# **Teхнически университет – Варна Факултет:** Факултет по изчислителна техника и автоматизация **Катедрата:** Софтуерни и интернет технологии **Специалността:** Софтуерни и интернет технологии

# **Тема на проекта:** Пътнически превози

# Студенти: Альона Бундева Фак. н. 19621832 Слави Лолов Фак. н. 19621675

1. **Пълно задание (условие) на проекта.**

Да се разработи информационна система, предоставяща услуга пътнически превози.

Програмата съхранява и обработва данни за разпространение на билети.

Системата поддържа два вида потребители администратор и клиенти (пътническа

компания, разпространител, касиер) с различни роли за достъп до функционалностите

в системата.

Операции за работа с потребители:

* Създаване на пътническа компании от администратор;
* Създаване на разпространители от администратор;
* Създаване на касиери от разпространител
* Поддържане на профили с характеристики на клиентите в системата (хонорар и
* др…)
* Рейтинговане на клиентите в системата

Системата поддържа операции за работа със пътувания:

* Добавяне на ново пътуване от организатор (вид на пътуването, дестинация,
* дата на заминаване и пристигане, брой места, вид(ове) транспорт, ограничение
* в закупуването на билет от едно лице и др…);
* Заявяване на билети за продажба от разпространител и потвърждаване от
* пътническа компания
* Продаване на билети за пътуване от касиер, създаване на формуляр за
* закупуване (информация за купувача, избор на място и др…)

Системата поддържа Справки по произволен период за:

* Клиентите в системата:
* Пътническа компания с наличните пътувания (за разпространители);
* Разпространители
* Касиер
* Закупени билети (дата, статус, и др…);
* Пътувания;

Пътническата компания достъпва справки само за пътувания, на който е организатор.

Разпространителя достъпва справки за всички актуални пътувания. Касиерите има

право на справки само за пътувания чиито билето-разпространители са.

Системата поддържа Известия за:

* Новопостъпила заявка за пътуване (в профила на разпространител);
* Периодично уведомление за продадени билети от пътуване (в профила на
* собственика).
* Отменено пътуване (в профила на касиера и разпространителя)
* Наближаващо пътуване с не продадени билети (собственик, разпространител)

# **Функционални изисквания.**

* Скрипт за създаване на базата данни, всички таблици и връзките между тях, с предварителна проверка за наличието им.
* Диаграма на създадената база данни.
* Системата винаги разполага с един главен администратор, от който могат да се регистрират други регистратори.
* Системата позволява на потребител, логнат като администратор, да създава пътническа компания и разпространители, и да прави оглед на рейтинг на касиерите.
* Потребител, който се логва като компания има следните функционалности:
* Създаване на ново пътуване. То включва:
  + - * Задаване тип на пътуване (екскурзия, обикновено пътуване и т.н.).
      * Задаване на дестинация.
      * Задаване на транспорта, с който ще се пътува (автобус, маршрутка, микробус).
      * Брой места, с които ще разполага съответното
      * Задаване на дата на тръгване.
      * Задаване на дата на пристигане.
* Създаване на билети. То включва:
* Задаване на пътуването.
* Задаване на цена за пътуването.
* Задаване на станцията, от която ще се тръгва.
* Броят билети за съответното пътуване.
* Оглед на билетите, изпратени на станциите. Така потребителят, логнат като компания може да провери статуса на билетите, дали са приети или отхвърлени от съответната станция. При статус “disagree’, потребителят натискайки “confirm” ще ги изтрие. При статус “agree”, билета си сменя статуса на “not bought” и е в очакване на закупуване.
* Оглед на всички активни пътувания.
* Оглед на всички станции (разпространители).
* Потребителят логнат като станция (разпостренител) може да създава касиери, да прави оглед на пътуванията и касиерите, както и да приема изпратени билети от компаниите, извършили изпращането. Ако станцията откаже изпратените билети, статуса им се променя на “disagree” и компанията може да ги изтрие.
* Потребителят логнат като касиер има две функционалности.
* Закупуване на билет. Потребителят задава данните на клиента (име и фамилия), дестинацията, до която клиента ще пътува, мястото заедно с цената за билета
* Оглед на активните пътувания.
* Системата поддържа логове. Записва съобщенията, генерирани от програмата в текстов файл с име „application”. Така потребителят може да наблюдава, какво се случва с програмата.

