NSS - Semestrálka

Informačný systém pre spoločnosti dopravou nadrozmerných nákladov - Juraj Orenčák, Petr Pejchal

## O klientovi (obchodný prínos)

Firma nášho klienta sa zaoberá logistikou. To zahŕňa spracovanie objednávky, expedíciu tovaru a samozrejme zaistenie dopravenia tovaru k zákazníkovi. Zhodnotili sme, že jednotný informačný systém, ktorý by umožnil, jednoduché prehliadanie zakázok, vozidiel a ich vodičov dispečerom by nášmu klientovi značne ušetrilo prácu a náklady. Taktiež by nasadenie nášho systému pomohlo klientovi s udržiavaním evidencie o pohyboch vo firme.

## Súčasný stav

Momentálne je štruktúra spravovania zakázok relatívne chaotická. Klientova firma je relatívne malá, preto sa na správu využívajú rôzne open-sourcové nástroje ako sú napríklad google sheets, prípadne sa niektoré veci riešia na papieri (správy o závadách a podobne). Tento stav je nežiadúci, keďže to kladie veľký dôraz na precíznosť práce dispečerov. Fungovanie firmy v budúcnosti a pokračovanie využívania takéhoto systému je prakticky nezlúčiteľné vzhľadom na to, že väčšina konkurencie používa digitalizované systémy len málo odlišné od nášho riešenia.

## Ciele projektu (stav “to be”)

Cieľom projektu bude vytvoriť informačný systém, ktorým sa uľahčí správa expedovania zakázok zákazníka. Samotný systém bude zabezpečovať komunikáciu dispečera s vodičom kamiónu (umožní mu vodiča informovať o samotnej zakázke v resp. stave ciest, dávať mu rady ako sa vyhnúť zdržaniu a zároveň dohliadať na uskutočnenie zakázky ako takej). Zároveň bude systém umožňovať dispečerom management expedovania zakázok.

## SWOT analýza

### Strengths

* Programátori majú dosť jasný obraz o tom, ako by mal výsledný systém vyzerať
* Skúsenosť vývojárov s technológiami, ktoré využijú

### Weaknesses

* Silná konkurencia
* Veľa bočných projektow

### Opportunities

* zisk dlhodobého zákazníka
* potenciálne ďalšie zákazky
* možnosť využitia častí implementácie tohto projektu v budúcnosti

### Threats

* Nedodržanie časového plánu (odovzdanie po deadline)
* Ohrozenie vývoja vzhľadom na celosvetovú pandémiu
* Nespokojnosť zákazníka s finálnym produktom

## 5F analýza

### Konkurencia

* V tomto sektore je konkurencia relatívne veľká, čo by mohlo ohroziť náš projekt

### Vstup nových firiem

* Na náš projekt vstup nových firiem nebude mať žiadny dopad

### Dodávatelia

* Náš systém bude používaný interne, takže dodávatelia tento projekt vôbec neovplyvnia

### Odoberatelia

* Systém je cielený na firmy zaoberajúce sa logistikou, takže nám poskytuje veľké množstvo nových zákazníkov v budúcnosti

### Substitúcia

* Hrozba nahradenia je relatívne vysoká, vzhľadom na konkurenciu a systémy, ktoré používa

## Analýza PEST(E)

### Politika

* Súčasný stav, vzhľadom na výskyt pandémie je otázny, avšak tento sektor je postihnutý minimálne

### Ekonomika

* Všetko naznačuje príchod ekonomickej krízy, čo znamená že vývoj systému môže byť v najbližších mesiacoch skomplikovaný

### Spoločnosť

* Tento bod zodpovedá bodu o súčasnej Politike

### Technológia

* Súčasná situácia je stabilná

### Ekológia

* Náš projekt má v tomto okruhu minimálny, až nulový dopad

## Funkčné požiadavky + Zoznam užívateľov

Bežný užívateľ (vodič) bude môcť:

* pristupovať do systému pomocou jeho prihlasovacích údajov
* vidieť informácie o jeho účte, vozidle a zakázke, ktorá mu je momentálne priradená
* meniť stav zakázky na ukončenú/neukončenú (v prídape výskytu komplikácie)
* vidieť updaty od dispečera

Admin (dispečer) bude môcť:

* pristupovať k systému pomocou prihlasovacích údajov
* vytvárať/upravovať/mazať účty vodičov
* vytvárať nové zakázky a priradzovať ich vodičom
* komunikovať s vodičmi pomocou message boardu

## Nefunkčné požiadavky

* (Desktopová) Aplikácia bude bežať na operačnom systéme windows 7 a novšom
* Aplikácia bude vyvíjaná za použitia: Java, Springboot, JavaFX
* DtAplíkacia bude deploynutá na servere Tomcat, správu dát zaistí PostgreSQL databáza
* Grafické rozhranie bude natoľko jednoduché, aby sa s ním každý oboznámil čo najrýchlejšie

# Případy užití

## Editovat poruchy

Řidič si bude moct zobrazit a zaeditovat novou poruchu na svém vozidle.

## Zobrazit informace

Řidič bude moct zobrazit všechny informace týkající se jeho výkonu práce.

## Měnit stav zakázky

Řidič bude moct měnit stav zakázky na nedokončenou, dokončenou. Také bude moct přidat popis k zakázce.

## Komunikovat přes messageboard

Uživatelé budou moct vytvářet, zobrazovat zprávy mezi sebou pomocí messageboardu.

## Vytváření zakázek a vozidel

Dispečeři budou moct vytvářet, mazat a přiřazovat řidičům zakázky a vozidla.

