

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
на учебную практику
по ПМ.11 «Разработка, администрирование и защита баз данных»

Студент Багрова П.А. _____ Группа № У2331
(Фамилия И. О.)

Руководитель Ефимова Т.Н., преподаватель факультета СПО
Говоров А.И., преподаватель факультета СПО

Тема задания: Проектирование и реализация базы данных.

Сроки прохождения практики: 02.02.2020 -02.07.2020

Место прохождения практики: Факультет СПО

1. Виды работ и требования к их выполнению:

Учебная практика проводится распределенно (понедельно в течение семестра) на базе факультета СПО в лаборатории разработки баз данных. В ходе прохождения практики выполняются следующие виды работ:

- I. Вводный инструктаж по технике безопасности и общим целям, и задачам практики.
- II. Анализ поставленной задачи
- III. Выполнение индивидуального задания: проектирование БД, разработка прототипа веб-приложения.
- IV. Формирование отчета по учебной практике.

2. Виды отчетных материалов и требования к их оформлению:

По результатам прохождения практики составляется отчет, в котором представляются индивидуальное задание, модель базы данных, перечень использованных технологий, программных средств, использованных паттернов (шаблонов) проектирования программ, программный код, описание результатов работы программы. Оформление отчета должно соответствовать Рекомендациям по оформлению технических документов факультета СПО Университета ИТМО.

3. ПЛАН-ГРАФИК

№ эта па	Наименование этапа	Срок завершения этапа	Виды работ	Форма отчетности
1	2	3	4	5
1.	Вводный инструктаж	02.02.2020 – 09.02.2020	Ознакомление с инструкцией по технике безопасности. Ознакомление с целями и задачами производственной практики	Журнал по технике безопасности
2.	Постановка задачи	09.02.2020 – 09.03.2020	Анализ индивидуального задания. Обследование предметной области согласно индивидуальной теме учебной практики.	Отчет по практике: индивидуальное задание Дневник практики
3.	Моделирование базы данных и реализация	10.03.2020 – 31.03.2020	Описание предметной области. Создание диаграммы классов Создание таблиц Заполнение таблиц данными (команды)	Отчет по практике: индивидуальное задание Дневник практики
4.	Реализация модели данных средствами Django ORM	01.04.2020 – 07.04.2020	Создание модели Django в соответствии с моделью данных и настройка связи между таблицами	Отчет по практике: индивидуальное задание Дневник практики
5.		82.04.2020 – 21.04.2020	Реализация элементов инфраструктуры Django, в соответствии с архитектурным паттерном Model-View-Template или сокращенно MVT. Реализация интерфейсов к системе средствами Django Templates или сторонними средствами.	Отчет по практике: индивидуальное задание Дневник практики
6.	Подготовка отчетных материалов	17.06.2020 – 23.06.2020	Формирование отчета о практике	Отчет по практике: индивидуальное задание Дневник практики
7.	Защита результатов практики	24.06.2020 – 02.07.2020	Защита результатов практики в форме устного собеседования и представления результатов практики	

Задание утверждено председателем выпускающей комиссии факультета СПО
 Председатель выпускающей комиссии факультета СПО _____ Королев В.В.
 «___» _____ 20__ г

Дата выдачи задания: _____

Руководитель от факультета _____
 (подпись руководителя)

Задание принял к
 исполнению _____
 (подпись студента)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

Направление подготовки (специальность) 09.02.07 Информационные системы и программирование

О Т Ч Е Т

**об учебной практике по профессиональному модулю
ПМ.11 «Разработка, администрирование и защита баз данных»**

Тема задания: Разработка и реализация базы данных по предметной области

Обучающийся Багрова П.А. Группа У2331
(Фамилия И.О.) (номер группы)

Руководитель практики: Ефимова Т.Н., преподаватель факультета СПО
Говоров А.И., преподаватель факультета СПО

Ответственный за практику от университета: Королев В.В.зам. директора
факультета СПО

Практика пройдена с оценкой _____

Подписи членов комиссии _____ (_____) (подпись)

_____ (А.И.Говоров) (подпись)

_____ (Т.Н.Ефимова) (подпись)

Дата _____

Санкт-Петербург
2020

СОДЕРЖАНИЕ

Введение.....	5
1 Индивидуальное задание	6
1.1 Формулировка поставленной задачи	6
1.2 Описание предметной области	6
1.3 Функциональные требования к системе.....	6
2 Проектирование Базы данных	7
3 Технологии и программные средства.....	8
3.1 Использованные технологии	8
3.2 Программные средства	8
3.3 Шаблон проектирования MVT	8
4 Программная реализация	10
Заключение	17
Список литературы	18
Приложение А	20
Приложение Б.....	26

ВВЕДЕНИЕ

Целью учебной практики по профессиональному модулю ПМ.11 «Разработка, администрирование и защита баз данных» является углубление знаний и практических умений и получение начального практического опыта по основным видам деятельности «Разработка, администрирование и защита баз данных» и овладение соответствующими общими и профессиональными компетенциями: ОК 1, ОК 2, ОК 3, ОК 4, ОК 5, ОК 6, ОК 7, ОК 8, ОК 9, ОК 10, ОК 11, ПК 11.1., ПК 11.2., ПК 11.3, ПК 11.4., ПК 11.5., ПК 11.6. (см. рабочая программа и фонд оценочных средств по производственной практике).

Учебная практика проводится на базе факультета СПО Университета ИТМО.

Результатом практики является разработка прототипа веб-приложения по заданной предметной области, использующего реляционную базу данных.

Задачи:

1. Спроектировать базу данных;
2. Описать модель данных приложения;
3. Описать методы получения, вставки, редактирования и удаления данных;
4. Описать внешнюю оболочку приложения;
5. Упаковать приложение в Docker.

1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1.1 Формулировка поставленной задачи

Разработать веб-приложение на Django в соответствии с вариантом.

Средства разработки:

- Django + Django REST framework(DRF)
- Vue.js или React

1.2 Описание предметной области

«Автопарк». Автопарк осуществляет обслуживание заказов на перевозку грузов, используя для этой цели свой парк автомашин и своих водителей. Водитель, выполнивший заказ, получает 20 % от стоимости перевозки.

Управление автопарком должно иметь сведения:

об автомашинах: номер машины, марка, пробег на момент приобретения, грузоподъемность;

о водителях: табельный номер, фамилия водителя, категория, стаж, адрес, год рождения;

о выполненных заказах: дата, фамилия водителя, номер машины, километраж, масса груза, стоимость перевозки.

1.3 Функциональные требования к системе

1. Вход по логину и паролю;
2. Разделение привилегий для авторизованных и нет пользователей;
3. Интерфейсы просмотра для 5 таблиц;
4. Интерфейсы добавления данных о заказах, клиентах и путевых листах;
5. Интерфейсы редактирования для вышеуказанных данных;
6. Интерфейсы удаления для вышеуказанных данных;
7. Сортировка данных по полям в таблицах;
8. Упаковка в Docker.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

В процессе проектирования базы данных была использована методология ERD (Entity Relationship Diagrams – диаграмма «сущность-связь») и соответствующая ей нотация.

при анализе предметной области были выделены следующие сущности, обозначенные в инфологической модели на рисунке А.1 в приложении А:

- модели автомобилей,
- автомобили,
- водители,
- путевые листы,
- заказы,
- клиенты,
- состав заказа,
- товары.

Логическая модель соответствует первой нормальной форме, так как все атрибуты атомарные. Описание логической модели в приложении А в таблицах А.1-А.8.

3 ТЕХНОЛОГИИ И ПРОГРАММНЫЕ СРЕДСТВА

3.1 Используемые технологии

При разработке системы были использованы следующие технологии:

- технология баз данных - SQLite,
- веб-технологии - node.js, Rest.

3.2 Программные средства

При проектировании базы данных был использован MySQL Workbench 8.0 CE – унифицированный визуальный инструмент для разработки и администрирования баз данных.

Для реализации системы были использованы следующие программные средства:

- JetBrains PyCharm 2019.2.4 x64 – IDE для профессиональной разработки на Python
- Docker – открытая платформа для разработки, доставки и запуска приложений.

3.3 Шаблон проектирования MVT

Фреймворк Django реализует архитектурный паттерн Model-View-Template или сокращенно MVT, который является модификацией распространенного в веб-программировании паттерна MVC (Model-View-Controller).

На рисунке 1 представлена архитектура MVT в Django.

