

# **Rozšíření dispečinkového systému o modul zpoždění**

Bc. Dalibor Dobeš



\*\*\* Nascanované zadání, strana 1 \*\*\*

\*\*\* Nascanované zadání, strana 2 \*\*\*

## **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

.....

podpis autora

## **ABSTRAKT**

Cílem diplomové práce je analyzovat, doplnit a rozšířit stávající dispečinkový systém o nové zdroje dat zpoždění a polohy spojů. To bude zahrnovat připojení na SOAP [1] API společnosti SŽDC odkud budem načítat aktuální zpoždění a další doplňkové informace. Pro účeli návrhu výpočtu zpoždění a polohy vozů se bude potřeba připojit k RESTovému api Polohavozu.cz . Jako poslední pak bude potřeba načítat strukturu vlakového spoje pomocí JavaMail api, to vše se bude ukládat do databaze stávajícího dispečinkového systému.

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

## **ABSTRACT**

Text of the abstract

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

Život je nepřetržitá řada průserů nepravidelně za sebou jdoucích.

## 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 .....	13
1.4.1    WSDL.....	13
1.4.2    SoapUI.....	13
1.5    REST .....	14
1.5.1    JSON .....	14
1.5.2    Postman.....	15
1.6    JAVAMAIL .....	15
1.7    CSV .....	15
1.8    POSTGIS .....	16
<b>2    DALŠÍ NADPIS .....</b>	<b>17</b>
2.1    PODNADPIS .....	17
2.1.1    Podpodnadpis .....	17
2.1.2    Podpodnadpis .....	17
<b>II   ANALYTICKÁ ČÁST .....</b>	<b>17</b>
<b>3    ANALÝZA STÁVAJÍCÍHO DISPEČINKOVÉHO SYSTÉMU .....</b>	<b>19</b>
3.1    PODNADPIS .....	19
<b>4    NAVRHNĚTE TECHNICKÉ ŘEŠENÍ PRO VÝPOČET ZPOŽDĚNÍ...</b>	<b>20</b>
4.1    PODNADPIS .....	20
<b>III   PROJEKTOVÁ ČÁST .....</b>	<b>20</b>
<b>5    PŘIPOJENÍ NOVÝCH ZDROJŮ DAT .....</b>	<b>22</b>
5.1    PŘIPOJENÍ SŽDC API .....	22
5.1.1    Datové rozhraní GRAPP .....	22
5.1.2    Podnadpis .....	23
5.2    PŘIPOJENÍ POLOHAVOZU.CZ.....	23
5.2.1    Podnadpis .....	23
5.3    NAČÍTÁNÍ STRUKTURY VLAKU DO DISPEČINKOVÉHO SYSTÉMU .....	23
5.3.1    Podnandpis.....	24

<b>6</b>	<b>DISTRIBUCE ZPOŽDĚNÍ.....</b>	<b>25</b>
6.1	ZOBRAZENÍ NA ZPOZDENI SPOJE.....	25
6.1.1	Podnandpis.....	25
6.2	ZOBRAZENÍ ÚDAJŮ O ZPOŽDĚNÍ.....	25
6.2.1	Podnandpis.....	25
	<b>ZÁVĚR.....</b>	<b>26</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>27</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>30</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>31</b>
	<b>SEZNAM TABULEK .....</b>	<b>32</b>
	<b>SEZNAM PŘÍLOH .....</b>	<b>33</b>



## ÚVOD

První odstavec pod nadpisem se neodsazuje, ostatní ano (pouze první řádek, odsazení vertikální mezy odstavci je typické pro anglickou sazbu; czech babel toto respektuje, netřeba do textu přidávat jakékoliv explicitní formátování, viz ukázka sazby tohoto textu s následujícím odstavcem).

Formátování druhého odstavce. Text text text text text text text text text text text.