# **Структура на проекта.**

* **MS SQL SERVER** – база данни.
* **Hibernate** – работа с база данни.
* **JavaFX** – Реализация на бизнес логика и графичен интерфейс.
* **Log4J** – логове за контролерите и проследяване на приложението, какви съобщения връща.
* **Junit** – тестове.

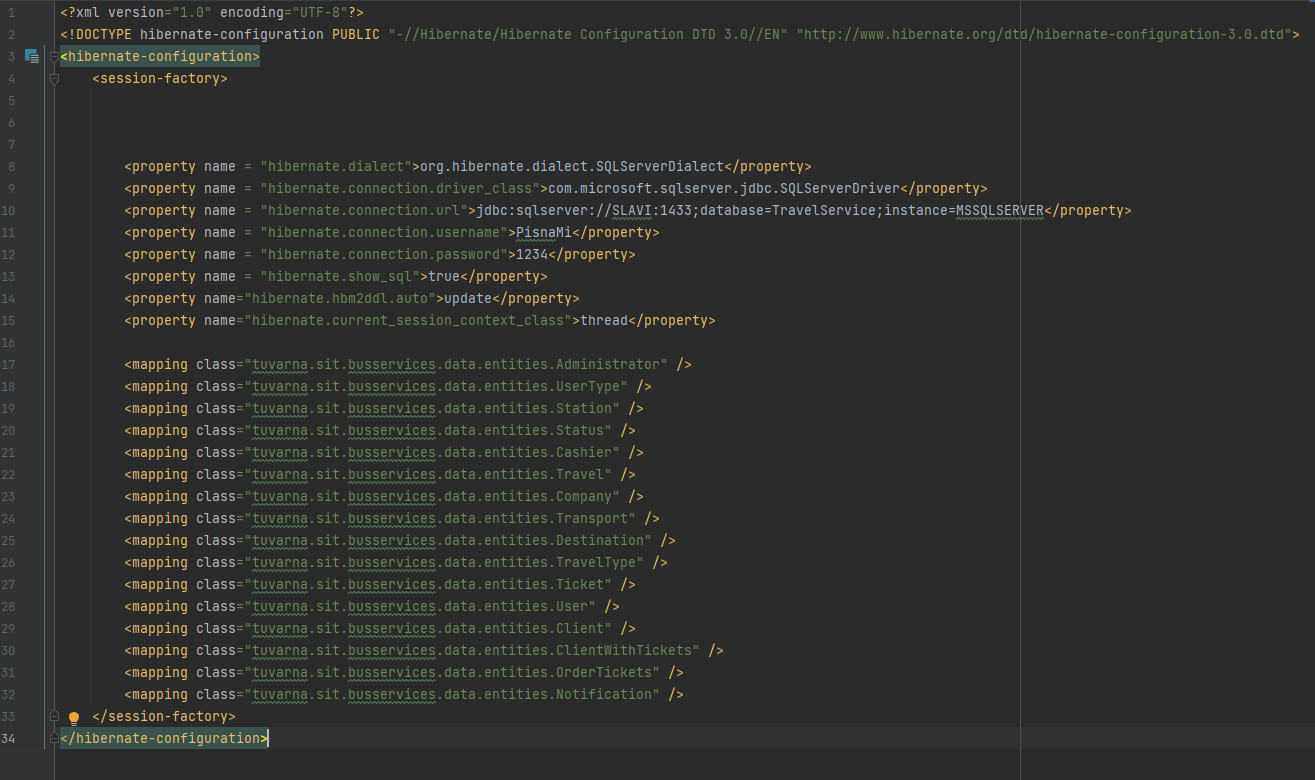
# **Дефиниция на модулите на системата.**