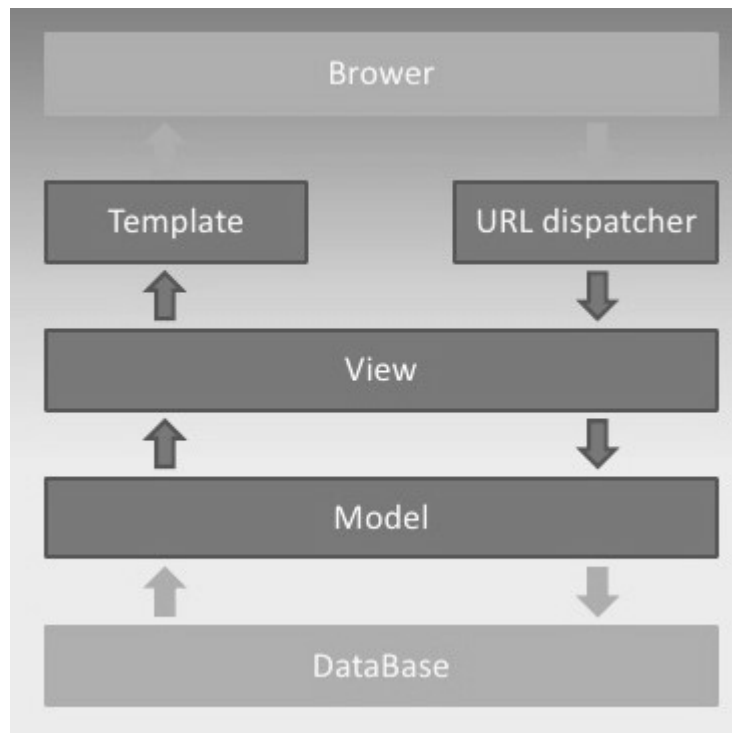


Рисунок 1 – Архитектура MVT

URL dispatcher на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.

View получает запрос, обрабатывает его и отправляет пользователю некоторый ответ. Если для обработки запроса необходимо обращение к модели и базе данных, то View взаимодействует с ними. Для создания ответа может применять Template или шаблоны.

Model: описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.

Template: представляет логику представления в виде сгенерированной разметки html.

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Для реализации системы была описана модель данных на языке Python с использованием фреймворка Django и DjangoRestFramework, представленная в листинге Б.1 приложения Б.

Затем были описаны все методы взаимодействия внешнего пользователя с базой данных через views и serializers (листинг Б.2), а именно методы API: GET (получение данных), POST (вставка данных), PUT (редактирование данных) и DELETE (удаление данных), представленные в листинге Б.3 приложения Б.

После чего были разработаны пользовательские интерфейсы. Программный код представлен в приложении Б.

На рисунке 1 представлен интерфейс авторизации.

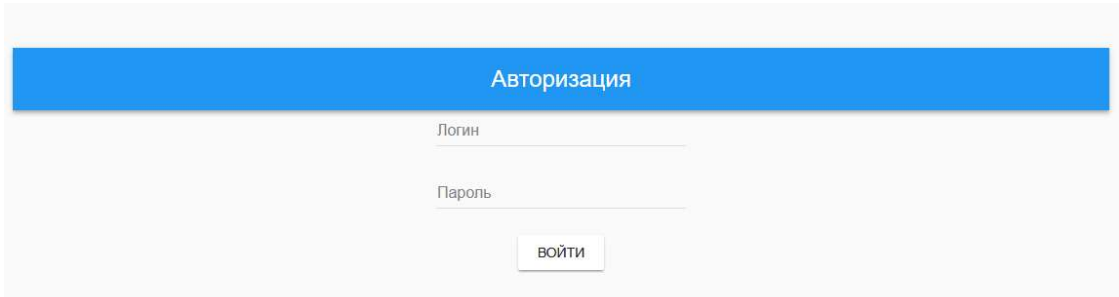


Рисунок 1 – Авторизация

Так как неавторизованный пользователь не может увидеть всю информацию, представленную в системе, в интерфейсе главной страницы было предусмотрено разграничение прав. Главная страница для неавторизованных пользователей представлена на рисунке 2, для авторизованных – на рисунке 3.

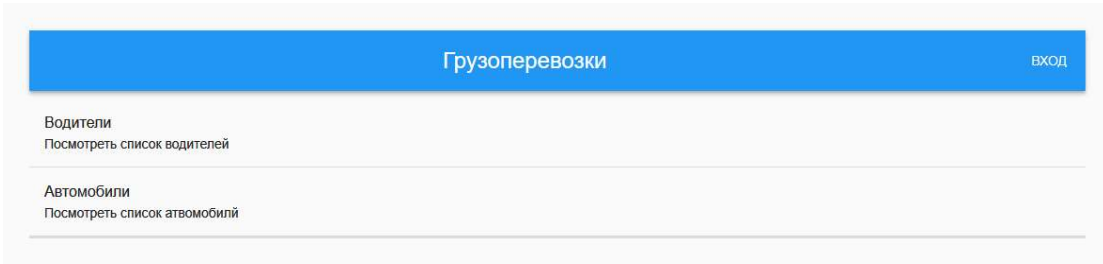


Рисунок 2 – Вид главной страницы без авторизации

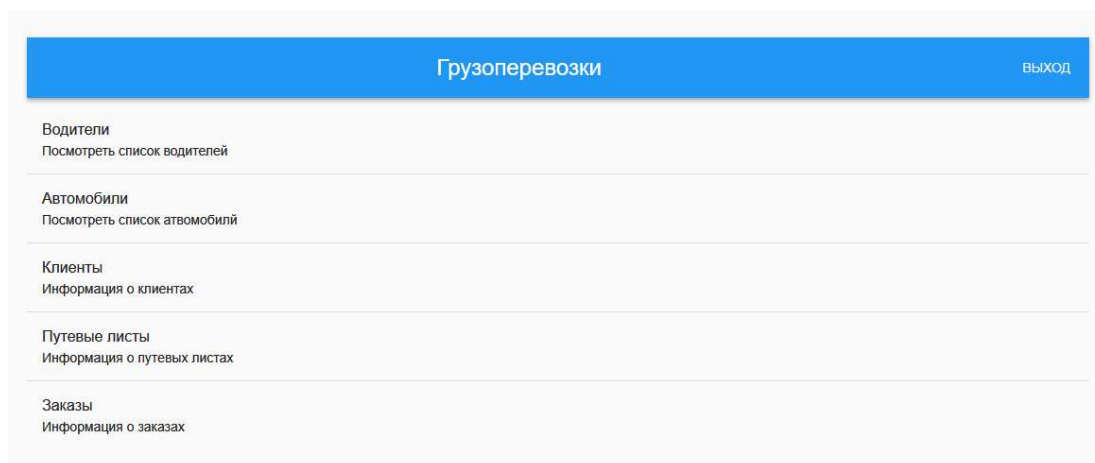


Рисунок 3 – Вид главной страницы с авторизацией

Для неавторизованных пользователей доступна информация только о водителях и автомобилях, зарегистрированных в системе. Так как это общедоступная информация, то ее изменение возможно только на сервере администратором. На рисунках 4 и 5 представлены страницы с информацией о водителях и автомобилях.

Водители						ГЛАВНАЯ
Фамилия	Имя	Категория	Опыт	Год рождения	Адрес	
Bagrova	Polina	B	5	1995	SPb, Narodnaya 65, 20	
Rashevskii	Vyacheslav	B	7	1993	SPb, Gorohovaya 65	
Bulkin	Petr	C	5	1994	SPb, Liteinii 78	

Рисунок 4 – Страница с данными о водителях

Автомобили			ГЛАВНАЯ
Государственный номер	Модель	Пробег	
T645AK	XC60	12000	
B897OT	Polo	5000	
K645OT	Polo	18200	
B451MC	XC60	7500	

Рисунок 5 – Страница с данными об автомобилях

Для авторизованных пользователей доступна информация о клиентах, заказах, путевых листах, а также функции добавления, редактирования и удаления.

На рисунках 6-8 представлены интерфейсы с данными о клиентах. Редактирование реализовано путем диалогового окна, появляющегося при нажатии на клиента. В этом же окне реализовано удаление.

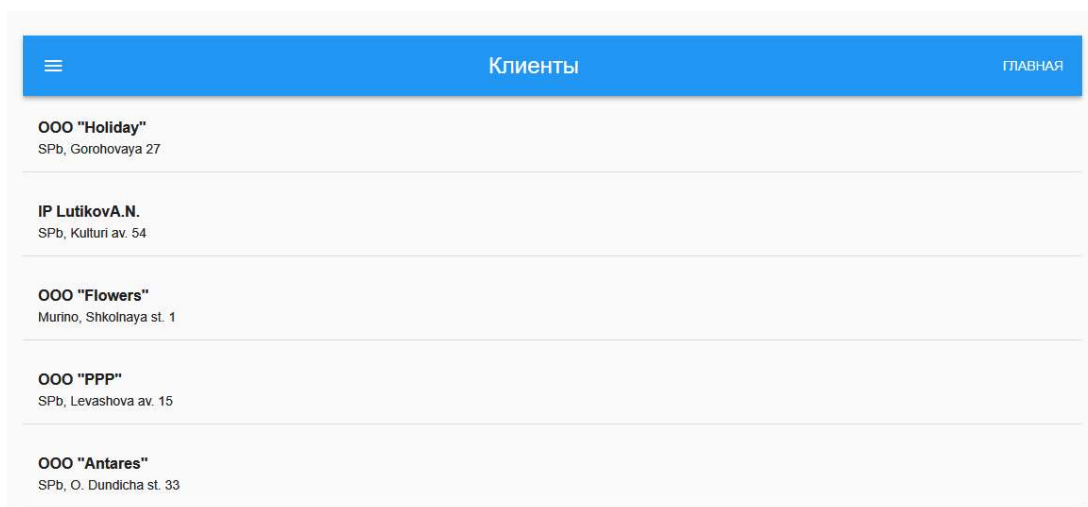


Рисунок 6 – Просмотр данных о клиентах

A screenshot of a form for adding a new client. The form has two input fields: 'Название' (Name) and 'Адрес' (Address). Below the 'Адрес' field is a green button labeled 'ДОБАВИТЬ' (Add).

