

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНA

Факултет: ФИТА  
Катедра: КНТ  
Специалност: КСТ

КУРСОВ ПРОЕКТ

Разработване на информационна система, предоставяща услуга за пътнически превози.

|  |  |
| --- | --- |
| Изготвили: Айджан Хасанов Ахмедов ФН: 18621409  Николай Деянов Иванов ФН: 18621408 | Проверил:  доц. д-р. инж. Хр.Ненов |

1. Пълно задание на проекта:

Да се разработи информационна система, предоставяща услуга пътнически превози. Програмата съхранява и обработва данни за разпространение на билети.

Системата подържа 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 пътнически компании от администратор
* Създаване на разпространител от администратор
* Създаване на касиери от разпространител
* Поддържане на профили с характеристики на клиенти 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, видове 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 закупувана (информация за купувача, избор на място и др.)

Системата поддържа Справки по произволен период за:

* Клиенти 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 имат право на справки само за пътувания чиито билето-разпространители са

Системата подържа известия за:

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

2. Анализ на продукта

2.1 Функционални изисквания

Анализ на клиентите:  
Системата поддържа 3 вида клиента: компания, дистрибутор и касиер. Като за всеки клиент системата предлага набор от различни възможности за работа.

Администратор: Администраторът в системата е един, той има правото да добавя нови компании и нови дистрибутори в системата, както и да редактира техните данни по всяко време

Компания: Компанията в системата има възможност да създава нови пътувания и да ги променя при необходимост. Като за всяко ново пътуване е необходимо да попълни данни като: вид на пътуването, дата и час на заминаване, маршрут по който минава и видове транспорт. Има възможност да преглежда получените заявки от дистрибутор съответно да ги приема или да ги отказва. Получава справка с продадените билети за своите пътувания.

Разпространител: Разпространителят има свързваща роля в система като предоставя възможност за продаване на билети на своите касиери. Касиерите на даден разпространител могат да продават билети за предстоящо пътуване когато компанията собственик на пътуването одобри заявката на разпространителя. Разпространителят има право да добавя касиери в системата както и да ги модифицира. Той може да вижда всички предстоящи пътувания и да заявява желание за разпространяване на билетите им.

Касиер: Касиерът е крайният потребител в системата неговата главна цел е да продава билети за различни пътувания. Касиерът също разполага със справка за пътувания като по този начин избира за кое пътуване да продаде билет.

2.2 Структура на проекта



**Моделът (Model)** е централен компонент в шаблона. Това динамичната структура от данни на приложението, независима от потребителския интерфейс. Моделът управлява данните, логиката и правилата на приложението.

**Изгледът (View)** е изходящият поток от информация (това, което приложението изпраща като отговор до дисплея, респективно – до потребителя, в следствие на неговата заявка). Възможни са няколко различни изгледа на една и съща информация, като например различни диаграми за мениджмънт на даден ресурс или различни таблици.

**Контролерът (Controller)** е третата част от този шаблон. Той приема потребителския вход (т.е. данните, които потребителя въвежда, неговите заявки и т.н.) и ги преобразува в команди към модела или изгледа.

2.3 Дефиниции на модулите в системата

Repository: осъществява връзка с базата данни посредством Entity обекти, извиква се само от Service(Business logic) модула на системата.

Service: модулът, който съдържа бизнес логиката на системата.

Controller: този модул управлява всички визуални елементи в приложението и тяхната интеракция с бизнес слоя

3. Проектиране на системата

3.1 Проектиране на отделните модули

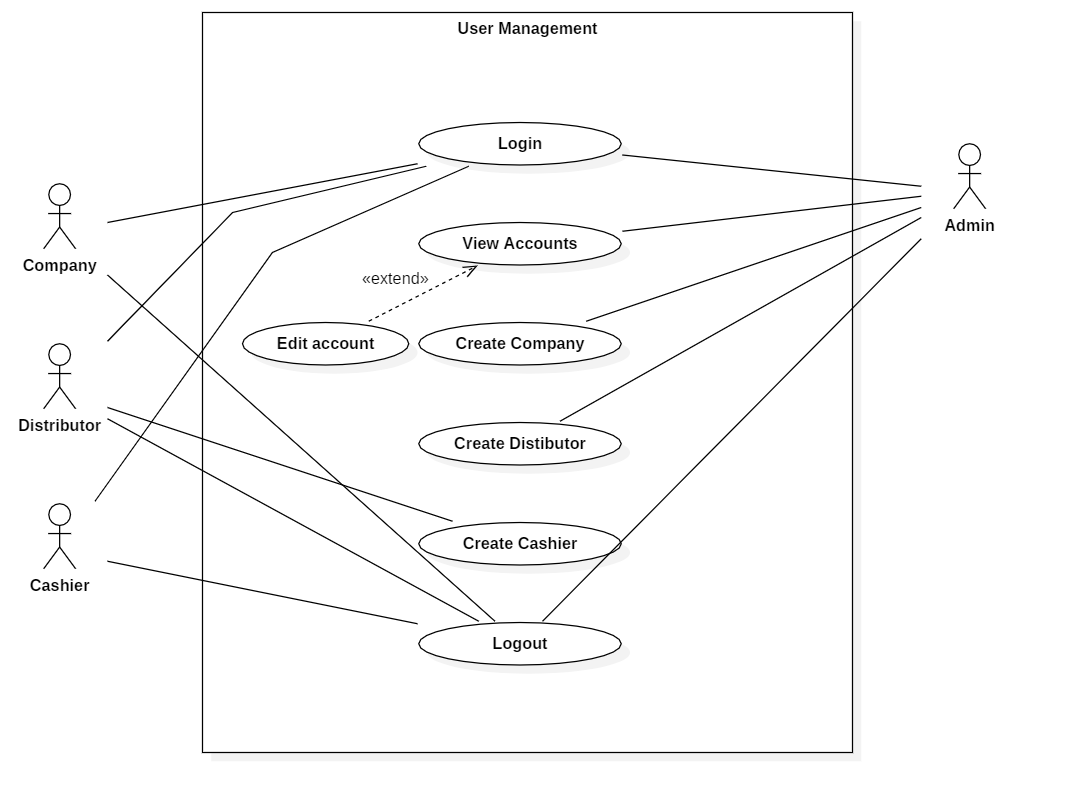
1. База от данни:

* Postgres като база от данни.
* Hibernate за реализация на даннов слой и комуникация с базата.