* **Data модул** – в този модул извършваме връзката с базата данни чрез класът Connection. Съхраняваме класовете в под-модула entities. Това са обекти, които биват съхранявани в базата данни и се извършват манипулации върху тях. В под-модула repository се извършва за всеки клас, обект в базата данни, основните функции – добавяне, изтриване, редактиране и извличане от базата данни. Имплентира се интерфейса **DAORepository**, в който се написват основните методи, споменати по-горе, за всяко entity. Тези методи препокриват нуждите на потребителя, защото чрез тях може да прави редакция на запис, добавяне на запис, изтриване на записи по даден критерии.
* **Business модул** – тук се съдържа под-модула service, в който за всеки клас от базата данни се подготвя бизнес логиката.
* **Controller модул** – тук се извършва цялата бизнес логика на приложението. Контролират се диалоговите прозорци като се извличат данните въведени от потребителя, валидират се и в последствие се изпращат надолу по йерархията за записване на данните в базата.

# **Диаграми.**

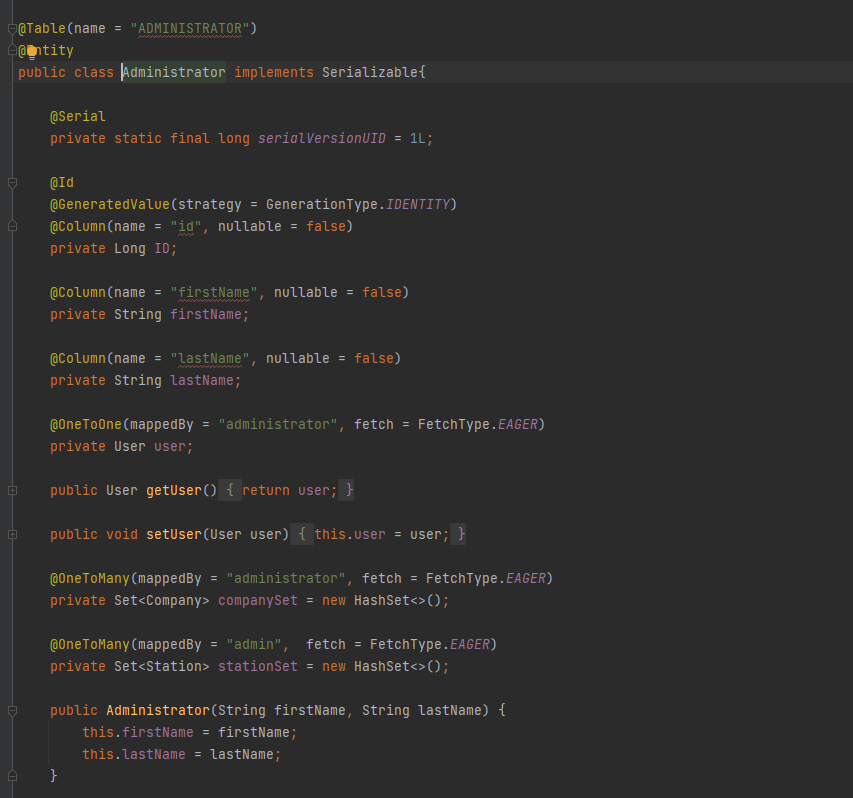
1. **Реализация на базата данни.**

* За реализацията на базата данни сме използвали **MS SQL SERVER**. Причината да работим на нея е, че последното нещо, което бяхме правили беше на нея, а и да пробваме нещо различно от предлаганият Oracle в университета.
* **Hibernate** е използваният framework за комуникация с базата данни. Той е разписан във файла hibernate.cfg.xml.

