
130
131     /*
132     * SZDC pouziva dva ruzne formaty cislovani zastavek jeden u SzdcInformationBoardMWSApiService (7 cislic) a druhý
133     * u SzdcPositionWSApiService (8 cislic), my tyto dva formaty sjednocujem a pouzivame prvni z nich (7 cislic).
134     */
135     StationD0 lPoint = new StationD0(sourceTrain.getLPoint().getStationName(), sourceTrain.getLPoint().getSR70() / 10);
136     trainPosition.setLastPoint(lPoint);
137     StationD0 fPoint = new StationD0(sourceTrain.getFPoint().getStationName(), sourceTrain.getFPoint().getSR70() / 10);
138     trainPosition.setFirstPoint(fPoint);
139     StationD0 lastConfirmedStation = new StationD0(sourceTrain.getLastConfirmedPoint().getStationName(),
140         (sourceTrain.getLastConfirmedPoint().getSR70() / 10));
141     trainPosition.setLastConfirmedPoint(lastConfirmedStation);
142     trainPosition.setArrival(sourceTrain.getLastConfirmedPoint().isArrival());
143     trainPosition.setReal(new DateTime(sourceTrain.getLastConfirmedPoint().getReal().toGregorianCalendar().getTime()));
144     trainPosition.setPlanned(new DateTime(sourceTrain.getLastConfirmedPoint().getPlanned().toGregorianCalendar().getTime()));
145     trainPosition.setDelay(sourceTrain.getLastConfirmedPoint().getDelay());
146     trainPosition.setDiverison(sourceTrain.getLastConfirmedPoint().isDiverison());
147     trainPosition.setSubstitution(sourceTrain.getLastConfirmedPoint().isSubstitution());
148     return trainPosition;
149 }
150
151 /**
152  * Konvertor pro View
153  * @param response
154  * @return List {@code <ViewD0>}
155  */
156 private List<ViewD0> convertView(ResponseInformationPanel response) {
157     Preconditions.checkNotNull(response);
158     List<ViewD0> viewD0s = new ArrayList<>();
159     for (View v : response.getView()) {
160         ViewD0 viewD0 = new ViewD0();
161         viewD0.setSr70(v.getHead().getSR70());
162         viewD0.setPL(v.getHead().isPL());
163         viewD0.setTr(v.getHead().isTr());
164         viewD0.setDesigns(convertDesigns(v.getDesign()));
165         viewD0s.add(viewD0);
166     }
167     return viewD0s;
168 }
169
170 /**
171  * Konvertor pro Design
172  * @param designs
173  * @return List {@code <DesignD0>}
174  */
175 private List<DesignD0> convertDesigns(List<Design> designs) {
176     Preconditions.checkNotNull(designs);
177     List<DesignD0> designD0s = new ArrayList<>();
178     for (Design d : designs) {
179         DesignD0 designD0 = new DesignD0();
180         designD0.setDeparture(d.isDeparture());
181         designD0.setValid(d.isValid());
182         designD0.setMessage(convertToMessage(d.getTextOrTrain()));
183         designD0.setTrainInformationInStations(convertToTrains(d.getTextOrTrain()));
184         designD0s.add(designD0);
185     }
186     return designD0s;

```

```

186 }
187
188 /**
189  * Konvertor pro message
190  * @param textOrTrain
191  * @return String
192  */
193 private String convertToMessage(List<Object> textOrTrain) {
194     Preconditions.checkNotNull(textOrTrain);
195     String message = "";
196     for (Object o : textOrTrain) {
197         if (o instanceof String) {
198             return message= (String) o;
199         }
200     }
201     return message;
202 }
203
204 /**
205  * Konvertor pro Train
206  * @param textOrTrain
207  * @return List {@code <TrainInformationInStationD0>}
208  */
209 private List<TrainInformationInStationD0> convertToTrains(List<Object> textOrTrain) {
210     Preconditions.checkNotNull(textOrTrain);
211     List<TrainInformationInStationD0> trainInformationInStationD0s = new ArrayList<>();
212     for (Object o : textOrTrain) {
213         TrainInformationInStationD0 trainInformationInStationD0 = new TrainInformationInStationD0();
214         if (o instanceof cz.sa.szdcInformationBoard.api.Train) {
215             cz.sa.szdcInformationBoard.api.Train trainOnInformationBoard = (cz.sa.szdcInformationBoard.api.Train) o;
216
217             trainInformationInStationD0.setTrainNumber(trainOnInformationBoard.getNumber());
218             trainInformationInStationD0.setType(trainOnInformationBoard.getType());
219             trainInformationInStationD0.setTrainName(trainOnInformationBoard.getName());
220             trainInformationInStationD0.setCompany(trainOnInformationBoard.getCompany().getValue());
221             trainInformationInStationD0.setTime(trainOnInformationBoard.getTime());
222             trainInformationInStationD0.setLine(trainOnInformationBoard.getLine());
223             trainInformationInStationD0.setDelay(trainOnInformationBoard.getDelay());
224             trainInformationInStationD0.setDestination(trainOnInformationBoard.getDestination());
225             List<String> directions = trainOnInformationBoard.getDirection().getStation();
226             trainInformationInStationD0.setDirections(directions);
227             trainInformationInStationD0.setPlatform(trainOnInformationBoard.getPlatform());
228             trainInformationInStationD0.setTrack(trainOnInformationBoard.getTrack());
229
230             trainInformationInStationD0s.add(trainInformationInStationD0);
231         }
232     }
233     return trainInformationInStationD0s;
234 }
235 }
236 }

```

*Vytvoření rozhraní a implementace služby* lze nyní provést (Kod. 5.6) (Kod. 5.7) pro načítání a aktualizaci nového zdroje dat zpoždění. Pro identifikaci spojení, na které je nutné se dotázat datového rozhraní a doplnit do requestu, je nutné vytvořit databázový dotaz. Ten bude realizován pomocí Hibernate (Kod. 5.5).

### Kod. 5.5 SzdcDao.java

```

1 package cz.sa.ybus.server.infrastructure.dao.szdc;
2
3 import java.util.List;
4
5 import org.joda.time.DateTime;
6 import org.joda.time.Duration;
7 import org.springframework.stereotype.Repository;
8 import org.springframework.transaction.annotation.Transactional;
9
10 import cz.sa.ybus.domain.bus.VehicleCategory;
11 import cz.sa.ybus.domain.timetable.Country;
12 import cz.sa.ybus.server.infrastructure.db.daosupport.AbstractDao;
13 import cz.sa.ybus.server.infrastructure.db.daosupport.QueryBuilder;
14 import cz.sa.ybus.server.infrastructure.db.entity.bus.BusConnectionVO;
15 import cz.sa.ybus.server.infrastructure.db.entity.bus.BusStationVO;
16 import cz.sa.ybus.server.infrastructure.util.SzdcUtils;
17
18 @Repository
19 @Transactional(readOnly = true)
20 public class SzdcDao extends AbstractDao {
21
22     private static final Duration DEPARTURE_INTERVAL = Duration.standardMinutes(30);
23
24     @SuppressWarnings("unchecked")
25     public List<BusStationVO> getCurrentBusStations() {
26
27         QueryBuilder hql = QueryBuilder.hql();
28
29         DateTime now = DateTime.now();
30         DateTime departureUpperLimit = now.plus(DEPARTURE_INTERVAL);
31         DateTime departureLowerLimit = now.minus(SzdcUtils.MAX_EXPECTED_DELAY);
32
33         hql.append("select distinct bs from BusStationVO bs ");
34         hql.append("join fetch bs.busConnection bc");
35         hql.append("join fetch bs.station s");
36         hql.append("join fetch s.city c");
37         hql.append("join bc.vehicles bcv");
38         hql.append("join bcv.busType bt");
39         hql.append("where s.sr70Number is not null");
40         hql.append("and ((bs.departure >= :dateTimeNow)");
41         hql.append("and bs.departure <= :dateTimeIncrement)");
42         hql.append("or (bs.arrival >= :dateTimeNow)");
43         hql.append("and bs.arrival <= :dateTimeIncrement)");
44         hql.append("and c.country = :country");
45         hql.append("and bt.vehicleCategory = :vehicleCategory");
46         hql.setParameter("dateTimeNow", departureLowerLimit);
47         hql.setParameter("dateTimeIncrement", departureUpperLimit);
48         hql.setParameter("country", Country.CZ);
49         hql.setParameter("vehicleCategory", VehicleCategory.RAIL_CAR);
50
51         return hql.createQuery(getSession()).list();
52     }
53
54     /**
55      * Pri snaze optimalizovat tento hql dotaz tim, ze pomoci clause WITH vybere jen ty potrebne bs.departue, bs.arrival jsme narazily na
56      * omezeni hibernatu
57      * kdy dochazelo k chybovemu hlaseeni "with clause can only reference columns in the driving table". Tento problem zatim neni v hibernatu
58      * vyresen 8.8.2017
59      * @see <a href="https://hibernate.atlassian.net/browse/HHH-2772">hibernate.atlassian.net/browse/HHH-2772</a>
60      */
61     @SuppressWarnings("unchecked")
62     public List<BusConnectionVO> getTodayBusConnections() {

```