# I. TEORETICKÁ ČÁST

## 1 Vývojové nástroje, protokoly a aplikace

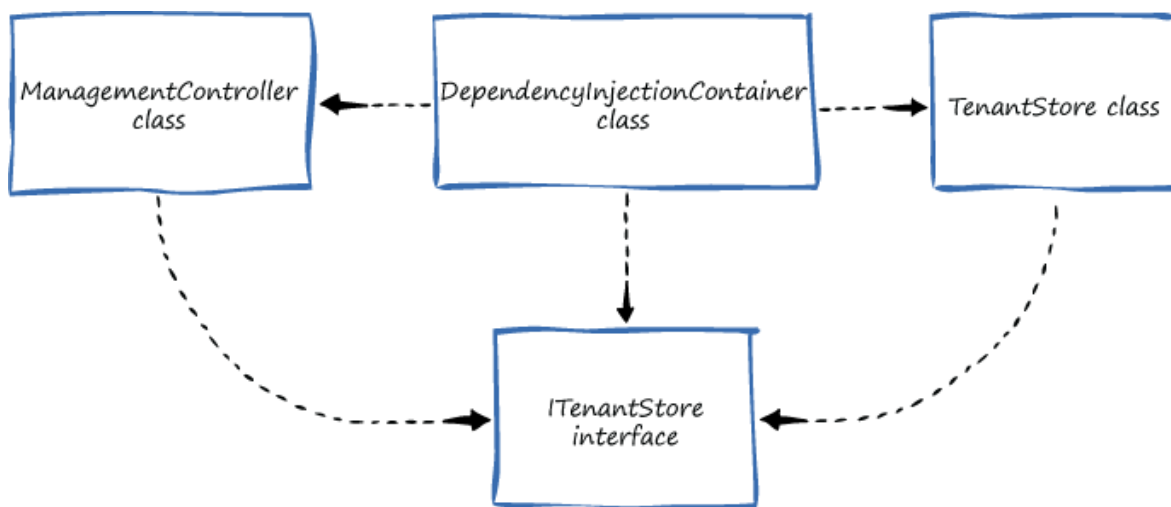
text

### 1.1 Spring

Spring Framework [2] (dále jen spring) je populární open-source [8] pro vývoj J2EE [3] aplikací. Jádru springu je využíván návrhový vzor IoC (Inversion of Control [4]) 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 [5] 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 [6]. Ty jsou vytvářeny typicky na základě načtení konfiguračního souboru ve formátu XML [7], 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 [9] je napsaný v jazyce Java [10], který umožňuje tzv. objektově-relační mapování (ORM [11]). Usnadňuje 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 Javovské 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.

Druhý způsob jak mapovat objekty je použít 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 [12] (viz JavaBean), ale pouze klasické objekty - tzv. POJO (Plain Old Java Object) [13]. Poté, co jsou objekty uloženy v databázi, se na ně lze dotazovat jazykem HQL (Hibernate Query Language), který je odvozen z SQL [14] a je mu tedy velice podobný.

## 1.3 Maven

Apache Maven [15] je nástroj pro správu, řízení a automatizaci buildů aplikací. Ačkoliv je možné použít tento nástroj pro projekty psané v různých programovacích jazycích, podporován je převážně jazyk Java. Základním principem fungování Mavenu je popsání projektu pomocí POM.

Tento model popisuje softwarový projekt nejen z pohledu jeho zdrojového kódu, ale včetně závislostí na externích knihovnách, popisu procesu buildování a různých funkcí s tím spojených (jako je spouštění testů, sbírání informací o zdrojových kódech a podobně). Maven sám je postaven na modulární architektuře a funguje na principu volání jednotlivých pluginů. Maven sám pouze obstarává dodání a spuštění nadefinovaných 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.

**Project Object Model koncept popisu projektu jako objektu.** Za tímto účelem je definovaná jednoduchá XML struktura, která definuje 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 definovat jeho závislosti na externích knihovnách.

Jednotlivé prvky Artifacts jsou jednoznačně definovány podle atributů `<groupId>` a `<artifactId>`. 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.

## 1.4 SOAP

SOAP [1] je protokolem pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP [16]. 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 [17] 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 [19] je open source aplikace pro testování webových služeb pro architektury orientované na služby (SOA [20]) a reprezentace státních přenosů (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 softwarem Eviware. V roce 2011 společnost Smart-Bear Software získala produkt Eviware. SoapUI byl zpočátku propuštěn do společnosti SourceForge v září 2005. Je to svobodný software, licencovaný na základě podmínek

veřejné licence Evropské unie.

Je postavena výhradně na platformě Java a používá rozhraní Swing pro uživatelské rozhraní. To znamená, že SoapUI je multiplatformní. Dnes SoapUI podporují všechny hlavní vývojová prostředí (IDE [18]), jako je např. IDEA, Eclipse nebo NetBeans a spousta dalších. SoapUI může testovat webové služby SOAP a REST, JMS [22], AMF [21], stejně jako volání HTTP(S) a JDBC [23].

- Výhody
  - je jednoduše čitelnější pro člověka.
- Nevýhody
  - velký zápis komunikace,
  - složitost,
  - pomalé zpracování jednotlivými systémy (složitě na parsování a validaci).