Рисунок 7 – Добавление данных о клиентах

A screenshot of a dialog window titled 'Редактирование' (Editing). The dialog contains two input fields: the first contains 'ООО "Antares"' and the second contains 'SPb, O. Dundicha st. 33'. At the bottom of the dialog are three buttons: a green 'СОХРАНИТЬ' (Save) button, a red 'УДАЛИТЬ' (Delete) button, and a blue 'ЗАКРЫТЬ' (Close) button.

Рисунок 8 – Редактирование и удаление данных о клиентах

Для удобства восприятия информация о заказах представлена в таблице. При нажатии на строку появляются кнопки редактирования и удаления. Интерфейс просмотра представлен на рисунке 9.

Направление	Второй пункт	Километраж	Вес	Стоимость
To client	LO, Nikolskoe, Kirpichnii zavod	45	160	550
To client	Murino, st	51	49.1	600
From client	Komarovo	85	55	2554
From client	Gatchina	55	30	1000

Рисунок 9 – Просмотр данных о заказах

Так как данных о заказе много, то интерфейс добавления реализован через диалоговое окно с функцией скроллинга (рис.10).

Добавление заказа

Клиент

Направление

Второй пункт

Километраж

Вес груза

ДОБАВИТЬ ЗАКРЫТЬ

Рисунок 10 – Интерфейс добавления данных о заказах

Редактирование реализовано аналогично добавлению (рис.11).

Редактирование заказа

Клиент

IP LutikovA.N.

Направление

From client

Второй пункт

Gatchina

Километраж

55

Вес груза

30

СОХРАНИТЬ

ЗАКРЫТЬ

Рисунок 11 – Интерфейс редактирования данных о заказах

Интерфейсы с данными аналогичны интерфейсам заказов (рис.12-14). На рисунке 15 представлен дополнительный интерфейс, который позволяет посмотреть прикрепленные к путевому листу заказы.

Путевые листы			ГЛАВНАЯ
ДОБАВИТЬ			
Дата	Водитель	Автомобиль	
2020-05-14	Bagrova Polina	T645AK	
СПИСОК ЗАКАЗОВ		РЕДАКТИРОВАТЬ	УДАЛИТЬ
2020-05-15	Bagrova Polina	T645AK	
2020-05-02	Bulkin Petr	B897OT	
2020-05-13	Bulkin Petr	K645OT	
2020-06-02	Bulkin Petr	B897OT	
2020-06-03	Bulkin Petr	B897OT	

Рисунок 12 – Интерфейс просмотра данных о путевых листах

Добавление путевого листа происходит на пользователя, который вошел в систему.

Добавление путевого листа

Путевой лист будет оформлен на данного пользователя.

Дата

Автомобиль

ДОБАВИТЬ

ЗАКРЫТЬ

Рисунок 13 – Интерфейс добавления данных о путевых листах

Редактирование путевого листа

Дата

2020-05-14

Водитель

Bagrova

Автомобиль

T645AK

СОХРАНИТЬ

ЗАКРЫТЬ

Рисунок 14 – Интерфейс редактирования данных о путевых листах

Заказы

Направление	Второй пункт	Километраж	Вес	Стоимость
To client	LO, Nikolskoe, Kipichnii zavod	45	160	550

ЗАКРЫТЬ

Рисунок 15 – Интерфейс просмотра прикрепленных заказов

После любого действия, изменяющего данные в системе появляется уведомление об успешности выполнения или об ошибке аналогичное представленному на рисунке 16.

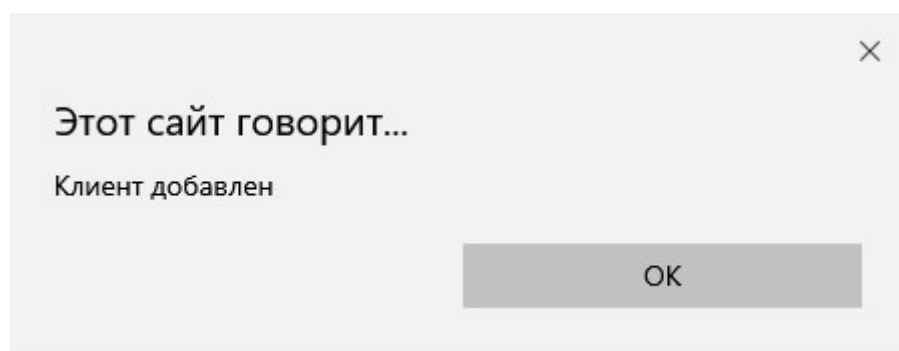


Рисунок 16 - Уведомление

При выходе пользователя из системы, главная страница перезагружается и скрывает данные, недоступные для неавторизованного пользователя.

ЗАКЛЮЧЕНИЕ

В ходе изучения учебной практики по профессиональному модулю ПМ.11 «Разработка, администрирование и защита баз данных» были углублены знания и получен начальный опыт по основным видам деятельности «Разработка, администрирование и защита баз данных». Был разработан прототип веб-приложения по заданной предметной области, использующего реляционную базу данных.

В ходе разработки была спроектирована база данных, которая описана в модели данных приложения, описаны методы взаимодействия с базой данных, описана внешняя оболочка веб-приложения. Готовый прототип был упакован в Docker.

СПИСОК ЛИТЕРАТУРЫ

1. Методология проектирования баз данных [Электронный ресурс] // Библиотека Карагандинского Государственного Технического Университета URL: http://lib.kstu.kz:8300/tb/books/Baz@i_dann@ih_Rad@mko/Theory/10_2.htm (дата обращения: 25.06.2020)
2. MySQL Workbench: Visual Database Design [Электронный ресурс] // MySQL URL: <https://www.mysql.com/products/workbench/design/> (дата обращения: 25.06.2020)
3. Информатика – курс лекций [Электронный ресурс] // Учебно-методический центр кафедры информационных технологий URL: http://dit.isuct.ru/IVT/sitanov/Literatura/InformLes_4.html (дата обращения: 25.06.2020)
4. PyCharm [Электронный ресурс] // JetBrains URL: <https://www.jetbrains.com/ru-ru/pycharm/> (дата обращения: 25.06.2020)
5. Get Docker [Электронный ресурс] // docker docs URL: <https://docs.docker.com/get-docker/> (дата обращения: 25.06.2020)
6. Введение в Django [Электронный ресурс] // METANIT.COM сайт о программировании URL: <https://metanit.com/python/django/1.1.php> (дата обращения: 25.06.2020)
7. Django rest framework [Электронный ресурс] // YouTube URL: <https://www.youtube.com/playlist?list=PLF-NY6ldwAWqP9PqPU3LA7mX2KJVyLhC> (дата обращения: 07.06.2020)
8. Создание Django API используя Django Rest Framework часть 1 [Электронный ресурс] // Еще один блог веб разработчика URL: <https://webdevblog.ru/sozдание-django-api-ispolzuya-django-rest-framework-apiview/> (дата обращения: 11.06.2020)
9. Создание Django API используя Django Rest Framework часть 2 [Электронный ресурс] // Еще один блог веб разработчика URL:

<https://webdevblog.ru/sozdanie-django-api-ispolzuya-djangorestframework-chast-2/> (дата обращения: 11.06.2020)

10. Muse-UI [Электронный ресурс] Muse-UI URL: <https://muse-ui.org/#/en-US> (дата обращения: 11.06.2020)

ПРИЛОЖЕНИЕ А

На рисунке А.1 представлена инфологическая схема спроектированной базы данных.