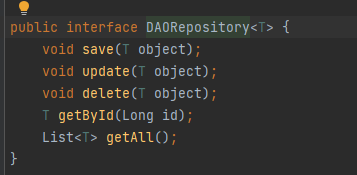


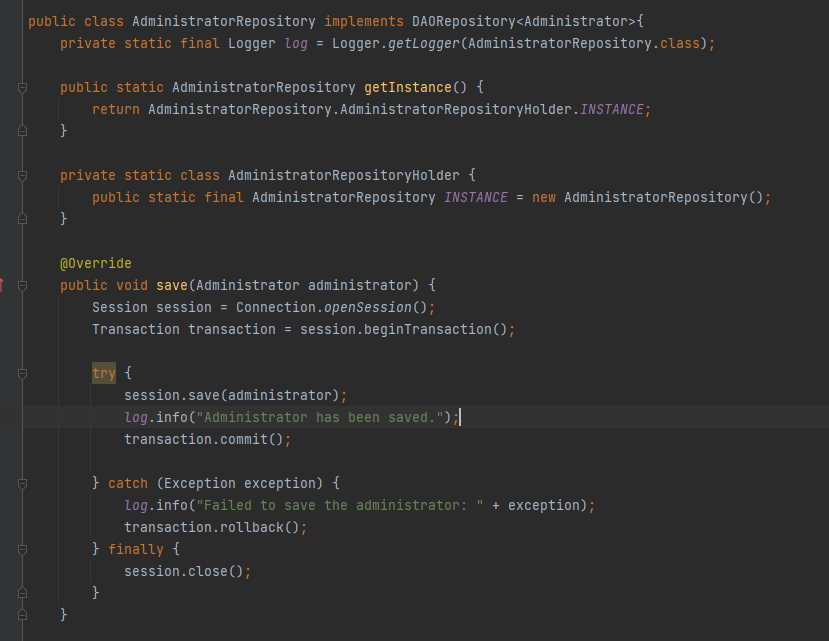
Отварянето на сесия за извършва от разписаният метод openSession() в класът Connection. Той е публичен и се вика навсякъде, където трябва да се отваря сесия за комуникация с базата. 

* Всяка таблица от базата има съответно разписан клас (**Entity**) в сорса. Описани са всички колони на таблицата, както и връзката (**foreign key**) на таблицата с друга таблица. Зададена е и каква е връзката (много към едно, едно към много и т.н.) Пример ще дадем с класът **Administrator**, разписан в модулът **entities**.



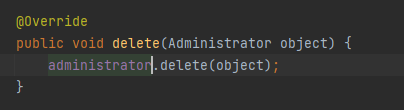
1. **Реализация на слоя за работа с базата данни.**

* ****Работата с **DAO** се извършва в под-модула **repository**. Създаден е интерфейс, който съдържа основните методи, които манипулират данните в базата. Той се имплементира на всяки клас от базата данни, който е означен така: **<име на класа>Repository**. Ето и описанието на интерфейса.
* Пример за клас, имплементиращ **DAORepository<T>** ще дадем с AdministratorRepository.