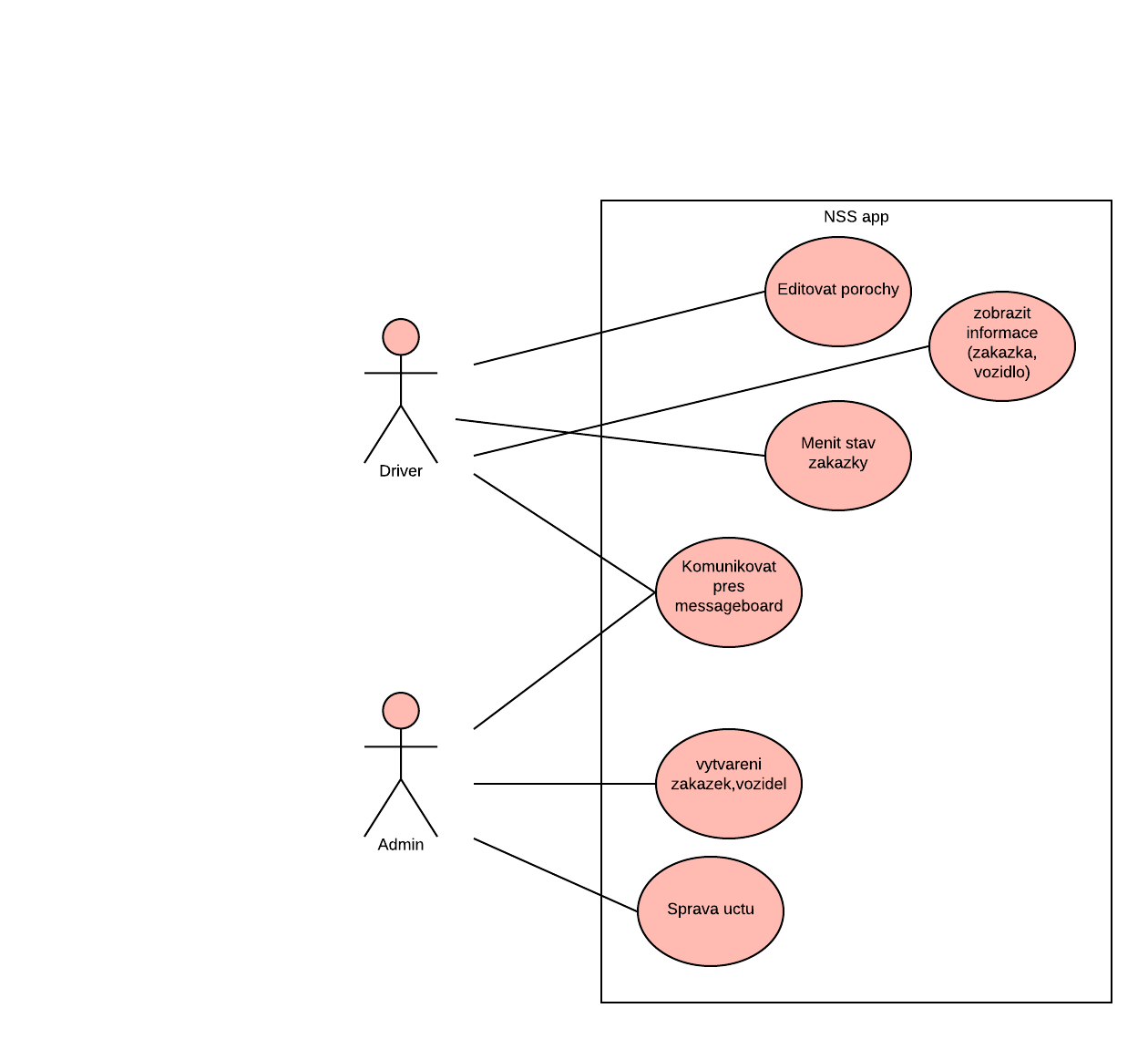
## Správa účtů

Dispečeři budou moct vytvářet, mazat a editovat účty jak dispečerů tak řidičů.

## 

## 

## 

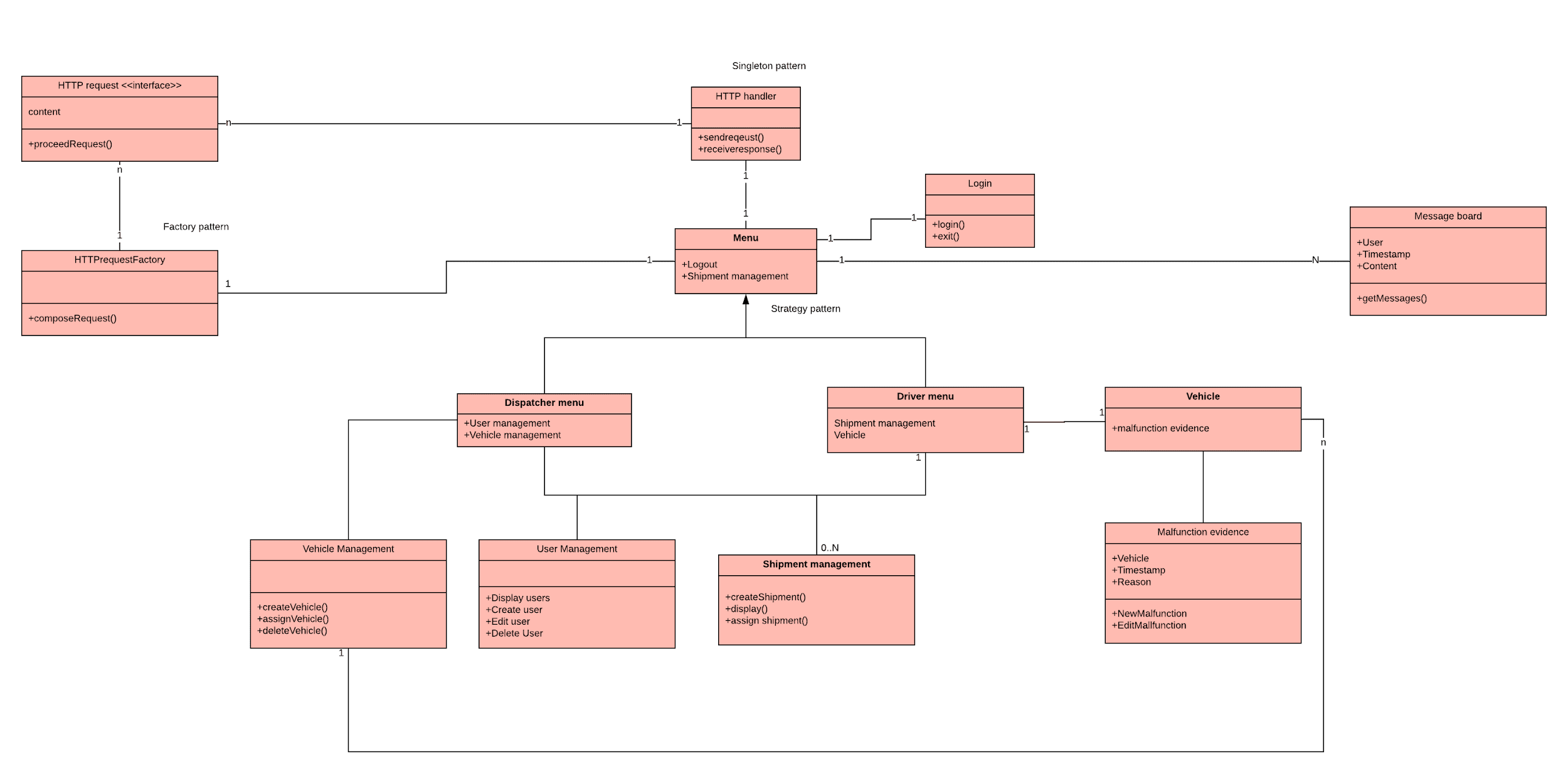


# 

# Architektura aplikace

## Class diagram klientské části

Klientská aplikace bude desktopová aplikace. GUI bude vytvořené pomocí JavaFX.