```

61
62     QueryBuilder hql = QueryBuilder.hql();
63
64     DateTime now = DateTime.now();
65
66     hql.append("select distinct bc from BusConnectionVO bc ");
67     hql.append("join bc.busStations bs");
68     hql.append("join bs.station s");
69     hql.append("join fetch bc.busStations bs2");
70     hql.append("join fetch bs2.station s2");
71     hql.append("join s.city c");
72     hql.append("join fetch s2.city c2");
73     hql.append("join bc.vehicles bcv");
74     hql.append("join bcv.busType bt");
75     hql.append("where s.sr70Number is not null");
76     hql.append("and ((bs.departure >= :dateTimeStart)");
77     hql.append("and bs.departure <= :dateTimeEnd)");
78     hql.append("or (bs.arrival >= :dateTimeStart)");
79     hql.append("and bs.arrival <= :dateTimeEnd)");
80     hql.append("and c.country = :country");
81     hql.append("and bt.vehicleCategory = :vehicleCategory");
82     hql.setParameter("dateTimeStart", now.minus(SzdcUtils.MAX_EXPECTED_DELAY));
83     hql.setParameter("dateTimeEnd", now);
84     hql.setParameter("country", Country.CZ);
85     hql.setParameter("vehicleCategory", VehicleCategory.RAIL_CAR);
86
87     return hql.createQuery(getSession()).list();
88 }
89
90 }

```

## Kod. 5.6 SzdcService.java

```

1  package cz.sa.ybus.core.szdc.service;
2
3  import cz.sa.ybus.core.common.service.Service;
4
5  public interface SzdcService extends Service {
6
7      void updatePlatforms();
8
9      void updateDelays();
10
11 }

```

## Kod. 5.7 SzdcServiceImpl.java

```

1  package cz.sa.ybus.core.szdc.service;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6  import java.util.Map;
7  import java.util.Objects;
8  import java.util.Set;
9  import java.util.regex.Matcher;
10 import java.util.regex.Pattern;
11 import java.util.stream.Collectors;
12
13 import org.apache.commons.lang3.StringUtils;
14 import org.apache.commons.lang3.text.StrBuilder;
15 import org.joda.time.LocalDate;
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.stereotype.Service;
18
19 import com.google.common.collect.ImmutableSet;
20 import com.google.common.collect.Maps;
21
22 import cz.sa.ybus.core.common.service.AbstractService;
23 import cz.sa.ybus.core.szdc.service.dos.DesignDO;
24 import cz.sa.ybus.core.szdc.service.dos.StationDO;
25 import cz.sa.ybus.core.szdc.service.dos.TrainInformationInStationDO;
26 import cz.sa.ybus.core.szdc.service.dos.ViewDO;
27 import cz.sa.ybus.core.szdc.service.enums.DirectionEnum;
28 import cz.sa.ybus.domain.timetable.Country;
29 import cz.sa.ybus.server.domain.busdao.BusService;
30 import cz.sa.ybus.server.domain.property.Property;
31 import cz.sa.ybus.server.domain.property.PropertyRepository;
32 import cz.sa.ybus.server.domain.user.User;
33 import cz.sa.ybus.server.domain.user.UserRepository;
34 import cz.sa.ybus.server.infrastructure.dao.szdc.SzdcDao;
35 import cz.sa.ybus.server.infrastructure.db.entity.TimetableUtils;
36 import cz.sa.ybus.server.infrastructure.db.entity.bus.BusConnectionVO;
37 import cz.sa.ybus.server.infrastructure.db.entity.bus.BusStationVO;
38 import cz.sa.ybus.server.infrastructure.db.entity.timetable.StationVO;
39 import cz.sa.ybus.server.infrastructure.provider.szdc.SzdcProvider;
40 import cz.sa.ybus.server.infrastructure.util.MailType;
41 import cz.sa.ybus.server.infrastructure.util.Mailer;
42 import cz.sa.ybus.server.logging.LogBefore;
43
44
45 @Service
46 public class SzdcServiceImpl extends AbstractService implements SzdcService {
47     @Autowired
48     private SzdcProvider szdcProvider;
49     @Autowired
50     transient private BusService busService;
51     @Autowired
52     private SzdcDao szdcDao;
53     @Autowired
54     transient private Mailer mailer;
55     @Autowired
56     transient private PropertyRepository propertyRepository;
57     @Autowired
58     private UserRepository userRepository;
59
60     private Map<Integer, LocalDate> errorStationSendedEmail = Maps.newHashMap();
61
62     /**
63      * Vsechna Sr70 cisla zastavek, pres ktera jezdi RJ_CZ vlaky, ale uz jsou mimo YBUS a mimo nase klienty.
64      * Napr. PrahaSmichov, kde se vlak pouze otaci.
65      * 54572263, 54583666, 54572289, 54581173-<br>
66      * SZDC pouziva dva ruzne formaty cislovani zastavek jeden u SzdcInformationBoardWSApiService (7 cislic) a druhu
67      * u SzdcPositionWSApiService (8 cislic), my tyto dva formaty sjednocujem a pouzivame prvni z nich (7 cislic).
68      */
69     private static final Set<Integer> EXTERNAL_SR70_NUMBERS = ImmutableSet.of(5457226, 5458366, 5457228, 5458117);
70
71     @Override
72     @LogBefore
73     public void updatePlatforms() {
74         List<TrainStationInfo> stationInfos = getCurrentDepartingStationsInfo();
75
76         /**
77          * z listu stationInfos je potreba vyselektovat pro uceli dotazu do SZDC
78          * cisla zastavek sr70number a odstranit duplicity
79          */
80         List<Integer> sr70numbers = stationInfos.stream().map(sr70 -> sr70.getSr70Number()).distinct().collect(Collectors.toList());
81

```

```
82 List<ViewD0> viewD0s = szdcProvider.getViews(sr70numbers, DirectionEnum.BOTH);
83
84 if (viewD0s.size() != sr70numbers.size()) {
85     List<Integer> sr70numbersFromViewD0s = viewD0s.stream().map(viewD0 -> viewD0.getSr70()).collect(Collectors.toList());
86     List<TrainStationInfo> unavailableStations = stationInfos.stream().filter(si -> !sr70numbersFromViewD0s.contains(si.getSr70Number()));
87     collect(Collectors.toList());
88     sendErrorMail(unavailableStations);
89 }
90
91 Map<Long, String> platformInfosForUpdate = new HashMap<>();
92
93 for (ViewD0 viewD0 : viewD0s) {
94     /**
95     * Pokud SZDC zobrazuje v zastavce na informacni tabuli nastupiste potom Pl = true
96     */
97     Boolean pl = viewD0.getPl();
98     /**
99     * Pokud SZDC zobrazuje v zastavce na informacni tabuli kolej potom Tr = true
100     */
101     Boolean tr = viewD0.getTr();
102     for (DesignD0 designD0 : viewD0.getDesigns()) {
103         if (designD0.isValid()) {
104             for (TrainInformationInStationD0 trainInformationInStationD0 : designD0.getTrainInformationInStations()) {
105                 String trainNumber = trainInformationInStationD0.getTrainNumber();
106                 for (TrainStationInfo stationInfo : stationInfos) {
107                     if (trainNumber.equals(stationInfo.getTrainNumber())
108                         && (viewD0.getSr70().intValue() == stationInfo.getSr70Number().intValue())) {
109                         StringBuilder platformInfo = new StringBuilder();
110                         String platform = trainInformationInStationD0.getPlatform();
111                         String track = trainInformationInStationD0.getTrack();
112                         platformInfo.append(pl && platform.length() > 0 ? platform : "");
113                         platformInfo.append("/");
114                         platformInfo.append(tr && track.length() > 0 ? track : "");
115                         platformInfosForUpdate.put(stationInfo.getBusStationId(), platformInfo.toString());
116                     }
117                 }
118             }
119         } else {
120             log.error("Valid" + designD0.isValid());
121         }
122     }
123 }
124 for (Map.Entry<Long, String> entry : platformInfosForUpdate.entrySet()) {
125     busService.updateBusStationPlatform(entry.getKey(), entry.getValue());
126 }
127 }
128
129 private List<TrainStationInfo> getCurrentDepartingStationsInfo() {
130     List<TrainStationInfo> trainStationInfos = new ArrayList<>();
131
132     List<BusStationV0> busStationV0s = szdcDao.getCurrentBusStations();
133     for (BusStationV0 busStationV0 : busStationV0s) {
134         String connectionCode = busStationV0.getBusConnection().getConnectionCode(); // z DB se nam vrati 'RJ 1001' a potrebujem pouze '1001'
135         Pattern pattern = Pattern.compile("\\d+");
136         Matcher m = pattern.matcher(connectionCode);
137         String code = null;
138         while (m.find()) {
139             code = m.group();
140         }
141         if (code != null) {
142             TrainStationInfo tsi = new TrainStationInfo();
143             tsi.setBusStationId(busStationV0.getId());
144             tsi.setSr70Number(convertSr70toSzdFormat(busStationV0.getStation().getSr70Number()));
145             tsi.setTrainNumber(code);
146             tsi.setStationName(busStationV0.getStation().getCity() + " " + busStationV0.getStation().getName());
147             trainStationInfos.add(tsi);
148         }
149     }
150     return trainStationInfos;
151 }
152
153 /**
154 * V YBUS DB mame ulozene cisla sr70 ve tvaru, ktery SZDC nepodporuje ale
155 * pomoci vhodne konverze jej lze upravit na tvar, ktery SZDC uz prijme
156 */
157 @param Long
158 sr70number
159 @return Integer sr70number
160 */
161 private Integer convertSr70toSzdFormat(Long sr70number) {
162     String uicCountryCode = Country.CZ.getUicCountryCode();
163     return Integer.valueOf(uicCountryCode + String.valueOf(sr70number / 10));
164 }
165
166 private static class TrainStationInfo {
167     private Integer sr70Number;
168     private Long busStationId;
169     private String trainNumber;
170     private String stationName;
171
172     public String getStationName() {
173         return stationName;
174     }
175
176     public void setStationName(String stationName) {
177         this.stationName = stationName;
178     }
179
180     public Integer getSr70Number() {
181         return sr70Number;
182     }
183
184     public void setSr70Number(Integer sr70Number) {
185         this.sr70Number = sr70Number;
186     }
187
188     public Long getBusStationId() {
189         return busStationId;
190     }
191
192     public void setBusStationId(Long busStationId) {
193         this.busStationId = busStationId;
194     }
195
196     public String getTrainNumber() {
197         return trainNumber;
198     }
199
200     public void setTrainNumber(String trainNumber) {
201         this.trainNumber = trainNumber;
202     }
203 }
204
205 @Override
206 @LogBefore
207 public void updateDelays() {
208     /**
209     * Vypnuti/zapnuti nacistani zpozdeni od szdc
210     * do logu se vypiseje kdy byla sluzba vynuta a kym
211     */
212     Property property = propertyRepository.getProperty("szdc.updateDelayEnable");
213     if (!property.getBooleanValue()) {
214         User user = userRepository.getUser(property.getUpdatedByUserId());
215     }
216 }
```