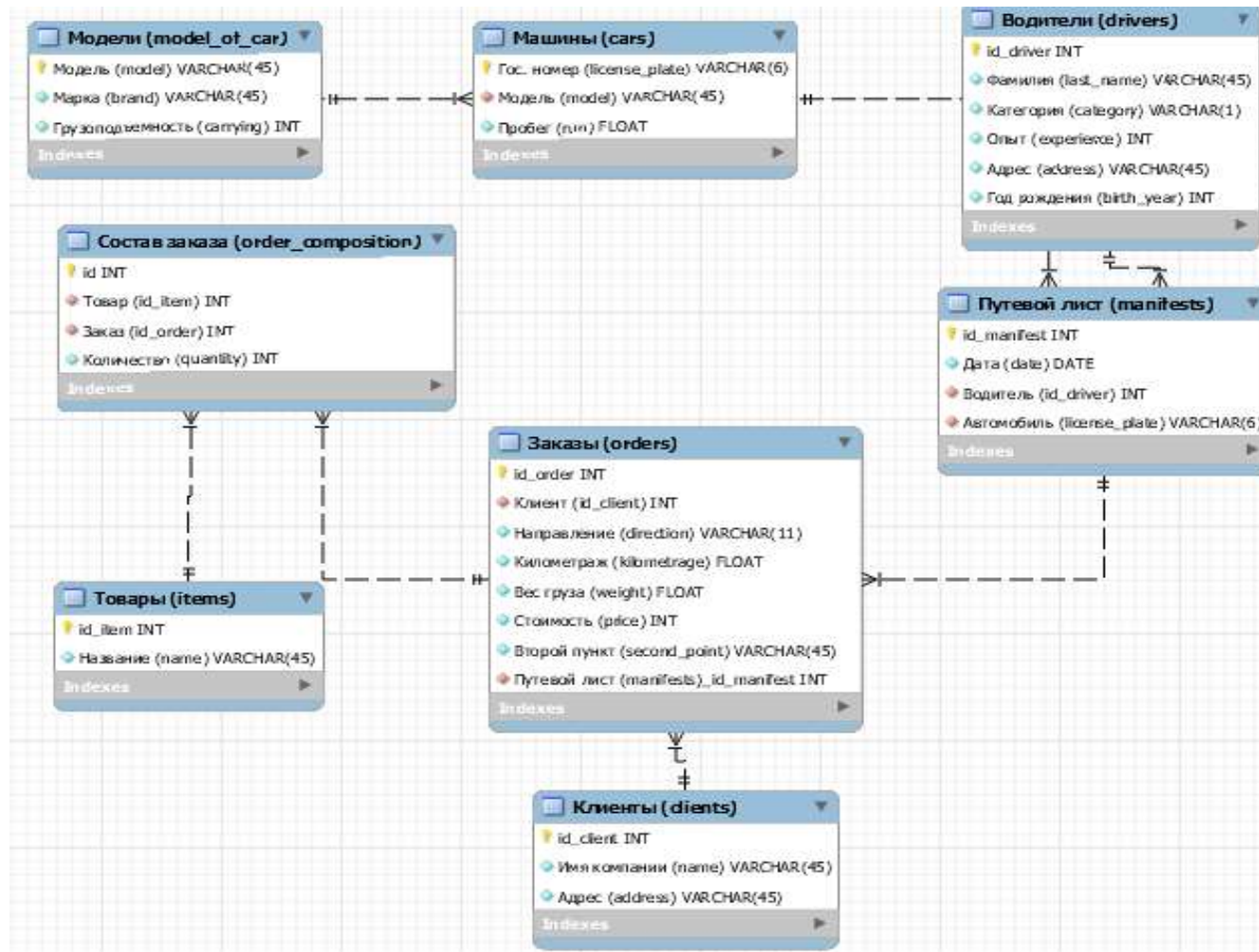


Рисунок А.1 – Инфологическая схема

В таблицах А.1-А.8 представлены описания всех полей таблиц.

Таблица А.1 – model_of_car (Модель автомобиля)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
model	VARCHAR(45)	+	+	-	Уникален, буквы, цифры, дефис	Модель автомобиля	Solaris
brand	VARCHAR(45)	+	-	-	Строковое значение	Брэнд автомобиля	Hyundai
carrying	INT	+	-	-	Положительное число	Грузоподъемность в кг	0.5

Таблица А.2 – auto (автомобиль)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
license_plate	VARCHAR(6)	+	+	-	Уникален, буквы и цифры	Номер автомобиля	M345AT
model	VARCHAR(45)	+	-	+	Строковое значение	Модель автомобиля	Solaris
run	FLOAT	+	-	-	Положительное число	Пробег на момент покупки в км	12000

Таблица A.3 – driver (водитель)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
id_driver	INT	+	+	-	Уникален, целое положительное число	Табельный номер	1
last_name	VARCHAR(45)	+	-	-	Строковое значение	Фамилия	Petrov
category	VARCHAR(1)	+	-	-	Буквы (A, B, C, D, E, M)	Категория водителя	B
experience	INT	+	-	-	Целое число ≥ 0	Опыт вождения	5
address	VARCHAR(45)	+	-	-	Буквы и цифры	Адрес	Narodnaya st 14
birth_year	INT	+	-	-	Целое положительное число, < 2002	Год рождения	1997

Таблица A.4 – manifests (Путевой лист)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
1	2	3	4	5	6	7	8
id_manifest	INT	+	+	-	Уникален, целое положительное число	Идентификатор	1
date	DATE	+	-	-	Формат YYYY-MM-DD	Дата совершения поездок	2020-02-25

Продолжение таблицы А.4

1	2	3	4	5	6	7	8
li- cense_plate	VARCHAR(6)	+	-	+	Буквы и цифры	Номер машины	M345AT
id_driver	INT	+	-	+	Целое положительное число	Табельный номер водителя	3

Таблица А.5 – orders (заказы)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
id_order	INT	+	+	-	Уникален, целое положительное число	Идентификатор	1
client	INT	+	-	+	Целое положительное число	Идентификатор клиента	1
direction	VAR- CHAR(11)	+	-	-	Выбор из 'To client', 'From client'	Направление поездки	To client
kilometrage	FLOAT	+	-	-	Положительное число	Километраж	500
weight	FLOAT	+	-	-	Положительное число	Масса груза в кг	5
price	INT	+	-	-	Целое положительное число	Стоимость перевозки	6000
second_point	VAR- CHAR(45)	+	-	-	Буквы и цифры	Адрес	Narodnaya st 14
id_manifest	INT	+	-	+	Целое положительное число	Идентификатор путевого листа	3

Таблица A.6 – clients (клиенты)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
id_client	INT	+	+	-	Уникален, целое положительное число	Идентификатор	1
name	VAR-CHAR(45)	+	-	-	Строковое значение	Имя компании	ООО “PPP”
address	VAR-CHAR(45)	+	-	-	Буквы и цифры	Адрес клиента	Narodnaya st. 14

Таблица A.7 – order_composition (состав заказа)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
id	INT	+	+	-	Уникален, целое положительное число	Идентификатор	1
id_item	INT	+	-	+	Положительное число	Идентификатор товара	1
id_order	INT	+	-	+	Положительное число	Идентификатор заказа	1
quantity	INT	+	-	-	Положительное число	Количество товара	2

Таблица A.8 – items (товары)

Имя поля	Тип данных	Обязательность	Первичный ключ	Внешний ключ	Ограничения	Пояснения	Пример заполнения поля
id_item	INT	+	+	-	Уникален, целое положительное число	Идентификатор	1
name	VAR-CHAR(45)	+	-		Строковое значение	Название товара	Carpet

ПРИЛОЖЕНИЕ Б

В листингах Б.1-Б.10 представлен программный код разработки.

Листинг Б.1 – Модель данных

```
from django.db import models
from django.contrib.auth.models import User

# from djoser.urls.base import User

class Model_of_car(models.Model):
    """Модель автомобиля"""
    model = models.CharField("Модель", max_length=45,
primary_key=True)
    brand = models.CharField("Брэнд", max_length=45)
    carrying = models.IntegerField("Грузоподъемность")

class Cars(models.Model):
    """Автомобили"""
    license_plate = models.CharField(max_length=6, pri-
mary_key=True)
    model = models.ForeignKey(Model_of_car, on_de-
lete=models.CASCADE)
    run = models.FloatField()

class Drivers(models.Model):
    """Водители"""
```

```

CATEGORY = [
    ('A', 'Motorcycle'),
    ('B', 'Car'),
    ('C', 'Truck'),
    ('D', 'Bus'),
    ('M', 'Moped'),
]

last_name = models.CharField(max_length=45)
first_name = models.CharField(max_length=45,
null=True)
category = models.CharField(max_length=1,
choices=CATEGORY, default='B')
experience = models.IntegerField()
address = models.CharField(max_length=100)
birth_year = models.IntegerField()
user = models.ForeignKey(User, null=True, on_delete=models.CASCADE)

```

```

class Manifests(models.Model):
    """Путевые листы"""
    date = models.DateField()
    driver = models.ForeignKey(Drivers, on_delete=models.CASCADE)
    car = models.ForeignKey(Cars, on_delete=models.CASCADE)

```

```

class Clients(models.Model):
    """Клиенты"""

```

```

name = models.CharField(max_length=45)
address = models.CharField(max_length=45)

class Items(models.Model):
    """Товары"""
    name = models.CharField(max_length=45)

class Orders(models.Model):
    """Заказы"""
    DIRECT = [
        ('From client', 'From client'),
        ('To client', 'To client'),
    ]
    client = models.ForeignKey(Clients, on_delete=models.CASCADE)
    direction = models.CharField(max_length=11,
choices=DIRECT, default='From client')
    kilometrage = models.FloatField()
    weight = models.FloatField()
    price = models.FloatField()
    second_point = models.CharField(max_length=45)
    manifest = models.ForeignKey(Manifests, null=True,
on_delete=models.CASCADE)

class Order_composition(models.Model):
    """Состав заказа"""