## 

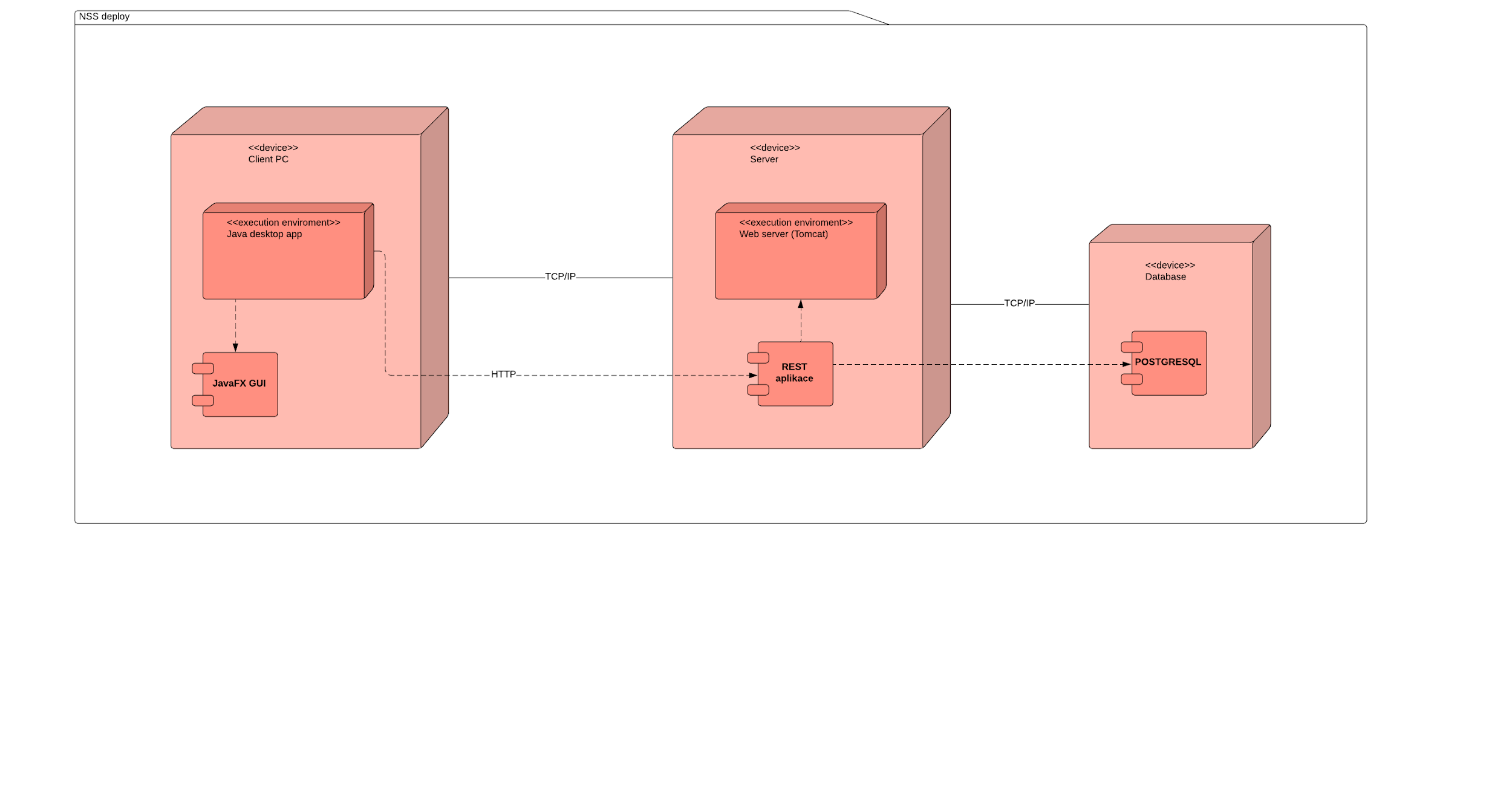
## Komponentový diagram serverové části

Serverová část bude implementovaná pomocí Java Spring boot. Deployovaná bude na serveru Tomcat.

### 

## Diagram nasazení

Aplikace bude vystavěná na principu komunikace klient-server. Komunikace bude zajištěna technologií REST na serverové části.



## WBS

Po pôvodnom rozbore sme si prácu rozdelili na tieto aktivity:

1. Vytvorenie modelu
2. Použitie vytvoreného modelu pri vytvárani serveru (dao, service, rest package)
3. Vytvorenie základných funkcionalít klienta (komunikácia so serverom)
4. Návrh a príprava UI
5. Adaptácia klienta na server (vytvorenie controllerov)
6. Dokončenie UI
7. Vypracovanie unit testov na serveri
8. Použitie Hazelcast v klientovi
9. Deploy serveru na Heroku

## 

## Matice zodpovědnosti

|  |  |  |
| --- | --- | --- |
| **Task podľa WBS** | **Juraj Orenčák** | **Petr Pejchal** |
| **1.** | X | X |
| **2.** | X | X |
| **3.** |  | X |
| **4.** | X |  |
| **5.** |  | X |
| **6.** | X |  |
| **7.** | X | X |
| **8.** | X |  |
| **9.** | X | X |

## Gantt diagram

## Analýza rizík FMEA

\* časti A,B,C sú hodnotené na škále 1 - 10. Pri tejto analýze existuje veľká podobnosť s predošlými analýzami, definíciami rizík a ich riešení, preto sme nezachádzali do hĺbky

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Failure mode | A - Severity | B - Probability of occurence | C - Probability of detection | Risk preference number (A\*B\*C) |
| Nedodržanie časového plánu | 10 | 3 | 10 | 300 |
| Ohrozenie vývoja epidémiou | 5 | 4 | 9 | 180 |
| Nespokojnosť zákazníka s produktom | 8 | 2 | 9 | 144 |

## Deploy

Pre spustenie klienta treba zadať vhodné vm options a parametre pri spúšťaní. Aplikácia sa štartuje pomocou custom configurácie (“Application”).

Main class: **cz.cvut.fel.pjv.cms.client.Client**

VM options: **-Djava.library.path="C:\WINDOWS\Sun\Java\bin" --add-exports javafx.controls/com.sun.javafx.scene.control.behavior=com.jfoenix --add-exports javafx.controls/com.sun.javafx.scene.control=com.jfoenix --add-exports javafx.base/com.sun.javafx.binding=com.jfoenix --add-exports javafx.base/com.sun.javafx.event=com.jfoenix**

Parameters: **-XstartOnFirstThread**

## SQL skript

Pre vytvorenie potrebnej DB používame tento skript

create table regularuser

(

username varchar(255) not null

constraint regularuser\_pkey

primary key,

fullname varchar(255),

licensenumber varchar(255),

password varchar(255),

vehicleid integer

);

create table systemmanager

(

username varchar(255) not null

constraint systemmanager\_pkey

primary key,

fullname varchar(255),

password varchar(255)

);

create table vehicle

(

licenseplate varchar(255) not null

constraint vehicle\_pkey

primary key,

availability boolean,

shipmentid bigint,

id serial not null,

driver varchar(255)

constraint vehicle\_regularuser\_\_fk

references regularuser

);

alter table regularuser

add constraint regularuser\_vehicle\_id\_fk

foreign key (vehicleid) references vehicle (id);

create table shipment

(

id integer not null

constraint shipment\_pkey

primary key,

cargo varchar(255),

destination varchar(255),

status varchar(255),

vehicle varchar(255)

constraint vehicle

references vehicle,

shippingdate date

);

create unique index vehicle\_id\_uindex

on vehicle (id);

create table message

(

driver varchar(255)

constraint message\_regularuser\_\_fk

references regularuser,

dispatcher varchar(255)

constraint message\_systemmanager\_\_fk

references systemmanager,

time timestamp,

sender varchar(10),

content varchar,

id integer not null

constraint message\_pk

primary key

);

create table archive

(

id bigint not null

constraint archive\_pkey

primary key,

assignedvehicle varchar(255),

cargo varchar(255),

description varchar(255),

destination varchar(255),

status varchar(255)