```

218     log.info("Aktualizace zpozdění je dočasne vypnuta od : " + property.getDateUpdated() + ", uzivatelem : " + user.getLogin());
219     return;
220 }
221
222 Map<Long, TrainPositionD0> delayInfosForUpdate = new HashMap<>();
223
224 List<BusConnectionV0> busConnectionV0s = szdcDao.getTodayBusConnections();
225
226 List<TrainPositionD0> trainPositionD0s = szdcProvider.getTrainPositions();
227 for (TrainPositionD0 trainPositionD0 : trainPositionD0s) {
228     for (BusConnectionV0 busConnectionV0 : busConnectionV0s) {
229         if (isBusConnectionMatching(busConnectionV0, trainPositionD0)) {
230             delayInfosForUpdate.put(busConnectionV0.getId(), trainPositionD0);
231         }
232     }
233 }
234 for (Map.Entry<Long, TrainPositionD0> entry : delayInfosForUpdate.entrySet()) {
235     final Long busConnectionId = entry.getKey();
236     final TrainPositionD0 trainPosition = entry.getValue();
237     final Integer delay = trainPosition == null ? null : trainPosition.getDelay();
238     final StationD0 lastConfirmedPoint = trainPosition == null ? null : trainPosition.getLastConfirmedPoint();
239     final String stationName = lastConfirmedPoint == null ? null : lastConfirmedPoint.getStationName();
240     final String messageInfo = BusService.SZDC_DELAY_INFO + " (" + StringUtils.stripAccents(stationName) + ")";
241     final Integer sr70 = lastConfirmedPoint == null ? null : lastConfirmedPoint.getSr70();
242     busService.updateBusConnectionDelay(busConnectionId, delay, messageInfo, sr70);
243 }
244
245 }
246
247 /**
248  * Pro porovani cisel vlaku musime upravit SZDC cislo na format vhodny pro YBUS
249  * @param trainNumber
250  * @return String
251  */
252 private String convertSzdcTrainNumberToYbus(Integer trainNumber) {
253     return "RJ " + trainNumber;
254 }
255
256 /**
257  * Posle mail o chybe informacni tabule v zastavkach na prislusnou adresu jednou za den
258  * @param unavailableStations
259  */
260 private void sendErrorMail(List<TrainStationInfo> unavailableStations) {
261     LocalDate today = LocalDate.now();
262     //vsechny zaznamy, ktere jsou uz den stare budou promazany
263     errorStationSendedEmail.values().removeIf(oldValue -> oldValue.isBefore(today));
264     String subject = "SZDC: Nedostupne informace (data) z informacni tabule";
265     String emails = propertyRepository.getProperty("szdc.errorMailTo").getValue();
266     for (TrainStationInfo us : unavailableStations) {
267         LocalDate lastSentDate = errorStationSendedEmail.get(us.getSr70Number().intValue());
268         //kontrola jestli byl uz dnes poslan email
269         if (Objects.equals(today, lastSentDate)) {
270             continue;
271         }
272         String msg = "K informacni tabuli v zastavce: "+us.getStationName()+" , SZDC kod ( "+us.getSr70Number()+" ) nemame opraveni k cteni dat";
273         mailer.sendMultiMail(MailType.INTERNAL, null, emails, subject, msg, msg);
274         errorStationSendedEmail.put(us.getSr70Number(), today);
275     }
276 }
277
278 private boolean isBusConnectionMatching(BusConnectionV0 busConnectionV0, TrainPositionD0 trainPositionD0) {
279     String trainNumber = convertSzdcTrainNumberToYbus(trainPositionD0.getTrainNumber());
280     if (!busConnectionV0.getConnectionCode().equals(trainNumber)) {
281         return false;
282     }
283     if (Objects.equals(trainPositionD0.getLastConfirmedPoint().getSr70(), busConnectionV0.getLastPostionSzdc())) {
284         return false;
285     }
286
287     // Vlak muze jet i mimo trasu definovanou Ybusem. (Např. zajet do točny v PrahaSmichov.) Tam uz ale neprevazi klienty a tj. uz nas nezajima
288     // zpozdění.
289     return isPositionAtYbusRoute(busConnectionV0, trainPositionD0);
290 }
291
292 // Pozn. Metoda je naimplementovana na miru pro (k zari 2017) jedinou vlakovou trasu Praha-Havírov (pripadne protazenou dal),
293 // na ktere nam SZDC poskytuje info. o zpozdění.
294 private boolean isPositionAtYbusRoute(BusConnectionV0 busConnectionV0, TrainPositionD0 trainPositionD0) {
295     if (EXTERNAL_SR70_NUMBERS.contains(trainPositionD0.getLastConfirmedPoint().getSr70())) {
296         return false;
297     }
298
299     StationV0 firstStationV0 = TimetableUtils.getFirstBusStation(busConnectionV0.getBusStations()).getStation();
300     StationV0 lastStationV0 = TimetableUtils.getLastBusStation(busConnectionV0.getBusStations()).getStation();
301
302     if (isMatchingPosition(trainPositionD0, firstStationV0)) {
303         return !trainPositionD0.getArrival();
304     }
305     if (isMatchingPosition(trainPositionD0, lastStationV0)) {
306         return trainPositionD0.getArrival();
307     }
308     return true;
309 }
310
311 private boolean isMatchingPosition(TrainPositionD0 trainPositionD0, StationV0 stationV0) {
312     // SZDC mame jen v Cesku
313     if (stationV0.getCity().getCountry() != Country.CZ) {
314         return false;
315     }
316     return (Objects.equals(trainPositionD0.getLastConfirmedPoint().getSr70(), convertSR70toSzdcFormat(stationV0.getSr70Number())));
317 }
318 }

```

*Automatické spuštění* je posledním krokem k dokončení operace. Pomocí CRONu vytvoříme periodické spuštění dotazu na datové rozhraní a následné uložení do data-báze.

### Kod. 5.8 applicationContex-cron-position.xml

```

1     <bean class="org.springframework.scheduling.quartz.CronTriggerFactoryBean">
2         <property name="jobDetail">
3             <bean class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
4                 <property name="targetObject" ref="szdcServiceImpl" />
5                 <property name="targetMethod" value="updateDelays" />
6                 <property name="concurrent" value="false" />
7             </bean>
8         </property>
9         <property name="cronExpression" value="${szdcUpdateDelay.cronExpression}" />
10    </bean>

```

### 5.1.2 Datové rozhraní „Informační tabule“

Další možností načítání zdroje dat zpoždění je datové rozhraní Informační tabule. Tato část implementace nakonec nebyla realizována. Její přínos pro nový dispečerský systém byl nulový. Toto rozhraní je primárně využito k načítání informací o zastávkách a nástupišťích kudy spoj projíždí.

*System Informační TABULE* vlastní opět společnost SŽDC. Tento systém poskytuje datové rozhraní formou webové služby pro další systémy. Cílem je poskytování dat z informačního systému TABULE pro třetí strany. Aplikace třetích stran se mohou dotazovat do informačního systému TABULE na aktuální data týkající se odjezdových a příjezdových tabulí dopravních bodů na dopravní síti SŽDC, které mohou tato data poskytovat.

## 5.2 Připojení Polohavozu.cz

V první fázi byla napojena datová rozhraní od SŽDC, která bohužel neobsahovala všechna potřebná data. Chybějícím prvkem byla poloha, souřadnice GPS, potřebná k určení, kde se vůz nachází a pomoci ní vypočítat zpoždění.

### *Tvorba providera* (inteface, implementace)

Kod. 5.9 VehiclePositionProvider.java