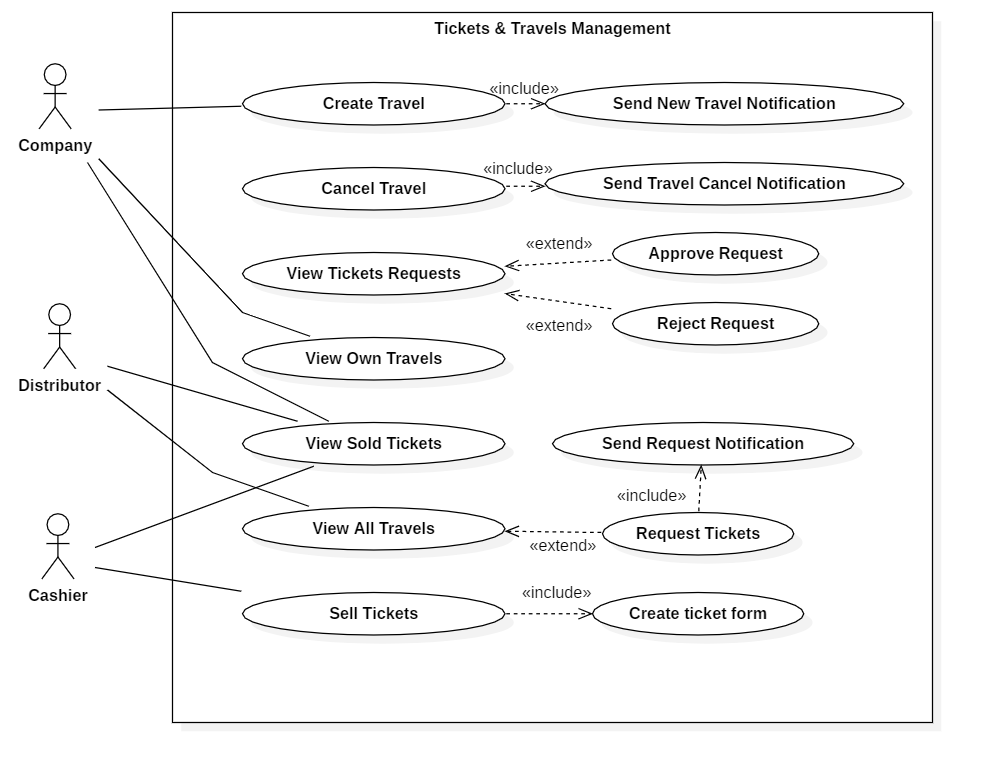
1. GUI: JavaFX 15 с fxml модула
2. Бизнес логика: Пакетиране на бизнес логиката в service пакет.