);

create table sequence

(

seq\_name varchar(50) not null

constraint sequence\_pkey

primary key,

seq\_count numeric(38)

);

## Znovupoužiteľnosť

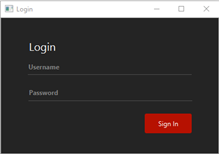
Klient - celý desktopový klient je zameraný na funkcionality, ktoré slúžia logistickým firmám, takže pri malom prerobení kódu (za predpokladu že sa server tiež zachová) môže byť použitý rôznymi firmami

Server - server je basic aplikácia určená na komunikáciu s sql databázou a po zmene modelu je aplikovateľná v ľubovolnej situácii

# **Užívateľská príručka**

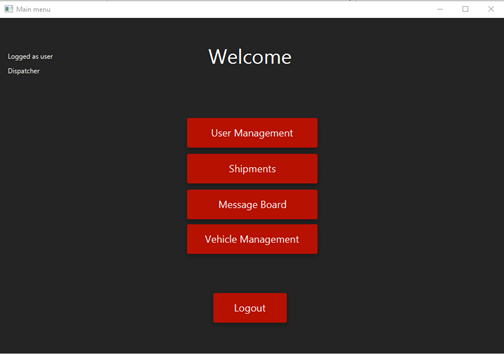
## **Login**

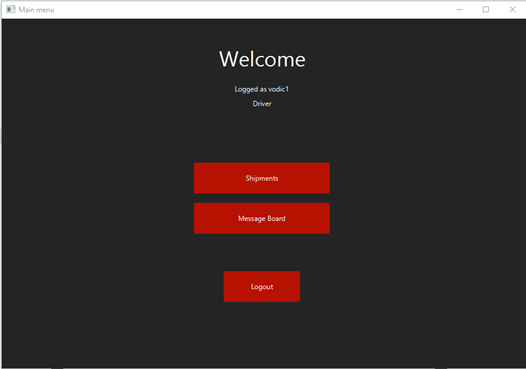
Po spustení aplikácie sa zobrazí uvítacie okno, k torom sa nachádza prihlasovací formulár. Zadá username, a heslo.



## **Main menu V/D**

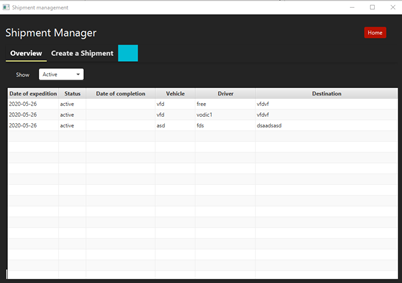
Po prihlásení je užívateľ presmerovaný do hlavného menu, kde si môže vyberať z funkcionalít aplikácie podľa potreby. Rozdiely medzi dispečerským a vodičským hlavným menu sú minimálne.

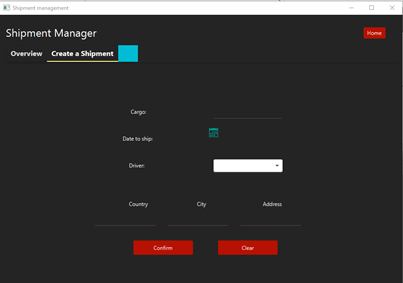


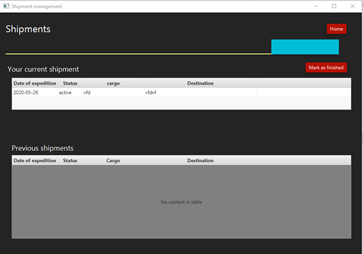


## **Shipment management**

Vodič: má možnosť sledovať zakázky a taktiež po dokončení súčasnej ju označiť za dokončenú, pričom dispečer môže vytvárať nové a priradzovať im vozidlá

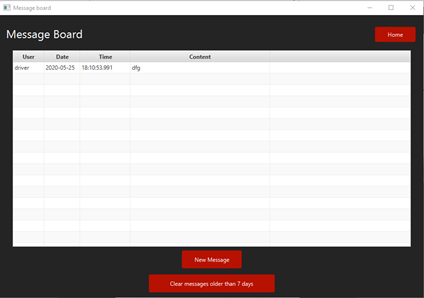






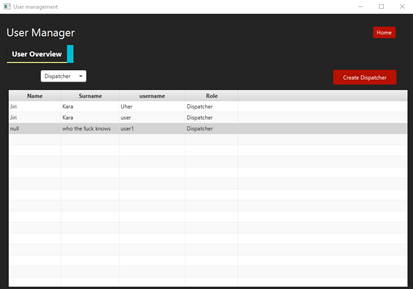
## **Message board**

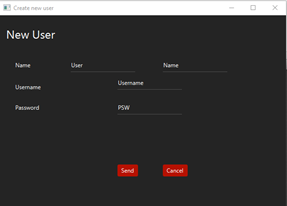
Jednotný pre všetkých užívateľov, umožňuje užívateľom vzájomné posúvanie inštrukcíí a informácii o cestách



## **Create profile (len pre dispečera)**

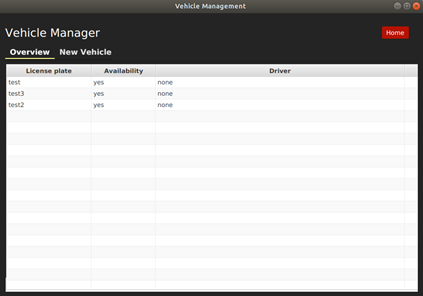
Umožňuje dispečerom vytvárať nové užívateľské účty. Najprv užívateľ vyberie typ účtu a následne po vyplnení formulára zaradí nový účet do databázy.