```
1 package cz.sa.ybus.server.infrastructure.provider.vehiclePosition;
2
3 import java.util.List;
4 import java.util.Map;
5
6 import cz.sa.ybus.core.vehiclePosition.service.dos.PositionDO;
7 import cz.sa.ybus.core.vehiclePosition.service.dos.VehicleInfoDO;
8
9 public interface VehiclePositionProvider {
10
11     List<VehicleInfoDO> getVehicleInfos(); //vrati info o vozech
12
13     Map<Long, PositionDO> getPositions(); // vrati poluhu bodu
14 }
```

Kod. 5.10 VehiclePositionProviderImpl.java

```
1 package cz.sa.ybus.server.infrastructure.provider.vehiclePosition.impl;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import java.util.Collections;
6 import java.util.HashMap;
7 import java.util.List;
8 import java.util.Map;
9
10 import javax.ws.rs.core.MediaType;
11
12 import org.apache.cxf.interceptor.LoggingInInterceptor;
13 import org.apache.cxf.jaxrs.client.ClientConfiguration;
14 import org.apache.cxf.jaxrs.client.WebClient;
15 import org.codehaus.jackson.JsonParseException;
16 import org.codehaus.jackson.jaxrs.JacksonJaxbJsonProvider;
17 import org.codehaus.jackson.map.JsonMappingException;
18 import org.codehaus.jackson.map.ObjectMapper;
19 import org.codehaus.jackson.map.type.MapType;
20 import org.codehaus.jackson.map.type.TypeFactory;
21 import org.slf4j.Logger;
22 import org.slf4j.LoggerFactory;
23 import org.springframework.beans.factory.annotation.Autowired;
24 import org.springframework.stereotype.Service;
25
26 import com.google.common.collect.Maps;
27
28 import cz.sa.ybus.core.common.annotations.NotNull;
29 import cz.sa.ybus.core.vehiclePosition.service.dos.PositionDO;
30 import cz.sa.ybus.core.vehiclePosition.service.dos.VehicleInfoDO;
31 import cz.sa.ybus.impl.common.general.Config;
32 import cz.sa.ybus.impl.common.general.ConfigKeys;
33 import cz.sa.ybus.server.infrastructure.provider.vehiclePosition.VehiclePositionProvider;
34 import cz.sa.ybus.server.infrastructure.provider.vehiclePosition.domain.Position;
35 import cz.sa.ybus.server.infrastructure.provider.vehiclePosition.domain.Vehicle;
36 import cz.sa.ybus.server.infrastructure.provider.vehiclePosition.domain.VehicleInfos;
37 import cz.sa.ybus.server.infrastructure.provider.vehiclePosition.domain.VehiclePosition;
38
39 @Service
40 public class VehiclePositionProviderImpl implements VehiclePositionProvider {
41
42     @Autowired
43     private Config config;
44
45     private static final String INIT_VEHICLES_PATH = "/init-vehicles/";
46     private static final String DATA = "/data/";
47     private static final String JSON = "json";
48
49     private final Logger log = LoggerFactory.getLogger(getClass());
```

```

50
51 private WebClient getWebClient() {
52
53     final String vehiclesEndpoint = config.getProperty(ConfigKeys.VEHICLE_POSITION_ENDPOINT);
54
55     final WebClient webClient = WebClient.create(vehiclesEndpoint, Collections.singletonList(new JacksonJaxbJsonProvider()));
56     ClientConfiguration clientConfiguration = WebClient.getConfig(webClient);
57     clientConfiguration.getInInterceptors().add(new LoggingInInterceptor());
58
59     return webClient;
60 }
61
62 @Override
63 public List<VehicleInfoDO> getVehicleInfos() {
64
65     final String code = config.getProperty(ConfigKeys.VEHICLE_POSITION_CODE);
66
67     final WebClient webClient = getWebClient();
68     webClient.path(INIT_VEHICLES_PATH + code);
69     webClient.type(MediaType.APPLICATION_JSON_TYPE);
70     webClient.accept(MediaType.APPLICATION_JSON_TYPE);
71
72     final String resp = webClient.get(String.class);
73     final ObjectMapper mapper = new ObjectMapper();
74     final TypeFactory typeFactory = mapper.getTypeFactory();
75
76     List<VehicleInfos> listVehicleInfos = new ArrayList<>();
77
78     try {
79         listVehicleInfos = mapper.readValue(resp, typeFactory.constructCollectionType(List.class, VehicleInfos.class));
80     }
81     catch (JsonParseException e) {
82         log.error("Deserialization error: " + e);
83     }
84     catch (JsonMappingException e) {
85         log.error("Deserialization error: " + e);
86     }
87     catch (IOException e) {
88         log.error("Deserialization error: " + e);
89     }
90
91     final List<Vehicle> vehicles = new ArrayList<>();
92
93     for (final VehicleInfos vehicleInfos : listVehicleInfos) {
94         for (final Vehicle vehicle : vehicleInfos.getVehicles()) {
95             vehicles.add(vehicle);
96         }
97     }
98
99     return convertVehicles(vehicles);
100 }
101
102 @NotNull
103 private List<VehicleInfoDO> convertVehicles(@NotNull final List<Vehicle> vehicles) {
104     final List<VehicleInfoDO> vehicleInfoDOs = new ArrayList<>();
105     for (final Vehicle vehicle : vehicles) {
106         final VehicleInfoDO vehicleInfoDO = new VehicleInfoDO();
107         vehicleInfoDO.setVehicleId(Long.parseLong(vehicle.getVehicleId()));
108         vehicleInfoDO.setVehicleName(vehicle.getVehicleTitle());
109         vehicleInfoDOs.add(vehicleInfoDO);
110     }
111
112     return vehicleInfoDOs;
113 }
114
115 @Override
116 public Map<Long, PositionDO> getPositions() {
117
118     final String code = config.getProperty(ConfigKeys.VEHICLE_POSITION_CODE);
119
120     final WebClient webClient = getWebClient();
121     webClient.path(DATA + code + "." + JSON);
122     webClient.type(MediaType.APPLICATION_JSON_TYPE);
123     webClient.accept(MediaType.APPLICATION_JSON_TYPE);
124
125     final String resp = webClient.get(String.class);
126     final ObjectMapper mapper = new ObjectMapper();
127     final TypeFactory typeFactory = mapper.getTypeFactory();
128     final MapType mapType = typeFactory.constructMapType(HashMap.class, String.class, VehiclePosition.class);
129
130     Map<String, VehiclePosition> positionMap = Maps.newHashMap();
131
132     try {
133         positionMap = mapper.readValue(resp, mapType);
134     }
135     catch (JsonParseException e) {
136         log.error("Deserialization error: " + e);
137     }
138     catch (JsonMappingException e) {
139         log.error("Deserialization error: " + e);
140     }
141     catch (IOException e) {
142         log.error("Deserialization error: " + e);
143     }
144
145     return convertPositions(positionMap);
146 }
147
148 @NotNull
149 private Map<Long, PositionDO> convertPositions(@NotNull final Map<String, VehiclePosition> positionMap) {
150     final Map<Long, PositionDO> positionMapDO = Maps.newHashMap();
151     for (final Map.Entry<String, VehiclePosition> entry : positionMap.entrySet()) {
152         final PositionDO positionDO = new PositionDO();
153         final Long id = Long.parseLong(entry.getKey());
154         final Position position = entry.getValue().getPosition();
155         positionDO.setLatitude(Double.parseDouble(position.getLatitude()));
156         positionDO.setLongitude(Double.parseDouble(position.getLongitude()));
157         positionDO.setDateTime(Long.parseLong(position.getTime()));
158         positionMapDO.put(id, positionDO);
159     }
160
161     return positionMapDO;
162 }
163 }

```

## Kod. 5.11 PositionDO.java

```

1 package cz.sa.ybus.core.vehiclePosition.service.dos;
2
3 import java.io.Serializable;
4
5 public class PositionDO implements Serializable {
6
7     private Double latitude;
8     private Double longitude;
9     private Long dateTime;
10
11     public Double getLatitude() {
12         return latitude;
13     }
14
15     public void setLatitude(Double latitude) {

```

```

16     this.latitude = latitude;
17 }
18
19 public Double getLongitude() {
20     return longitude;
21 }
22
23 public void setLongitude(Double longitude) {
24     this.longitude = longitude;
25 }
26
27 public Long getDateime() {
28     return dateTime;
29 }
30
31 public void setDateime(Long dateTime) {
32     this.dateTime = dateTime;
33 }
34 }

```

## Kod. 5.12 TrackedVehiclePathDO.java

```

1 package cz.sa.ybus.core.vehiclePosition.service.dos;
2
3 import java.io.Serializable;
4 import java.util.List;
5
6 public class TrackedVehiclePathDO extends VehicleInfoDO implements Serializable {
7
8     private PositionDO lastPosition;
9     private List<PositionDO> positionDOs;
10
11     public PositionDO getLastPosition() {
12         return lastPosition;
13     }
14
15     public void setLastPosition(PositionDO lastPosition) {
16         this.lastPosition = lastPosition;
17     }
18
19     public List<PositionDO> getPositionDOs() {
20         return positionDOs;
21     }
22
23     public void setPositionDOs(List<PositionDO> positionDOs) {
24         this.positionDOs = positionDOs;
25     }
26
27 }

```

## Kod. 5.13 VehicleInfoDO.java