3.2 Диаграми

Управление на клиентите в системата – Use Case

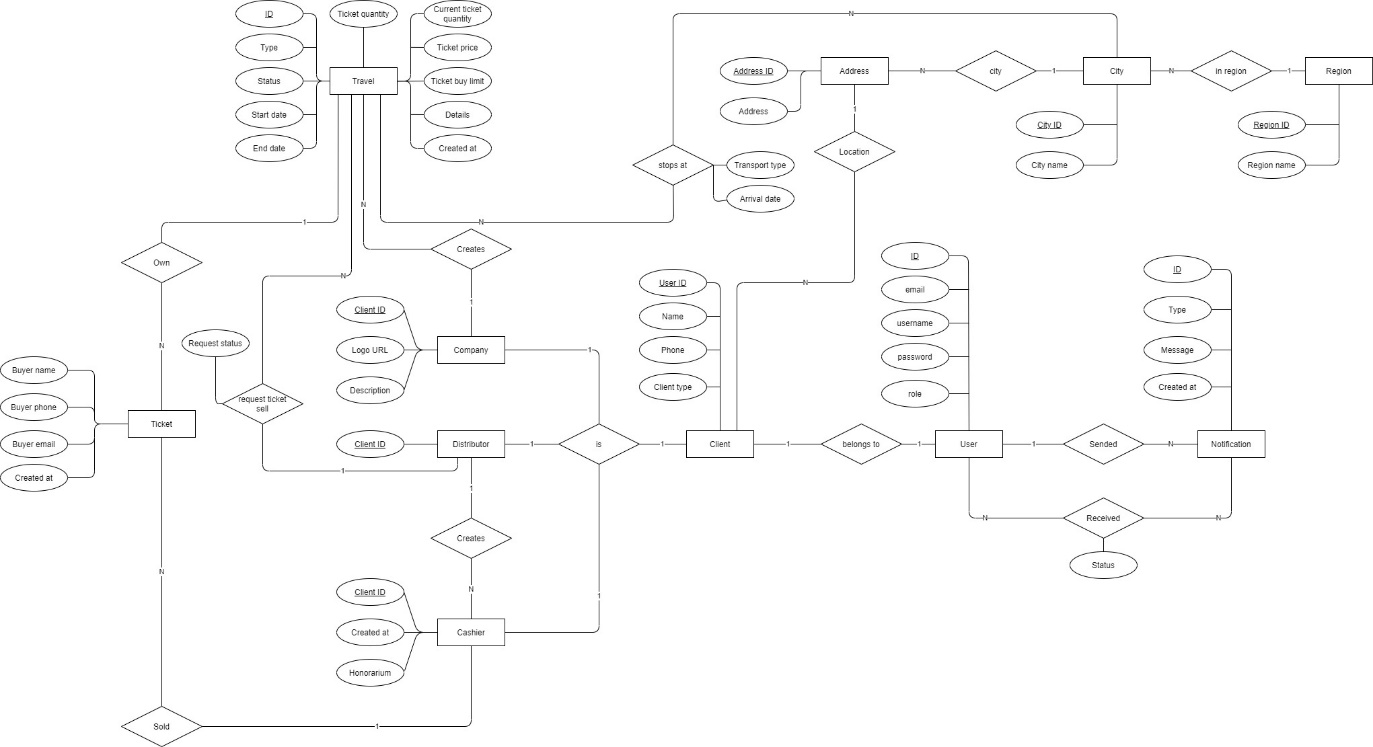


Управление на билети и пътувания – Use Case



3.3 Концептуален модел на базата данни

[ER диаграма](https://i.imgur.com/P5G24YX.jpeg) – модел на чен



4. Реализация на системата

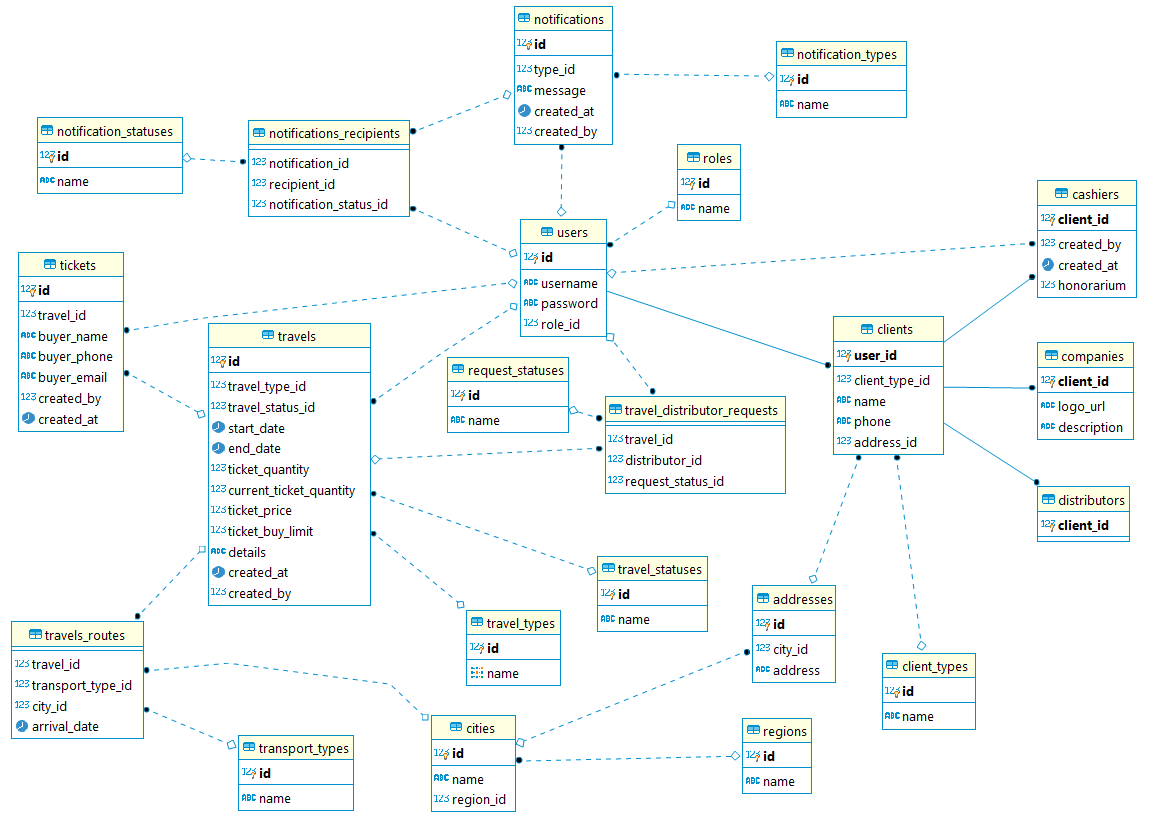
Използва се gradle, като инструмент за сглобяване на проекта.

Класовете в проекта са разделени в пакети, които дефинират различните слоеве или помощни класове:

* Common: съдържа общи класове, които се използват на много места в проекта, като: общи константи, конфигурация на проекта (AppConfig.java), поддържани езици за локализация (SupportedLanguage.java), както и класове, държащи информация за екраните и менюто.
* Control: съдържа класове, които наследяват javaFX компоненти и разширяват техните функционалности.
* Controller: съдържа всички контролер класове.
* Entity: съдържа всички ентитита.
* Repository: съдържа всички класове и методи, които комуникират с базата.
* Schedule: съдържа класове, които имплементират Runnable и тяхната задача е да изпълняват ендо нещо периодично (като проверки за известия).
* Service: съдържа всички класове, които дефинират бизнес логиката на системата.
* Util: съдържа помощни класове.

4.1 Реализация на базата от данни

Релационна диаграма:



4.2 Реализация на слоя за работа с базата данни

За работа с базата данни се използва Hibernate като имплементация на JPA. Всички класове, които комуникират с базата са в repository пакета и имплементират singleton pattern.

Дефиниран е базов клас (GenericCrudRepository), който съдържа основни (CRUD – create, read, update, delete) методи.

Създадени са 2 базови класа за ентитита:

1. BaseEntity: базово зласа всички ентитита, които имат id.
2. BaseAuditAbstractEntity: базов клас за всички ентитита, които имат created\_by и created\_at колона, имплементира и метод, който слуша за PrePersist Event и попълва полетата на класа преди запис.

Дефинирани са 2 помощни класа за работа с Hibernate:

1. EntityManagerUtil: съдържа помощни методи за работа с EntityManager.
2. JpaOperationsUtil: съдържа помощни методи за работа с JPA (Hibernate), като работа с транзакции.

