МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

Специализация 1–40 01 01 Программное обеспечение информационных технологий (программирование интернет – изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Система грузоперевозок»

Выполнил студент Мозолевский Александр Дмитриевич

(Ф.И.О.)

Руководитель проекта ст. преп, Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант:  ст. преп, Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преп, Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

Реферат

Пояснительная записка курсового проекта содержит 30 страниц пояснительной записки, 10 иллюстраций, 5 источников литературы, 6 приложений.

JAVA, SPRING FRAMEWORK, ORACLE 21C, SPATIAL AND GRAPH, INSOMNIA REST, TYPESCRIPT, ANGULAR, JSON IMPORT, JSON EXPORT.

Цель курсового проекта: разработка базы данных для системы грузоперевозок с применением технологии Oracle Spatial and Graph.

Первая глава описывает похожие сервисы, а также их недостатки, которые необходимо устранить в приложении.

Вторая глава описывает процесс разработки и настройки базы данных, с которой будет происходить взаимодействие.

В третьей главе приведены тестирование и способы оптимизации запросов.

Четвёртая глава описывает изученные и использованные в процессе разработки технологии.

В пятой главе приведено руководство пользователя по взаимодействию с приложением для каждой из ролей.

В заключении приведены результаты проделанной работы.

Abstract

The course project explanatory work consists of 30 pages of explanatory note, 10 illustrations, 5 literature sources, 6 appendixes.

JAVA, SPRING FRAMEWORK, ORACLE 21C, SPATIAL AND GRAPH, INSOMNIA REST, TYPESCRIPT, ANGULAR, JSON IMPORT, JSON EXPORT.

The goal of course project is development of a database for freight transportation system using Oracle Spatial and Graph technology.

The first chapter describes similar services and their shortcomings that need to be removed in the application.

The second chapter describes the process of developing and configuring the database with which the interaction will take place.

The third chapter provides testing and ways to optimize queries.

The fourth chapter describes the technologies, that studied and used in the development process.

The fifth chapter provides a user guide for interacting with the application for each of the roles.

In conclusion, the results of the work are presented.

**Содержание**

[Введение 5](#_Toc121913750)

[1 Аналитический обзор литературы 6](#_Toc121913751)

[1.1 Аналитический обзор источников 6](#_Toc121913752)

[1.2 Обзор аналогов 6](#_Toc121913753)

[2 Разработка программной системы 8](#_Toc121913754)

[2.1 Таблицы базы данных 8](#_Toc121913755)

[2.2 Роли и пользователи базы данных 11](#_Toc121913756)

[2.3 Процедуры и функции для поставленных задач 12](#_Toc121913757)

[2.3.1 Пакет для покупателя 12](#_Toc121913758)

[2.3.2 Пакет для менеджера 14](#_Toc121913759)

[2.3.3 Пакет для водителя 15](#_Toc121913760)

[2.4 Триггеры базы данных 16](#_Toc121913761)

[2.5 Представления базы данных 17](#_Toc121913762)

[3 Тестирование и улучшение производительности 18](#_Toc121913763)

[4 Описание технологии 19](#_Toc121913764)

[4.1 Импорт пространственных данных 19](#_Toc121913765)

[4.2 Просмотр пространственных данных 20](#_Toc121913766)

[5 Руководство пользователя 21](#_Toc121913767)

[Заключение 24](#_Toc121913768)

[Список литературы 25](#_Toc121913769)

[Приложение А 26](#_Toc121913770)

[Приложение Б 27](#_Toc121913771)

[Приложение В 28](#_Toc121913772)

[Приложение Г 29](#_Toc121913773)

[Приложение Д 30](#_Toc121913774)

[Приложение Е 31](#_Toc121913775)

Введение

Целью данной работы является разработка приложения с реляционной базой данных, предназначающейся для пользования компании грузоперевозок. База данных должна быть составлена с целью возможности пользования ей пользователями, водителями и администраторами с разным уровнем доступа к встроенным функциям.

База данных – организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных – база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Oracle 12c, в связи с её высокой производительностью и надёжностью.

Также необходимо разработать приложение, для демонстрации работы базы данных. Для разработки серверной части приложения используется язык программирования Java и фреймворк Spring. Java – строго типизированный язык программирования общего назначения. Spring Framework – универсальный фреймворк с открытым исходным кодом для Java-платформы.

Для разработки клиентской части приложения используется язык программирования TypeScript и фреймворк Angular. TypeScript – язык программирования для веб-разработки, основанный на JavaScript. Angular – фреймворк, написанный на TypeScript от компании Google, используемый для создания масштабируемых веб-приложений.

Для взаимодействия приложения с базой данных используется технология JDBC. JDCB – платформенно независимый промышленный стандарт взаимодействия Java-приложения с различными СУБД, реализованный в виде пакета Java.

В основной части будут затронуты аспекты разработки проекта и обоснованны технические приёмы, которые используются в приложении.

Для обеспечения безопасности пользователей приложения в проекте используется технология хэширования паролей от аккаунта перед записью в базу данных.

Основные требования к приложению:

* хранение информации;
* дополнение информации;
* реализация трех ролей: пользователя, водителя и администратора;
* поиск информации по уровню доступа роли;
* взаимодействие с базой данных с помощью хранимых процедур;
* нахождение наиболее оптимального маршрута доставки.

В пояснительной записке находится информация об аналогах, архитектуре и реализации проекта, а также руководство пользователя.

1 Аналитический обзор литературы

1.1 Аналитический обзор источников

В ходе подготовки пояснительной записки была изучена специальная техническая, учебно-методическая и справочная литература, статьи и материалы, опубликованные в сети интернет.

Основная информация о работе с фреймворком Spring была получена из официальной документации.

Принципы работы с SQL были получены из статьи «Подключение к базе данных». В статье было рассмотрено подключение необходимых библиотек, работа с SQL.

Информация о работе с технологией Spatial And Graph была получения из официальной документации Oracle.

1.2 Обзор аналогов

Система грузоперевозок позволяет совершать перемещение каких-либо объектов (груза) с помощью транспорта. В наше время различная деятельность разных компаний не обходится без очень тесного сотрудническая с компаниями, которые занимаются грузоперевозками.

Для создания нового решения в виде программного продукта, используемого для системы грузоперевозок, необходимо проанализировать уже существующие программные средства в данной сфере.

Анализ достоинств и недостатков этих сервисов позволит сформировать требования к проектируемому программному средству. В качестве исследуемых аналогов были выбраны программные продукты, которые можно отнести к сервисам по доставке товаров. Все подходящие сервисы были найдены в сети Интернет.

Одним из основных сервисов по оптимизации путей является сервис Relog, основная страница которого представлена на рисунке 1.1.

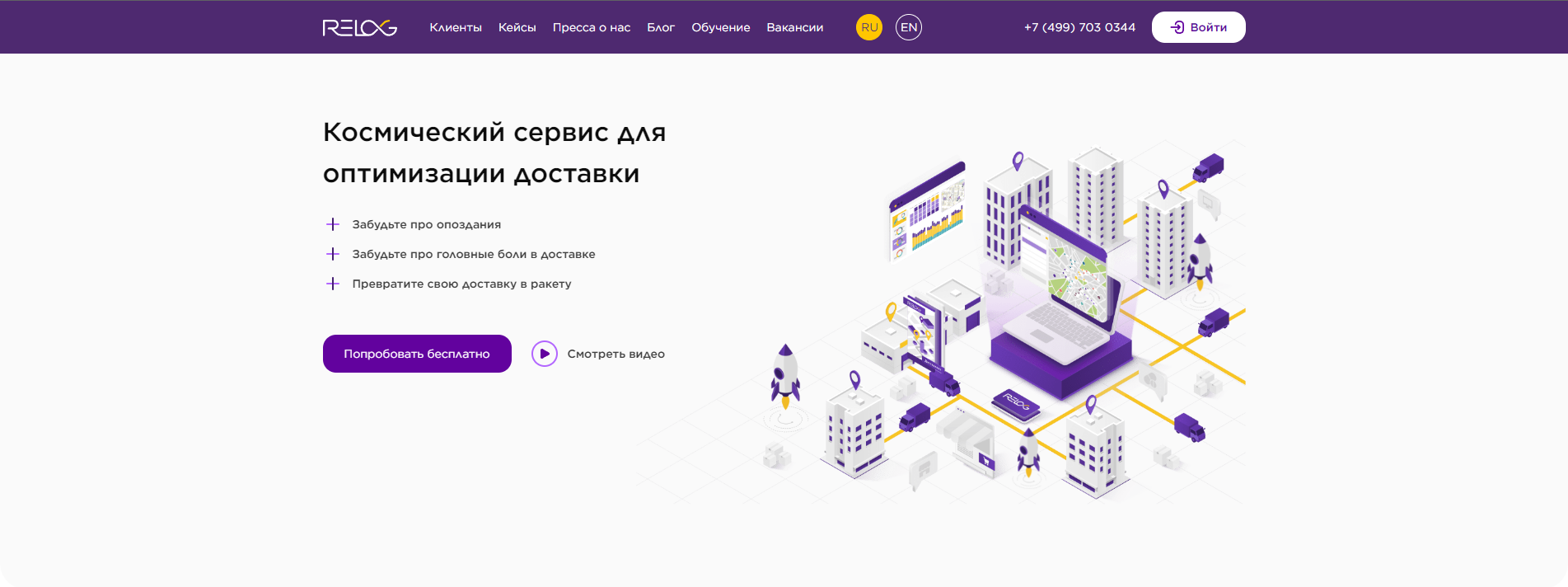


Рисунок 1.1 – Домашняя страница Relog

С помощью этого сервиса пользователь может указать в своем личном кабинете исходный пункт и пункт назначения. Затем сервис подберёт наиболее оптимизированные маршруты для данных пунктов.

Самым популярным сервисом грузоперевозок в Беларуси является Transinfo, домашняя страница которого представлена на рисунке 1.2.