```

1 package cz.sa.ybus.core.vehiclePosition.service.dos;
2
3 import java.io.Serializable;
4
5 public class VehicleInfoDO implements Serializable {
6
7     private Long vehicleId;
8     private String vehicleName;
9
10     public Long getVehicleId() {
11         return vehicleId;
12     }
13
14     public void setVehicleId(Long vehicleId) {
15         this.vehicleId = vehicleId;
16     }
17
18     public String getVehicleName() {
19         return vehicleName;
20     }
21
22     public void setVehicleName(String vehicleName) {
23         this.vehicleName = vehicleName;
24     }
25 }

```

### 5.3 Načítání struktury vlaku do dispečerského systému

Pro správné dotazování se na polohu vozů je třeba nejprve získat strukturu složení vlaku. Jedná se o aktuální zapojení vagonů. Načítat se bude identifikátor vagonu (jedinčná hodnota typu String). Takto identifikované vozy pak jsou přiřazeny jednotlivým spojům a uloženy do databáze. Podle nich lze přesně získat data o poloze.

#### *Tvorba Provideru (inteface, implementace)*

## Kod. 5.14 TrafficControlProvider.java

```

1 package cz.sa.ybus.server.infrastructure.provider.trafficcontrol;
2
3 import java.util.function.Consumer;
4
5 import cz.sa.ybus.core.common.annotations.NotNull;
6 import cz.sa.ybus.core.trafficcontrol.service.dos.TrainCompositionDO;
7
8 public interface TrafficControlProvider {
9
10     void updateTrainCompositions(@NotNull final Consumer<TrainCompositionDO> trainCompositionUpdater);
11
12 }

```

## Kod. 5.15 TrafficControlProviderImpl.java

```
1 package cz.sa.ybus.server.infrastructure.provider.trafficcontrol.impl;
2
3 import java.io.IOException;
4 import java.io.StringReader;
5 import java.io.UnsupportedEncodingException;
6 import java.util.ArrayList;
7 import java.util.Collections;
8 import java.util.Comparator;
9 import java.util.List;
10 import java.util.StringTokenizer;
11 import java.util.function.Consumer;
12 import java.util.regex.Pattern;
13
14 import org.apache.commons.csv.CSVFormat;
15 import org.apache.commons.csv.CSVParser;
16 import org.apache.commons.csv.CSVRecord;
17 import org.joda.time.LocalDate;
18 import org.joda.time.format.DateTimeFormat;
19 import org.joda.time.format.DateTimeFormatter;
20 import org.slf4j.Logger;
21 import org.slf4j.LoggerFactory;
22 import org.springframework.beans.factory.annotation.Autowired;
23 import org.springframework.stereotype.Component;
24
25 import cz.sa.ybus.core.common.annotations.NotNull;
26 import cz.sa.ybus.core.trafficcontrol.service.dos.TrainCompositionDO;
27 import cz.sa.ybus.core.trafficcontrol.service.dos.WagonDO;
28 import cz.sa.ybus.server.infrastructure.provider.trafficcontrol.ExcelToCsvConverter;
29 import cz.sa.ybus.server.infrastructure.provider.trafficcontrol.TrafficControlMailDownloader;
30 import cz.sa.ybus.server.infrastructure.provider.trafficcontrol.TrafficControlProvider;
31
32 /**
33  *
34  * @author dalibor.dobes
35  *
36  */
37 @Component
38 public class TrafficControlProviderImpl implements TrafficControlProvider {
39
40     @Autowired
41     private TrafficControlMailDownloader mailDownloader;
42     @Autowired
43     private ExcelToCsvConverter excelToCsvConverter;
44
45     public static final char WAGON_SEPARATOR = ',';
46     public static final char DEFAULT_SEPARATOR = ';';
47
48     private static final Pattern TRAIN_CODE_PATTERN = Pattern.compile("[0-9]+");
49
50     public static final DateTimeFormatter DATE_TIME_FORMATTER = DateTimeFormat.forPattern("yyyy-MM-dd");
51
52     private final Logger log = LoggerFactory.getLogger(getClass());
53
54     @Override
55     @NotNull
56     public void updateTrainCompositions(@NotNull final Consumer<TrainCompositionDO> trainCompositionUpdater) {
57
58         final byte[] attachmentBytes = mailDownloader.downloadEmailAttachments();
59         if (attachmentBytes != null) {
60             final String csvString = excelToCsvConverter.convertTrainStructure(attachmentBytes);
61             if (csvString != null) {
62                 try (final CSVParser csvParser = new CSVParser(new StringReader(csvString), CSVFormat.DEFAULT.withDelimiter(DEFAULT_SEPARATOR));
63                    ) {
64                     for (CSVRecord csvRecord : csvParser.getRecords()) {
65                         final String csvDate = csvRecord.get(0);
66                         final String csvTrainCode = csvRecord.get(1);
67                         final String csvTrainStructure = csvRecord.get(2);
68
69                         final TrainCompositionDO tcDO = new TrainCompositionDO();
70                         tcDO.setLocalDate(LocalDate.parse(csvDate, DATE_TIME_FORMATTER));
71                         if (!TRAIN_CODE_PATTERN.matcher(csvTrainCode).matches()) {
72                             log.error("The train number must not contain letters");
73                             continue;
74                         }
75                         final Integer trainCode = Integer.parseInt(csvTrainCode);
76                         tcDO.setTrainName(trainCode);
77                         tcDO.setWagons(getTrainStructure(csvTrainStructure, trainCode));
78                         trainCompositionUpdater.accept(tcDO);
79                     }
80                 }
81                 catch (final UnsupportedEncodingException e) {
82                     log.error("UnsupportedEncoding for CSV file");
83                 }
84                 catch (final IOException e) {
85                     log.error("Error while parsing CSV", e);
86                 }
87             }
88         }
89     }
90
91     @NotNull
92     private List<WagonDO> getTrainStructure(@NotNull final String csvTrainStructure, @NotNull final Integer trainName) {
93
94         final List<WagonDO> wagons = new ArrayList<>();
95         final StringTokenizer stringTokenizer = new StringTokenizer(csvTrainStructure, String.valueOf(WAGON_SEPARATOR));
96         int position = 1;
97         while (stringTokenizer.hasMoreElements()) {
98             final String wagonId = stringTokenizer.nextToken();
99             if (!wagonId.isEmpty()) {
100                 if (!Character.isDigit(wagonId.charAt(0))) {
101                     final WagonDO wDO = new WagonDO();
102                     wDO.setPositionInTrain(position);
103                     wDO.setWagonId(wagonId);
104                     wagons.add(wDO);
105                     position++;
106                 }
107             }
108         }
109         if (trainName % 2 != 0) {
110             int countOfWagons = wagons.size();
111             final int controlValue = wagons.size() + 1;
112             Collections.sort(wagons, Comparator.comparingInt(WagonDO::getPositionInTrain));
113             for (final WagonDO wagonDO : wagons) {
114                 if (wagonDO.getPositionInTrain() + countOfWagons == controlValue) {
115                     wagonDO.setPositionInTrain(countOfWagons);
116                     countOfWagons--;
117                 }
118             }
119         }
120         return wagons;
121     }
122 }
```