```

```

        item = models.ForeignKey(Items, on_delete=models.CASCADE)
        order = models.ForeignKey(Orders, on_delete=models.CASCADE)
        quantity = models.IntegerField()

```

Листинг Б.2 – Сериализация модели данных

```

from rest_framework import serializers

from autopark.models import *
from django.contrib.auth.models import User

class UserSerializer(serializers.ModelSerializer):
    """Сериализация пользователя"""

    class Meta:
        model = User
        fields = ("id", "username")

class ModelsSerializer(serializers.ModelSerializer):
    """Сериализация моделей автомобилей"""

    class Meta:
        model = Model_of_car
        fields = ("model", "brand")

class CarsSerializer(serializers.ModelSerializer):
    """Сериализация машин"""

```

```

class Meta:
    model = Cars
    fields = ("license_plate", "model", "run")

class ClientsSerializer(serializers.ModelSerializer):
    """Сериализация клиентов"""

    class Meta:
        model = Clients
        fields = ("id", "name", "address")

    def update(self, instance, validated_data):
        instance.name = validated_data.get('name', instance.name)
        instance.address = validated_data.get('address', instance.address)
        instance.save()
        return instance

class DriversSerializer(serializers.ModelSerializer):
    """Сериализация водителя"""
    user = UserSerializer()

    class Meta:
        model = Drivers
        fields = ("last_name", "first_name", "category", "experience", "address", "birth_year", "user")

```

```

class DriverNameSerializer(serializers.ModelSerializer):
    """Сериализация имени водителя"""

    class Meta:
        model = Drivers
        fields = ("last_name", "first_name")

class ManifestsSerializer(serializers.ModelSerializer):
    """Сериализация путевых листов"""
    driver = DriverNameSerializer()

    class Meta:
        model = Manifests
        fields = ("id", "date", "driver", "car")

    def update(self, instance, validated_data):
        instance.date = validated_data.get('date', instance.date)
        instance.driver = validated_data.get('driver', instance.driver)
        instance.car = validated_data.get('car', instance.car)
        instance.save()
        return instance

```

```

class ManifestsPostSerializer(serializers.ModelSerializer):
    """Сериализация путевых листов для POST-запроса"""

    class Meta:
        model = Manifests
        fields = ("date", "car")

class OrdersSerializer(serializers.ModelSerializer):
    """Сериализация заказов"""

    class Meta:
        model = Orders
        fields = ("id", "client", "direction", "kilometrage", "weight", "price", "second_point", "manifest")

    def update(self, instance, validated_data):
        instance.client = validated_data.get('client',
instance.client)
        instance.direction = validated_data.get('direction', instance.direction)
        instance.kilometrage = validated_data.get('kilometrage', instance.kilometrage)
        instance.weight = validated_data.get('weight', instance.weight)
        instance.price = validated_data.get('price', instance.price)
        instance.second_point = validated_data.get('second_point', instance.second_point)

```



```

        instance.manifest = validated_data.get('manifest', instance.manifest)
        instance.save()
        return instance

```

```

class ItemsSerializer(serializers.ModelSerializer):

```

```

    """Сериализация товаров"""

```

```

    class Meta:

```

```

        model = Items

```

```

        fields = ("name",)

```

```

class OrderCompositionSerializer(serializers.ModelSerializer):

```

```

    """Сериализация состава заказа"""

```

```

    item = ItemsSerializer()

```

```

    class Meta:

```

```

        model = Order_composition

```

```

        fields = ("item", "quantity")

```

Листинг Б.3 – Файл views.py

```

from django.shortcuts import render
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import permissions
from rest_framework.generics import get_object_or_404

```

```

from autopark.serializers import *

```

```

class DriversView(APIView):
    """Водители"""

    permission_classes = [permissions.AllowAny, ]

    def get(self, request):
        drivers = Drivers.objects.all()
        serializer = DriversSerializer(drivers,
many=True)
        return Response({"data": serializer.data})

class CarsView(APIView):
    """Машины"""

    permission_classes = [permissions.AllowAny, ]

    def get(self, request):
        cars = Cars.objects.all()
        serializer = CarsSerializer(cars, many=True)
        return Response({"data": serializer.data})

class ClientsView(APIView):
    """Клиенты"""

    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request, pk=None):

```

```

        clients = Clients.objects.all()
        if pk is not None:
            clients = Clients.objects.filter(pk=pk)
        serializer = ClientsSerializer(clients,
many=True)
        return Response({"data": serializer.data})

def post(self, request):
    client = ClientsSerializer(data=request.data)
    if client.is_valid():
        client.save()
        return Response(status=201)
    else:
        return Response(status=400)

def put(self, request, pk):
    saved_client = get_object_or_404(Clients.objects.all(), pk=pk)
    data = request.data
    serializer = ClientsSerializer(instance=saved_client, data=data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response(status=201)
    else:
        return Response(status=400)

def delete(self, request, pk):
    client = get_object_or_404(Clients.objects.all(), pk=pk)

```

```

        client.delete()

        return Response(status=204)

class ManifestsView(APIView):
    """Путевой лист и номера заказов для него"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request, pk=None):
        if pk is not None:
            manifest = Manifests.objects.filter(pk=pk)
            serializerManifest = ManifestsSerializer(
                manifest, many=True)
            order = Orders.objects.filter(manifest_id=pk)
            serializerOrder = OrdersSerializer(order,
                many=True)

            return Response({"manifest": serializerManifest.data, "orders": serializerOrder.data})

        manifests = Manifests.objects.all()
        serializerManifest = ManifestsSerializer(manifests, many=True)

        return Response({"manifest": serializerManifest.data})

    def post(self, request):
        id = User.objects.values_list('id', flat=True).get(username=request.user)
        driver = Drivers.objects.values_list('id', flat=True).get(user=id)

```

```

        manifest = ManifestsPostSerializer(data=request.data)

        if manifest.is_valid():
            manifest.save(driver_id=driver)
            return Response(status=201)
        else:
            return Response(status=400)

    def put(self, request, pk):
        saved_manifest = get_object_or_404(Manifests.objects.all(), pk=pk)
        data = request.data
        serializer = ManifestsSerializer(instance=saved_manifest, data=data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(status=201)
        else:
            return Response(status=400)

    def delete(self, request, pk):
        manifest = get_object_or_404(Manifests.objects.all(), pk=pk)
        manifest.delete()
        return Response(status=204)

class OrdersView(APIView):
    """Заказы и их состав"""
    permission_classes = [permissions.IsAuthenticated, ]

```

```

def get(self, request, pk=None):
    orders = Orders.objects.all()
    if pk is not None:
        orders = Orders.objects.filter(pk=pk)
        serializerOrder = OrdersSerializer(orders,
many=True)
        client = Clients.objects.filter(id=Orders.objects.values_list('client',
flat=True).get(pk=pk))
        serializerClient = ClientsSerializer(client,
many=True)
        composition = Order_composition.objects.filter(order_id=pk)
        serializerComposition = OrderCompositionSerializer(composition, many=True)
        return Response(
            {"orders": serializerOrder.data,
"items": serializerComposition.data, "client": serializerClient.data})
        serializerOrder = OrdersSerializer(orders,
many=True)
        return Response({"orders": serializerOrder.data})

def post(self, request):
    order = OrdersSerializer(data=request.data)
    if order.is_valid():
        order.save()
        return Response(status=201)

```

```

        else:
            return Response(status=400)

    def put(self, request, pk):
        saved_order = get_object_or_404(Orders.objects.all(), pk=pk)
        data = request.data
        serializer = OrdersSerializer(instance=saved_order, data=data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(status=201)
        else:
            return Response(status=400)

    def delete(self, request, pk):
        order = get_object_or_404(Orders.objects.all(), pk=pk)
        order.delete()
        return Response(status=204)

```

Листинг Б.4 – Учетная запись администратора

```

from django.contrib import admin
from autopark.models import *

class DriversAdmin(admin.ModelAdmin):
    list_display = ("last_name", "first_name", "category", "user")

```

```

class CarsAdmin(admin.ModelAdmin):
    list_display = ("license_plate", "model")

class ManifestsAdmin(admin.ModelAdmin):
    list_display = ("id", "date", "driver", "car")

class ClientsAdmin(admin.ModelAdmin):
    list_display = ("name", "address")

class ModelsAdmin(admin.ModelAdmin):
    list_display = ("model", "brand")

class OrdersAdmin(admin.ModelAdmin):
    list_display = ("id", "client", "direction", "second_point")

class ItemsAdmin(admin.ModelAdmin):
    list_display = ("name",)

class OrderCompositionAdmin(admin.ModelAdmin):
    list_display = ("order", "item", "quantity")

admin.site.register(Drivers, DriversAdmin)