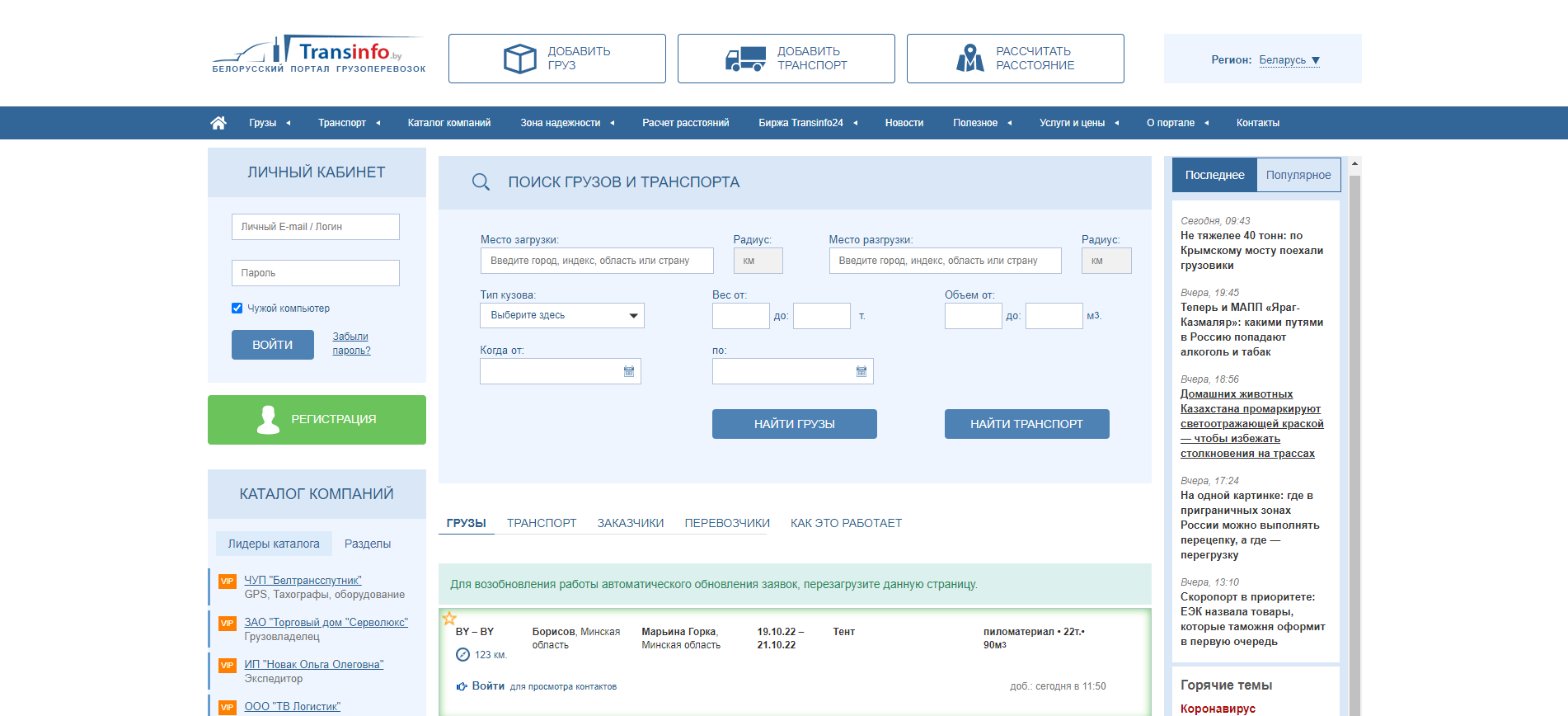


Рисунок 1.2 – Домашняя страница Transinfo

Данный сервис позволяет пользователям заказывать перевозку товаров с указанием начального и конечного пунктов. В зависимости от расстояния перевозки, массы и объема будут предложены возможные варианты доставки товара, которые может выбрать пользователь. Пример вариантов доставки представлен на рисунке 1.3.

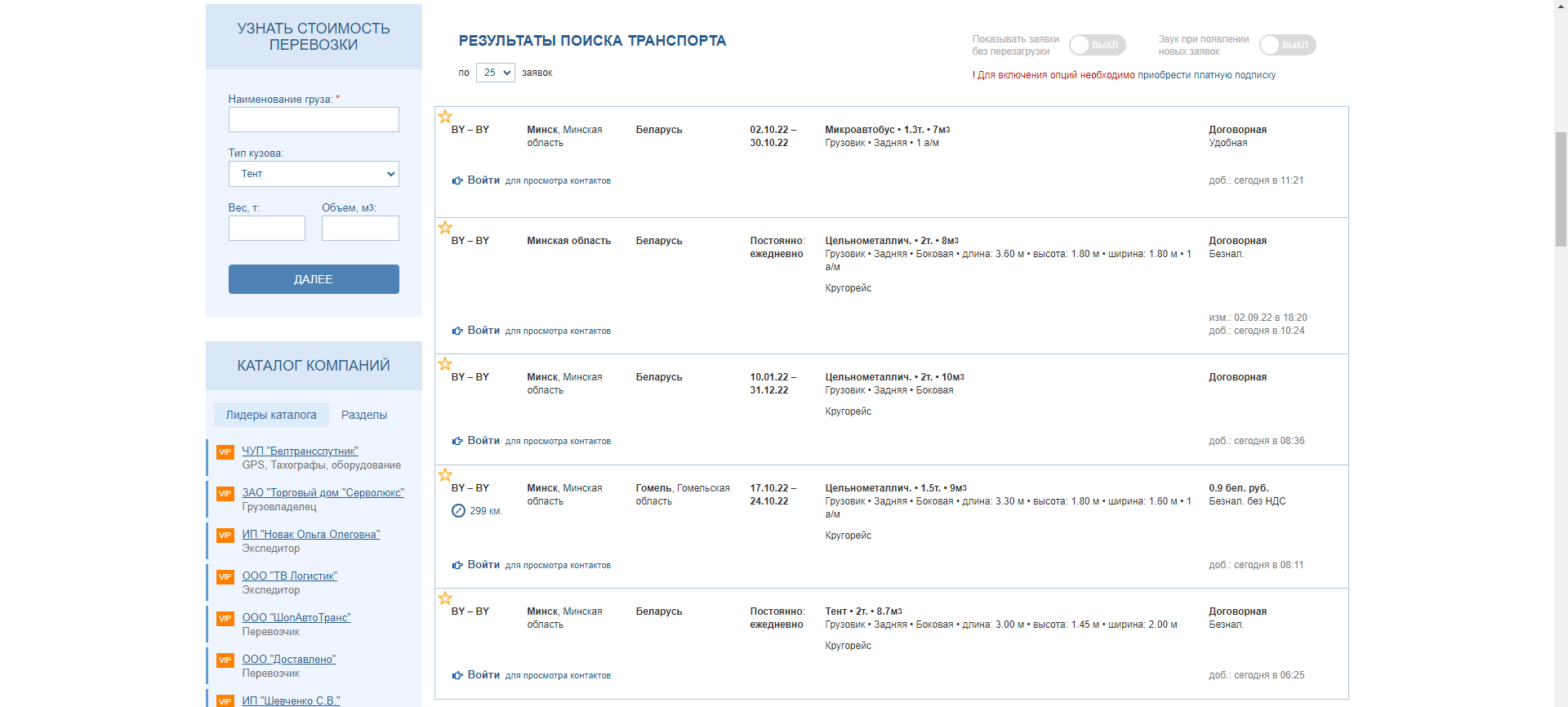


Рисунок 1.3 – Пример вариантов доставки

Из недостатков стоит отметить, что пользователю не предоставляется информация о водителе и он не может оценить его работу.

2 Разработка программной системы

При разработке приложения для курсового проекта была использована база данных Oracle 21c.

2.1 Таблицы базы данных

Для реализации базы данных системы грузоперевозок было разработано 12 таблиц, диаграмма со структурой связей которых представлена в приложении А.

При разработке приложение получило рабочее название «Deliverty». В структуру базы данных входят следующие таблицы: boundaries, cities, roads, user\_roles, users, drivers, drivers\_ratings, order\_statuses, orders, order\_items, user\_notifications, admin\_notifications. Далее представлено описание каждой из них.

Таблица boundaries используется для хранения полигонов, которые представляют собой границы стран, а также их названия. Её структура представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы boundaries

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| name | varchar2(50) | Not null |
| boundary | sdo\_geometry | Not null |

Таблица cities используется для хранения географических точек, координаты которых соответствуют городам, а также их названия. Её структура представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы cities

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| name | varchar2(50) | Not null |
| point | sdo\_geometry | Not null |

Таблица roads используется для хранения ломанных, координаты которых соответствуют дорогам стран. Её структура представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы roads

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| line | sdo\_geometry | Not null |

Таблица user\_roles является хранилищем ролей, которое содержит их названия. Её структура представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы user\_roles

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| name | varchar2(10) | Not null |

Таблица users используется для хранения информации о пользователях, а именно их логина, хэше пароля, имени, фамилии, телефона и роли. Её структура представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы users

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| username | varchar2(50) | Unique, not null |
| password\_hash | varchar2(300) | Not null |
| first\_name | varchar2(30) |  |
| last\_name | varchar2(30) |  |
| telephone | varchar2(30) |  |
| user\_role\_id | number | Foreign key |

Таблица drivers используется для хранения информации о водителях, а именно их логина, хэше пароля, имени, фамилии, телефона и рейтинга. Её структура представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы drivers

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| username | varchar2(50) | Unique, not null |
| password\_hash | varchar2(300) | Not null |
| first\_name | varchar2(30) |  |
| last\_name | varchar2(30) |  |
| telephone | varchar2(30) |  |
| rating | number(3, 1) |  |

Таблица drivers\_ratings используется для хранения отзывов от покупателей, а именно о самом отзыве и оценке работы. Её структура представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы drivers\_ratings

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| driver\_id | number | Foreign key |
| rating | number | Not null |
| message | varchar2(300) | Not null |

Таблица order\_statuses является хранилищем статусов заказа, которое содержит их названия. Её структура представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы order\_statuses

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| status | varchar2(20) | Not null |

Таблица orders используется для хранения заказов пользователей, содержит следующую информацию: идентификатор пользователя, который создал заказ, идентификатор водителя, который исполняет заказ, идентификатор статуса заказа, начальный город, конечный город, цена, дата создания заказа, дата выполнения заказа. Её структура представлена в таблице 2.9.

Таблица 2.9 – Структура таблицы orders

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| user\_id | number | Foreign key |
| driver\_id | number | Foreign key |
| order\_status\_id | number | Foreign key |
| start\_point | varchar2(50) | Not null |
| end\_point | varchar2(50) | Not null |
| price | number(10, 2) |  |
| start\_date | date | Not null |
| end\_date | date |  |

Таблица order\_items используется для хранения информации о грузе, а именно его названии, весе, объеме и идентификаторе заказа, которому он принадлежит. Её структура представлена в таблице 2.10.

Таблица 2.10 – Структура таблицы order\_items

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| order\_id | number | Foreign key |
| name | varchar2(50) | Not null |
| weight | number | Not null |
| volume | number | Not null |

Таблица user\_notifications используется для хранения уведомлений для покупателей, а именно даты создания уведомления, которое принадлежит покупателю и сообщения. Её структура представлена в таблице 2.11.

Таблица 2.11 – Структура таблицы user\_notifications

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| operation\_date | date | Not null |
| message | varchar2(1000) | Not null |
| user\_id | number | Foreign key |

Таблица admin\_notifications используется для хранения уведомлений для менеджера, а именно даты создания уведомления и сообщения. Её структура представлена в таблице 2.12.

Таблица 2.12 – Структура таблицы admin\_notifications

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| id | number | Primary key |
| name | varchar2(50) | Not null |
| point | sdo\_geometry | Not null |

2.2 Роли и пользователи базы данных

База данных содержит 4 основных пользователя: programmer, customer, manager и driver. Пользователь programmer используется для создания инфраструктуры и объектов на подключаемой базе данных. Остальные пользователи могут взаимодействовать с базой данных благодаря полномочий на вызов процедур и функций из указанных пакетов, которые описаны в главе 2.3.

Каждый пользователь имеет профиль, который представлен в листинге 2.1.

|  |
| --- |
| create profile users\_profile limit  password\_life\_time 180  sessions\_per\_user 20  failed\_login\_attempts 5  password\_lock\_time 60  password\_reuse\_time 10  connect\_time 180  idle\_time 60; |

Листинг 2.1 – Профиль для всех пользователей

Для каждого пользователя создается его собственная роль. Роль – поименованный набор привилегий. Роль для пользователя customer представлен в листинге 2.2.

|  |
| --- |
| create role customer\_role;  grant connect to customer\_role;  grant execute on programmer.user\_package to customer\_role; |

Листинг 2.2 – Создание роли для пользователя customer

После создания роли и назначения ей привилегий, необходимо создать пользователя и присвоить ему роль, созданную ранее. Создание пользователя customer приведено в листинге 2.3.

|  |
| --- |
| create user customer  identified by qwerty11  default tablespace course\_ts  profile users\_profile  quota unlimited on course\_ts;  grant customer\_role to customer; |

Листинг 2.3 – Создание пользователя customer

Остальные пользователи базы данных создаются по тому же принципу, но их ролям присваивается привилегия на вызов процедур и функций из других пакетов.

2.3 Процедуры и функции для поставленных задач

Для управления данными, пользователи вызывают процедуры и функции, которые располагаются в пакетах. Пакет Oracle PL/SQL – это объект БД, который группирует логически связанные типы, элементы и подпрограммы.

В базе данных используется три пакета: user\_package, admin\_package и driver\_package, описание которых представлено ниже, графическое описание функций приведено в приложении Б.

2.3.1 Пакет для покупателя

Пакет user\_package содержит 15 процедур и функций, предназначенных для взаимодействия с базой данных на уровне покупателя, описание этого пакета приведено в приложении В.

Функция encrypt\_password позволяет зашифровать пароль пользователя. Она принимает один параметр: пароль пользователя. В данном курсовом проекте используется кодировка base64.

Процедура register\_user выполняет регистрацию пользователя. Эта процедура принимает два параметра: пароль и логин пользователя. В случае, если пользователь с таким логином уже существует, вызывается и обрабатывается исключение. В случае уникальности логина, пароль преобразуется с помощью вышеуказанной функции encrypt\_password и заносится в таблицу users.

Для авторизации пользователя в приложении разработана функция login\_user. У неё есть два входных параметра: логин и пароль пользователя. Если указанная комбинация логина и пароля отсутствует в таблице users, то вызывается исключение. Если исключение не было вызвано, то в курсор заносится логин и название роли, которые извлекаются из соединения таблиц users и user\_roles.

Заполнение данных происходит путём вызова процедуры fulfill\_user\_data, которая принимает следующие параметры: логин, имя, фамилия и телефон пользователя. Процедура находит строку пользователя в базе данных по его логину, в случае успешного обнаружения изменяет остальные его столбцы на указанные в процедуре. В случае отсутствия пользователя с таким логином вызывает исключение.

Чтобы получить данные пользователя необходимо вызвать функцию get\_user\_data, которая принимает логин пользователя. Функция находит строку пользователя в таблице users по его логину и заносит в курсор его логин, имя, фамилию и телефон. В случае отсутствия пользователя с таким логином вызывает исключение.

Процедура add\_item\_to\_order принимает следующие параметры: идентификатор заказа, название, вес и объем груза. Процедура обрабатывает исключение, которое может возникнуть в случае передачи неверного идентификатора заказа. Если заказ, с указанным идентификатором существует, то данные, переданные через параметры, будут занесены в таблицу order\_items.

Для создания нового заказа используется процедура create\_order. Она принимает следующие параметры: логин пользователя, начальная и конечная точки заказа. После чего соответствующие данные заносятся в таблицу orders.

Информация о заказе заносится в таблицу orders, при этом в столбец start\_date заносится текущее время, а в статус заказа указывается идентификатор из таблицы order\_statuses, который соответствует значению pending. Остальные столбцы получают значение null.

Функция get\_user\_orders используется для получения информации о всех заказах пользователя. Для её вызова необходимо указать логин пользователя. Если пользователь с таким логином существует, то функция вернет курсор, который содержит следующую информацию: идентификатор заказа, город отправления, город прибытия, цену, время создания заказа, время завершения заказа, имя, фамилию и телефон водителя, а также, текущий статус заказа.

Чтобы получить название всех стран, в которых работает сервис, необходимо вызвать функцию get\_countries, которая не принимает параметров и возвращает курсор, содержащий список стран, в которых можно заказывать доставку.

Для получения городов необходимо вызвать функцию get\_cities, которая принимает параметром страну и возвращает курсор, содержащий все города, находящиеся в пределах страны. Для этого в теле функции вызывается встроенная функция sdo\_relate, которая возвращает все пространственные объекты, которые имеют конкретное пространственное взаимодействие с данным объектом [1]. Затем из полученных строк таблицы cities извлекается значения столбца name.

Чтобы оценить работу водителя, вызывается процедура rate\_driver, которая принимает следующие параметры: идентификатор заказа, рейтинг и отзыв. Если заказ с указанным идентификатором существует, то из него извлекается в переменную driver\_id идентификатор водителя. Далее в таблицу drivers\_ratings заносится информация об оценке и отзыве, относящаяся к водителю, выполнившего заказ. Если такой водитель не был найден, то выбрасывается исключение.

Функция get\_order\_items используется для получения всех грузов заказа. Она получает параметром идентификатор заказа, и если такой заказ существует, то возвращает курсор, который содержит информацию о названии, весе и объеме груза. Если такой заказ не был найден, то выбрасывается исключение.

Функция get\_user\_notifications используется для просмотра уведомлений из таблицы user\_notifications. Она принимает параметром логин пользователя, если пользователь с таким логином существует, то его идентификатор заносится в переменную user\_id. Затем курсор заполняется данными о времени и сообщении для пользователя, идентификатор которого соответствует переменной user\_id.

Чтобы удалить просмотренное уведомление необходимо вызывать процедуру remove\_user\_notification, которая принимает идентификатор уведомления параметром. Если уведомления с указанным идентификатором существует, то оно будет удалено.

Функция get\_order\_status принимает идентификатор заказа параметром. Если заказ с указанным идентификатором существует, то таблица orders соединяется с таблицей order\_statuses по столбцам id и order\_status\_id, откуда извлекается статус. Если такого заказа нет, то выбрасывается исключение.

2.3.2 Пакет для менеджера

Пакет admin\_package содержит 18 процедур и функций, предназначенных для взаимодействия с базой данных на уровне менеджера, описание этого пакета приведено в приложении Г.

Функция evaluate\_order\_price принимает идентификатор заказа параметром, если заказ с таким идентификатором существует, то его цена считается с использованием следующей формулы:

Информация о весе и объеме всех товаров извлекается из таблицы order\_items, которая соединяется с таблицей orders. Их суммарный объем и вес заносится в соответствующие переменные. Дистанция доставки вычисляется с помощью встроенной функции sdo\_distance, которая используется для получения расстояния между двумя точками в указанных единицах, в нашем случае это километры. Результат формулы (1) заносится в переменную order\_price и возвращается функцией.

Для обновления стоимости заказа используется функция update\_order\_price, которая принимает идентификатор заказа параметром. Если такой заказ существует, то в переменную order\_price заносится результат функции evaluate\_order\_price, описанной выше. Затем в строке заказа изменяется столбец price на полученное значение.

Процедура accept\_order используется менеджером для изменения статуса заказа. Она принимает параметром идентификатор заказа. В теле процедуры извлекается идентификатор статуса заказа со значением accepted и заносится в столбец order\_status\_id таблицы orders.

Менеджер может нанимать водителей с использованием процедуры hire\_driver, которая принимает следующие параметры: логин водителя и пароль. В переменную v\_password\_hash заносится результат функции encrypt\_password, которая описана в главе 2.3.2. Затем в таблицу drivers заносится новая строка с указанным логином и полученным хэшем пароля.

Для получения информации о пользователях вызывается функция get\_all\_users, которая не принимает параметров и возвращает курсор, в который заносятся значения логина, имени, фамилии и телефона каждой строки таблицы users.

Чтобы получить информацию о водителях необходимо вызывать функцию get\_all\_drivers, которая не принимает параметров. В результате функции возвращается курсор со значениями логина, имени, фамилии, телефона и рейтинга каждого водителя.

Функция get\_all\_orders не принимает параметров и возвращает курсор с информацией о всех заказах. В курсор заносятся значения идентификатора заказа, начального города, конечного города, цены, время начала заказа, время завершения заказа, статус заказа, логина пользователя и логина водителя, которые получены из соединения таблиц orders, order\_statuses, users и drivers.

Функция get\_all\_order\_by\_user принимает параметром идентификатор пользователя. Если такой пользователь существует, то возвращается курсор, содержащий значения идентичные функции get\_all\_orders, но извлекаются только те строки, где параметр равен идентификатору пользователя из таблицы orders.

По тому же принципу работает функция get\_all\_order\_by\_driver, но вместо идентификатора пользователя необходимо передать идентификатор водителя, затем проверка на равенство идентификаторов проводится по таблице drivers.

Чтобы отклонить заказ необходимо вызывать процедуру reject\_order, которая принимает параметром идентификатор заказа. Если заказ с таким идентификатором существует, то он удаляется из таблицы orders.

Для просмотра уведомления менеджера необходимо вызывать функцию get\_admin\_notifications, которая не принимает параметров. Результатом функции является курсор, содержащий идентификатор, время и сообщение уведомления.

Чтобы удалить просмотренное уведомление, следует вызывать функцию remove\_notification, которая принимает идентификатор уведомления параметром и удаляется соответствующую строку из таблицы admin\_notifications.

Функция get\_driver\_rating используется для просмотра всех отзывов от пользователей для указанного водителя. При её вызове необходимо передать логин водителя, чьи отзывы надо просмотреть. Если такой водитель существует, то функция вернет курсор, содержащий идентификатор отзыва, рейтинг и сообщение всех отзывов для выбранного водителя.

Для удаления отзыва используется процедура remove\_driver\_rating, которая принимает параметром идентификатор отзыва. Если отзыв с таким идентификатором существует в таблице drivers\_ratings, то удаляет его.

Чтобы изменить рейтинг водителя, необходимо вызывать процедуру confirm\_driver\_rating, которая принимает параметром логин водителя. В теле процедуры высчитывается среднее значение рейтинга среди всех отзывов водителя и заносится в переменную driver\_rating. Если такой водитель существует, то значение его рейтинга в таблице drivers изменяется на значение переменной driver\_rating.

Функция get\_all\_users\_pagination используется для реализации механизма пагинации в таблице users. Пагинация – разделение большого массива данных на отдельные страницы. Функция принимает следующие параметры: номер страницы и размер страницы. В курсор заносится информация о пользователях, порядковый номер которых меньше или равен номеру страницы, умноженной на размер страницы, и больше предыдущей странице, умноженной на размер страницы.

Процедура export\_orders принимает следующие параметры: начальная и конечная дата. В результате процедуры создается файл формата JSON, который содержит все заказы, которые были выполнены в промежутке начальной и конечной даты. Если конечная дата раньше начальной, то обрабатывается исключение.

2.3.3 Пакет для водителя

Пакет driver\_package содержит 10 процедур и функций, предназначенных для взаимодействия с базой данных на уровне водителя, описание этого пакета приведено в приложении Д.

Процедура fulfill\_driver\_data принимает следующие параметры: логин водителя, имя, фамилию и телефон. Если водитель с таким логином существует, то в соответствующей строке таблицы drivers изменяются поля на указанные в параметрах.

Функция get\_accepted\_orders не принимает параметров и возвращает курсор, который содержит строки со следующими значениями: идентификатор заказа, начальный город, конечный город, цена, время создания заказа, статус заказа и логин пользователя, создавшего заказ. Извлекаются только те строки, идентификатор статуса которых указывает на строку со значением accepted.

Функция get\_order\_items используется для получения всех грузов заказа. Она получает параметром идентификатор заказа, и если такой заказ существует, то возвращает курсор, который содержит информацию о названии, весе и объеме груза. Если такого заказа нет, то обработается исключение.

Чтобы получить информацию о водителе необходимо вызвать функцию get\_driver\_data, которая принимает параметром логин водителя. Если водитель с таким логином существует, то функция вернет курсор со строками, содержащими значения столбцов username, first\_name, last\_name, telephone и rating.

Для того, чтобы начать выполнения заказа необходимо вызвать процедуру start\_order, которая принимает следующие параметры: идентификатор заказа и логин водителя. Если такой заказ и водитель существуют, то соответствующая строка таблицы orders изменит значения следующих столбцов: order\_status\_id и driver\_id. В столбец order\_status\_id занесется значение идентификатора из таблицы order\_statuses, который соответствует строке со значением in\_progress. В столбец driver\_id занесется идентификатор из таблицы drivers, соответствующий указанному в параметре логину.

Чтобы закончить выполнение заказа необходимо вызвать процедуру finish\_order, которая принимает идентификатор заказа параметром. Если такой заказ был найден, то идентификатор статуса заказа будет изменен на идентификатор, соответствующий значению completed из таблицы order\_statuses.

Функция get\_driver\_orders принимает логин водителя параметром. Если такой водитель существует, то функция вернет курсор, содержащий строки со следующими значениями: идентификатор заказа, начальный город, конечный город, цена, дата создания заказа, дата завершения заказа и статус заказа. Извлекаются только те строки таблицы orders, идентификатор водителя которых равен идентификатору водителя из таблицы drivers, который указывает на логин водителя, заданный параметром.

Для обновления пароля водителя необходимо вызвать процедуру update\_driver\_password, которая принимает логин водителя и новый пароль. Если водитель с таким логином существует, то значение его пароля изменится на хэш пароля, который был указан в качестве параметра.

2.4 Триггеры базы данных

Для решения задачи об оповещении пользователей о действиях в базе данных были использованы триггеры. Триггер – именованный программный блок, выполняемый в ответ на происходящие в базе данных события [2].

Для реализации системы оповещений были созданы два триггера: notify\_delete\_trigger и notify\_insert\_trigger. Реализация этих триггеров приведена в приложении Е.

Триггер notify\_delete\_trigger используется для информирования покупателя об отмене заказа. Он срабатывает при удалении на таблице orders. В блоке before происходит формирование сообщения и извлечение идентификатора покупателя для дальнейшего занесения в таблицу. В блоке after происходит вставка строки в таблицу user\_notifications. В неё заносятся текущее время, сообщение уведомления и идентификатор покупателя, кому это уведомление принадлежит.

Триггер notify\_inset\_trigger используется для информирования менеджеров о создании нового заказа. В блоке before формируется сообщение, которое включает начальный и конечный города заказа, и информация что заказ создан. В блоке after происходит вставка строки в таблицу admin\_notifications. В неё заносятся текущее время и сообщение уведомления.

2.5 Представления базы данных

Для удобства и повышения производительности при обращении с базой данных были разработаны 7 представлений.

Представление v\_completed\_orders соединяет между собой таблицы orders, users, drivers и order\_statuses. Возвращает идентификатор заказа, цену, логин пользователя и логин водителя. Выборка происходит из тех строк идентификатор статуса заказа которых равен идентификатору статуса completed.

Представление v\_total\_profit извлекает суммарную стоимость по всем заказам.

Представление v\_total\_profit\_by\_driver соединяет между собой таблицы orders и drivers, затем группирует строки по логину водителя и извлекает суммарную стоимость всех заказов, выполненных водителем.

Представление v\_total\_profit\_by\_customer соединяет между собой таблицы orders и users, затем группирует строки по логину покупателя и извлекает суммарную стоимость всех заказов, оформленных покупателем.

Представление v\_total\_weight\_and\_volume извлекает суммарный вес грузов и их объем среди всех строк таблицы order\_items.

Представление v\_total\_orders\_by\_customer соединяет между собой таблицы users и orders, затем группирует строки по логину покупателя и извлекает количество заказов, принадлежащих покупателю.

Представление v\_total\_order\_by\_driver соединяет между собой таблицы drivers и orders, затем группирует строки по логину водителя и извлекает количество заказов, которые выполнил или выполняет водитель.

3 Тестирование и улучшение производительности

Для тестирования номинальности триггеров, были выполнены запросы к таблицам. Сначала был выполнен select-запрос, который извлекает все города из указанной страны. После выполнения запроса был просмотрен его план, стоимость которого равнялась 19213.0.

Затем на таблицы cities и boundaries были добавлены пространственные индексы, после чего стоимость запроса стала равна 6.0.

Также были проведены тесты на скорость запросов к API с помощью приложения Insomnia REST. Insomnia REST – бесплатное кроссплатформенное приложение для тестирования REST-приложений.

Соответствующий запрос на получение городов к API был выполнен без использования пространственного индекса, тогда его скорость составила 6 секунд. После создания пространственных индексов скорость запроса стала составлять 620 миллисекунд, что представлено на рисунке 3.1.

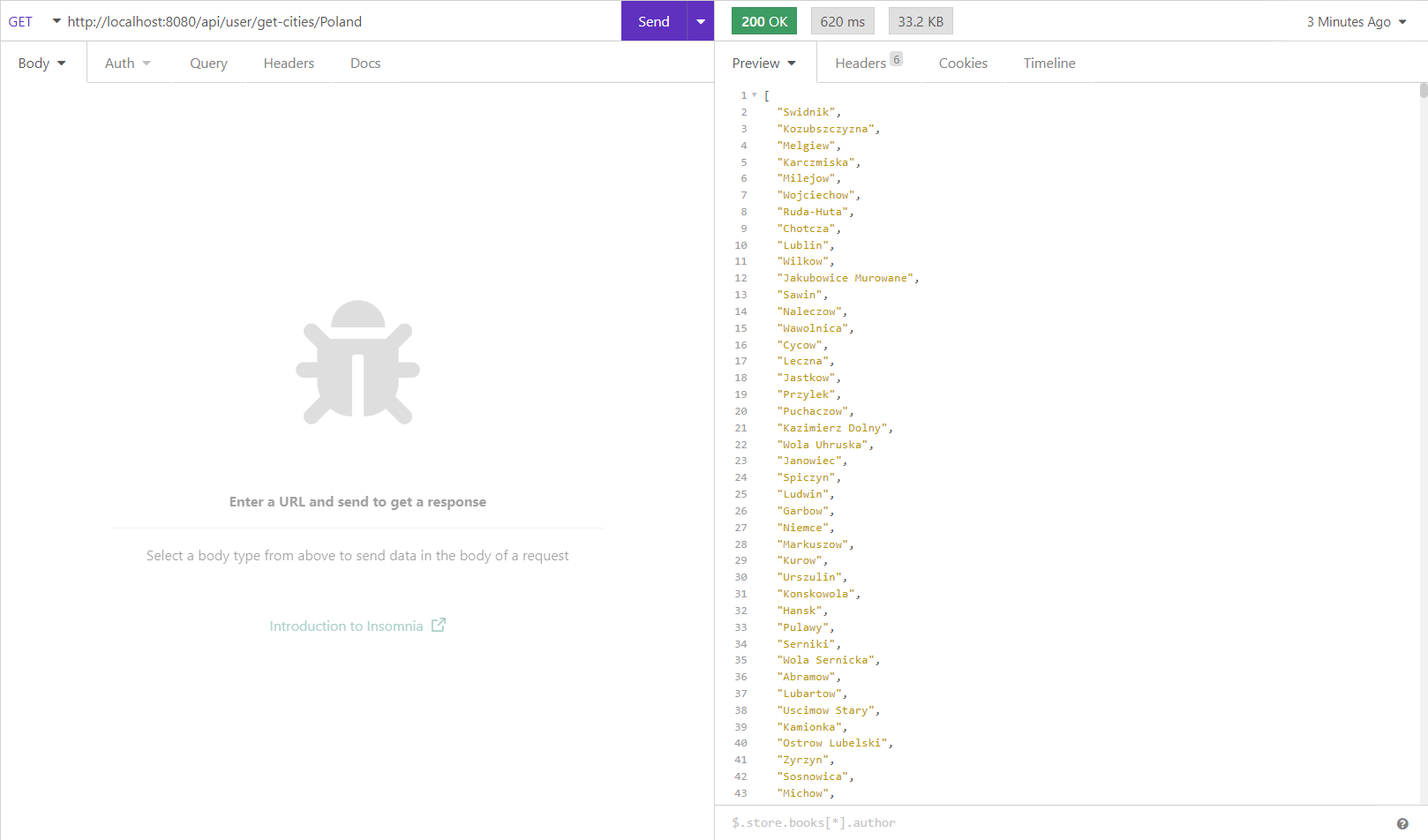


Рисунок 3.1 – Результат запроса на получение городов

4 Описание технологии

В качестве основной технологии данного курсового проекта была выбрана Oracle Spatial and Graph. Эта технология включает в себя расширенные функции для пространственных данных и анализа.

Пространственные функции в Oracle Spatial and Graph состоят из набора типов объектов, функций и процедур, использующих эти типы. Геометрия хранится как объект в одной строке в столбце типа sdo\_geometry.

В данном курсовом проекте технология Oracle Spatial and Graph является необходимой, так как было принято решение хранить реальные географические данные в базе данных. Вследствие чего нужно использовать методы нахождения расстояния по карте и способы хранения координат. Данная технология позволяет это реализовать.

4.1 Импорт пространственных данных

Чтобы загрузить пространственные данные в проект было принято решение использовать формат GeoJSON – открытый формат, предназначенный для хранения географических структур данных, основан на JSON. Соответствующие файлы были взяты с сайта opendatasoft.

Для импорта данных создается таблица с названием json\_documents, которая содержит столбец data с типом данных clob и проверкой, что тип файла json. Код создания таблицы json\_documents представлен в листинге 4.1.

|  |
| --- |
| create table json\_documents  (  id raw(16),  data clob,  constraint json\_documents\_pk primary key (id),  constraint json\_documents\_json\_chk check (data is json)  ); |

Листинг 4.1 – Создание таблицы json\_documents

Затем необходимо создать директорию, в которой располагаются документы для импорта и загрузить их содержимое в таблицу json\_documents. Затем извлечь необходимые данные в нужные таблицы. Этот процесс представлен в листинге 4.2.

|  |
| --- |
| create directory json\_dir as 'C:\app\Administrator\admin\orcl\json\_dir';  insert into json\_documents (id, data)  values (sys\_guid(), bfilename('JSON\_DIR', 'europe.geojson'));  -- извлечение из файла только названий и полигонов стран  insert into boundaries (name, boundary)  select jt.name, jt.sdo\_val  from json\_documents j nested data.features[\*]  columns (name varchar2(50) path '$.properties.NAME',  sdo\_val sdo\_geometry path '$.geometry') jt;  commit; |

Листинг 4.2 – Процесс извлечения данных в таблицы

4.2 Просмотр пространственных данных

Для просмотра пространственных данных использовался встроенный компонент СУБД SQL Developer под названием Map View. Map View – инструмент рендера карт, который изображает различные типы пространственных данных.

Чтобы открыть Map View необходимо выбрать соответствующий пункт во вкладке View СУБД SQL Developer. Затем нужно написать select-запросы к таблицам, которые нужно отобразить на карте. Результат рендера карты из таблиц boundaries и roads представлен на рисунке 4.1.

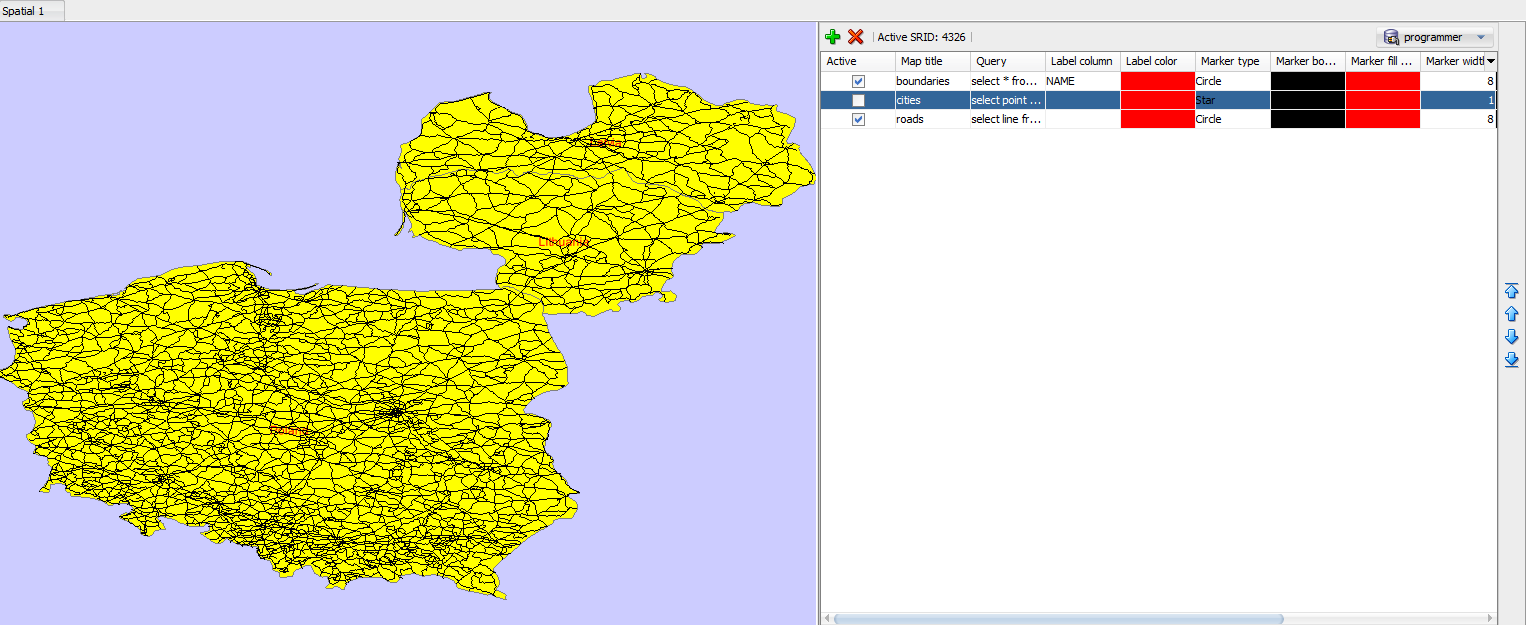


Рисунок 4.1 – Рендер каты из таблиц boundaries и roads

Рендер карты с добавлением таблицы cities представлен на рисунке 4.2.

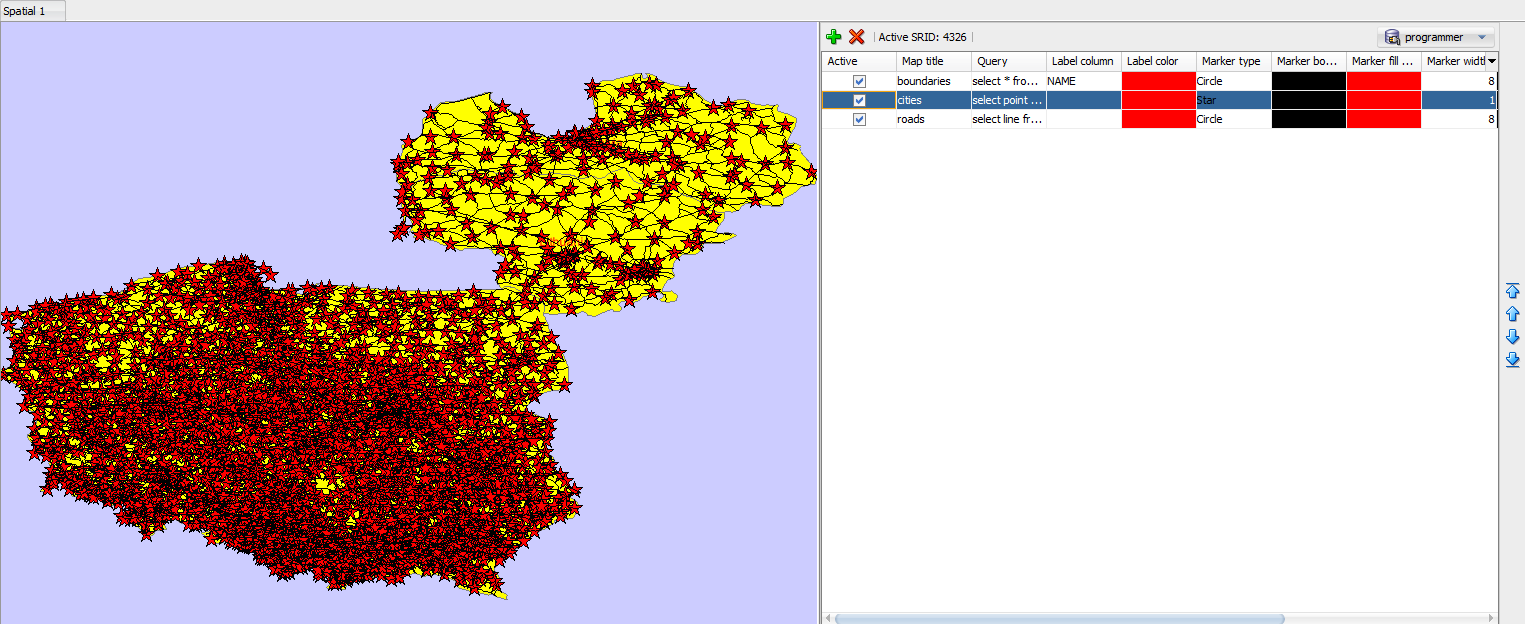


Рисунок 4.2 – Рендер карты с добавлением таблицы cities

5 Руководство пользователя

Разработанное в рамках курсового проекта приложение поддерживает три роли: покупатель, водитель и менеджер. Заходя на главную страницу приложения пользователю представляется форма для авторизации, а также кнопки отправки формы и перехода к странице регистрации. Главная страница приложения представлена на рисунке 5.1.

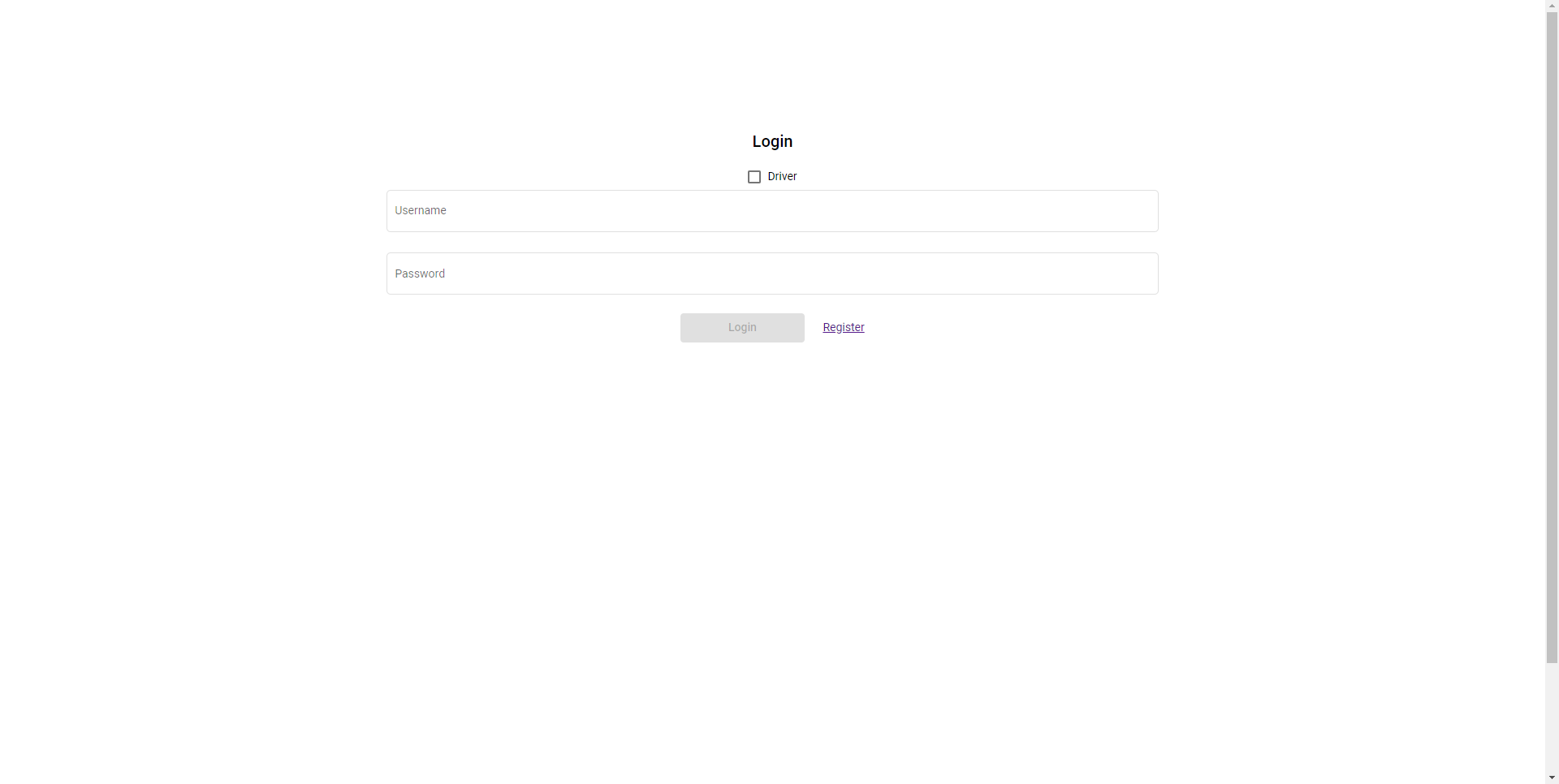


Рисунок 5.1 – Главная страница приложения

После успешной авторизации пользователь попадает на свою основную страницу, которая содержит список всех его заказов. На этой странице пользователь может создать новый заказ, нажав на соответствующую кнопку. Также пользователь при нажатии на заказ может добавить в него грузы, которые необходимо перевезти.

На навигации у пользователя имеется возможность перейти в свой личный кабинет для заполнения данных, а именно имени, фамилии и телефона. Также имеются кнопки для просмотра уведомлений и выхода из аккаунта. Основная страница для пользователя представлена на рисунке 5.2.

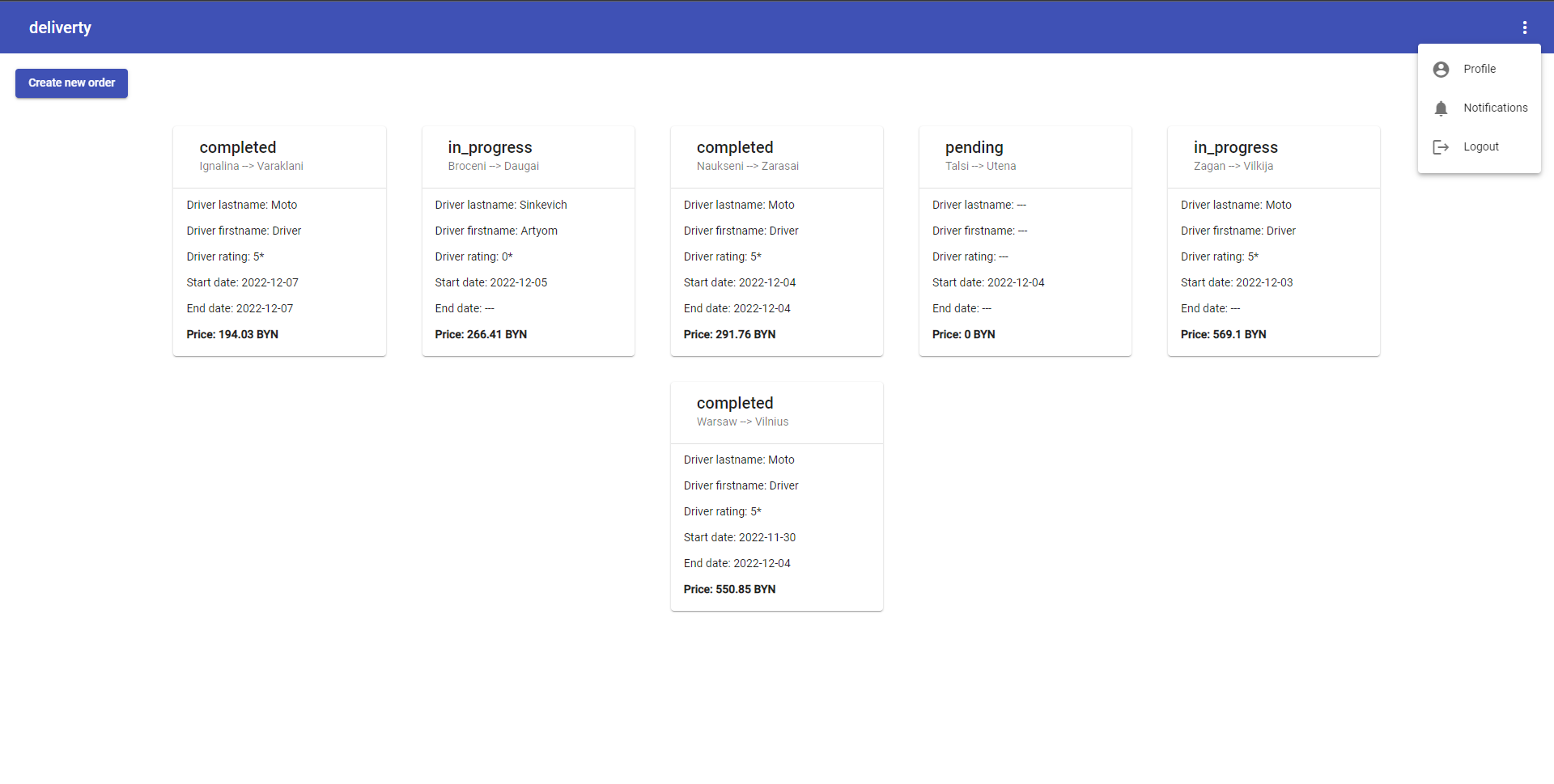


Рисунок 5.2 – Основная страница пользователя

Если авторизация была выполнена под менеджером, то приложение перенаправит на страницу со всеми пользователями, зарегистрированными в приложении. На навигации для администратора присутствуют кнопки просмотра всех пользователей, просмотра всех водителей, просмотра всех заказов, просмотра уведомлений менеджера и выхода из приложения.

При просмотре всех пользователей они будут выведены постранично, по 15 пользователей. Переключение между страницами осуществляется с помощью кнопок под таблицей. Также есть возможность нажать на строку с пользователем и получить все заказы, которые он создал.

При просмотре всех водителей выведется таблица, содержащая информацию о них. Также есть возможность нажать на строку с водителем и получить список всех отзывов от покупателей о работе данного водителя. При просмотре отзывов менеджер может удалять отдельные отзывы, которые не прошли проверку. После проверки менеджер может принять отзывы и рассчитать рейтинг водителя.

При просмотре всех заказов выведется таблица, содержащая информацию о них и кнопки, которые активны при статусе заказа pending. Всего имеется 3 кнопки: высчитать цену заказа, принять заказ и отклонить заказ. При нажатии на строку заказа выведется таблица с информацией о всех его грузах.

При просмотре уведомлений выведется таблица, содержащая дату создания и сообщение уведомления. После просмотра менеджер может удалить уведомление нажав на соответствующую кнопку.

Пример интерфейса для страницы менеджера приведен на рисунке 5.3.

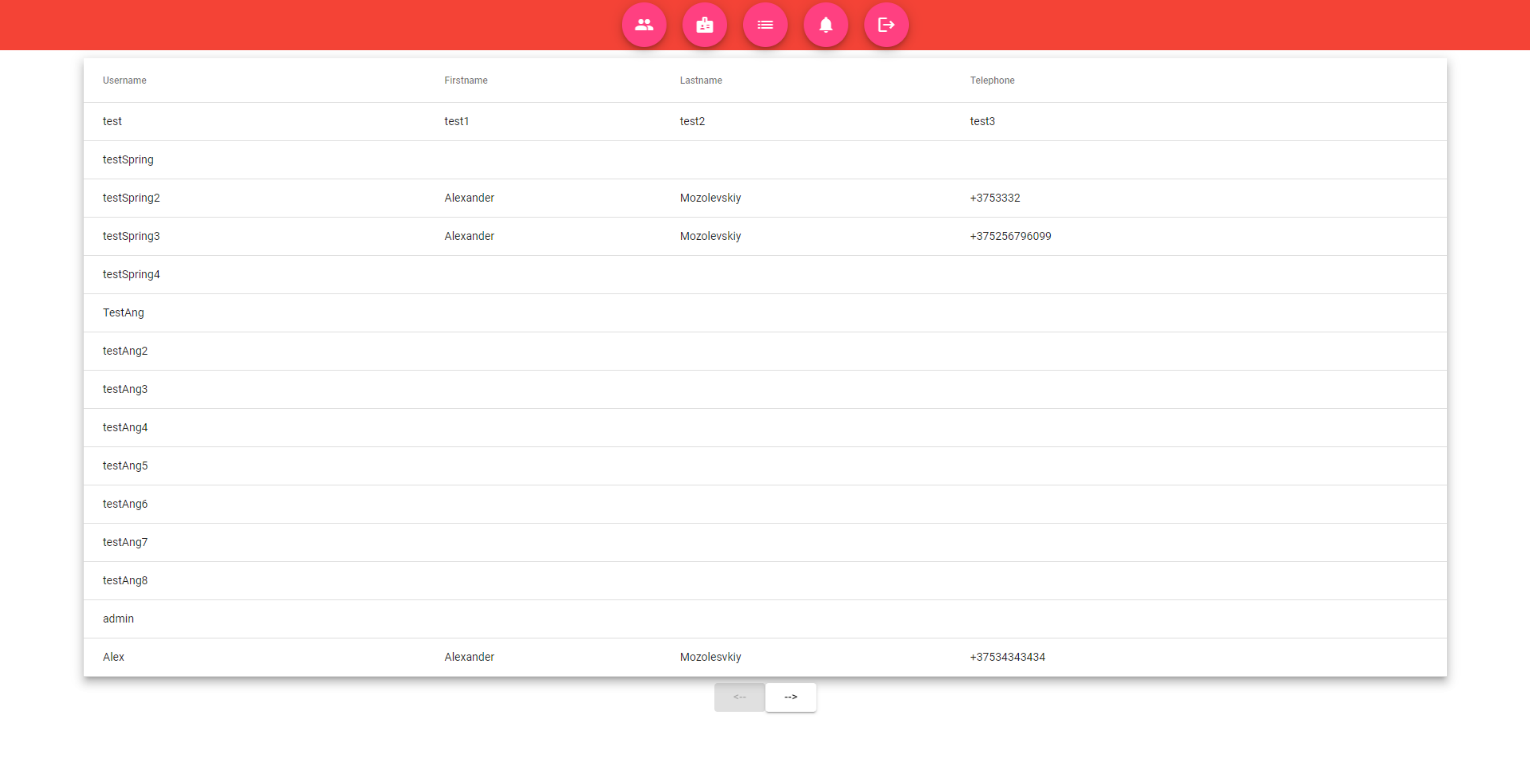


Рисунок 5.3 – Страница просмотра всех пользователей

Также авторизация может быть выполнена под водителем, если выбрать соответствующий пункт на главной странице приложения. Если авторизация была выполнена успешно, то произойдет перенаправление на страницу с заказами, которые были приняты менеджером.

На навигации для водителя присутствуют кнопки просмотра заказов, принятых менеджером, просмотра заказов, которые на данный момент выполняются или уже выполнены текущим водителем, для просмотра информации о текущем водителе и выхода из приложения.

При просмотре заказов, принятых менеджером, выведется таблица, которая содержит информацию о заказе и кнопка, позволяющая начать выполнение заказа. Также есть возможность нажать на строку с заказом и просмотреть информацию о всех грузах, которые необходимо доставить.

При просмотре заказов, которые выполняются или выполнены текущим водителем, выведется таблица, содержащая информацию о заказе и кнопка, позволяющая указать, что заказ был выполнен текущим водителем. Также есть возможность нажать на строку с заказом и просмотреть информацию о всех грузах, которые необходимо доставить.

При просмотре информации о водителе, выведется форма, в которой водитель может заполнить или заменить личные данные, такие как имя, фамилия и телефон. Также на странице присутствует поле для смены пароля.

Пример страницы для водителя представлен на рисунке 5.4.

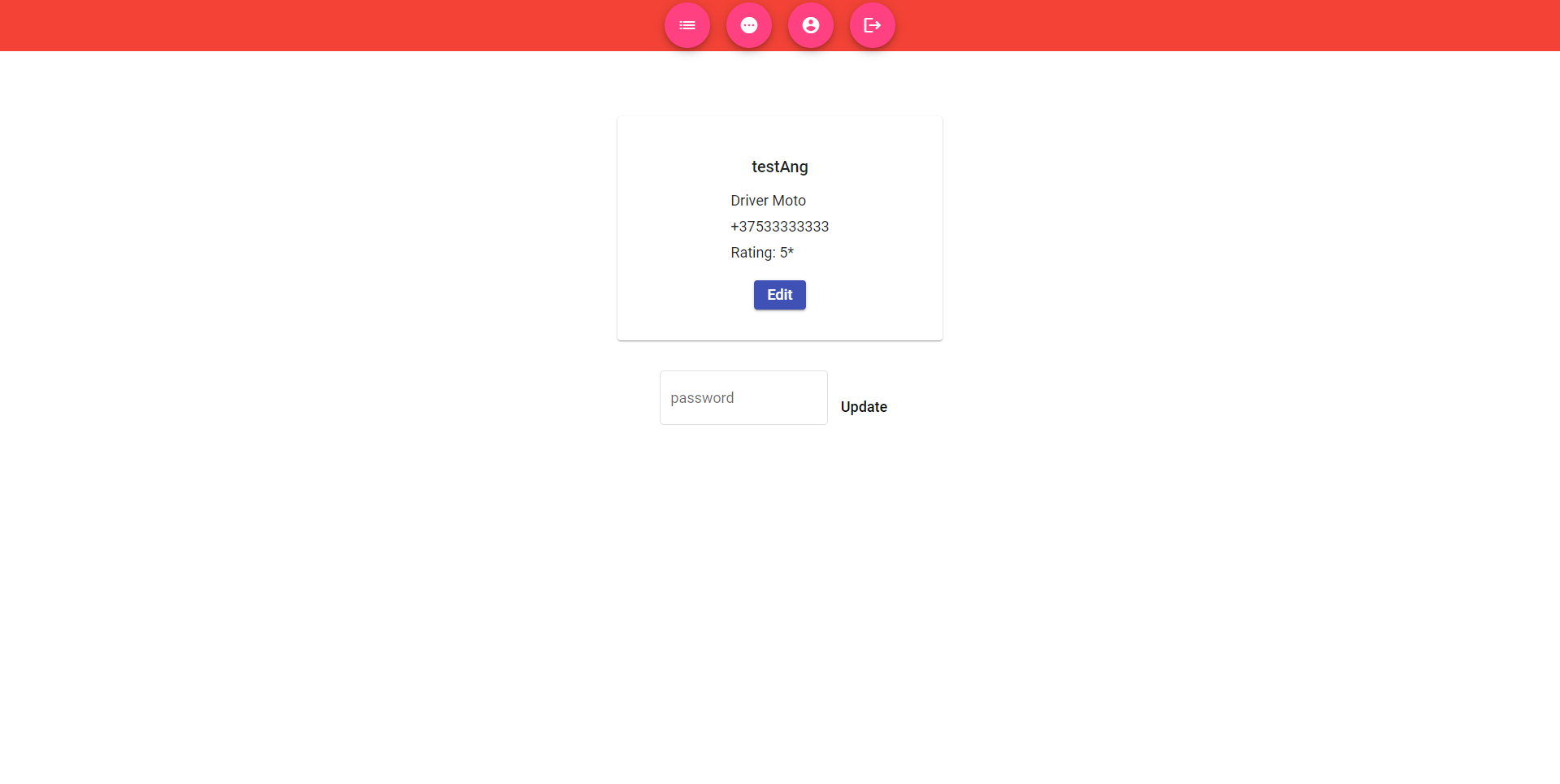


Рисунок 5.4 – Страница просмотра информации о водителе

Заключение

Результатом выполнения курсового проекта стало вэб-приложение с использованием технологий Java Spring для взаимодействия с базой данных и предоставления собственного API, и TypeScript Angular для клиентской части. В ходе курсового проекта была использована база данных Oracle 21c, которая спроектирована для системы грузоперевозок.

Были реализованы основные требования, указанные в листе задания вместе со следующими пунктами:

* регистрация и авторизация пользователей с учётом выданных им ролей;
* пароли при регистрации шифруются;
* пространственные данные соответствуют реальным географическим объектам, как страны, города и дороги;
* возможность изменения личной информации покупателю и водителю;
* создание заказов покупателем;
* расчёт стоимости выполнения заказа менеджером;
* возможность принятия или отмены заказа менеджером;
* возможность начала выполнения заказа водителем;
* отметка о завершении выполнения заказа водителем;
* оценка выполненной работы покупателем;
* возможность нанимать новых водителей менеджером;
* смена своего пароля водителем;
* просмотр уведомлений покупателя и менеджера;
* просмотр отзывов о работе водителя менеджером;
* удаление некорректных отзывов и расчёт рейтинга водителя менеджером;
* в процессе разработки производительность приложения была протестирована с помощью Insomnia REST.

Данный курсовой проект является примером разработки базы данных и приложения для работы с ней. Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

Список литературы

1. Oracle Spatial and Graph Documentation [Электронный ресурс] / Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/19/spatl/index.html – Дата доступа: 10.11.2022.

2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru – Дата доступа: 27.11.2022.

3. Все о триггерах Oracle [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/256655/ – Дата доступа: 07.12.2022

4. How to Export Oracle Query Results to a JSON File [Электронный ресурс] / Режим доступа: https://database.guide/how-to-export-oracle-query-results-to-a-json-file-when-using-sqlcl/ – Дата доступа: 06.12.2022

5. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm – Дата доступа: 26.10.2022.

Приложение А

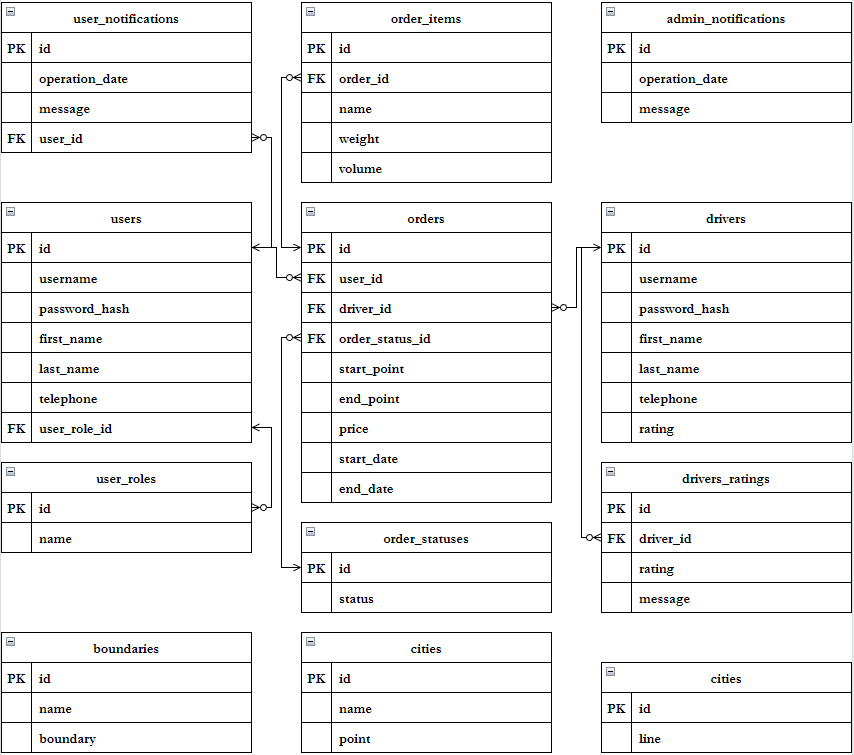


Схема базы данных

Приложение Б

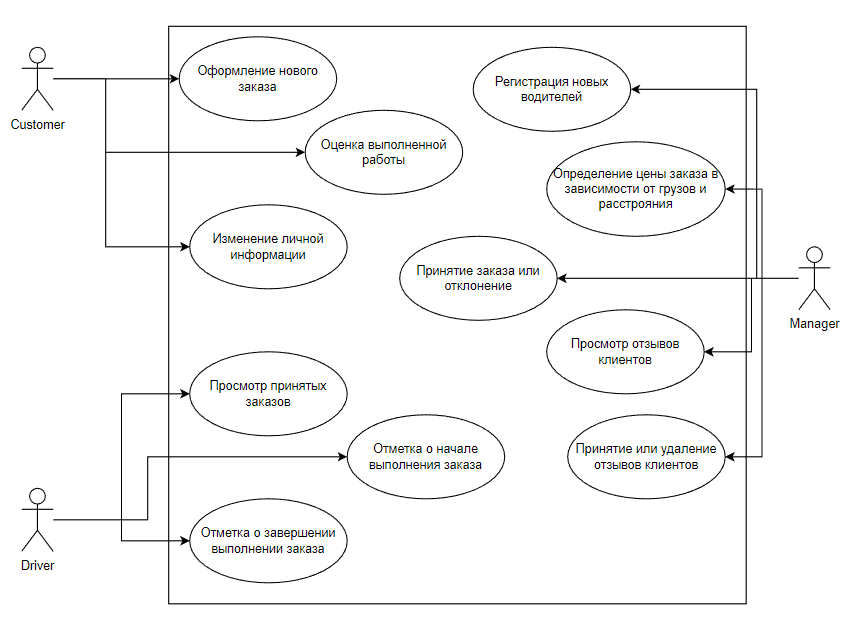


Диаграмма использования

Приложение В

Описание процедур и функций пакета «user\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| OBJECT\_NAME | ARGUMENT\_NAME | DATA\_TYPE | IN\_OUT |
| LOGIN\_USER |  | REF CURSOR | OUT |
| LOGIN\_USER | USERNAME\_IN | VARCHAR2 | IN |
| LOGIN\_USER | PASSWORD | VARCHAR2 | IN |
| GET\_USER\_DATA |  | REF CURSOR | OUT |
| GET\_USER\_DATA | USERNAME\_IN | VARCHAR2 | IN |
| GET\_CITIES |  | REF CURSOR | OUT |
| GET\_CITIES | COUNTY \_IN | VARCHAR2 | IN |
| GET\_COUNTRIES |  | REF CURSOR | OUT |
| RATE\_DRIVER | ORDER\_ID\_IN | NUMBER | IN |
| RATE\_DRIVER | RATING\_IN | NUMBER | IN |
| RATE\_DRIVER | MESSAGE\_IN | VARCHAR2 | IN |
| GET\_ORDER\_ITEMS |  | REF CURSOR | OUT |
| GET\_ORDER\_ITEMS | ORDER\_ID\_IN | NUMBER | IN |
| REGISTER\_USER | USERNAME\_IN | VARCHAR2 | IN |
| REGISTER\_USER | PASSWORD | VARCHAR2 | IN |
| ADD\_ITEM\_TO\_ORDER | ORDER\_ID\_IN | NUMBER | IN |
| ADD\_ITEM\_TO\_ORDER | ITEM\_NAME\_IN | VARCHAR2 | IN |
| ADD\_ITEM\_TO\_ORDER | ITEM\_WEIGHT\_IN | NUMBER | IN |
| ADD\_ITEM\_TO\_ORDER | ITEM\_VOLUME\_IN | NUMBER | IN |
| GET\_USER\_NOTIFICATIONS |  | REF CURSOR | OUT |
| GET\_USER\_NOTIFICATIONS | USERNAME\_IN | VARCHAR2 | IN |
| FULFILL\_USER\_DATA | USERNAME\_IN | VARCHAR2 | IN |
| FULFILL\_USER\_DATA | FIRST\_NAME\_IN | VARCHAR2 | IN |
| FULFILL\_USER\_DATA | LAST\_NAME\_IN | VARCHAR2 | IN |
| FULFILL\_USER\_DATA | TELEPHONE | VARCHAR2 | IN |
| GET\_USER\_ORDERS |  | REF CURSOR | OUT |
| GET\_USER\_ORDERS | USERNAME\_IN | VARCHAR2 | IN |
| REMOVE\_NOTIFICATION | NOTIFIC\_ID\_IN | NUMBER | IN |
| GET\_ORDER\_STATUS |  | VARCHAR2 | OUT |
| GET\_ORDER\_STATUS | ORDER\_ID\_IN | NUMBER | IN |
| CREATE\_ORDER | USERNAME\_IN | VARCHAR2 | IN |
| CREATE\_ORDER | START\_POINT\_IN | VARCHAR2 | IN |
| CREATE\_ORDER | END\_POINT\_IN | VARCHAR2 | IN |
| ENCRYPT\_PASSWORD |  | VARCHAR2 | OUT |
| ENCRYPT\_PASSWORD | PASSWORD | VARCHAR2 | IN |

Приложение Г

Описание процедур и функций пакета «admin\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| OBJECT\_NAME | ARGUMENT\_NAME | DATA\_TYPE | IN\_OUT |
| ACCEPT\_ORDER | ORDER\_ID\_IN |  | IN |
| EVALUATE\_ORDER\_PRICE |  | NUMBER | OUT |
| EVALUATE\_ORDER\_PRICE | ORDER\_ID\_IN |  | IN |
| GET\_ALL\_ORDERS\_BY\_USER |  | REF CURSOR | OUT |
| GET\_ALL\_ORDERS\_BY\_USER | USERNAME\_IN |  | IN |
| REJECT\_ORDER | ORDER\_ID\_IN |  | IN |
| GET\_ALL\_ORDERS |  | REF CURSOR | OUT |
| REMOVE\_DRIVER\_RATING | RATING\_ID\_IN |  | IN |
| REMOVE\_NOTIFICATION | NOTIFIC\_ID\_IN |  | IN |
| CONFIRM\_DRIVER\_RATING | DRIVER\_USRNAME |  | IN |
| ENCRYPT\_PASSWORD |  | VARCHAR2 | OUT |
| ENCRYPT\_PASSWORD | PASSWORD |  | IN |
| GET\_NOTIFICATIONS |  | REF CURSOR | OUT |
| UPDATE\_ORDER\_PRICE | ORDER\_ID\_IN |  | IN |
| GET\_DRIVER\_RATING |  | REF CURSOR | OUT |
| GET\_DRIVER\_RATING | DRIVER\_USRNAME |  | IN |
| GET\_USERS\_PAGINATION |  | REF CURSOR | OUT |
| GET\_USERS\_PAGINATION | PAGE\_IN |  | IN |
| GET\_USERS\_PAGINATION | PAGE\_SIZE\_IN |  | IN |
| GET\_ALL\_USERS |  | REF CURSOR | OUT |
| GET\_ORDER\_ITEMS |  | REF CURSOR | OUT |
| GET\_ORDER\_ITEMS | ORDER\_ID\_IN |  | IN |
| HIRE\_DRIVER | DRIVER\_USRNAME |  | IN |
| HIRE\_DRIVER | PASSWORD\_IN |  | IN |
| EXPORT\_ORDERS | START\_DATE\_IN |  | IN |
| EXPORT\_ORDERS | END\_DATE\_IN |  | IN |
| GET\_ALL\_DRIVERS |  | REF CURSOR | OUT |
| GET\_ORDERS\_BY\_DRIVER |  | REF CURSOR | OUT |
| GET\_ORDERS\_BY\_DRIVER | DRIVER\_USRNAME |  | IN |

Приложение Д

Описание процедур и функций пакета «driver\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| OBJECT\_NAME | ARGUMENT\_NAME | DATA\_TYPE | IN\_OUT |
| GET\_DRIVER\_ORDERS |  | REF\_CURSOR | OUT |
| GET\_DRIVER\_ORDERS | DRIVER\_USRNAME |  | IN |
| LOGIN\_DRIVER |  | REF\_CURSOR | OUT |
| LOGIN\_DRIVER | DRIVER\_USRNAME |  | IN |
| LOGIN\_DRIVER | PASSWORD |  | IN |
| GET\_ACCEPTED\_ORDERS |  | REF\_CURSOR | OUT |
| UPDATE \_PASSWORD | DRIVER\_USRNAME |  | IN |
| UPDATE \_PASSWORD | NEW\_PASSWORD |  | IN |
| ENCRYPT\_PASSWORD |  | VARCHAR2 | OUT |
| ENCRYPT\_PASSWORD | PASSWORD |  | IN |
| FULFILL\_DRIVER\_DATA | DRIVER\_USRNAME |  | IN |
| FULFILL\_DRIVER\_DATA | FIRST\_NAME\_IN |  | IN |
| FULFILL\_DRIVER\_DATA | LAST\_NAME\_IN |  | IN |
| FULFILL\_DRIVER\_DATA | TELEPHONE\_IN |  | IN |
| GET\_DRIVER\_DATA |  | REF\_CURSOR | OUT |
| GET\_DRIVER\_DATA | DRIVER\_USRNAME |  | IN |
| GET\_ORDER\_ITEMS |  | REF\_CURSOR | OUT |
| GET\_ORDER\_ITEMS | ORDER\_ID\_IN |  | IN |
| FINISH\_ORDER | ORDER\_ID\_IN |  | IN |
| START\_ORDER | ORDER\_ID\_IN |  | IN |
| START\_ORDER | DRIVER\_USRNAME |  | IN |

Приложение Е

|  |
| --- |
| create or replace trigger notify\_delete\_order  for delete  on orders  compound trigger  message varchar2(1000);  v\_user\_id number;  before each row is  begin  message := 'Order ' || :old.start\_point || '-->' || :old.end\_point  || ' rejected';  v\_user\_id := :old.user\_id;  end before each row;  after statement is  begin  if message is not null then  insert into user\_notifications (operation\_date, message, user\_id) values (sysdate, message, v\_user\_id);  end if;  end after statement;  end notify\_delete\_order;  create or replace trigger notify\_insert\_order  for insert  on orders  compound trigger  message varchar2(1000);  before each row is  begin  message := 'Order ' || :new.start\_point || '-->' || :new.end\_point  || ' created';  end before each row;  after statement is  begin  if message is not null then  insert into admin\_notifications(operation\_date, message) values (sysdate, message);  end if;  end after statement;  end notify\_insert\_order; |

Листинг – Создание триггеров