## **Vehicle Management**

Umožňuje dispečerovi spravovať vozidlá v systéme



## **Tlačidlo logout**

Self explanatory... umožňuje užívateľom návrat k prihlasovaciemu oknu

# Analýza kódu

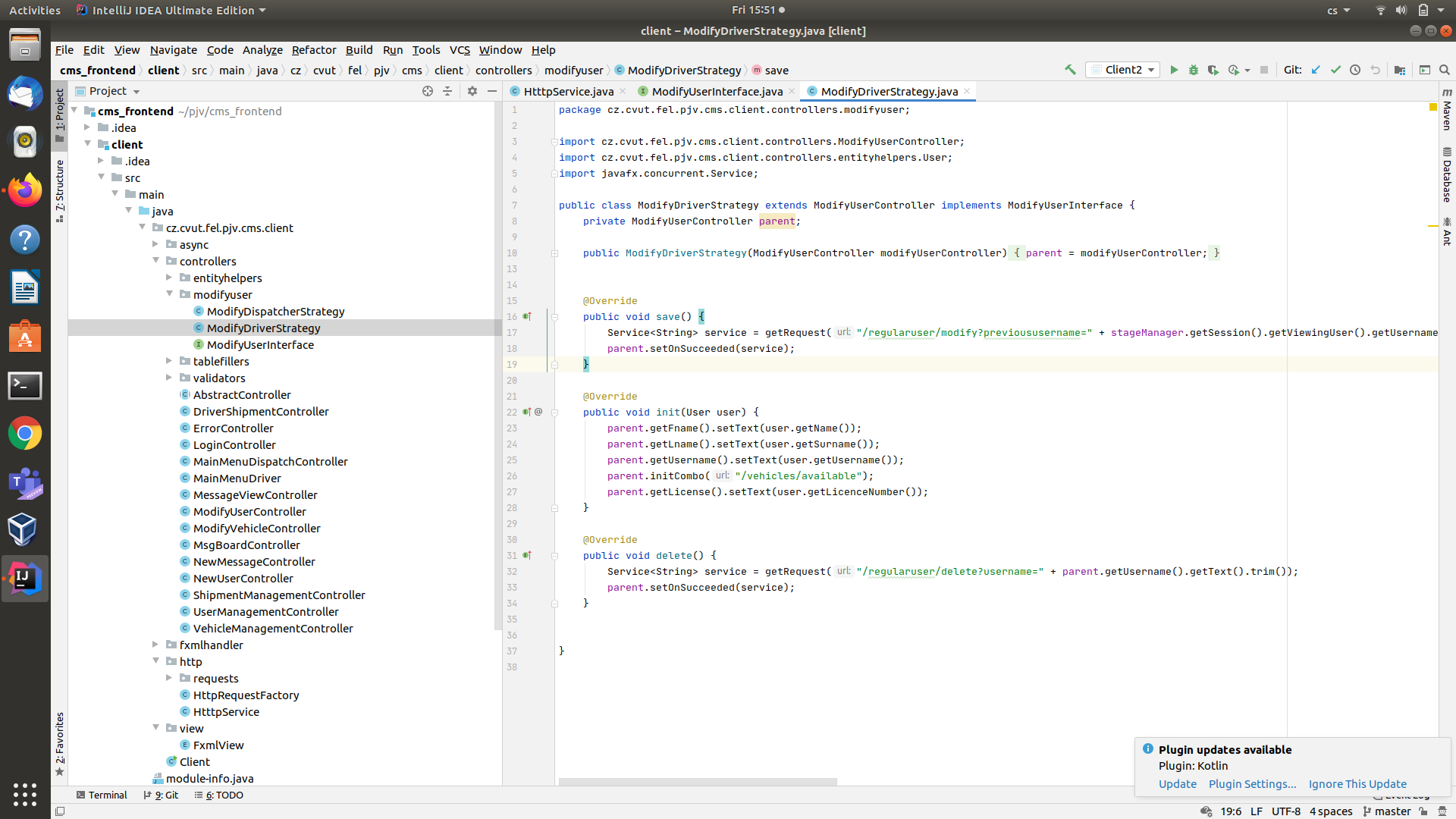
V této kapitole se podíváme na využití design patternů a jejich ukázek.

## Strategy

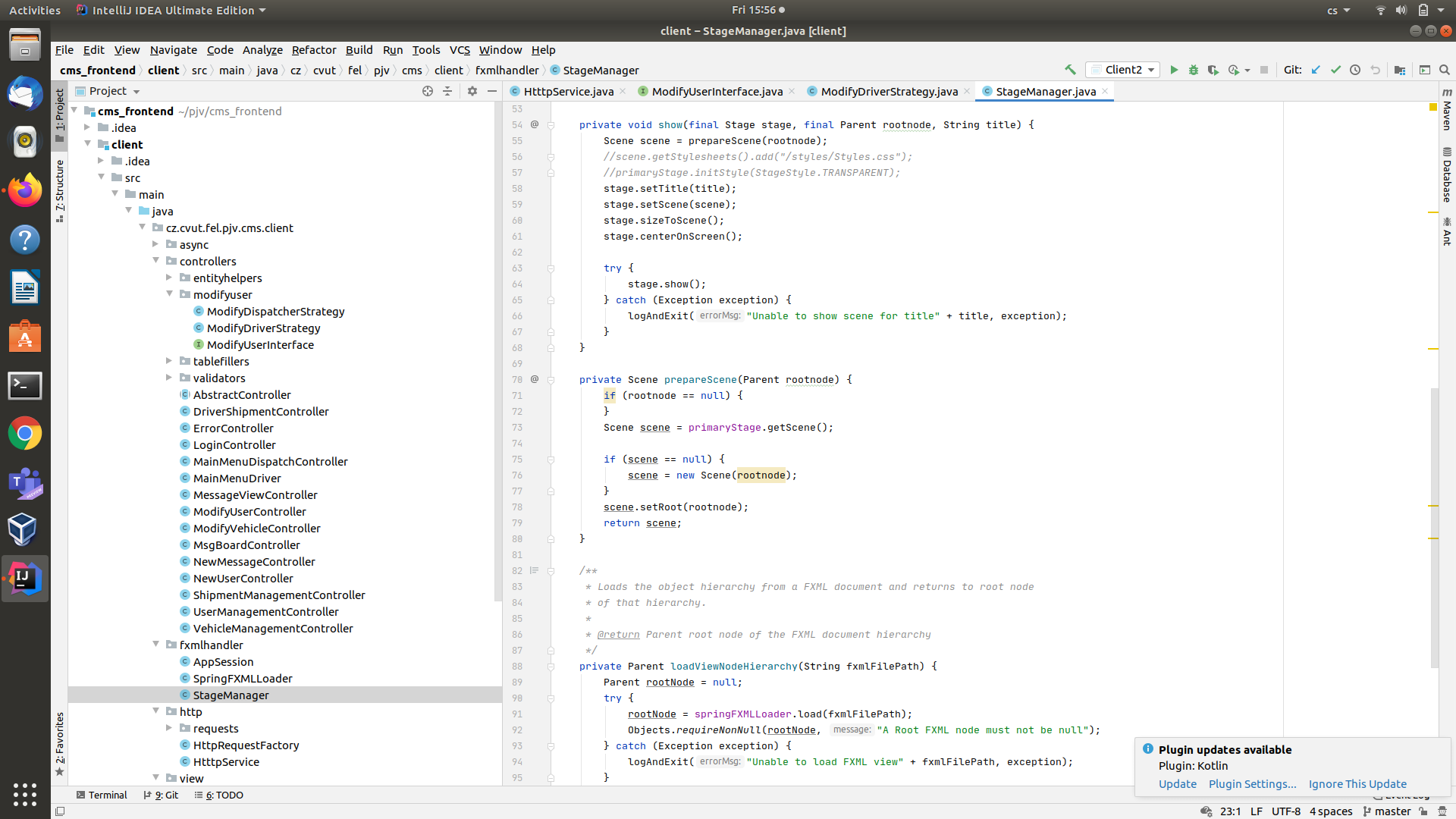
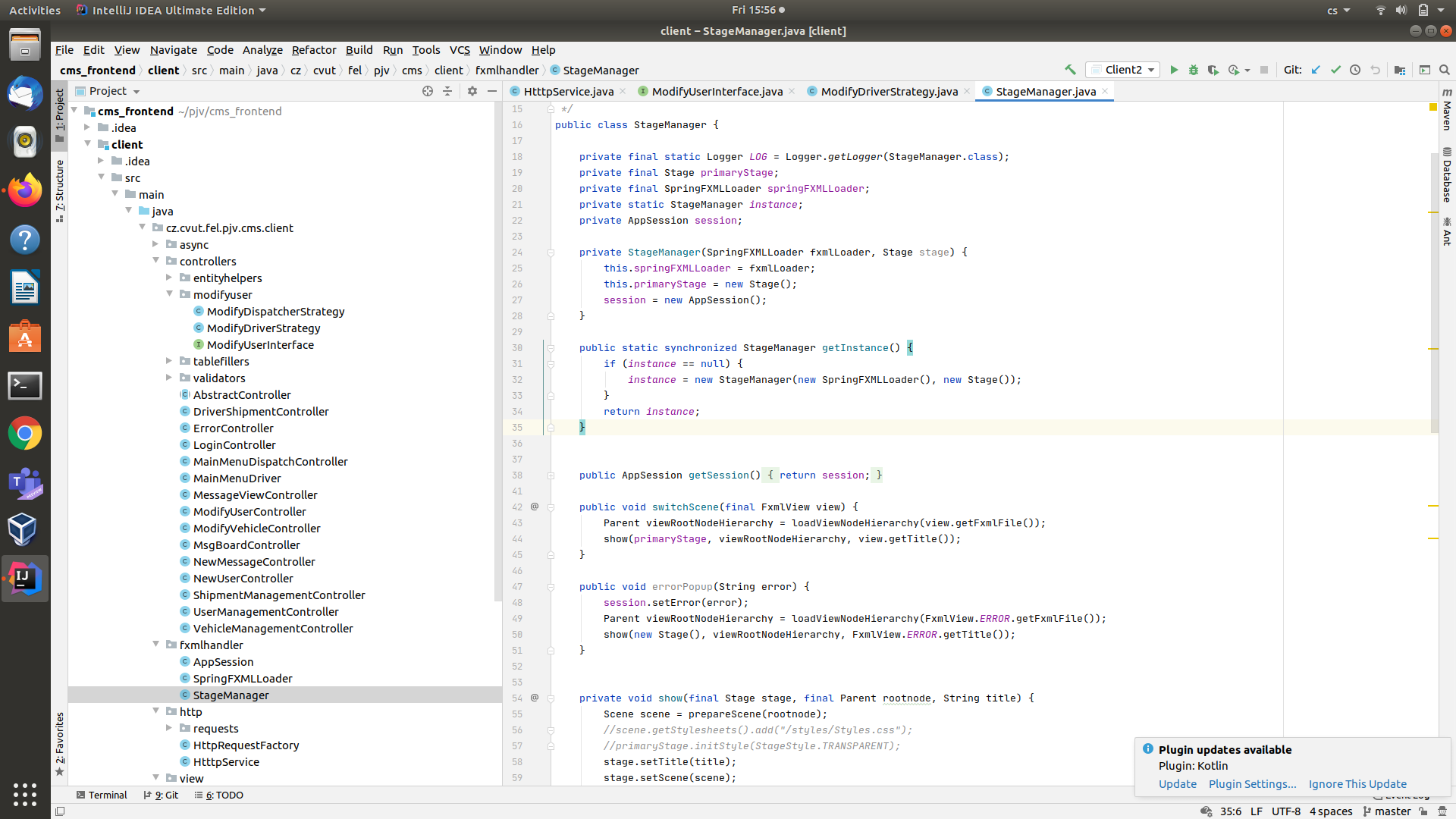
Strategy pattern je využit na více místech, kde je potřeba rozdělit chování aplikace podle toho, jestli se jedná o řidiče, nebo dispečera. Jako příklad jsem vybral úpravu uživatelského účtu.