За релациите ‘към много’, които съдържат допълнителни колони, са създадени отделни ентитита (NotificationRecipient, TravelDistributorRequest, TravelRoute).

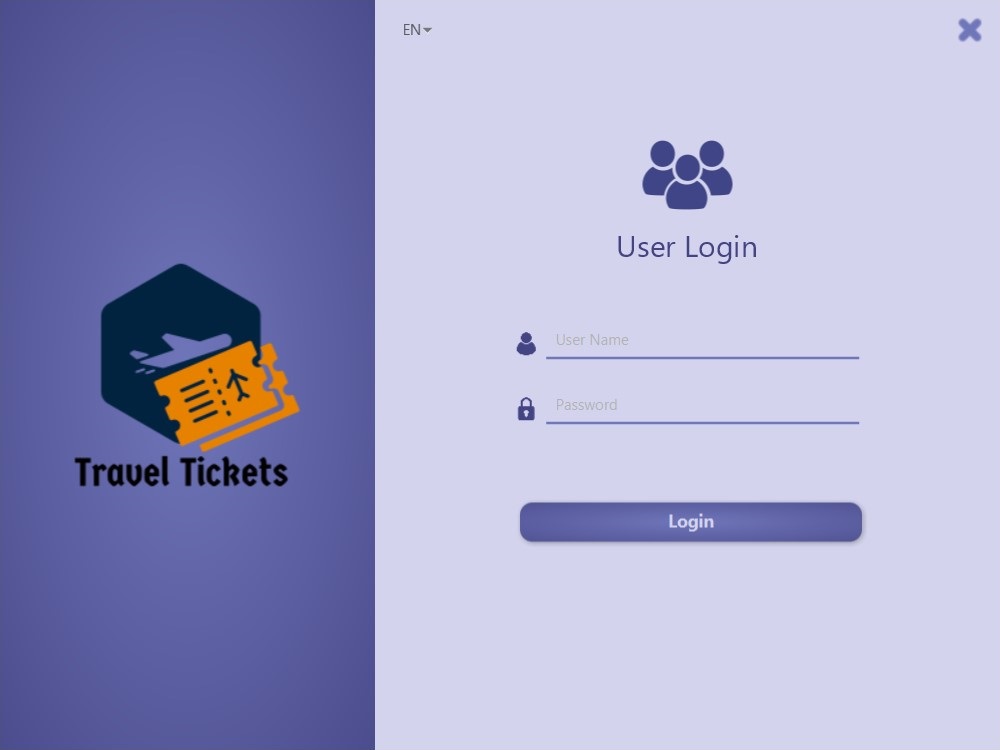
Създадено е ентити за клиента, което се наследява от всички клиент ентитита (Distributor, Company, Cashier).

4.3 Реализация на бизнес логика и графичен интерфейс

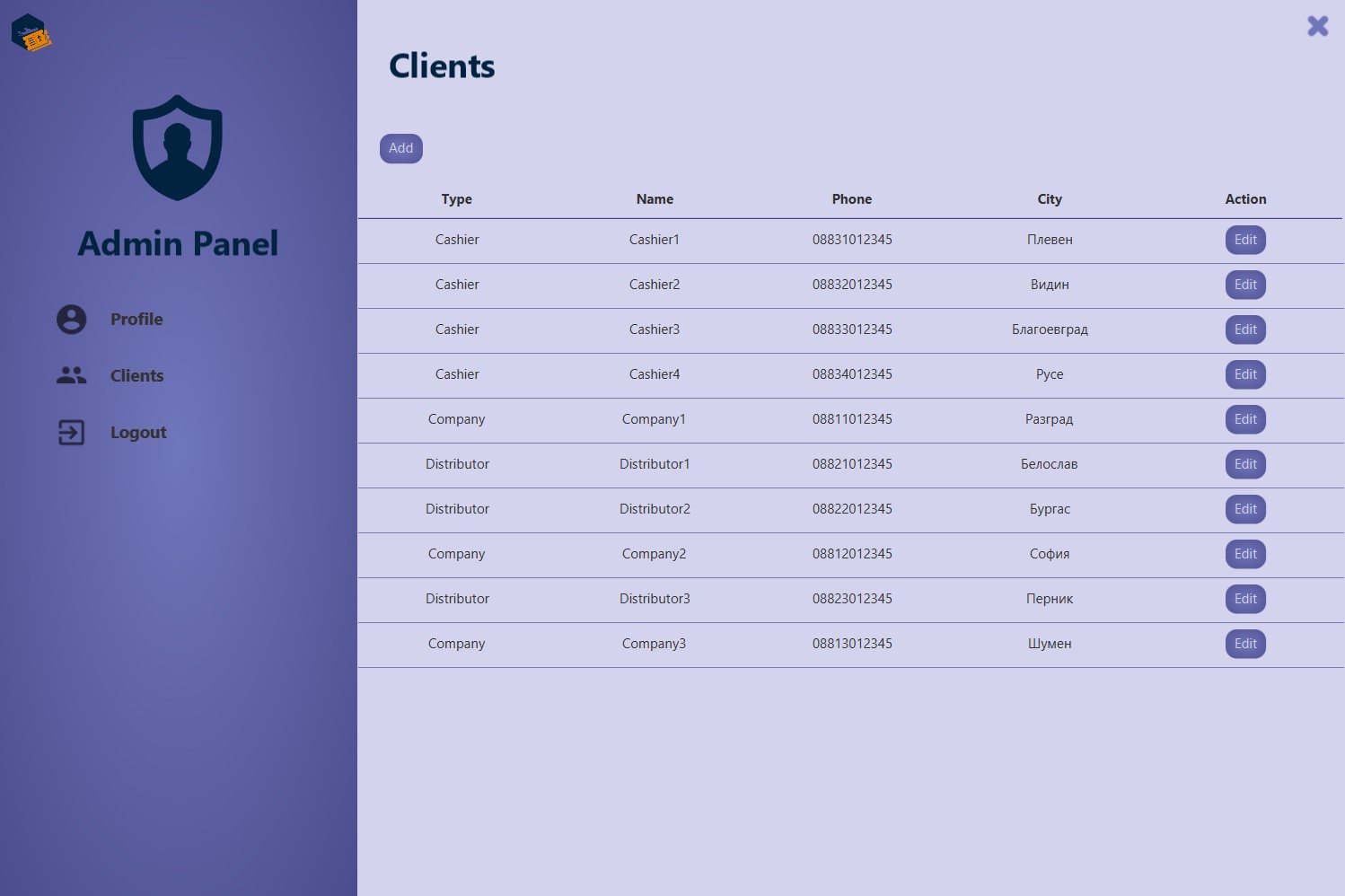
Всички класове за бизнес логиката са в service пакета и имплементират singleton pattern.