```



```

admin.site.register(Cars, CarsAdmin)
admin.site.register(Model_of_car, ModelsAdmin)
admin.site.register(Manifests, ManifestsAdmin)
admin.site.register(Orders, OrdersAdmin)
admin.site.register(Clients, ClientsAdmin)
admin.site.register(Items, ItemsAdmin)
admin.site.register(Order_composition,      OrderComposi-
tionAdmin)

```

Листинг Б.5 – urls.py приложения

```

from django.urls import path
from autopark.views import *

urlpatterns = [
    path('drivers/', DriversView.as_view()),
    path('cars/', CarsView.as_view()),
    path('clients/', ClientsView.as_view()),
    path('clients/<int:pk>', ClientsView.as_view()),
    path('manifests/', ManifestsView.as_view()),
    path('manifests/<int:pk>',          Mani-
festView.as_view()),
    path('orders/', OrdersView.as_view()),
    path('orders/<int:pk>', OrdersView.as_view()),
]

```

Листинг Б.6 – urls.py проекта

```

"""laboratiry_work_1 URL Configuration

The `urlpatterns` list routes URLs to views."""
from django.contrib import admin
from django.urls import path, include

```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('autopark/', include("autopark.urls")),
    path('auth/', include('djoser.urls')),
    path('auth/', include('djoser.urls.authtoken')),
    path('auth/', include('djoser.urls.jwt')),
]
```

Листинг Б.7 – Объявление страниц во внешней оболочке

```
import Vue from 'vue'
import Router from 'vue-router'
import Autopark from '../components/Autopark'
import Login from "../components/Login"
import Clients from "../components/Clients"
import Drivers from "../components/Drivers"
import Cars from "../components/Cars"
import Manifests from "../components/Manifests"
import Orders from "../components/Orders"

Vue.use(Router);

export default new Router({
  routes: [
    {
      path: '/',
      name: 'Autopark',
      component: Autopark
    },
    {
      path: '/login',
```

```

        name: 'Login',
        component: Login
    },
    {
        path: '/clients',
        name: 'Clients',
        component: Clients
    },
    {
        path: '/drivers',
        name: 'Drivers',
        component: Drivers
    },
    {
        path: '/cars',
        name: 'Cars',
        component: Cars
    },
    {
        path: '/manifests',
        name: 'Manifests',
        component: Manifests
    },
    {
        path: '/orders',
        name: 'Orders',
        component: Orders
    },
]
}))

```

```
<template>
  <mu-container>
    <mu-appbar style="width: 100%;" color="primary">
      Грузоперевозки
      <mu-button flat v-if="!auth" slot="right"
@click="goLogin">Вход</mu-button>
      <mu-button flat v-else slot="right" @click="log-
out">Выход</mu-button>
    </mu-appbar>

    <mu-list textline="two-line">
      <mu-list-item avatar :ripple="false" button
@click="goDrivers">
        <mu-list-item-content>
          <mu-list-item-title>Водители</mu-list-item-
title>
          <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">Посмотреть список водителей</mu-list-item-
sub-title>
        </mu-list-item-content>
      </mu-list-item>
      <mu-divider></mu-divider>

      <mu-list-item avatar :ripple="false" button
@click="goCars">
        <mu-list-item-content>
          <mu-list-item-title>Автомобили</mu-list-item-
title>
```

```

        <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">Посмотреть список автомобилей
    </mu-list-item-sub-title>
</mu-list-item-content>
</mu-list-item>
<mu-divider></mu-divider>

    <mu-list-item v-if="auth" avatar :ripple="false"
button @click="goClients">
        <mu-list-item-content>
            <mu-list-item-title>Клиенты</mu-list-item-ti-
tle>
                <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">Информация о клиентах</mu-list-item-sub-ti-
tle>
            </mu-list-item-content>
        </mu-list-item>
    <mu-divider></mu-divider>

    <mu-list-item v-if="auth" avatar :ripple="false"
button @click="goManifests">
        <mu-list-item-content>
            <mu-list-item-title>Путевые листы</mu-list-
item-title>
                <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">Информация о путевых листах</mu-list-item-
sub-title>
            </mu-list-item-content>
        </mu-list-item>
    <mu-divider></mu-divider>

```

```

        <mu-list-item v-if="auth" avatar :ripple="false"
button @click="goOrders">
        <mu-list-item-content>
            <mu-list-item-title>Заказы</mu-list-item-ti-
tle>
            <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">Информация о заказах</mu-list-item-sub-ti-
tle>
        </mu-list-item-content>
    </mu-list-item>
</mu-list>
</mu-container>
</template>

```

```

<script>
    export default {
        name: "Autopark",
        computed: {
            auth() {
                if (sessionStorage.getItem("auth_to-
ken")) {
                    return true
                }
            },
        },
        methods: {
            goLogin() {
                this.$router.push({name: "Login"})
            },
        },
    }

```

```

        logout() {
            sessionStorage.removeItem("auth_token")
            window.location = '/'
        },
        goDrivers() {
            this.$router.push({name: "Drivers"})
        },
        goCars() {
            this.$router.push({name: "Cars"})
        },
        goClients() {
            this.$router.push({name: "Clients"})
        },
        goManifests() {
            this.$router.push({name: "Manifests"})
        },
        goOrders() {
            this.$router.push({name: "Orders"})
        },
    },
}
</script>

```

```

<style scoped>

```

```

</style>

```

Листинг Б.9 - Авторизация

```

<template>

```

```

  <mu-container>

```

```

    <mu-appbar style="width: 100%;" color="primary">

```

```

        Авторизация
    </mu-appbar>

    <mu-text-field v-model="login" type="text" place-
holder="Логин"></mu-text-field> <br>

    <mu-text-field v-model="password" type="password"
placeholder="Пароль"></mu-text-field> <br>

    <mu-button @click="setLogin">Войти</mu-button>
</mu-container>
</template>

<script>
    import $ from 'jquery'

    export default {
        name: "Login",
        data() {
            return {
                login: '',
                password: '',
            }
        },
        methods: {
            setLogin() {
                $.ajax({
                    url:
"http://127.0.0.1:8000/auth/token/login/",
                    type: "POST",
                    data: {
                        username: this.login,
                        password: this.password,

```



```

        },
        success: (response) => {
            alert("Успешный вход")
            session-
Storage.setItem("auth_token",      response.data.attrib-
utes.auth_token)
            this.$router.push({name:      "Au-
topark"})
        },
        error: (response) => {
            if (response.status === 400) {
                alert("Логин или пароль не
верен")
            }
        },
    })
},
}
}
</script>

```

```

<style scoped>

</style>

```

Листинг Б.10 – Меню

```

<template>
  <mu-menu slot="left">
    <mu-button icon flat color="primary">
      <mu-icon value="menu"></mu-icon>
    </mu-button>

```

```

    <mu-list slot="content">
      <mu-list-item button @click="goDrivers">
        <mu-list-item-title>Водители</mu-list-item-ti-
title>
      </mu-list-item>
      <mu-list-item button @click="goCars">
        <mu-list-item-title>Автомобили</mu-list-item-
title>
      </mu-list-item>
      <mu-list-item v-if="auth" button @click="goCli-
ents">
        <mu-list-item-title>Клиенты</mu-list-item-ti-
title>
      </mu-list-item>
      <mu-list-item v-if="auth" button @click="goMani-
fests">
        <mu-list-item-title>Путевые листы</mu-list-
item-title>
      </mu-list-item>
      <mu-list-item v-if="auth" button
@click="goOrders">
        <mu-list-item-title>Заказы</mu-list-item-title>
      </mu-list-item>
    </mu-list>
  </mu-menu>
</template>

<script>
  export default {
    name: "Menu",

```

```

        computed: {
            auth() {
                if (sessionStorage.getItem("auth_token")) {
                    return true
                }
            },
            methods: {
                goDrivers() {
                    this.$router.push({name: "Drivers"})
                },
                goCars() {
                    this.$router.push({name: "Cars"})
                },
                goClients() {
                    this.$router.push({name: "Clients"})
                },
                goManifests() {
                    this.$router.push({name: "Manifests"})
                },
                goOrders() {
                    this.$router.push({name: "Orders"})
                },
            },
        }
    }
</script>

<style scoped>

```

```
</style>
```

Листинг Б.11 – Страница с водителями

```
<template>
  <mu-container>
    <mu-appbar style="width: 100%;" color="primary">
      <Menu slot="left"></Menu>
      Водители
      <mu-button flat slot="right"
@click="goHome">Главная</mu-button>
    </mu-appbar>
    <mu-paper :z-depth="1" align="center">
      <mu-data-table :columns="columns"
:sort.sync="sort"
@sort-change="handleSortChange"
:data="drivers">
        <template slot-scope="scope">
          <td>{{scope.row.last_name}}</td>
          <td>{{scope.row.first_name}}</td>
          <td class="is-center">{{scope.row.cate-
gory}}</td>
          <td class="is-center">{{scope.row.experi-
ence}}</td>
          <td class="is-cen-
ter">{{scope.row.birth_year}}</td>
          <td class="is-center">{{scope.row.ad-
dress}}</td>
        </template>
      </mu-data-table>
    </mu-paper>
  </mu-container>
```