### Strategy interface

### Jedna z jeho implementací



## Facade

Facade pattern je využit pro přepínání zobrazovaných obrazovek třídou, která se jmenuje stageManager.

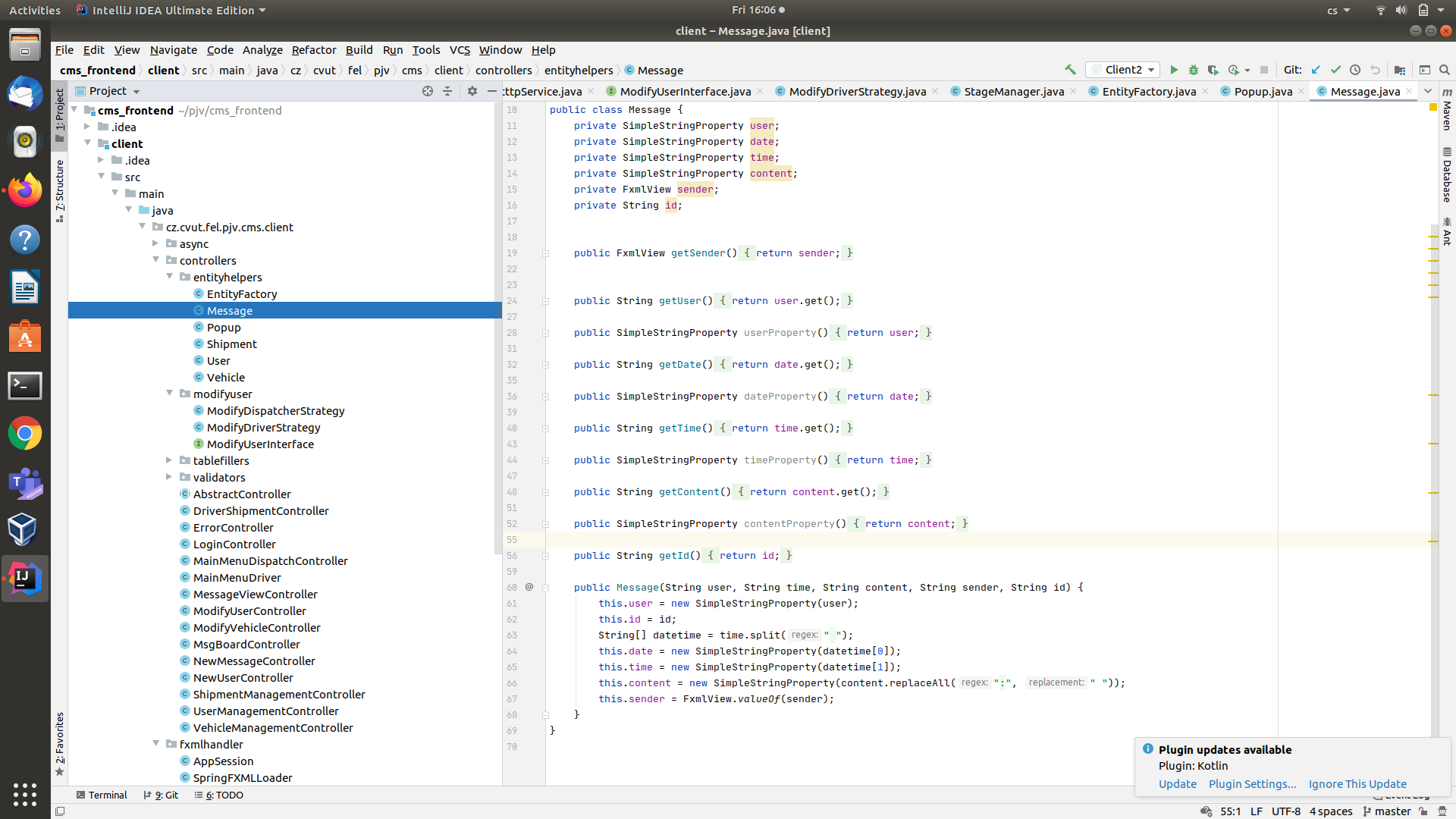
## Factory

Factory pattern stejně jako strategy pattern je využit na více místech. Pro demonstraci jsem vybral třídu EntityFactory, která má za úkol z listu stringů vytvořit list objektů, které následně slouží jako výplň tabulek.