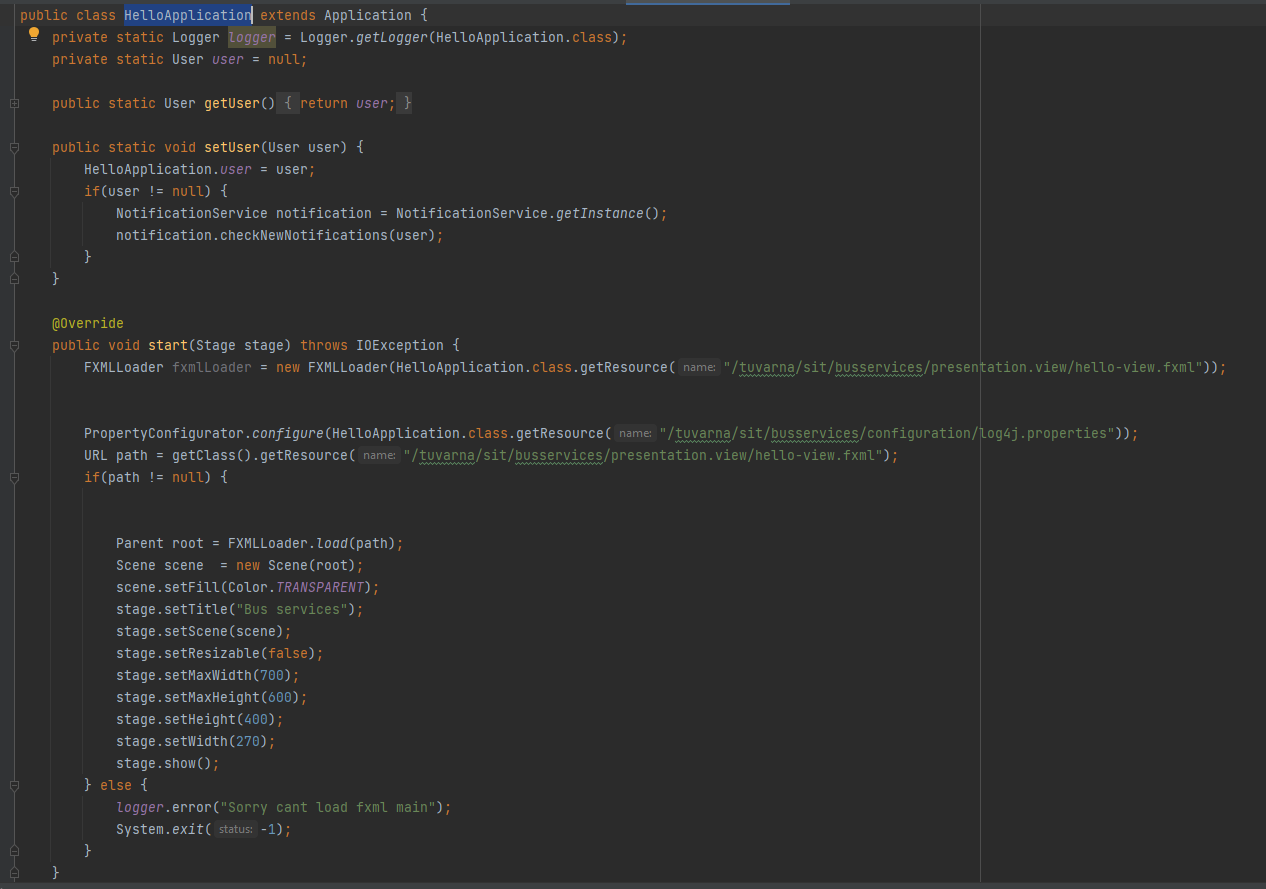
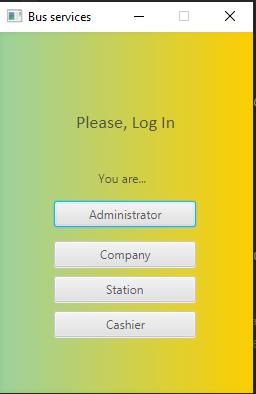


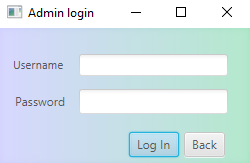


* Тези методи се викат в сервисите (**services**), където се извършва **бизнес логиката** на приложението. Ето и пример на използване метода **delete.**