```
</template>
```

```
<script>
```

```
import $ from 'jquery'
```

```
import Menu from "../Menu";
```

```
export default {
```

```
  name: "Drivers",
```

```
  components: {
```

```
    Menu
```

```
  },
```

```
  data() {
```

```
    return {
```

```
      drivers: '',
```

```
      sort: {
```

```
        name: '',
```

```
        order: 'asc'
```

```
      },
```

```
      columns: [
```

```
        {title: 'Фамилия', name:
```

```
'last_name', width: 150, sortable: true},
```

```
        {title: 'Имя', name: 'first_name',
```

```
width: 150, sortable: true},
```

```
        {title: 'Категория', name: 'category', width: 150, align: 'center', sortable: true},
```

```
        {title: 'Опыт', name: 'experience', width: 150, align: 'center', sortable: true},
```

```
        {title: 'Год рождения', name: 'birth_year', width: 150, align: 'center', sortable: true},
```

```

                {title: 'Адрес', name: 'address',
width: 300, align: 'center'},
            ],
        },
    },
    created() {
        this.loadDrivers()
    },
    methods: {
        loadDrivers() {
            $.ajax({
                url: "http://127.0.0.1:8000/au-
topark/drivers/",
                type: "GET",
                success: (response) => {
                    this.drivers = re-
sponse.data.data
                }
            })
        },
        goHome() {
            this.$router.push({name: "Autopark"})
        },
        handleSortChange({name, order}) {
            if (name === 'category' || name ===
'last_name' || name === 'first_name') {
                this.drivers = this.driv-
ers.sort(function (a, b) {
                    var nameA = a[name].toLower-
Case(), nameB = b[name].toLowerCase()

```

```

        if (order === 'asc') {
            if (nameA < nameB) {
                return 1
            }
            if (nameA > nameB) {
                return -1
            }
            return 0
        } else {
            if (nameA < nameB)
                return -1
            if (nameA > nameB)
                return 1
            return 0
        }
    }
}

)
} else {
    this.drivers = this.drivers.sort((a,
b) => order === 'asc' ? b[name] - a[name] : a[name] -
b[name]);
    }
},
}
}
}
</script>

<style scoped>

</style>

```

```
<template>

  <mu-container>

    <mu-appbar style="width: 100%;" color="primary">
      <Menu slot="left"></Menu>
      Автомобили
      <mu-button flat slot="right"
@click="goHome">Главная</mu-button>
    </mu-appbar>

    <mu-paper :z-depth="1" align="center">
      <mu-data-table :columns="columns"
:sort.sync="sort"
@sort-change="handleSortChange"
:data="cars">
        <template slot-scope="scope">
          <td class="is-center">{{scope.row.li-
cense_plate}}</td>
          <td class="is-cen-
ter">{{scope.row.model}}</td>
          <td class="is-center">{{scope.row.run}}</td>
        </template>
      </mu-data-table>
    </mu-paper>
  </mu-container>
</template>

<script>
  import $ from 'jquery'
  import Menu from "../Menu";
```



```

export default {
  name: "Cars",
  components: {
    Menu
  },
  data() {
    return {
      cars: '',
      sort: {
        name: '',
        order: 'asc'
      },
      columns: [
        {
          title: 'Государственный номер',
          name: 'license_plate',
          width: 200,
          align: 'center',
          sortable: true
        },
        {title: 'Модель', name: 'model',
width: 150, align: 'center', sortable: true},
        {title: 'Пробег', name: 'run',
width: 150, align: 'center', sortable: true},
      ],
    }
  },
  created() {
    this.loadCars()
  }
}

```

```

    },
    methods: {
        loadCars() {
            $.ajax({
                url: "http://127.0.0.1:8000/autopark/cars/",
                type: "GET",
                success: (response) => {
                    this.cars = response.data.data
                }
            })
        },
        goHome() {
            this.$router.push({name: "Autopark"})
        },
        handleSortChange({name, order}) {
            if (name === 'license_plate' || name === 'model') {
                this.cars = this.cars.sort(function
(a, b) {
                    var nameA = a[name].toLowerCase(),
                    nameB = b[name].toLowerCase()
                    if (order === 'asc') {
                        if (nameA < nameB) {
                            return 1
                        }
                        if (nameA > nameB) {
                            return -1
                        }
                        return 0
                    }
                })
            }
        }
    }
}

```

```

        } else {
            if (nameA < nameB)
                return -1
            if (nameA > nameB)
                return 1
            return 0
        }
    }
}
)
} else {
    this.cars = this.cars.sort((a, b) =>
order === 'asc' ? b[name] - a[name] : a[name] - b[name]);
    }
},
}
}
</script>

```

```
<style scoped>
```

```
</style>
```

Листинг Б.13 – Страница с клиентами

```

<template>
  <mu-container>
    <mu-appbar style="width: 100%;" color="primary">
      <Menu slot="left"></Menu>
      Клиенты
      <mu-button flat slot="right"
@click="goHome">Главная</mu-button>
    </mu-appbar>

```

```

    <mu-list textline="two-line" v-for="client in cli-
ents">
      <mu-list-item button @click="openEditDialog(cli-
ent.name, client.address)">
        <mu-list-item-content>
          <mu-list-item-title><strong>{{cli-
ent.name}}</strong></mu-list-item-title>
          <mu-list-item-sub-title style="color: rgba(0,
0, 0, .87)">{{client.address}}</mu-list-item-sub-title>
          <mu-container align="left">
            <mu-dialog title="Редактирование"
width="500" :open.sync="openEdit">
              <mu-text-field v-model="editForm.name"
type="text" placeholder="Название"></mu-text-field>
              <br>
              <mu-text-field v-model="editForm.address"
type="text" placeholder="Адрес"></mu-text-field>
              <br>
              <mu-button slot="actions" color="green"
@click="updateClient(client.id)">Сохранить</mu-button>
              <mu-button slot="actions" color="red"
@click="deleteClient(client.id)">Удалить</mu-button>
              <mu-button slot="actions" color="primary"
@click="closeEditDialog">Закрыть</mu-button>
            </mu-dialog>
          </mu-container>
        </mu-list-item-content>
      </mu-list-item>
    </mu-list>
  </mu-divider>
</mu-list>

```

```

    <mu-container align="left">
      <mu-form :model="form" class="mu-demo-form" :label-position="top" label-width="100">
        <mu-form-item prop="input" label="Название">
          <mu-text-field v-model="form.name"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="input" label="Адрес">
          <mu-text-field v-model="form.address"></mu-text-field>
        </mu-form-item>
      </mu-form>
      <mu-button color="green" @click="addClient">Добавить</mu-button>
    </mu-container>
  </mu-container>
</template>

```

```

<script>
  import $ from 'jquery'
  import Menu from "./Menu";

  export default {
    name: "Clients",
    components: {
      Menu
    },
    data() {
      return {
        clients: '',

```

```

        form: {
            name: '',
            address: '',
        },
        editForm: {
            name: '',
            address: '',
        },
        openEdit: false
    }
},
created() {
    $.ajaxSetup({
        headers: { 'Authorization': "Token " +
sessionStorage.getItem('auth_token') },
    });
    this.loadClients()
},
methods: {
    goHome() {
        this.$router.push({name: "Autopark"})
    },
    openEditDialog(name, address) {
        this.editForm.name = name
        this.editForm.address = address
        this.openEdit = true
    },
    closeEditDialog() {
        this.openEdit = false;
    },

```

```

        loadClients() {
            $.ajax({
                url: "http://127.0.0.1:8000/au-
topark/clients/",
                type: "GET",
                success: (response) => {
                    this.clients = re-
sponse.data.data
                }
            })
        },
        addClient() {
            $.ajax({
                url: "http://127.0.0.1:8000/au-
topark/clients/",
                type: "POST",
                data: {
                    name: this.form.name,
                    address: this.form.address
                },
                success: (response) => {
                    alert("Клиент добавлен")
                    this.loadClients()
                    this.form = {
                        name: '',
                        address: ''
                    };
                },
                error: (response) => {
                    alert("Ошибка")
                }
            })
        }
    }
}

```

```

        }
    })
},
updateClient(id) {
    $.ajax({
        url:      "http://127.0.0.1:8000/au-
topark/clients/" + id,
        type: "PUT",
        data: {
            name: this.editForm.name,
            address: this.editForm.address
        },
        success: (response) => {
            alert("Клиент обновлен")
            this.loadClients()
            this.closeEditDialog()
        },
        error: (response) => {
            alert("Ошибка")
        }
    })
},
deleteClient(id) {
    $.ajax({
        url:      "http://127.0.0.1:8000/au-
topark/clients/" + id,
        type: "DELETE",
        success: (response) => {
            alert("Клиент удален")
            this.loadClients()

```