```

50 @Nullable
51 public byte[] downloadEmailAttachments() {
52
53     /* přihlasovací údaje */
54     final String userName = config.getProperty(ConfigKeys.TRAFFIC_CONTROL_TRAIN_USERNAME);
55     final String password = config.getProperty(ConfigKeys.TRAFFIC_CONTROL_TRAIN_PASSWORD);
56
57     /* nastavení properties pro session */
58     final Properties properties = getConnectionProperties();
59     final Session session = Session.getInstance(properties);
60
61     byte[] attachmentBytes = null;
62
63     try {
64         final Store store = session.getStore(PROTOCOL);
65
66         /* vytvoření spojení */
67         store.connect(HOST, userName, password);
68
69         /* složka s e-mailly */
70         final Folder folderInbox = store.getFolder("INBOX");
71         folderInbox.open(Folder.READ_WRITE);
72
73         /* vráti jen nepřečtené zprávy */
74         final Message messages[] = folderInbox.search(new FlagTerm(new Flags(Flag.SEEN), false));
75         if (messages.length == 0) {
76             log.info("There are no new emails on address ybus.dispecink@studentagency.cz");
77             return null;
78         }
79         boolean manyNewMails = messages.length > 1 ? true : false;
80         /* pokud máme více nových zpráv, tak chceme tu s nejnovějším datem. Cislování je od nejstarší po nejnovější, proto potřebujeme obrátit
            radu zpráv
            * aby nám první index (0) ukazoval na nejnovější */
81         if (manyNewMails) {
82             ArrayUtils.reverse(messages);
83         }
84         boolean foundEmailWithExcelAttachment = false;
85
86         for (int i = 0; i < messages.length; i++) {
87             /* chceme poslední přijatou zprávu */
88             final Message message = messages[i];
89             message.setFlag(Flag.SEEN, true);
90             /* po nalezení a přečtení nejnovější zprávy potřebujeme ostatním nastavit že jsou přečtené (pokud existují) */
91             if (foundEmailWithExcelAttachment) {
92                 continue;
93             }
94             final String contentType = message.getContentType();
95             /* mail musí obsahovat přílohy (indikuje se podle "multipart") */
96             if (contentType.contains("multipart")) {
97
98                 /* po přečtení mailu dojde k automatickému nastavení na Flag.SEEN "přečteno" ("message.getContentType()") */
99                 final Multipart multiPart = (Multipart) message.getContent();
100                 final int numberOfParts = multiPart.getCount();
101                 for (int partCount = 0; partCount < numberOfParts; partCount++) {
102                     final Part part = multiPart.getBodyPart(partCount);
103                     final String disposition = part.getDisposition();
104
105                     /* otevru pouze část mimeTypeu ve kterém je uložena příloha */
106                     if ((disposition != null)
107                         && ((disposition.equalsIgnoreCase(Part.ATTACHMENT) || (disposition.equalsIgnoreCase(Part.INLINE)))))) {
108                         final MimeBodyPart mimeBodyPart = (MimeBodyPart) part;
109                         final String fileName = MimeUtility.decodeText(mimeBodyPart.getFileName());
110                         /* může se stát že nám na mail dojde zpráva s nějakou přílohou ale mi potřebujeme pouze email s přílohou obsahující excel */
111                         if (EXCEL_SUFFIX.matcher(fileName).find()) {
112                             try (final InputStream is = mimeBodyPart.getInputStream()) {
113                                 attachmentBytes = IOUtils.toByteArray(is);
114                                 foundEmailWithExcelAttachment = true;
115                             }
116                         }
117                     }
118                 }
119             }
120         }
121     } catch (MessagingException | IOException ex) {
122         log.error("Error while accessing e-mail store or reading e-mail.", ex);
123     }
124     folderInbox.close(false);
125     store.close();
126 }
127
128 private Properties getConnectionProperties() {
129     final Properties properties = new Properties();
130     properties.setProperty(String.format("mail.%s.host", PROTOCOL), HOST);
131     properties.setProperty(String.format("mail.%s.port", PROTOCOL), PORT);
132     properties.setProperty(String.format("mail.store.protocol", PROTOCOL), PROTOCOL);
133     properties.setProperty(String.format("mail.%s.auth", PROTOCOL), "true");
134     properties.setProperty(String.format("mail.%s.socketFactory.class", PROTOCOL), "javax.net.ssl.SSLSocketFactory");
135     properties.setProperty(String.format("mail.%s.socketFactory.fallback", PROTOCOL), "false");
136     properties.setProperty(String.format("mail.%s.socketFactory.port", PROTOCOL), PORT);
137
138     return properties;
139 }
140
141 }

```

Další oblastí zájmu je načtení přijatých a odfiltrování přebytečných mailů, které jsou buď staré nebo neobsahují potřebnou přílohu. Dále aplikační logika nezbytná k otevření přílohy a načtení XLS souboru do streamu pro následné zpracování. Poslední částí zpracování je konverze pracovních dat (vybraných buněk) do formátu CSV.

### Kod. 5.19 ExcelToCsvConverter.java

```

1 package cz.sa.ybus.server.infrastructure.provider.trafficcontrol;
2
3 import java.io.ByteArrayInputStream;
4 import java.io.IOException;
5 import java.io.InputStream;
6 import java.util.Date;
7
8 import javax.annotation.Nullable;
9
10 import org.apache.commons.lang3.text.StrBuilder;
11 import org.apache.poi.ss.usermodel.DateUtil;
12 import org.apache.poi.xssf.usermodel.XSSFCell;
13 import org.apache.poi.xssf.usermodel.XSSFRow;
14 import org.apache.poi.xssf.usermodel.XSSFSheet;
15 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
16 import org.joda.time.LocalDate;

```

```

17 import org.slf4j.Logger;
18 import org.slf4j.LoggerFactory;
19 import org.springframework.stereotype.Component;
20
21 import cz.sa.ybus.core.common.annotations.NotNull;
22 import cz.sa.ybus.server.infrastructure.provider.trafficcontrol.impl.TrafficControlProviderImpl;
23
24 /**
25  * Trida napsana namiru nacisti struktury vlaku z excelu, po vytvoreni dispecerskeho systemu bude jiz nepotreba a prijde odstranit
26  * @author dalibor.dobes
27  *
28  */
29 @Component
30 public class ExcelToCsvConverter {
31
32     /* nazev listu v excelu */
33     private static final String XLS_TRAIN_STRUCTURE_SHEET_NAME = "Struktura";
34
35     /* list Struktura */
36     private static final int XLS_STRUCTURE_FIRST_ROW_NUM = 1;
37     private static final int XLS_STRUCTURE_FIRST_CELL_NUM = 0;
38     private static final int XLS_STRUCTURE_FIRST_WAGON_CELL_NUM = 2;
39     private static final int XLS_STRUCTURE_LAST_WAGON_CELL_NUM = 16;
40
41     /* oddelovace */
42     private static final String CSV_ROW_DELIMITER = "\n";
43     private static final String STRING_WRAPPER = "\"";
44
45     private final Logger log = LoggerFactory.getLogger(getClass());
46
47     @Nullable
48     public String convertTrainStructure(@NotNull final byte[] attachmentBytes) {
49
50         String csvString = null;
51
52         try {
53
54             final StrBuilder csvBuilder = new StrBuilder();
55
56             /* nacist excel */
57             try (final InputStream inputXls = new ByteArrayInputStream(attachmentBytes)) {
58
59                 /* otevrit dany list v excelu */
60                 final XSSFSheet xlsSheet = new XSSFWorkbook(inputXls).getSheet(XLS_TRAIN_STRUCTURE_SHEET_NAME);
61
62                 XSSFRow xlsRow = null;
63
64                 for (int row = XLS_STRUCTURE_FIRST_ROW_NUM; row < xlsSheet.getLastRowNum(); row++) {
65                     xlsRow = xlsSheet.getRow(row);
66                     final StrBuilder csvRowBuilder = new StrBuilder();
67
68                     XSSFCell xlsCell = null;
69
70                     /* bunka na pozici "0" je vzdy datum */
71                     xlsCell = xlsRow.getCell(XLS_STRUCTURE_FIRST_CELL_NUM);
72                     if (DateUtil.isCellDateFormatted(xlsCell)) {
73                         final Date cDate = xlsCell.getDateCellValue();
74                         if (cDate == null) {
75                             continue;
76                         }
77                         final LocalDate lDate = new LocalDate(cDate);
78                         final String cellDate = lDate.toString(TrafficControlProviderImpl.DATE_TIME_FORMATTER);
79                         csvRowBuilder.append(getWrappedString(cellDate));
80                         csvRowBuilder.append(TrafficControlProviderImpl.DEFAULT_SEPARATOR);
81                     }
82
83                     /* bunka na pozici "1" je nazev vlaku */
84                     xlsCell = xlsRow.getCell(XLS_STRUCTURE_FIRST_CELL_NUM + 1);
85                     final String cellTrain = xlsCell.getRowValue();
86                     if (cellTrain == null) {
87                         continue;
88                     }
89                     csvRowBuilder.append(getWrappedString(cellTrain));
90                     csvRowBuilder.append(TrafficControlProviderImpl.DEFAULT_SEPARATOR);
91                     csvRowBuilder.append(STRING_WRAPPER);
92
93                     /* nacisti kodu vagonu do jednoho stringu v CSV */
94                     for (int cellWagons = XLS_STRUCTURE_FIRST_WAGON_CELL_NUM; cellWagons < XLS_STRUCTURE_LAST_WAGON_CELL_NUM + 1; cellWagons++) {
95                         xlsCell = xlsRow.getCell(cellWagons);
96                         final String wagonCellValue = xlsCell.getRowValue();
97                         csvRowBuilder.append(wagonCellValue);
98                         csvRowBuilder.append(TrafficControlProviderImpl.WAGON_SEPARATOR);
99                     }
100                     csvRowBuilder.append(STRING_WRAPPER);
101                     csvRowBuilder.append(CSV_ROW_DELIMITER);
102                     csvBuilder.append(csvRowBuilder);
103                 }
104             }
105             csvString = csvBuilder.toString();
106         } catch (IOException e) {
107             log.error("Cannot convert Excel file to CSV");
108         }
109         return csvString;
110     }
111
112     private String getWrappedString(@NotNull final String string) {
113         return STRING_WRAPPER + string + STRING_WRAPPER;
114     }
115 }
116

```

*Tvorba service* již není tak obsahově svázaná. Striktní definici podléhá jen interface. Jeho implementace je již plně modifikovatelná. Lze využít např. funkcionálního interface z Java8 [1]. Přístup k DB je odstíněný pomocí DAO. Lze plně využít stávajících funkcí pro správu dat. Veškeré DB operace jsou zprostředkovány pomocí Hibernate.

#### Kod. 5.20 TrafficControlService.java

```

1 package cz.sa.ybus.core.trafficcontrol.service;
2
3 public interface TrafficControlService {
4
5     void updateTrainStructure();
6
7 }

```

## Kod. 5.21 TrafficControlServiceImpl.java