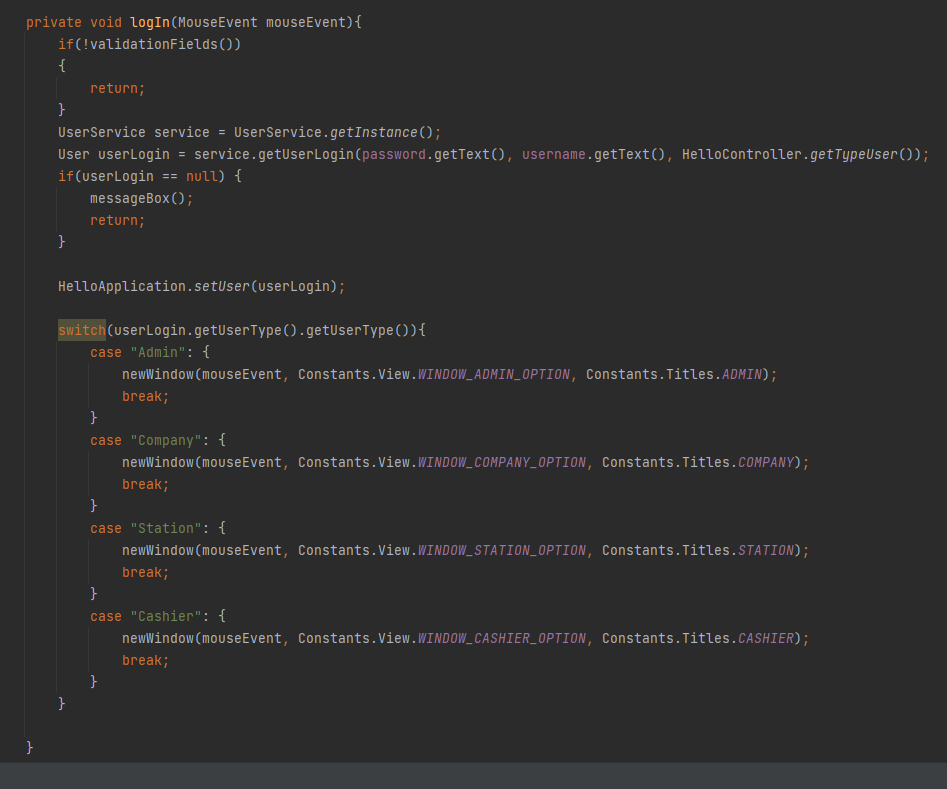
1. **Реализация на бизнес логика и графичен интерфейс.**

* **JavaFX е framework-ът**, който контролира интерфейса и бизнес логиката ни.
* Run класът, от който започва всичко е HelloApplication, който наследява Application. Методиката му на работа е следната: 
* При пускане на приложението се зарежда главният прозорец, от който започва всичко. Там потребителят избира под какво заглавие да се логва (администратор, компания, касиер и станция).
* Според избраният бутон (в примерния случай е Administrator) излиза диалогов прозорец, в който се вписват данните на потребителят (Administrator), който се логва.



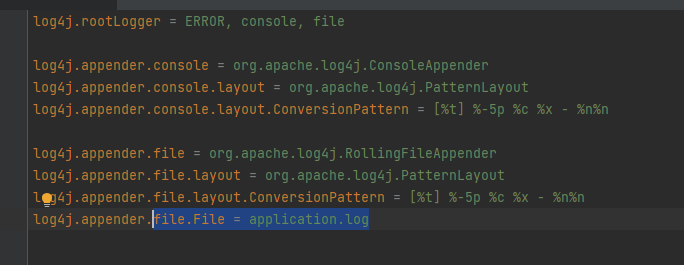
* Всеки следващ изглед е описан в папка ресурси, presentation.view и е с разширение .fxml.
* Всеки бутон си има handler, който извършва валидациите на данните, взети от текстовите полета на диалоговия прозорец и при правилно въведени данни, се изпращат към базата.
* Всеки прозорец си има контролер, който си отговаря на съответните събития, за които е създаден. В случаят за логване на администратора, прозореца си има контролер, който handle-ва текстовите полета и бутоните.
* Пример ще дадем с бутона „Log In” за диалога при логване на администратор.





1. **Реализация на модул за регистриране на събитията.**

* За регистрирането на събития, генерирани от приложението, се използва **библиотеката** **Log4J**. Тя извежда съобщения на конзолата и също така ги записва в текстов файл, който се намира в папката на приложението, с наименование „**application**”.



* С помощта на тази библиотека можем лесно да проследим, през какви проблеми преминава приложението ни, а най-важното е, че можем да видим, какви грешки ни хвърля то.