## 1.5 REST

REST je architektura rozhraní, navržená pro distribuované prostředí. REST navrhl a popsal v roce 2000 Roy Fielding (jeden ze spoluautorů protokolu http) v rámci disertační práce *Architectural Styles and the Design of Network-based Software Architectures*. 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 známy pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.

### 1.5.1 JSON

JSON je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat, která mohou být organizována v polích nebo agregována 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. Ta 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. Ten 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) and 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 je open source aplikace pro testování webových služeb pro architektury orientované na REST.

## 1.6 JavaMail

Knihovna pro přijímání a odesílání mailů. Javamail je rozhraní API jazyka Java, které slouží k odesílání a přijímání e-mailů prostřednictvím 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 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 tyto zdvojeny.

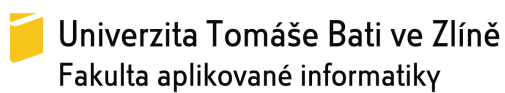
## 1.8 PostGIS

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



## 2 Další nadpis

Tato sekce obsahuje ukázkou vložení obrázku (Obr. 2.1).



Obr. 2.1 Popisek obrázku

### 2.1 Podnadpis

Tato sekce obsahuje ukázkou vložení tabulky (Tab. 2.1).

Tab. 2.1 Popisek tabulky

	1	2	3	4	5	Cena [Kč]
<i>F</i>	(jedna)	(dva)	(tři)	(čtyři)	(pět)	300

#### 2.1.1 Podpodnadpis

#### 2.1.2 Podpodnadpis

Citace knihy. [?]

## II. ANALYTICKÁ ČÁST

### **3 Analýza stávajícího dispečinkového systému**

Stávající systém založený na komunikaci pomocí SMS mezi dispečerem a obsluhou vozu (řidič, steward). SMS se parsují a ukládají do databáze.

#### **3.1 Podnadpis**

text

#### 4 Navrhnete technické řešení pro výpočet zpoždění

- Vytvoríme si základní trasu spoje pomocí GPS souřadnic
- Tuto trasu rozdělíme na trasy mezi zastávkami (aby nebyl příliš velký počet souřadnic)
- Pomocí Trasování spoje (napojení na [polohavozu.cz](http://polohavozu.cz)) získáme souřadnice GPS a čas kdy byli tyto data porizeny
- Vytvoríme modelovou trasu spoje s časem
- Porovnáním modelové trasy spoje a aktuální pozice vozu, můžeme vypočítat zpoždění
- Využijeme PostGIS k vrácení nejbližší souřadnice, u této souřadnice známe i čas průjezdu
- Zpoždění uložíme do databáze

##### 4.1 Podnadpis

text

## III. PROJEKTOVÁ ČÁST

## 5 Připojení nových zdrojů dat

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

### 5.1 Připojení SŽDC api

Prvním z nich je datové rozhraní GRAPP, ze které se bude načítat zpoždění a poslední potvrzené průjezdové místo vozu kontrolním bodem. Druhým je pak datové rozhraní Informační tabule, zde se budou načítat aktuální názvy nástupišť v dané zastávce pro daný spoj. Tyto datová rozhraní spravuje společnost Oltis group a.s. , která jej provozuje pro společnost SŽDC.

#### 5.1.1 Datové rozhraní GRAPP

Jako první upravíme pom.xml soubor do kterého přidáme v sekci <plugin> nové údaje. Pomocí tohoto přidaného kódu pak při opětovném sestavení projektu pomocí mavenu dojde k autovygenerování komunikačních zprávek, které pak využijem pro načítání dat.

```
<execution>
  <id>SzdcPositionWS</id>
  <goals>
    <goal>wsimport</goal>
  </goals>
  <configuration>
    <wsdlDirectory>${project.basedir}/src/main/resources/cz/sa/szdcPosition/api/</wsdlDirectory>
    <wsdlUrls>
      <wsdlUrl>http://provoz.szdc.cz/grappnv/trainposition.asmx?wsdl</wsdlUrl>
    </wsdlUrls>
    <wsdlLocation>SZDCPositionApiWebService.wsdl</wsdlLocation>
    <destDir></destDir>
    <sourceDestDir>${project.build.directory}/generated-sources/webservices</sourceDestDir>
    <packageName>cz.sa.szdcPosition.api</packageName>
    <keep>true</keep>
    <verbose>false</verbose>
    <extension>true</extension>
    <xnocompile>true</xnocompile>
  </configuration>
</execution>
```

Obr. 5.1 Úprava Pom.xml

- Tvorba Provideru (inteface, implementace)
- Vytvoreni wsdl factory trid, abstraktnich trid
- Tvorba dao vrstvy pro nacistani zastvaek a spoju podle typu spojeni a doby odjezdu (hibernate)