```

1 package cz.sa.ybus.core.trafficcontrol.service;
2
3 import java.util.List;
4 import java.util.stream.Collectors;
5
6 import org.joda.time.DateTime;
7 import org.joda.time.LocalDate;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.jmx.export.annotation.ManagedOperation;
12 import org.springframework.jmx.export.annotation.ManagedResource;
13 import org.springframework.stereotype.Service;
14
15 import com.sun.istack.Nullable;
16
17 import cz.sa.ybus.core.busconnection.service.BusConnectionVehicleService;
18 import cz.sa.ybus.core.busconnection.service.dos.BusID0;
19 import cz.sa.ybus.core.busconnection.service.dos.BusTypeID0;
20 import cz.sa.ybus.core.common.annotations.NotNull;
21 import cz.sa.ybus.core.common.service.AbstractService;
22 import cz.sa.ybus.core.trafficcontrol.service.dos.TrainCompositionD0;
23 import cz.sa.ybus.core.trafficcontrol.service.dos.WagonD0;
24 import cz.sa.ybus.domain.bus.BCVehicleID;
25 import cz.sa.ybus.domain.bus.BusConnectionID;
26 import cz.sa.ybus.domain.bus.BusID;
27 import cz.sa.ybus.domain.bus.BusTypeID;
28 import cz.sa.ybus.domain.bus.VehicleCategory;
29 import cz.sa.ybus.server.domain.busconnection.BusConnection;
30 import cz.sa.ybus.server.domain.busconnection.BusConnectionVehicle;
31 import cz.sa.ybus.server.domain.busconnection.BusConnectionsCache;
32 import cz.sa.ybus.server.domain.busdao.BusDao;
33 import cz.sa.ybus.server.domain.busdao.BusService;
34 import cz.sa.ybus.server.infrastructure.provider.trafficcontrol.TrafficControlProvider;
35 import cz.sa.ybus.server.logging.LogBefore;
36
37 /**
38  *
39  * @author dalibor.dobes
40  *
41  */
42 @ManagedResource
43 @Service
44 public class TrafficControlServiceImpl extends AbstractService implements TrafficControlService {
45
46     @Autowired
47     private TrafficControlProvider trafficControlProvider;
48     @Autowired
49     private BusDao busDao;
50     @Autowired
51     private BusService busService;
52     @Autowired
53     private BusConnectionsCache busConnectionsCache;
54     @Autowired
55     private BusConnectionVehicleService busConnectionVehicleService;
56
57     private final Logger log = LoggerFactory.getLogger(getClass());
58
59     private static final String RJ = "RJ";
60
61     @ManagedOperation
62     @Override
63     @LogBefore
64     public void updateTrainStructure() {
65
66         final DateTime now = DateTime.now();
67         final DateTime todayMidnight = now.withTimeAtStartOfDay();
68         final DateTime midnightLast = todayMidnight.plusDays(2).withTimeAtStartOfDay();
69         final VehicleCategory vehicleCategory = VehicleCategory.RAIL_CAR;
70
71         final List<Long> busConnectionV0s = busDao.getBusConnectionsByDepartureTimeAndVehicleType(todayMidnight, midnightLast,
72             vehicleCategory);
73
74         final List<BusConnection> busConnections = busConnectionsCache.getBusConnections(busConnectionV0s);
75         trafficControlProvider.updateTrainCompositions(tcD0 -> updateTrainVehicles(tcD0, busConnections));
76     }
77
78     private void updateTrainVehicles(@NotNull final TrainCompositionD0 trainCompositionD0, @NotNull final List<BusConnection> busConnections) {
79         for (BusConnection busConnection : busConnections) {
80             final BusConnectionID busConnectionID = busConnection.getID();
81             final LocalDate dateD0 = busConnection.getDeparture().toLocalDate();
82             final String trainCodeD0 = busConnection.getConnectionCode();
83             final List<BusConnectionVehicle> busConnectionVehicles = busConnection.getVehicles();
84             final List<WagonD0> wagonD0s = trainCompositionD0.getWagons();
85             if (wagonD0s.size() != busConnectionVehicles.size()) {
86                 log.warn("The number of wagons does not match: " + wagonD0s.size() + " is not the same " + busConnectionVehicles.size()
87                     + " from database");
88             }
89             final LocalDate dateCSV = trainCompositionD0.getLocalDate();
90             final String trainCodeCSV = RJ + " " + trainCompositionD0.getTrainName();
91             if ((dateD0.equals(dateCSV)) && (trainCodeD0.equals(trainCodeCSV))) {
92                 for (final BusConnectionVehicle busConnectionVehicle : busConnectionVehicles) {
93                     final BCVehicleID bcVehicleID = busConnectionVehicle.getID();
94                     final Integer vehiclePosition = busConnectionVehicle.getPosition();
95                     final BusTypeID0 busType = busConnectionVehicleService
96                         .getBusTypeID0(BusTypeID.forId(busConnectionVehicle.getBusTypeID()));
97                     for (final WagonD0 wagonD0 : wagonD0s) {
98                         final Integer wagonPositionCSV = wagonD0.getPositionInTrain();
99                         final String wagonId = wagonD0.getWagonId();
100                         if (vehiclePosition.intValue() == wagonPositionCSV.intValue()) {
101                             updateBusConnectionBusVehicle(busConnectionID, bcVehicleID, busType, wagonId);
102                         }
103                     }
104                 }
105             }
106         }
107     }
108
109     private void updateBusConnectionBusVehicle(
110         @NotNull final BusConnectionID busConnectionID,
111         @NotNull final BCVehicleID bcVehicleID,
112         @NotNull final BusTypeID0 busType,
113         @NotNull final String wagonId
114     ) {
115         /* bude potreba prvne uložit bus/wagon pokud jeste neexistuje */
116         final List<BusID0> allWagons = getAllWagons();
117
118         if (!allWagons.stream().anyMatch(bus -> bus.getSpz().equals(wagonId)) {
119             final Integer maxNumber = allWagons.stream().map(bus -> bus.getNumber()).max(Integer::compareTo)
120                 .orElse(BusService.INITIAL_WAGON_NUMBER - 1) + 1;
121             final BusID0 wagon = new BusID0();
122             wagon.setNumber(maxNumber);
123             wagon.setSpz(wagonId);
124             wagon.setBusType(busType);
125             wagon.setActive(true);
126
127             busService.saveBus(wagon);
128         }
129         changeBus(busConnectionID, bcVehicleID, wagonId);
130     }
131
132     @Nullable
133     private List<BusID0> getAllWagons() {
134         final List<BusID0> allBuses = busService.getAllBuses();
135         return allBuses.stream().filter(bus -> bus.getBusType().getVehicleCategory() == VehicleCategory.RAIL_CAR).collect(Collectors.toList());

```

```
136 }
137
138 private void changeBus(
139     @NotNull final BusConnectionID busConnectionID,
140     @NotNull final BCVehicleID bcVehicleID,
141     @NotNull final String wagonId
142 ) {
143     final List<BusID0> allWagons = getAllWagons();
144     final BusID0 wagon = allWagons.stream().filter(bus -> bus.getSpz().equals(wagonId)).findAny().get();
145     final BusID busID = BusID.forId(wagon.getId());
146     busService.changeBus(busConnectionID, bcVehicleID, busID);
147 }
148 }
```

*Tvorba CRON jobu* pro pravidelnou aktualizaci

### Kod. 5.22 applicationContext-cron-struktura.xml