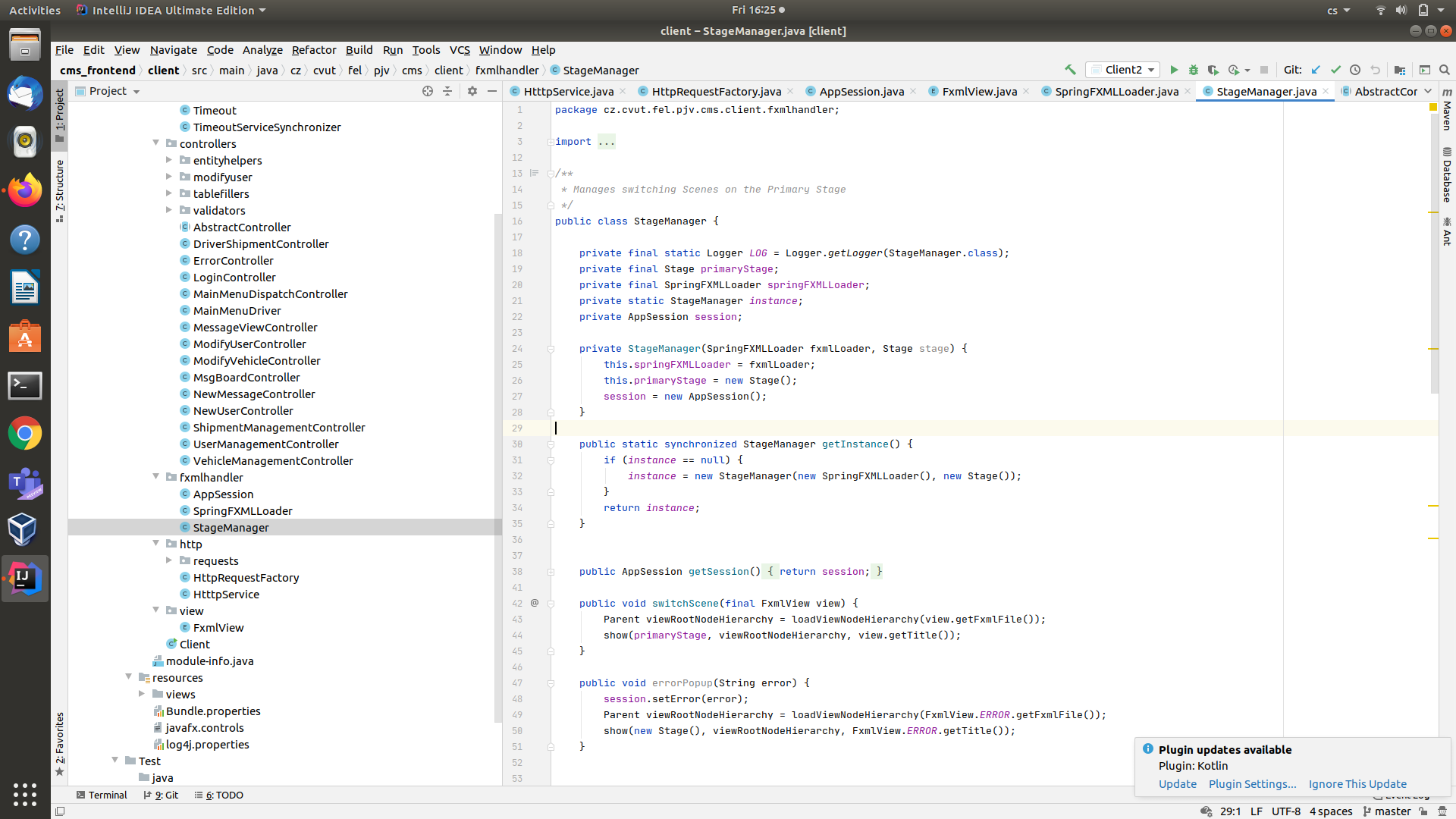
### Třída EntityFactory

Funkce jsou si velmi podobné a proto jako ukázka asi stačí takhle.

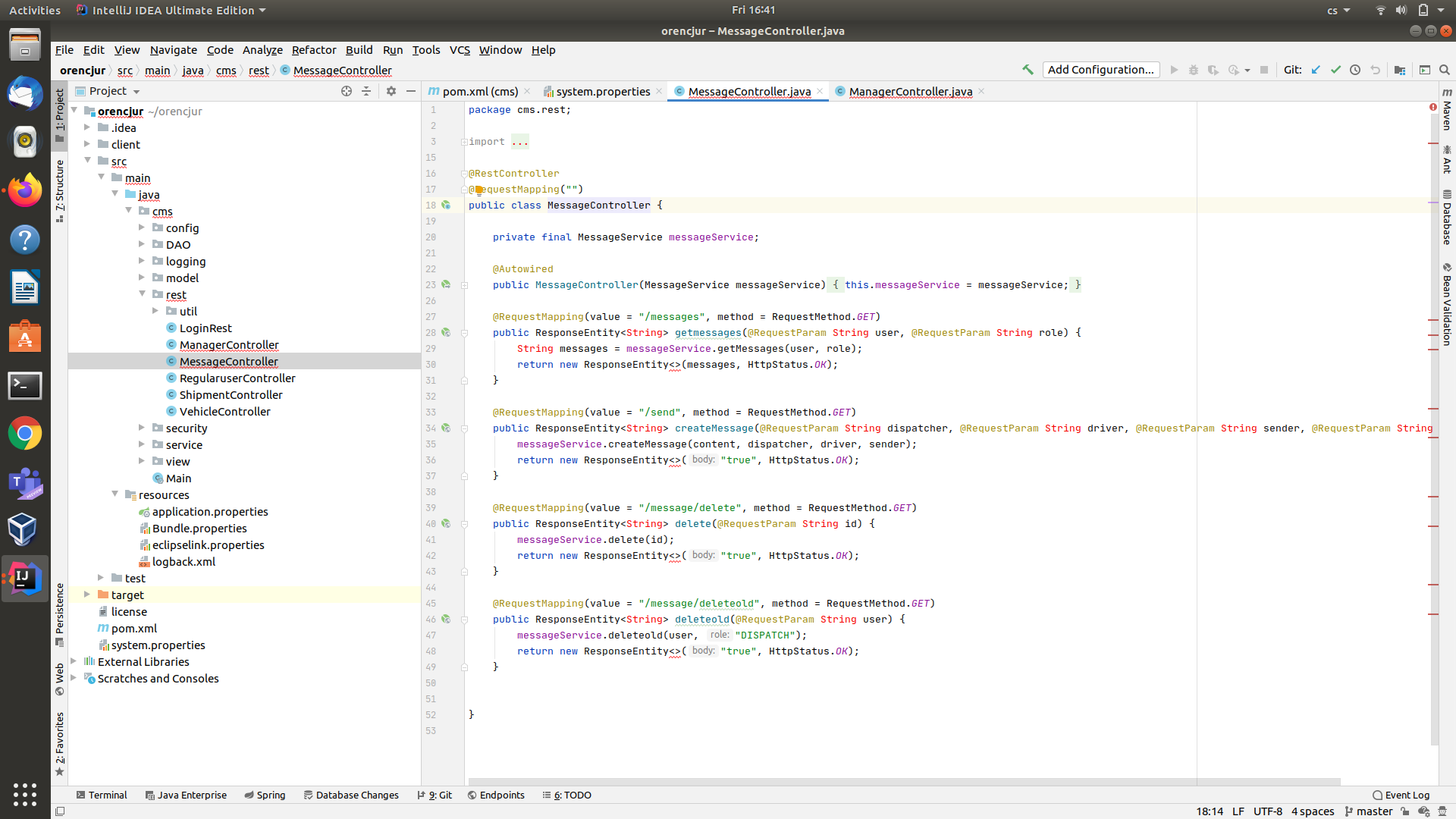
### Jedna z tříd vytvářených EntityFactory