Графичният интерфейс на приложението позволява на потребителя да избира между 2 езика Български и Английски.  
Изборът се осъществява от главния екран за вход.

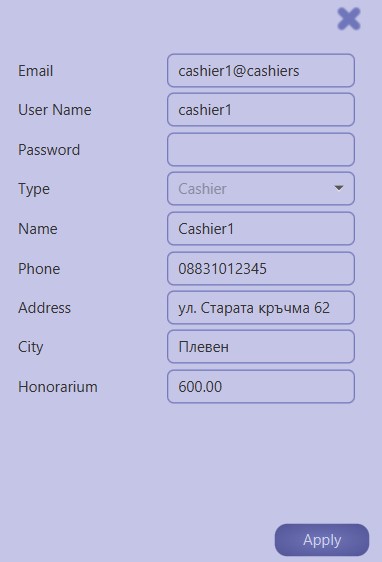
Главен екран за вход:



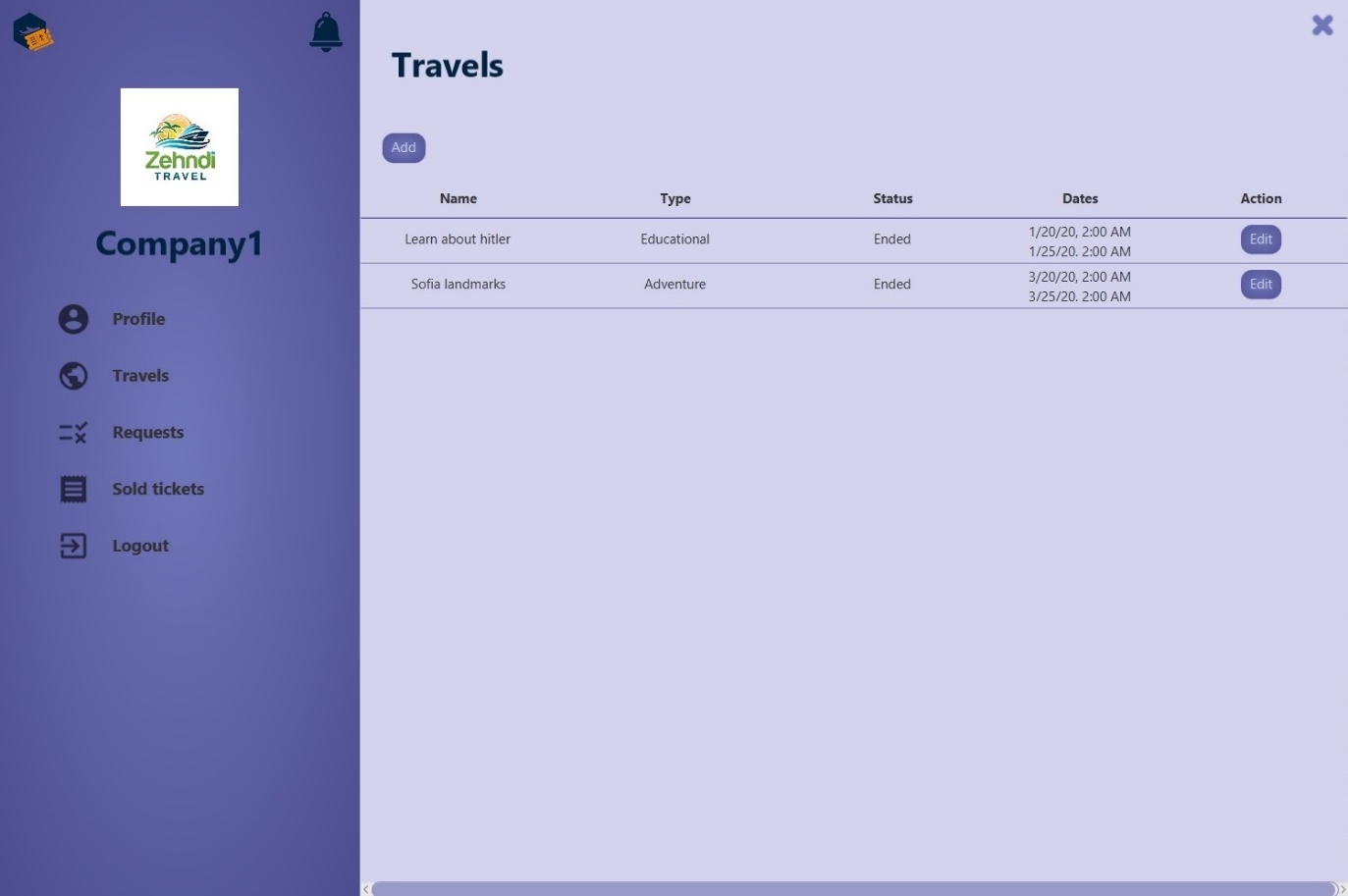
Изглед клиенти, позволява създавана и редактирана на клиенти:



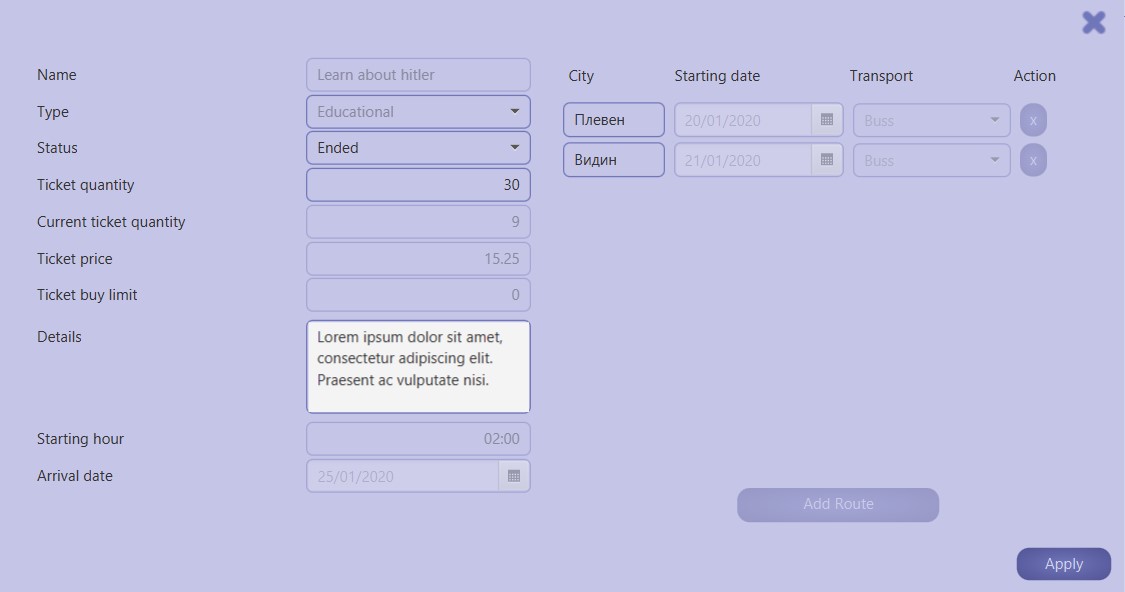
Диалог за визуализиране на данните за съответния клиент:



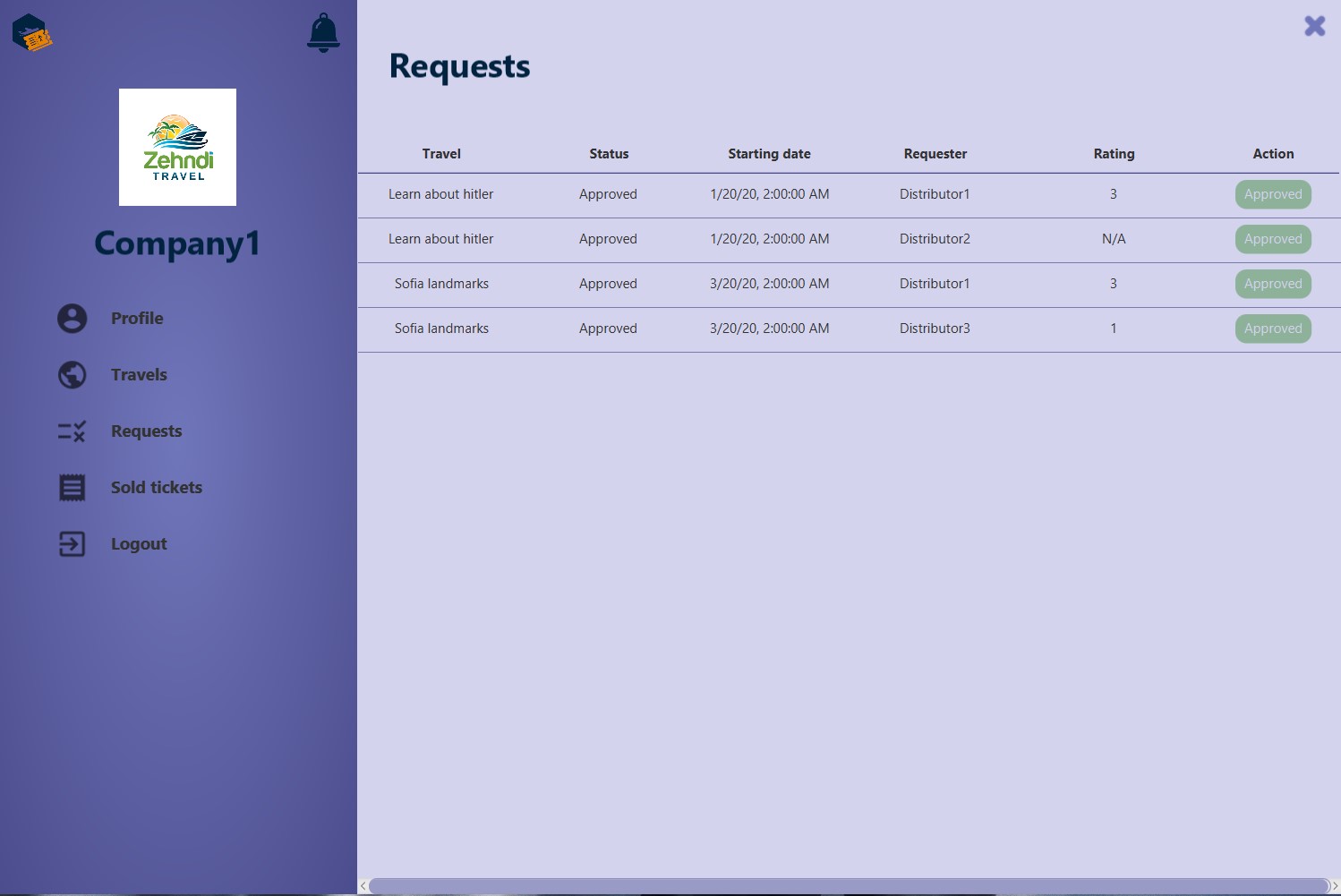
Изглед пътувания, предоставя различни възможности в зависимост от типа на вписания клиент:



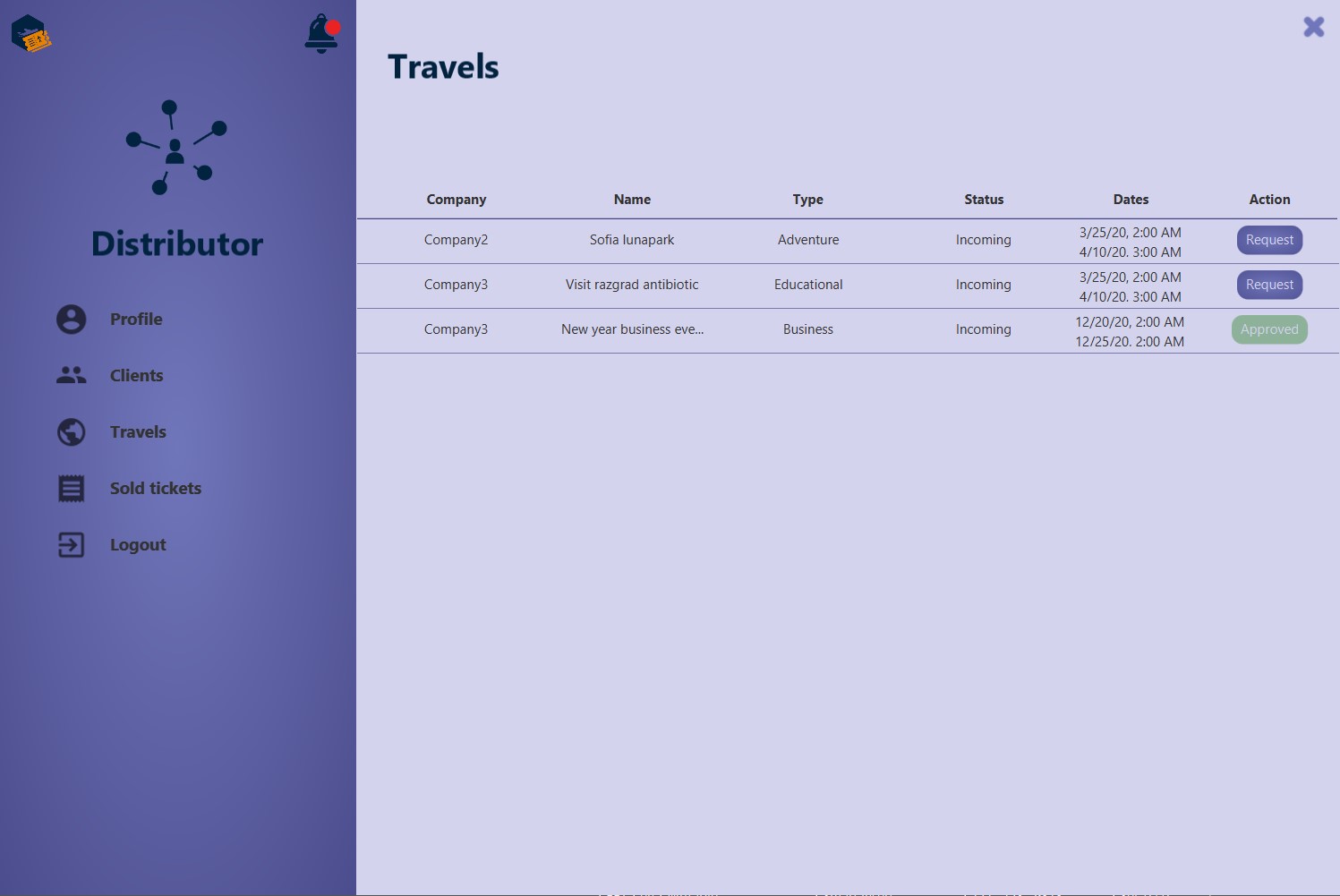
Диалог за добавяне ново пътуване:



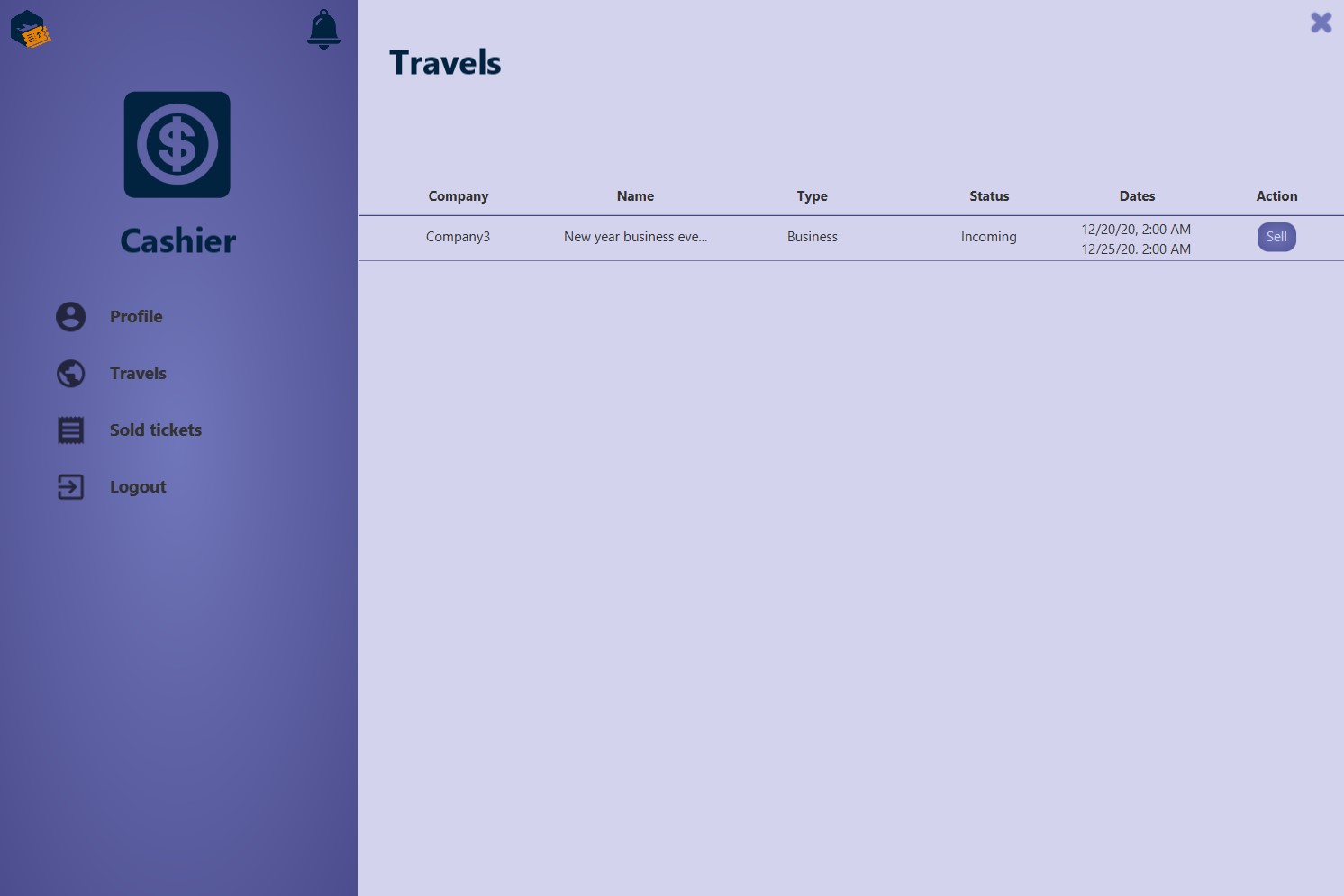
Изглед заявки предоставя възможност за преглеждане на получените заявки и съответно одобряване или отказване:



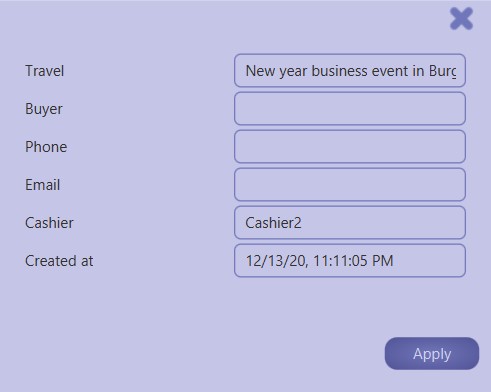
Изглед пътувания за разпространител позволява пращане на заявки:



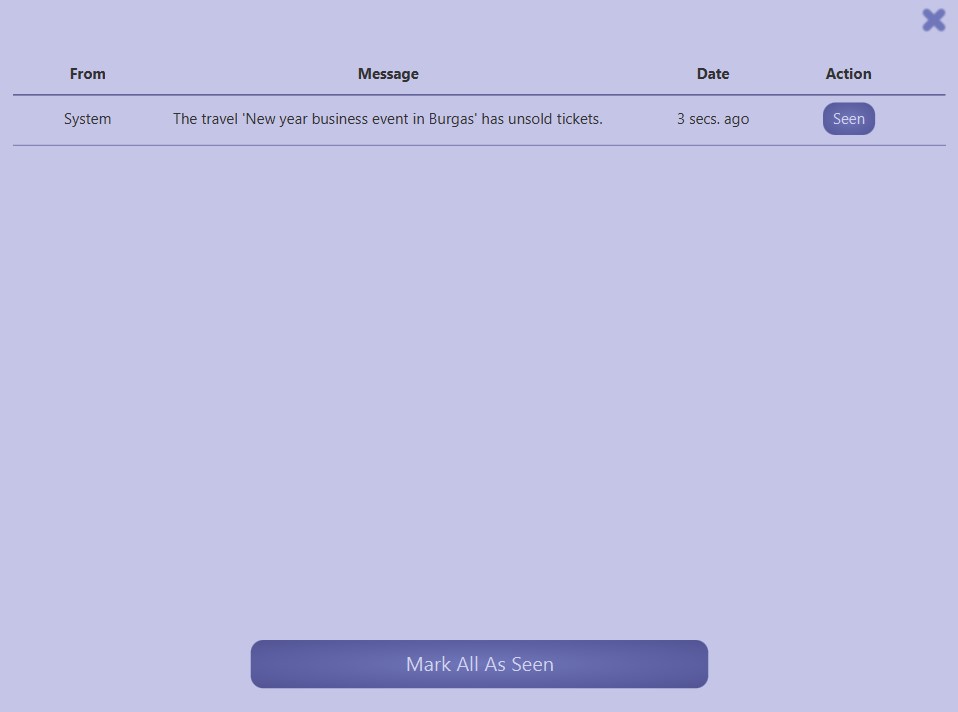
Изглед пътувания за касиер позволява продаване на билети за избрано пътуване



Диалог за продаване на билет:



Диалог за визуализиране на известията, предлага възможност за маркиране на всички известия като прочетени:



4.4 Реализация на модул за регистриране на събития в системата - log4j

Използва се log4j за регистриране на събития в системата, логовете се зписват във файл, /logs/travel\_tickets.log, като има логове от всеки слой (даннов, бизнес, gui).

Пример за логове:

