

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по индивидуальному домашнему заданию
по дисциплине « Введение в нереляционные базы данных»
Тема: ИС курьерской фирмы

Студенты гр. 9382

Иерусалимов Н.

Голубева В.П.

Сорокумов С.В.

Преподаватель

Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Иерусалимов Н.

Голубева В.П.

Сорокумов С.В.

Группа 9382

Тема проекта: Разработка информационной системы курьерской фирмы.

Исходные данные:

Необходимо реализовать приложение для управления курьерской фирмой для СУБД MongoDB.

Содержание пояснительной записки:

“Содержание”

“Введение”

“Качественные требования к решению”

“Сценарий использования”

“Модель данных”

“Разработка приложения”

“Вывод”

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 9382

Иерусалимов Н.

Голубева В.П.

Сорокумов С.В.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать в команде информационную систему курьерской фирмы. Для реализации была выбрана СУБД MongoDB. Во внимание будут приняты такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке:
<https://github.com/moevm/nosql2h22-courier>

ANNOTATION

As part of this course, it was supposed to develop an information system of a courier company in the team. MongoDB DBMS was chosen for implementation. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: <https://github.com/moevm/nosql2h22-courier>

Оглавление

1. Введение	7
2. Качественные требования к решению	7
3. Сценарий использования	7
4. Модель данных	16
5. Разработанное приложение	24
6. Вывод	25

1. Введение

Цель работы - создать высокопроизводительную и удобную систему для управлением курьерской фирмы.

Было решено разработать веб-приложение, которое позволит управлять компанией, взаимодействовать как клиентам, так и работникам.

2. Качественные требования к решению

Требуется разработать приложение с использованием СУБД MongoDB.

3. Сценарий использования

Макеты UI

1. Страница авторизации (Рис. 1).

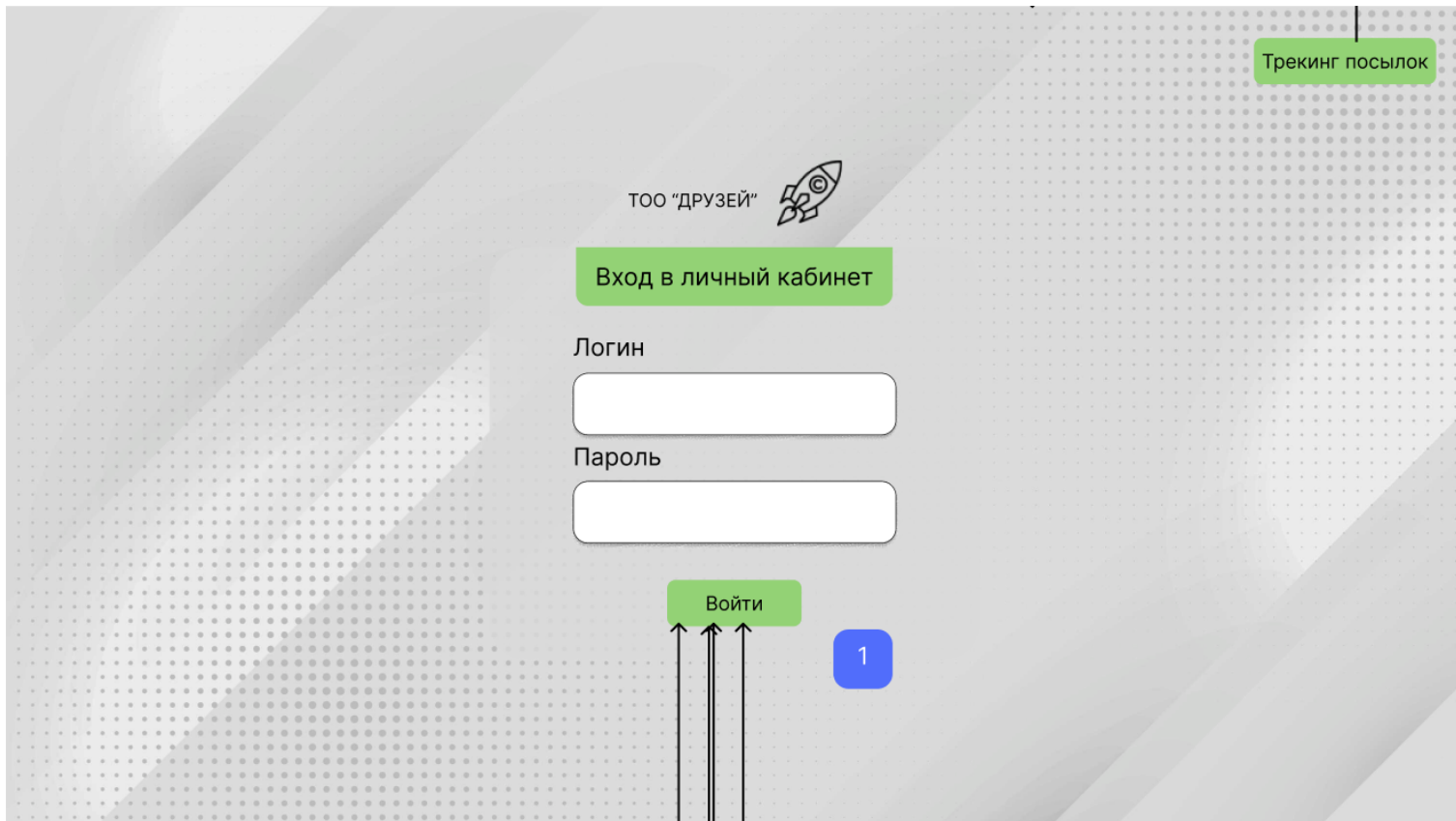


Рисунок 1 - Страница авторизации.

2. Страница для отслеживания посылки 1 (Рис. 2).

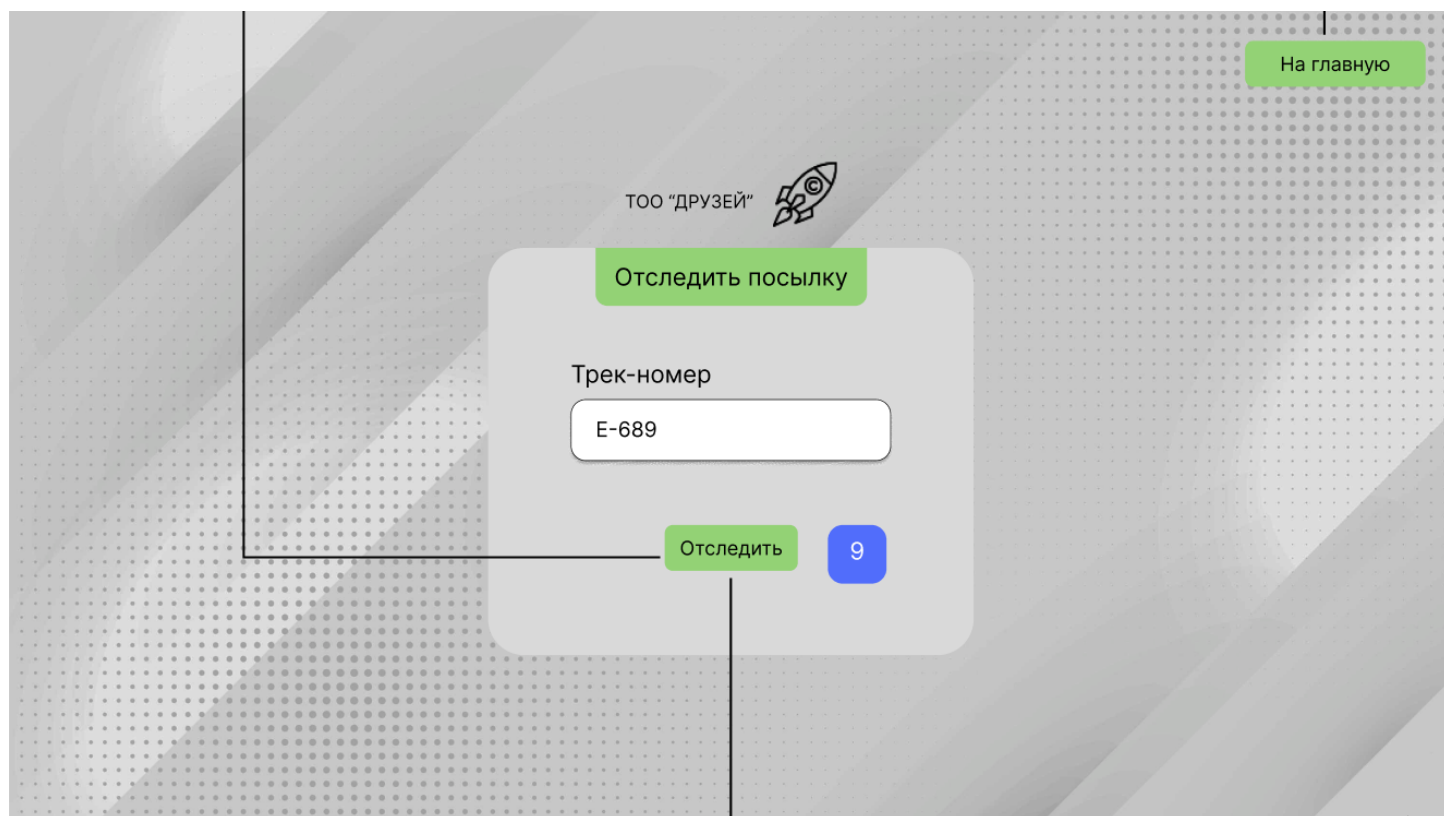


Рисунок 2 - Страница для отслеживания посылки 1.

3. Страница для отслеживания посылки 2 (Рис. 3).

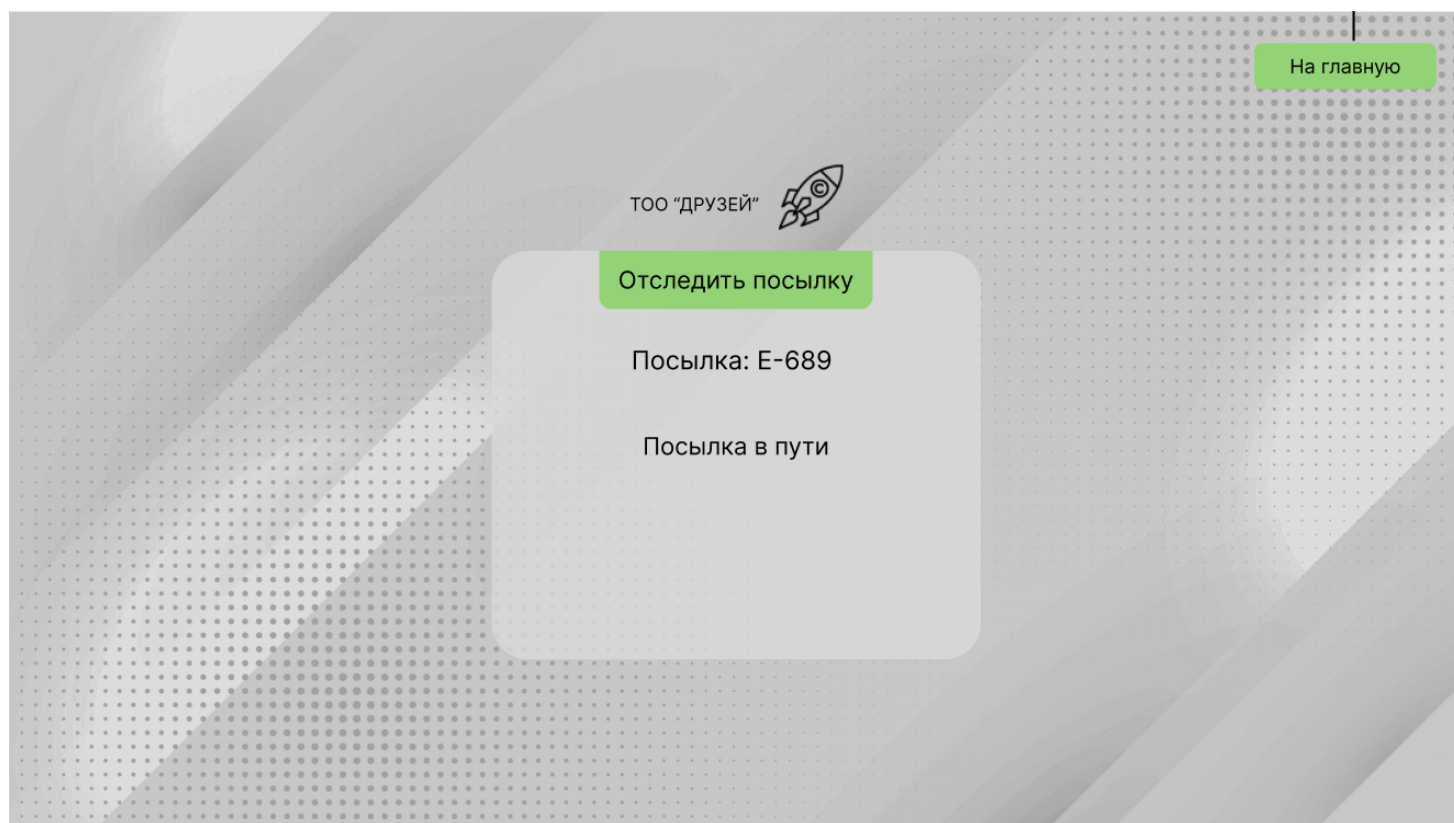


Рисунок 3 - Страница для отслеживания посылки 2.

4. Главная страница пользователя с ролью пользователь (Рис. 4).

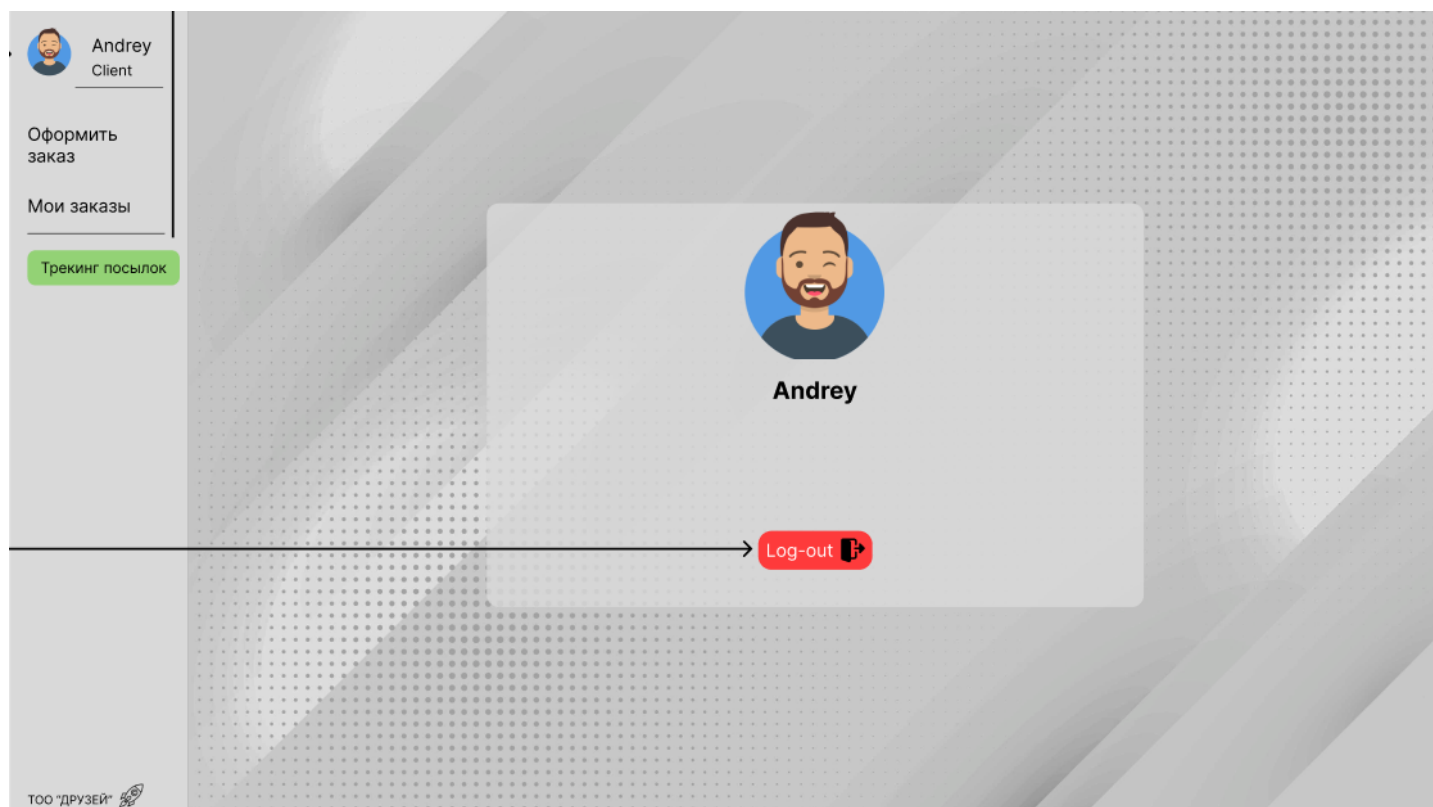


Рисунок 4 - Главная страница пользователя с ролью пользователь.

5. Страница оформления заказа (Рис. 5).

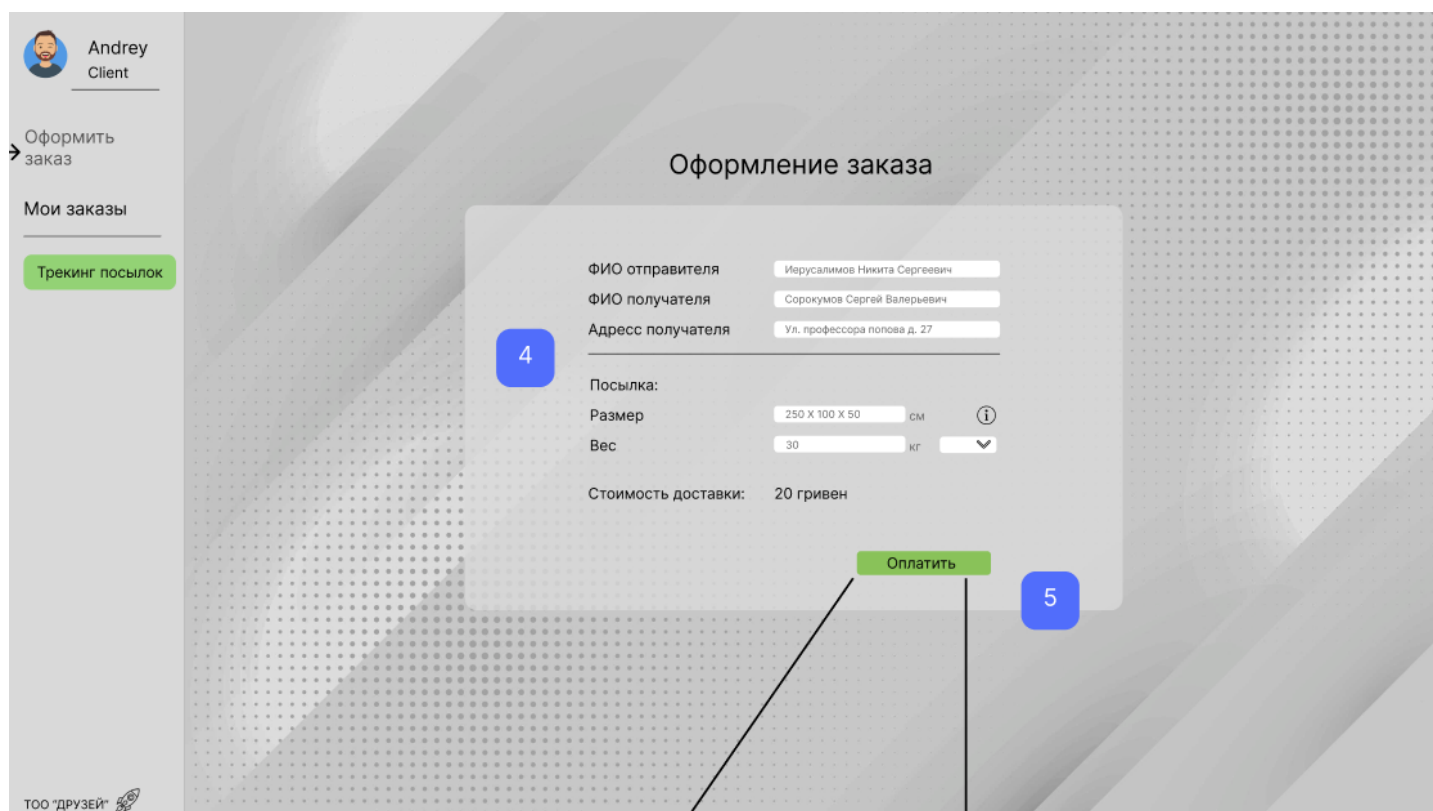


Рисунок 5 - Страница оформления заказа.

6. Страница “Мои заказы” (Рис. 6).



Рисунок 6 - Страница “Мои заказы”.

7. Главная страница пользователя с ролью администратор (Рис. 7).

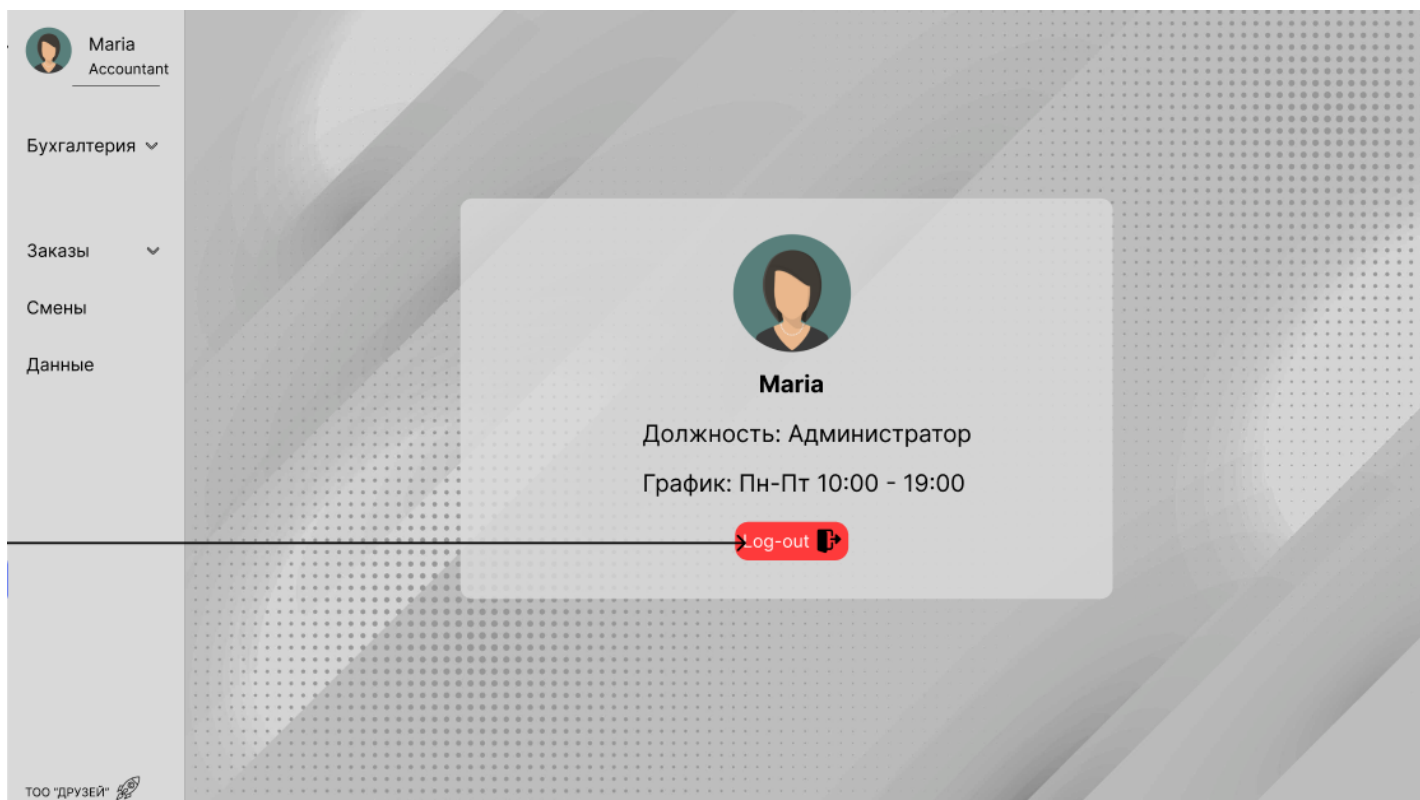
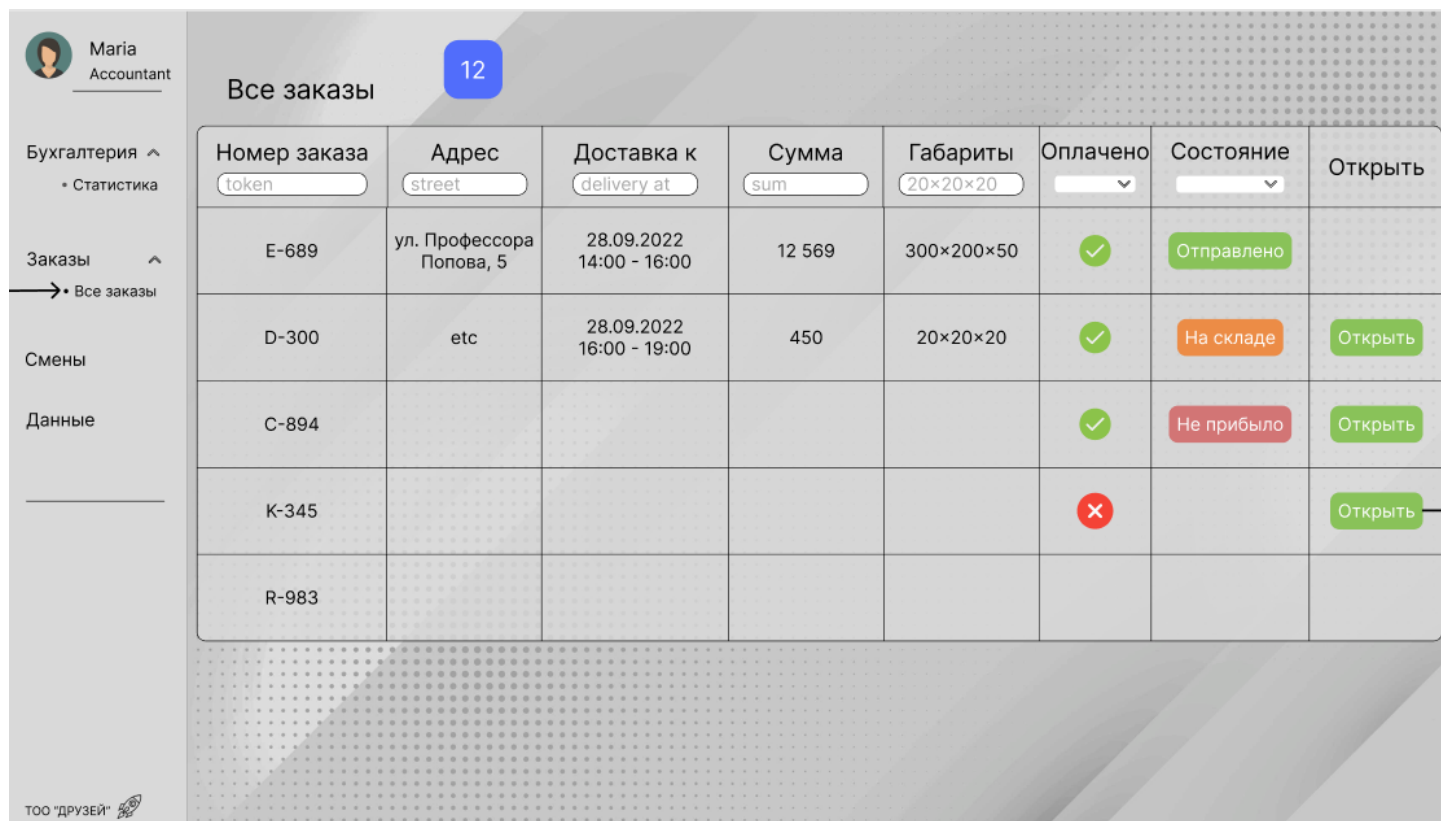


Рисунок 7 - Главная страница пользователя с ролью администратор.

8. Страница заказов от пользователя с ролью администратор (Рис. 8).



Номер заказа	Адрес	Доставка к	Сумма	Габариты	Оплачено	Состояние	Открыть
E-689	ул. Профессора Попова, 5	28.09.2022 14:00 - 16:00	12 569	300×200×50	✓	Отправлено	
D-300	etc	28.09.2022 16:00 - 19:00	450	20×20×20	✓	На складе	Открыть
C-894					✓	Не прибыло	Открыть
K-345					✗		Открыть
R-983							

Рисунок 8 - Страница заказов от пользователя с ролью администратор.

9. Страница статистики (Рис. 9).

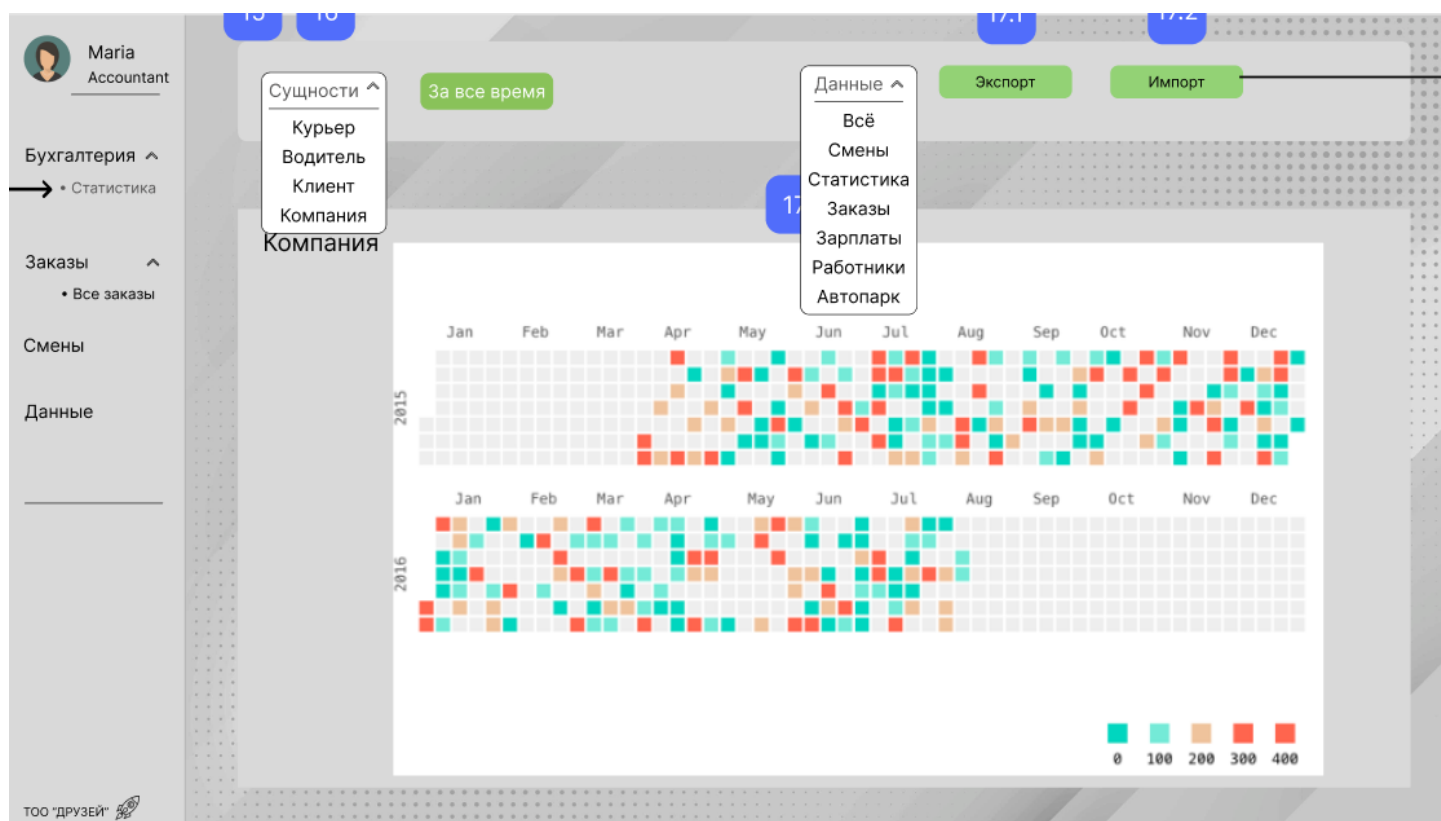


Рисунок 9 - Страница статистики.

10. Страница “Смены” (Рис. 10).

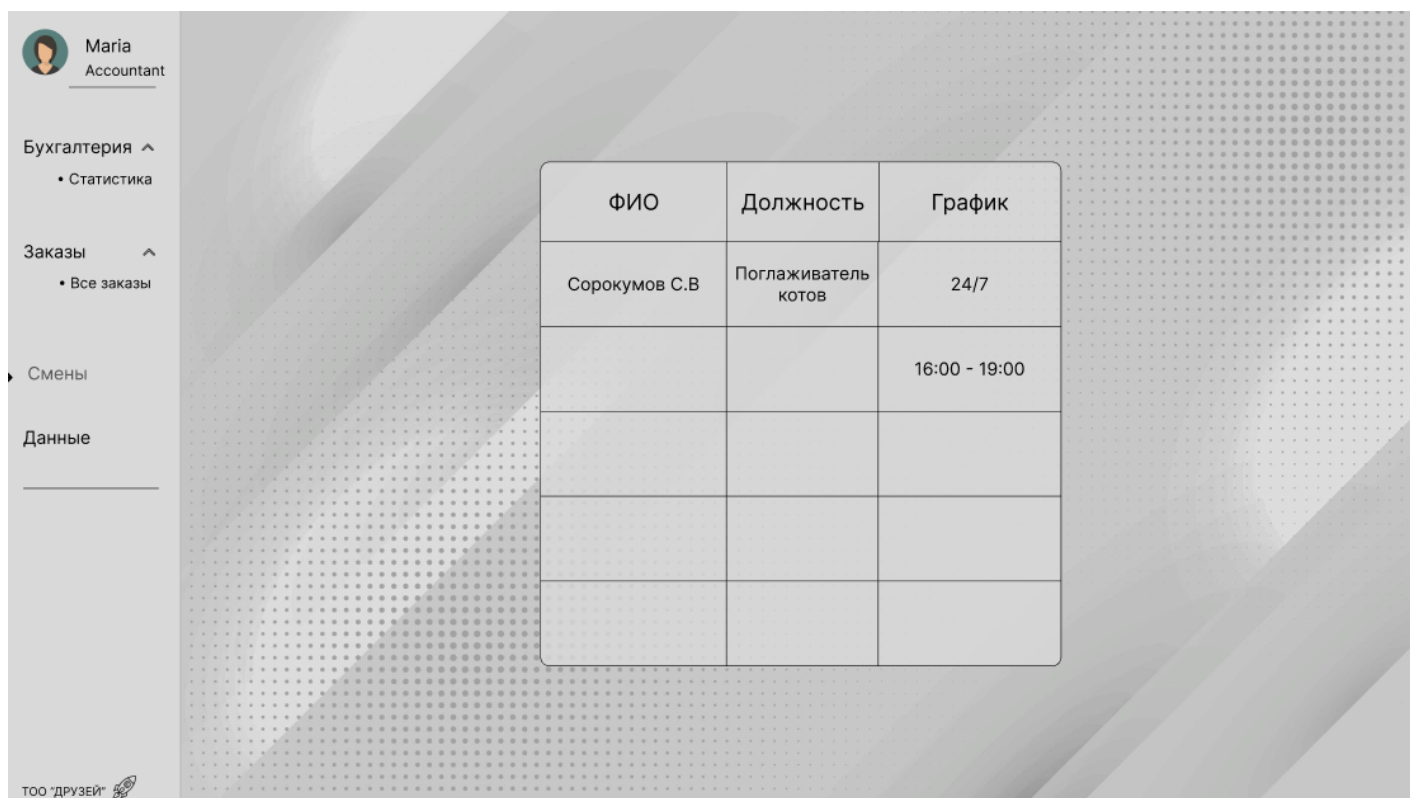


Рисунок 10 - Страница “Смены”.

11. Главная страница пользователя с ролью водитель (Рис. 11).

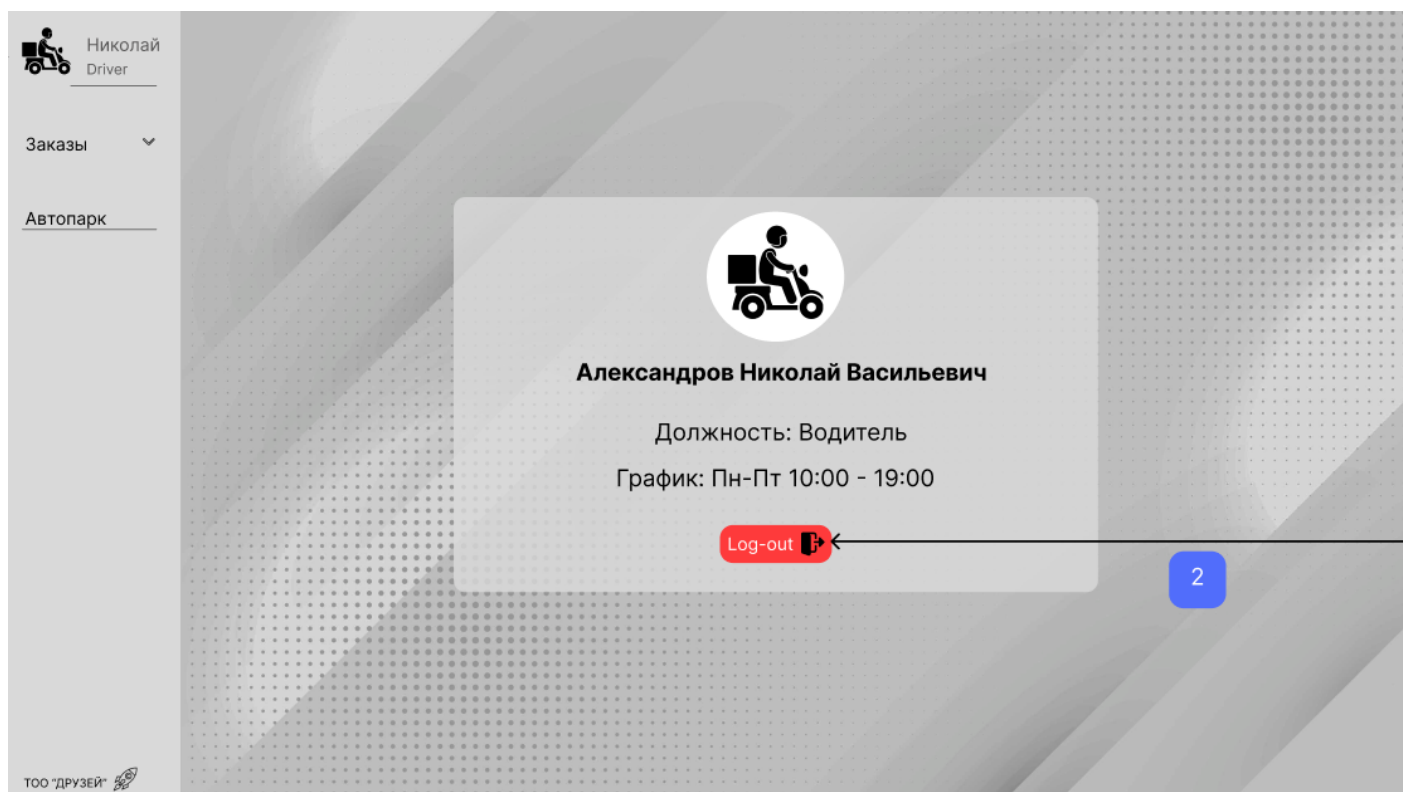



Рисунок 11 - Главная страница пользователя с ролью водитель.

12. Страница “Автопарк” (Рис. 12).


Mark Driver

Заказы

- Активные
- Завершеные

Автопарк

Автопарк

Номер	Тип	Данные	Готова к работе	Свободна	Действия
O0070A	Легковая	Открыть	Да	Да	Взять Отдать
A748DY	Грузовая	Открыть	Нет	Нет	

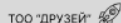



Рисунок 12 - Страница “Автопарк”.

13. Страница “Данные машины” (Рис. 13).



Mark Driver

Заказы

- Активные
- Завершеные

Автопарк

Данные машины A748DY



Номер
A748DY

Год выпуска
2016

Модель
Газель М-10

Статус
Занято Иванов И.И.

Готовность к работе
Готова/Не готова

Цвет
Синий

Взять
Отдать

21

←
К списку
→




Рисунок 13 - Страница “Данные машины”.

14. Главная страница пользователя с ролью курьер (Рис. 14).

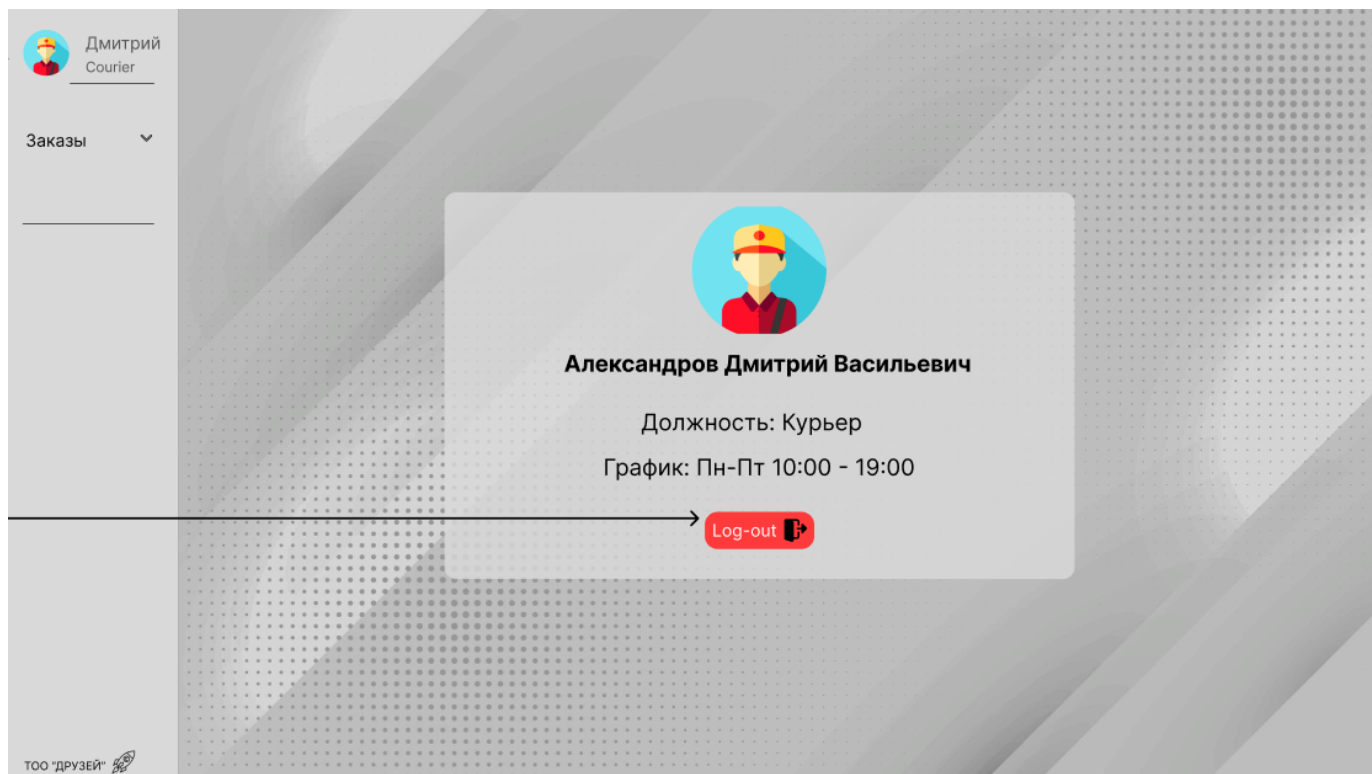


Рисунок 13 - Главная страница пользователя с ролью курьер.

Описание сценариев использования

USE-CASE Авторизация

1. Вход в личный кабинет: ЛЮБОЙ пользователь вводит логин и пароль, нажимает кнопку войти и переходит в личный кабинет
 - 1.1. Перейти в профиль пользователя
2. ЛЮБОЙ пользователь жмет кнопку log-out и завершить активную сессию

USE-CASE Оформление заказа

3. Пользователь жмет кнопку “Оформление заказа”
4. На странице с оформлением заказа пользователь вводит “ФИО получателя”, “ФИО отправителя”, “Адрес получателя”, “Размер посылки”, “вес посылки” => получает стоимость заказа (динамически после заполнения всех полей)
5. Пользователь жмет кнопку “Оформить”
6. Пользователь видит сообщение

6.1. Об успехе

6.2. О неудаче

USE-CASE Пользователь, Мои заказы

7. Пользователь получает список всех имеющихся заказов (сделать пример плашек с заказами)
8. Пользователь жмет кнопку “Трекинг посылок”
 - 8.1. Пользователь жмёт кнопку “Отследить” в списке заказов, получает формочку с заполненным номером, жмёт кнопку “Отследить”
9. Вводит “Трек-номер” заказа, жмет кнопку “Отследить”
10. Получает статус заказа
 - 10.1. Получает уведомление что такого заказа нет
 - 10.2. Возвращается в трекер через кнопку “К трекеру”

USE-CASE Администратор заказы

11. Администратор жмет кнопку заказы
12. Администратор выбирает заказ с помощью фильтра (нажимает “Открыть”)
13. При наличии галочки в поле “Оплачено” нажимает кнопку передать на склад, возвращается к списку заказов

USE-CASE Администратор Статистика

14. Администратор жмет кнопку “Бухгалтерия-> Статистика”
15. Администратор выбирает сущность, нажимая выбрать
16. Администратор устанавливает значения интервала, по которому хочет получить статистику, жмет кнопку получить
17. Администратор видит статистику
 - 17.1. Администратор нажимает на кнопку Export и в его файловую систему загружается файл с выбранными данными
 - 17.2. Администратор нажимает на кнопку Import
 - 17.3. Администратор загружает из файловой системы файл с данными для бд

USE-CASE Администратор Смены

18. Администратор жмет кнопку “Смены” -> Видит таблицу со сменами
- USE-CASE Водитель, Взять машину**
19. Водитель жмет “Автопарк”
20. Водитель выбирает машину в таблице и в этой строчке жмет открыть
21. Если машина была взята, жмет отдать. Если машина не была взята, жмет
взять

USE-CASE Водитель, Курьер

22. Водитель/Курьер жмет “Заказы->Завершенные”
23. Видит таблицу со всеми завершенными заказами и статистику по ним
24. Водитель/Курьер жмет “Заказы->Активные”
25. Водитель/Курьер видит таблицу с активными заказами
26. Водитель/Курьер жмет кнопку открыть у выбранного заказа
27. Если статус в таблице был галочка, Водитель/Курьер вводит код
28. Водитель/Курьер жмет проверить код
29. Если код верный, жмет завершить заказ
30. Если неверный ищет настоящего клиента
31. Если статус в таблице был крестик, жмет взять со склада

4. Модель данных

1. MongoDB

Графическое представление

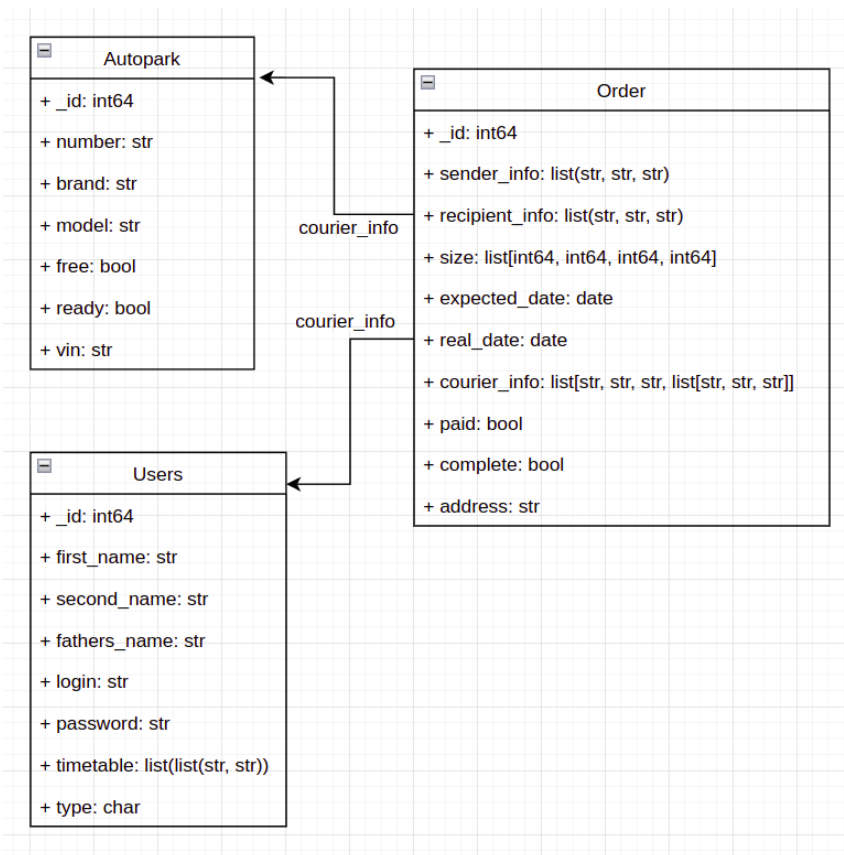


Рисунок 15 - Графическое представление MongoDB модели.

Коллекция Users связана с коллекцией Orders через поля `first_name` `second_name` `fathers_name`

Коллекция Autopark связана с коллекцией Orders через поля `number` `brand` `model`

Подробное описание

БД содержит 3 коллекции:

1. Autopark

- `_id` - уникальный идентификатор автомобиля
- `number` - гос. номер автомобиля
- `brand` - название марки авто
- `model` - название модели авто
- `free` - свободна ли машина
- `ready` - готова ли машина к работе
- `vin` - VIN номер автомобиля

2. Users

- `_id` - уникальный идентификатор пользователя
- `first_name` - Имя
- `second_name` - Фамилия
- `fathers_name` - Отчество
- `login` - логин, для входа
- `password` - пароль для входа
- `timetable` - Расписание работы (если пользователь -- работник)
- `role` - Тип пользователя

3. Orders

- `_id` - уникальный идентификатор пользователя
- `sender_info` - информация о отправителе
 - `first_name` - Имя
 - `second_name` - Фамилия
 - `fathers_name` - Отчество
- `recipient_info` - информация о получателе
 - `first_name` - Имя
 - `second_name` - Фамилия
 - `fathers_name` - Отчество
- `size` - размеры посылки
 - `height` - высота посылки
 - `width` - ширина посылки
 - `length` - длина посылки
 - `weight` - вес посылки
- `address` - адрес доставки
- `paid` - оплачена ли доставка
- `expected_date` - ожидаемая дата доставки
- `real_date` - реальная дата доставки
- `courier_info` - информация о курьере

- `first_name` - Имя
- `second_name` - Фамилия
- `fathers_name` - Отчество
- `car_info` - информация об автомобиле, если доставка большого груза
 - `number` - гос. номер автомобиля
 - `brand` - название марки авто
 - `model` - название модели авто
- `complete` - выполнен ли заказ
- `cost` - стоимость заказа

Оценка удельного объема информации, хранимой в модели

1. Autopark

- `_id` - int64 - 8b
- `number` - str (9 символов) - 9b
- `brand` - str - max 15b
- `model` - str - max 15b
- `vin` - str (17 символов) - 17b
- `free` - bool - 1b
- `ready` - bool - 1b Один элемент занимает 66b

2. Users

- `_id` - уникальный идентификатор пользователя
- `first_name` - str - max 35b
- `second_name` - str - max 35b
- `fathers_name` - str - max 35b
- `login` - str - max 35b
- `password` - str - max 35b
- `timetable` - list(list(str, str)) - max 100b
- `role` - char - 1b Один элемент занимает 276b

3. Orders

- `_id` - int64 - 8b
- `sender_info` - list[str, str, str] - 105b
- `recipient_info` - list[str, str, str] - 105b
- `size` - list[int64, int64, int64, int64] - 32b
- `address` - str - 100b
- `paid` - bool - 1b
- `expected_date` - date - 4b
- `real_date` - date - 4b
- `courier_info` - list[str, str, str, list[str, str, str]] - 144b
- `complete` - bool - 1b
- `cost` - int64 - 8b

Один элемент занимает 512b.

Пример модели

Моделирование компании с 30 автомобилями, 50 сотрудниками, 1000 пользователей и 50000 заказов - 25791780 ~ 25,8Mb

Выразим объем модели через количество заказов, на каждый заказ приходится по 3 пользователя, на каждый второй заказ приходится 1 автомобиль, тогда получим линейную зависимость равную

$$V_d(m) = 512m + m/3(276) + m/2(66) = 637m$$

Избыточность модели равна:

$$V_d(m) = \frac{512m + m/3(276) + m/2(66)}{158m + m/3(276) + m/2(66)} = \frac{637m}{283m} = \frac{637}{283} = 2.25$$

Запросы:

- Поиск пользователя при авторизации
`db.users.findOne({login: login, password: password})`
- Поиск всех свободных машин, готовых к работе
`db.autopark.find({ready: true, free: true})`
- Поиск заказов с опозданием

```
db.order.find({$where: function(){
    return obj.expected_date != obj.real_date}})
```

- Поиск расписания работника

```
db.users.findOne({_id: courier_id})['timetable']
```

- Поиск всех выполненных заказов

```
db.order.find({complete: True})
```

- Поиск всех заказов, выполненных определенным работником

```
courier = db.users.findOne({_id: id})
```

```
db.order.find({courier_info: [courier...], complete: True})
```

- Поиск определенного заказа

```
db.order.find({_id: id})
```

2. SQL

Графическое представление

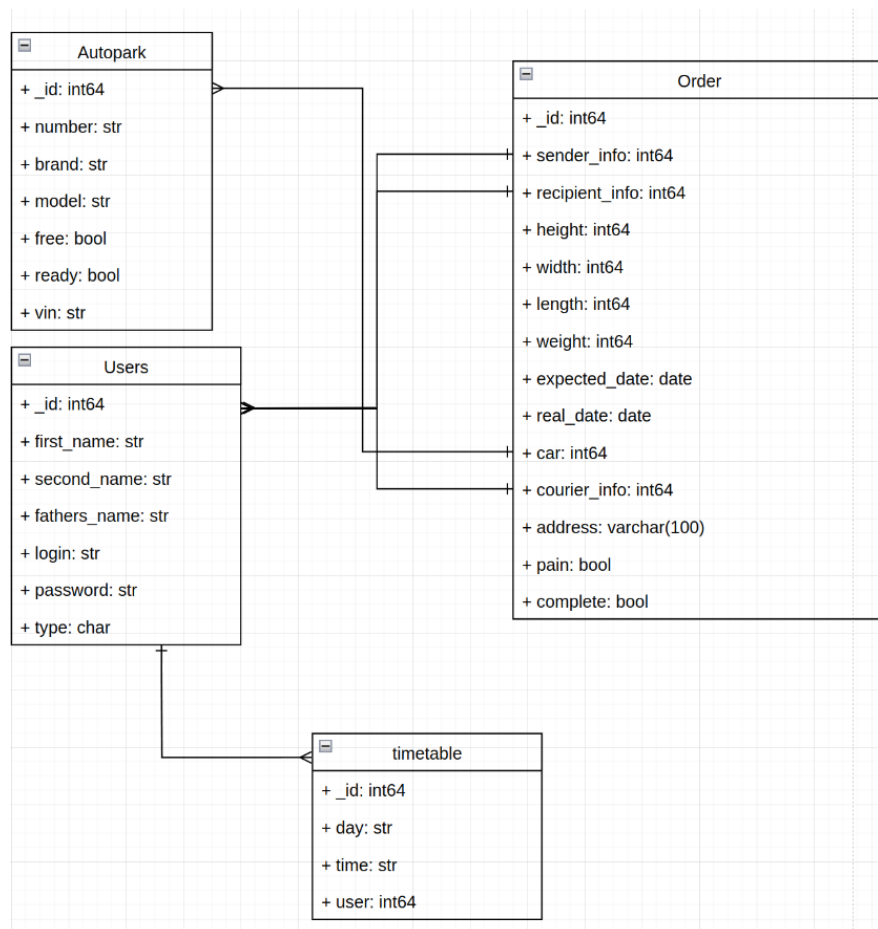


Рисунок 16 - Графическое представление SQL модели.

Подробное описание

1. Autopark

- `_id` - уникальный идентификатор автомобиля - int64 - 8b
- `number` - гос. номер автомобиля - varchar (9 символов) - 9b
- `brand` - название марки авто - varchar (15 символов) - 15b
- `model` - название модели авто - varchar (15 символов) - 15b
- `vin` - VIN номер автомобиля - varchar (17 символов) - 17b
- `ready` - готова ли машина - bool - 1b
- `free` - свободна ли машина - bool - 1b

Один элемент занимает 66b

2. Users

- `_id` - уникальный идентификатор пользователя - int64 - 8b
- `first_name` - Имя - varchar (35 символов) - 35b
- `second_name` - Фамилия - varchar (35 символов) - 35b
- `fathers_name` - Отчество - varchar (35 символов) - 35b
- `login` - логин, для входа - varchar (35 символов) - 35b
- `password` - пароль для входа - varchar (35 символов) - 35b
- `type` - Тип пользователя - char - 1b

Один элемент занимает 184b

3. Timetable

- `_id` - уникальный идентификатор расписания - int64 - 8b
- `day` - день недели - varchar (2 символов) - 2b
- `time` - время работы - varchar (11 символов) - 11b
- `user` - идентификатор пользователя, кому принадлежит расписание - int64, 8b

Один элемент занимает 29b

4. Order

- `_id` - уникальный идентификатор заказа - int64 - 8b
- `sender_info` - уникальный идентификатор отправителя - int64 - 8b

- recipient_info - уникальный идентификатор получателя - int64 - 8b
- height - высота посылки - int64 - 8b
- width - ширина посылки - int64 - 8b
- length - длина посылки - int64 - 8b
- weight - вес посылки - int64 - 8b
- address - адрес доставки - varchar - 100b
- paid - оплачена ли доставка - bool - 1b
- expected_date - ожидаемая дата доставки - date - 4b
- real_date - реальная дата доставки - date - 4b
- courier_info - информация о курьере - int64 - 8b
- complete - выполнен ли заказ - bool - 1b
- cost - стоимость заказа - int64 - 8b

Один элемент занимает 182b

Моделирование компании с 30 автомобилями, 50 сотрудниками (по 5 записей в расписании), 1000 пользователей и 50000 заказов - 9302430 ~ 9.3Mb

Запросы

- Поиск пользователя при авторизации

```
SELECT * FROM Users WHERE login = login AND password =
password;
```

- Поиск всех свободных машин, готовых к работе

```
SELECT * FROM Autopark WHERE ready = true AND free = true;
```

- Поиск заказов с опозданием

```
SELECT * FROM Order WHERE expected_date != real_date;
```

- Поиск расписания работника

```
SELECT * FROM Timetable WHERE user = id;
```

- Поиск всех выполненных заказов

```
SELECT * FROM Order WHERE complete = true;
```

- Поиск всех заказов, выполненных определенным работником

```
SELECT * FROM Order WHERE complete = true AND courier_info =
id;
```

- Поиск определенного заказа

```
SELECT * FROM Order WHERE _id = id;
```

Сравнение моделей

Как можно заметить из аналогичных запросов, количество запросов к моделям будет примерно идентичное. Количество памяти необходимое для mongodb примерно в 2.77 раза больше из-за дублирования информации для увеличения скорости работы. Для случаев, где необходимо минимизировать затраты память, лучше использовать sql модель. В нашем случае при работе с клиентами решающую роль будет играть скорость работы, обработки запросов, поэтому лучше использовать не реляционную модель.

5. Разработанное приложение

Краткое описание системы

Приложение состоит из двух компонент back-end и front-end.

Back-end представляет из себя python-приложение реализованное при помощи фреймворка flask. Основная суть заключается в обрабатывать запросы front-end`а для взаимодействия с базой данных.

Front-end js-приложение реализованное при помощи фреймворка React-js, которое использует API back-end приложения для работы с БД и визуализации данных для удобной работы пользователя.

Использованные технологии

1. Python 3.9
2. Flask
3. React-js
4. MongoDB
5. Docker
6. Docker-compose
7. pymongo

6. Вывод

В ходе выполнения работы была спроектирована СУБД MongoDB, произведена оценка удельного объема информации, хранимой в модели. Была представлена аналогичная реляционная модель и произведено сравнение, в результатах которого MongoDB модель показала себя лучше чем SQL модель.

Было реализовано web-приложение для курьерской фирмы.