```
1 <bean class="org.springframework.scheduling.quartz.CronTriggerFactoryBean">
2   <property name="jobDetail">
3     <bean class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
4       <property name="targetObject" ref="trafficControlServiceImpl" />
5       <property name="targetMethod" value="updateTrainStructure" />
6       <property name="concurrent" value="false" />
7     </bean>
8   </property>
9   <property name="cronExpression" value="${trafficControl.updateTrainStructure}" />
10 </bean>
```

---

## 6 Distribuce zpoždění

Vypočítané zpoždění spojů je vystaveno pomocí webové služby (REST) a je konzumováno cílovými platformami. Základní platforma jsou tzv. Plazmy (současné digitální informační tabule na provozních místech Student Agency). Dále je zpoždění konzumováno společností Google v rámci služby Google Transit (pro realizaci je použito napojení na GTFS-real time). Zákazníkům, kteří mají povolené v osobním nastavení notifikace (SMS nebo email) chodí průběžné informace o aktuálních změnách zpoždění spojů, na které mají zakoupený lístek.

---

**ZÁVĚR**

Cíle této diplomové práce, rozšířit dispečinkový systém, bylo dosaženo. Některé části této práce se podařilo nejen realizovat, ale také aplikovat do produkčního provozu výslednou implementací. Hlavní přínos je vystavení webových služeb pro poskytování dat o zpoždění třetím stranám. Dále nově systém sám konzumuje řadu výstupů systémů třetích stran (SŽDC, Polohavozu.cz).

Jiné části zůstaly ve fázi návrhu. Jedná se především o vizualizaci spoje na trase. Tato funkcionality nedostala dostatečnou součinnost, a to především proto, že podobných vizualizačních SW nástrojů je velké množství a jejich funkcionality je dostatečná.

Výsledkem práce je tedy rozšíření modulu zpoždění o nové zdroje dat a řádové zpřesnění aktuálního zpoždění jednotlivých spojů. K dalším rozšířením patří načítání struktury vlakové soupravy a určení polohy jednotlivých vozů pomocí GPS. Všechna nová data jsou zpracována a dále distribuována.

Řešení popsané v této diplomové práci nabízí řadu možností na rozšíření či úpravy.

---

SEZNAM POUŽITÉ LITERATURY

- [1] URMA, Raoul-Gabriel, Mario FUSCO a Alan MYCROFT. Java 8 in action: lambdas, streams, and functional-style programming. Shelter Island: Manning, 2015. ISBN 1617291994.
- [2] WALLS, Craig. Spring in action. Fourth Edition. Shelter Island, NY: Manning, 2015. ISBN 161729120x.
- [3] Maven: the definitive guide. Sebastopol: O'Reilly, c2008. ISBN 0596517335.
- [4] OBE, Regina O. a Leo S. HSU. PostGIS in action. Second edition. Shelter Island, NY: Manning, 2015. ISBN 1617291390.
- [5] PRICE, Maribeth Hughett. Mastering ArcGIS. Seventh edition. New York, NY: McGraw-Hill Education, 2016. ISBN 978-0078095146.
- [6] FIELDING, Roy Thomas. Architectural styles and the design of network-based software architectures. Irvine, 2000. ISBN 0-599-87118-0. Dostupné také z: [https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf). Doctoral Dissertation. University of California. Vedoucí práce Richard N. Taylor.
- [7] SOAP. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: <https://en.wikipedia.org/wiki/SOAP>.
- [8] *Spring Framework* [online]. Pivotal Software, 2018 [cit. 2018-05-18]. Dostupné z: <https://projects.spring.io/spring-framework/>.
- [9] Java EE. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://cs.wikipedia.org/wiki/Java\\_EE](https://cs.wikipedia.org/wiki/Java_EE).
- [10] Inversion of control. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Inversion\\_of\\_control](https://en.wikipedia.org/wiki/Inversion_of_control).
- [11] Dependency injection. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Dependency\\_injection](https://en.wikipedia.org/wiki/Dependency_injection).
- [12] JavaBeans. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: <https://cs.wikipedia.org/wiki/JavaBeans>.
- [13] XML. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: <https://en.wikipedia.org/wiki/XML>.

- 
- [14] Open-source software. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Open-source\\_software](https://en.wikipedia.org/wiki/Open-source_software).
- [15] Hibernate (framework). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Hibernate\\_\(framework\)](https://en.wikipedia.org/wiki/Hibernate_(framework)).
- [16] Java (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)).
- [17] Object-relational mapping. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Object-relational\\_mapping](https://en.wikipedia.org/wiki/Object-relational_mapping).
- [18] Enterprise JavaBeans. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Enterprise\\_JavaBeans](https://en.wikipedia.org/wiki/Enterprise_JavaBeans).
- [19] Plain old Java object. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Plain\\_old\\_Java\\_object](https://en.wikipedia.org/wiki/Plain_old_Java_object).
- [20] SQL. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: <https://en.wikipedia.org/wiki/SQL>.
- [21] Apache Maven. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-18]. Dostupné z: [https://en.wikipedia.org/wiki/Apache\\_Maven](https://en.wikipedia.org/wiki/Apache_Maven).
- [22] Hypertext Transfer Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol).
- [23] Web Services Description Language. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Web\\_Services\\_Description\\_Language](https://en.wikipedia.org/wiki/Web_Services_Description_Language).
- [24] Vývojové prostředí. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9\\_prost%C5%99ed%C3%AD](https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD).
- [25] *The Most Advanced REST & SOAP Testing Tool in the World* [online]. SmartBear Software, 2018 [cit. 2018-05-19]. Dostupné z: <https://www.soapui.org/>.

- [26] Service Oriented Architecture. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://cs.wikipedia.org/wiki/Service\\_Oriented\\_Architecture](https://cs.wikipedia.org/wiki/Service_Oriented_Architecture).
- [27] Action Message Format. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Action\\_Message\\_Format](https://en.wikipedia.org/wiki/Action_Message_Format).
- [28] Java Message Service. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Java\\_Message\\_Service](https://en.wikipedia.org/wiki/Java_Message_Service).
- [29] Java Database Connectivity. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://cs.wikipedia.org/wiki/Java\\_Database\\_Connectivity](https://cs.wikipedia.org/wiki/Java_Database_Connectivity).
- [30] SoapUI Testing: Overview of SoapUI Testing Tool. *Astegic* [online]. Falls Church, Virginie, U.S.A.: Astegic, 2017, 21.7.2017 [cit. 2018-05-19]. Dostupné z: <https://www.astegic.com/overview-soapui-testing-tool/>.
- [31] Representational state transfer. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer).
- [32] Hypertext Transfer Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol).
- [33] FRUZENSCHTEIN, Alex. *Hibernate for beginners. Introduction tutorial* [online]. Fuzenshtein's Notes, 2013, 2013-01-13 [cit. 2018-05-19]. Dostupné z: <http://fuzenshtein.com/hibernate-for-beginners/>.
- [34] Maven: Maven Central Repository. *Maven.apache.org* [online]. Wakefield, Massachusetts, U.S.A.: The Apache Software Foundation, 2018 [cit. 2018-05-19]. Dostupné z: <https://maven.apache.org/repository/>.
- [35] In:JSON. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: <https://en.wikipedia.org/wiki/JSON>.
- [36] *Postman* [online]. San Francisco, California, U.S.A.: Postdot Technologies, 2018 [cit. 2018-05-19]. Dostupné z: <https://www.getpostman.com/>
- [37] JavaMail. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: <https://en.wikipedia.org/wiki/JavaMail>.

- 
- [38] Comma-separated values. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: [https://en.wikipedia.org/wiki/Comma-separated\\_values](https://en.wikipedia.org/wiki/Comma-separated_values).
- [39] PostGIS. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-05-19]. Dostupné z: <https://en.wikipedia.org/wiki/PostGIS>.
- [40] Portál provozní dráhy: Náповěda-GRAPP. *SŽDC* [online]. Olomouc: OLTIS Group, 2018, 22.2.2018 [cit. 2018-05-20]. Dostupné z: <http://provoz.szdc.cz/Portal/ViewArticle.aspx?oid=1402364>.

---

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

AMF	Action Message Format
API	Application Programming Interface
CSV	Comma Separated Values
EJB	Enterprise JavaBeans
GIS	Geografický Informační Systém
GPS	Global Positioning System
GRAPP	GRAfická Prezentace Polohy
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HQL	Hibernate Query Language
IDE	Integrated Development Environment
IDEA	IntelliJ IDEA
IMAP	Internet Message Access Protocol
IoC	Inversion of Control
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
Java EE	Java Platform, Enterprise Edition
Java SE	Java Platform, Standard Edition
JMS	Java Message Service
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
POJO	Plain Old Java Object
POM	Project Object Model
POP3	Post Office Protocol
REST	REpresentational State Transfer
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SŽDC	Správa Železniční Dopravní Cesty
WSDL	Web Services Description Language
XML	eXtensible Markup Language

---

SEZNAM OBRÁZKŮ

Obr. 1.1	Náhled možností dependency injection . . . . .	11
Obr. 1.2	Příklad integrace Hibernate do projektu [33] . . . . .	12
Obr. 1.3	Příklad integrace Maven repository [34] . . . . .	13
Obr. 1.4	Podporované protokoly SOAP-UI [30] . . . . .	15
Obr. 1.5	Schéma užití Postman [36] . . . . .	16
Obr. 3.1	Vkládání zpoždění v administrátorském systému . . . . .	20
Obr. 3.2	Audit zpoždění . . . . .	20
Obr. 3.3	Pohled L1 na současnou HW architekturu . . . . .	21
Obr. 3.4	Funkční požadavky . . . . .	21
Obr. 3.5	Nefunkční požadavky . . . . .	22
Obr. 3.6	Případy užití . . . . .	22
Obr. 4.1	Náhled spoje trasovaného pomocí KML na mapy.cz . . . . .	23
Obr. 5.1	GRAPP [40] . . . . .	26
Obr. 5.2	Vizualizace WSDL pro SŽDC (getTrain position) . . . . .	27

## SEZNAM ZDROJOVÉHO KÓDU

1.1	pomExample.xml . . . . .	13
5.1	uprava-pom-szdc-position.xml . . . . .	27
5.2	SzdcProvider.java . . . . .	27
5.3	SzdcPositionWSApiService.java . . . . .	28
5.4	SzdcProviderImpl.java . . . . .	28
5.5	SzdcDao.java . . . . .	30
5.6	SzdcService.java . . . . .	31
5.7	SzdcServiceImpl.java . . . . .	31
5.8	applicationContex-cron-position.xml . . . . .	33
5.9	VehiclePositionProvider.java . . . . .	34
5.10	VehiclePositionProviderImpl.java . . . . .	34
5.11	PositionDO.java . . . . .	35
5.12	TrackedVehiclePathDO.java . . . . .	36
5.13	VehicleInfoDO.java . . . . .	36
5.14	TrafficControlProvider.java . . . . .	36
5.15	TrafficControlProviderImpl.java . . . . .	37
5.16	TrainCompositionDO.java . . . . .	38
5.17	WagonDO.java . . . . .	38
5.18	TrafficControlMailDownloader.java . . . . .	38
5.19	ExcelToCsvConverter.java . . . . .	39
5.20	TrafficControlService.java . . . . .	40
5.21	TrafficControlServiceImpl.java . . . . .	41
5.22	applicationContext-cron-struktura.xml . . . . .	42

---

SEZNAM PŘÍLOH

P I. CD s průvodními materiály

## **PŘÍLOHA P I. CD S PRŮVODNÍMI MATERIÁLY**

Na CD naleznete

- kompletní zdrojové kódy
- pomocné projekty (např. analýza v Enterprise Architect)