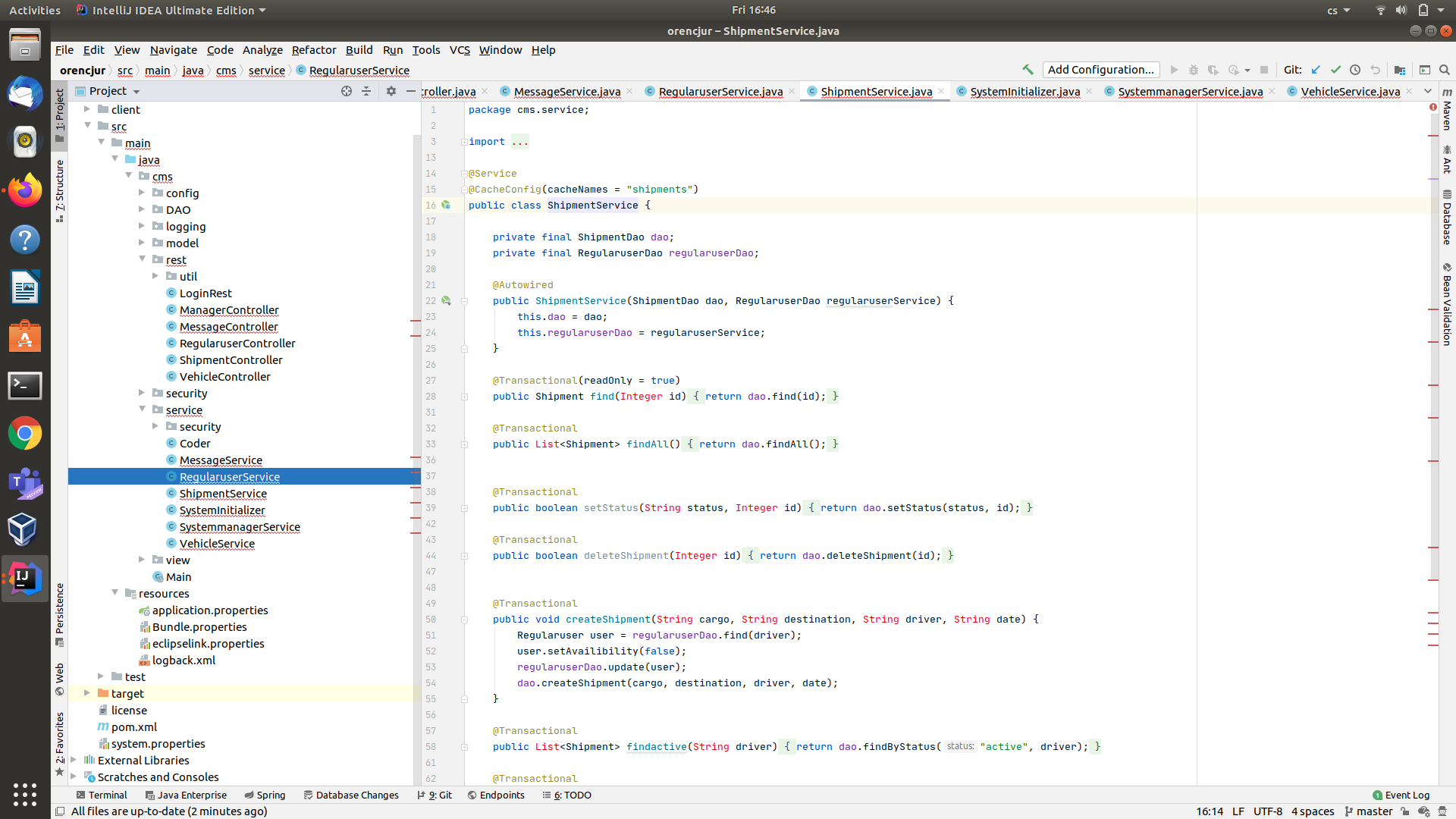
## Singleton

Singleton je využit v třídě StageManager, kterou jsme tu už jednou měli.

## REST

Komunikace s klientem je zajištěna REST rozhraním, jako příklad je uveden jeden z restových controllerů.

## Hazelcast

Pro zlepšení výkonu je použit caching pomocí hazelcastu na servisové vrstvě serveru. Například v třídě ShipmentService

# Nasadenie a podpora

Po úspešnom otestovaní aplikácie ju klient bude možný okamžite začať používať na svojich desktopových zariadeniach. Dodatočne mu bude poskytovaná podpora v podobe pravidelnej údržby systému a databázy. Pri vyskytnutí závady budeme schopní ponúknuť urýchlené odstránenie problému.

# Záver

Táto semestrálna práca bola vypracovaná za účelom vytvorenia a dokumentácie desktopovej client-server aplikácie. Server, ktorý komunikuje s PostgreSQL databázou a klientom sme zostrojili pomocou SpringBootu. Na komunikáciu s klientom bolo použité REST api a následne bol server deploynutý na Heroku (https://nsscms.herokuapp.com). Klient bol naprogramovaný v Jave, samotné UI v JavaFX. Vypracovanie tohto dokumentu bolo mierne problematické, vzhľadom na to, že po naplánovaní sme pri samotnej implementácii narazili na niekoľko prekážok a boli sme nútení mierne pozmeniť stratégiu. Avšak v konečnom dôsledku sme implementáciu boli schopní dokončiť viac ako týždeň pred deadlinom, čo bolo určite veľké pozitívum.