- Vytvoreni DO trid, datova struktura odpovedi
- Konvertor na tyto DO třídy
- Tvorba service (inteface, implementace)
- Nacitani zpozdeni jednotlivych spoju
- Nacitani nazvu kolej/nastupiste podle doby prijezdu do zastvaky
- Tvorba cron jobu pro tyto servisy

### 5.1.2 Podnadpis

text

## 5.2 Připojení Polohavozu.cz

- Tvorba Provideru (inteface, implementace)
- Pomoci Postman ziskame JSON fily ze kterych vytvorime POJO objekty
- dva endpointy
  - seznam vozu a jejich identifikace
  - GPS souradnice vozu obsahujici identifikator
- Konverze na DO objekty
- tvorba cron jobu pro pravidelý update
- Dao ukladani namere polohy do DB

### 5.2.1 Podnadpis

text

## 5.3 Načítání struktury vlaku do dispečinkového systému

- Tvorba Provideru (inteface, implementace)
- Vytvoreni mailu pro odebírání excel souboru se strukturou vlaku rozepsanou na den
- Pripojeni se na tuto mailovou adresu pomocí javamail knihovny
- Nacteni prijatých mailu a odfiltrovani prebytecných mailu, které jsou bud stare nebo neobsahují prilohu

- Otevření přílohy a načtení excelového souboru
- Otevření pracovního sešitu a načtení dat z excelových bunek do CSV souboru
- Parsování CSV souboru a uložení dat (konverze dat) do datových prepravek
- Dao funkce (hibernate dotaz do DB)
- Tvorba service (inteface, implementace)
- Pomocí stavajících funkcí uložíme data do DB
- Finalní úprava a kontrolní mechanismy na správnost číslování vozu
- tvorba cron jobu pro pravidelý update

### 5.3.1 Podnandpis

text



## 6 Distribuce zpoždění

Plazmy, GTFS-real time, SMS, mail

### 6.1 Zobrazení na zpozdeni spoje

- Vyuzit stavajici funkci pro zobrazovani na Plazmach
- Zasilani SMS a mailu o zpozdeni pro jednotlivy prodejce a dispecery podle skupin

#### 6.1.1 Podnandpis

text

### 6.2 Zobrazení údajů o zpoždění

- Tvorba noveho endpointu pro real-time GTFS
- GTFS-RT musi obsahovat stejne identifikatory jako staticke GTFS
- Google Protokolbuffer
- Pripojeni nacistani zpozdeni a poloha GPS

#### 6.2.1 Podnandpis

text

## ZÁVĚR

Text závěru

## SEZNAM POUŽITÉ LITERATURY

- [1] 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>.
- [2] *Spring Framework* [online]. Pivotal Software, 2018 [cit. 2018-05-18]. Dostupné z: <https://projects.spring.io/spring-framework/>.
- [3] 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).
- [4] 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).
- [5] 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).
- [6] 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>.
- [7] 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>.
- [8] 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).
- [9] 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)).
- [10] 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)).
- [11] 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).

- 
- [12] 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).
- [13] 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).
- [14] 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>.
- [15] 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).
- [16] 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).
- [17] 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).
- [18] 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).
- [19] *The Most Advanced REST & SOAP Testing Tool in the World* [online]. SmartBear Software, 2018 [cit. 2018-05-19]. Dostupné z: <https://www.soapui.org/>.
- [20] 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).
- [21] 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).
- [22] 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).

- 
- [23] 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).

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

AMF	Action Message Format
API	Application Programming Interface
CSV	Comma Separated Values
EJB	Enterprise JavaBeans
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HQL	Hibernate Query Language
IDE	Integrated Development Environment
IDEA	IntelliJ IDEA
IoC	Inversion of Control
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JMS	Java Message Service
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
POJO	Plain Old Java Object
POM	Project Object Model
REST	REpresentational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
WSDL	Web Services Description Language
XML	eXtensible Markup Language

**SEZNAM OBRÁZKŮ**

Obr. 1.1	Náhled možností dependency injection . . . . .	11
Obr. 2.1	Popisek obrázku . . . . .	17
Obr. 5.1	Úprava Pom.xml . . . . .	22

**SEZNAM TABULEK**

Tab. 2.1	Popisek tabulky . . . . .	17
----------	---------------------------	----



## SEZNAM PŘÍLOH

P I.	Název přílohy
------	---------------

## **PŘÍLOHA P I. NÁZEV PŘÍLOHY**

Obsah přílohy