```

        this.closeEditDialog()
    }
    })
    },
    }
    }
</script>

```

```

<style scoped>

```

```

</style>

```

Листинг Б.14 – Страница с заказами

```

<template>
  <mu-container>
    <mu-appbar style="width: 100%;" color="primary">
      <Menu slot="left"></Menu>
      Заказы
      <mu-button flat slot="right"
@click="goHome">Главная</mu-button>
    </mu-appbar>
    <mu-container align="left">
      <mu-dialog title="Добавление заказа" width="500"
scrollable :open.sync="openAdd">
        <mu-form :model="form" class="mu-demo-form" :la-
bel-position="top" label-width="100">
          <mu-form-item prop="select" label="Клиент">
            <mu-select v-model="form.client">
              <mu-option v-for="client in clients"
:key="client.name" :label="client.name"

```

```

                                :value="client.id"></mu-op-
tion>

        </mu-select>
    </mu-form-item>

    <mu-form-item                prop="select"                la-
bel="Направление">
        <mu-select v-model="form.direction">
            <mu-option v-for="direction in directions"
:key="direction" :label="direction"
                                :value="direction"></mu-op-
tion>

            </mu-select>
        </mu-form-item>

        <mu-form-item    prop="input"    label="Второй
пункт">

            <mu-text-field                v-model="form.sec-
ond_point"></mu-text-field>

            </mu-form-item>

            <mu-form-item prop="input" label="Километраж">
                <mu-text-field                v-model="form.kilo-
metrage"></mu-text-field>

                </mu-form-item>

                <mu-form-item prop="input" label="Вес груза">
                    <mu-text-field    v-model="form.weight"></mu-
text-field>

                    </mu-form-item>

                    <mu-form-item prop="input" label="Стоимость">
                        <mu-text-field    v-model="form.price"></mu-
text-field>

                        </mu-form-item>

```

```

        <mu-form-item    prop="select"    label="Путевой
лист">

        <mu-select v-model="form.manifest">
            <mu-option v-for="manifest in manifests"
:key="manifest.date" :label="manifest.date"
:value="manifest.id"></mu-op-
tion>

        </mu-select>
    </mu-form-item>
</mu-form>

    <mu-button    slot="actions"    color="green"
@click="addOrder">Добавить</mu-button>

    <mu-button    slot="actions"    color="primary"
@click="closeAddDialog">Закрыть</mu-button>
</mu-dialog>

    <mu-button    color="green"    @click="openAddDia-
log">Добавить</mu-button>
</mu-container>

    <mu-paper :z-depth="1" align="center">
        <mu-data-table                :columns="columns"
:sort.sync="sort"

                                @sort-change="handleSortChange"
:data="orders">
            <template slot="expand" slot-scope="prop">
                <div style="padding: 15px;">
                    <mu-button small slot="actions" color="pri-
mary"

```

```

        @click="openEditDia-
log(prop.row.client, prop.row.direction, prop.row.sec-
ond_point, prop.row.kilometrage, prop.row.weight,
prop.row.price, prop.row.manifest)">
        Редактировать
    </mu-button>
    <mu-button small slot="actions" color="red"
@click="deleteOrder(prop.row.id)">Удалить</mu-button>
    <mu-dialog title="Редактирование заказа"
width="500" scrollable :open.sync="openEdit">
        <mu-form :model="editForm" class="mu-demo-
form" :label-position="top" label-width="100">
            <mu-form-item prop="select" la-
bel="Клиент">
                <mu-select v-model="editForm.client">
                    <mu-option v-for="client in clients"
:key="client.name" :label="client.name"
:value="client.id"></mu-
option>
                </mu-select>
            </mu-form-item>
            <mu-form-item prop="select" la-
bel="Направление">
                <mu-select v-model="editForm direc-
tion">
                    <mu-option v-for="direction in di-
rections" :key="direction" :label="direction"
:value="direction"></mu-
option>
                </mu-select>

```

```

        </mu-form-item>
        <mu-form-item prop="input" label="Второй
пункт">
            <mu-text-field v-model="editForm.sec-
ond_point"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="input" la-
bel="Километраж">
            <mu-text-field v-model="editForm.kil-
ometrage"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="input" label="Вес
груза">
            <mu-text-field v-model="edit-
Form.weight"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="input" la-
bel="Стоимость">
            <mu-text-field v-model="edit-
Form.price"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="select" la-
bel="Путевой лист">
            <mu-select v-model="editForm.mani-
fest">
                <mu-option v-for="manifest in mani-
fests" :key="manifest.date" :label="manifest.date"
                    :value="mani-
fest.id"></mu-option>
            </mu-select>

```

```

        </mu-form-item>
    </mu-form>
    <mu-button slot="actions" color="green"
@click="updateOrder(prop.row.id)">Сохранить</mu-button>
    <mu-button slot="actions" color="primary"
@click="closeEditDialog">Закрыть</mu-button>
</mu-dialog>
</div>
</template>
<template slot-scope="scope">
    <td class="is-center">{{scope.row.direction}}</td>
    <td>{{scope.row.second_point}}</td>
    <td class="is-center">{{scope.row.kilometrage}}</td>
    <td class="is-center">{{scope.row.weight}}</td>
    <td class="is-center">{{scope.row.price}}</td>
</template>
</mu-data-table>
</mu-paper>
</mu-container>
</template>

<script>
    import $ from 'jquery'
    import Menu from "../Menu";

    export default {

```

```

name: "Orders",
components: {
    Menu
},
data() {
    return {
        orders: '',
        sort: {
            name: '',
            order: 'asc'
        },
        columns: [
            {title: 'Направление', name: 'direction', width: 150, sortable: true},
            {title: 'Второй пункт', name: 'second_point', width: 300, align: 'center'},
            {title: 'Километраж', name: 'kilometrage', width: 100, align: 'center', sortable: true},
            {title: 'Вес', name: 'weight', width: 100, align: 'center', sortable: true},
            {title: 'Стоимость', name: 'price', width: 100, align: 'center', sortable: true},
        ],
        openAdd: false,
        openEdit: false,
        form: {
            client: '',
            direction: '',
            second_point: '',
            kilometrage: '',

```

```

        weight: '',
        price: '',
        manifest: ''
    },
    editForm: {
        client: '',
        direction: '',
        second_point: '',
        kilometrage: '',
        weight: '',
        price: '',
        manifest: ''
    },
    directions: [
        'To client',
        'From client'
    ],
    clients: '',
    manifests: '',
    composition: '',
    client: '',
    }
},
created() {
    $.ajaxSetup({
        headers: {'Authorization': "Token " +
sessionStorage.getItem('auth_token')},
    });
    this.loadOrders()
    this.loadClients()

```



```

        this.loadManifests()
    },
    methods: {
        loadOrders() {
            $.ajax({
                url: "http://127.0.0.1:8000/au-
topark/orders/",
                type: "GET",
                success: (response) => {
                    this.orders = response.data.or-
ders
                }
            })
        },
        addOrder() {
            $.ajax({
                url: "http://127.0.0.1:8000/au-
topark/orders/",
                type: "POST",
                data: {
                    client: this.form.client,
                    direction: this.form.direction,
                    second_point: this.form.sec-
ond_point,
                    kilometrage: parse-
Float(this.form.kilometrage),
                    weight: parse-
Float(this.form.weight),
                    price: parse-
Float(this.form.price),

```

```

        manifest: this.form.manifest
    },
    success: (response) => {
        alert("Заказ добавлен")
        this.loadOrders()
        this.closeAddDialog()
        this.form = {
            client: '',
            direstion: '',
            second_point: '',
            kilometrage: '',
            weight: '',
            price: '',
            manifest: ''
        };
    },
    error: (response) => {
        alert("Ошибка")
    }
}))
},
updateOrder(id) {
    $.ajax({
        url: "http://127.0.0.1:8000/autopark/orders/" + id,
        type: "PUT",
        data: {
            client: this.editForm.client,
            direction: this.editForm.direction,

```

```

                second_point:      this.edit-
Form.second_point,
                kilometrage:      parse-
Float(this.editForm.kilometrage),
                weight:      parseFloat(this.edit-
Form.weight),
                price:      parseFloat(this.edit-
Form.price),
                manifest:      this.editForm.mani-
fest
            },
            success: (response) => {
                alert("Заказ обновлен")
                this.loadOrders()
                this.closeEditDialog()
            },
            error: (response) => {
                alert("Ошибка")
            }
        })
    },
    deleteOrder(id) {
        $.ajax({
            url:      "http://127.0.0.1:8000/au-
topark/orders/" + id,
            type: "DELETE",
            success: (response) => {
                alert("Заказ удален")
                this.loadOrders()
            },

```

```

        error: (response) => {
            alert("Ошибка")
        }
    })
},
loadClients() {
    $.ajax({
        url:      "http://127.0.0.1:8000/au-
topark/clients/",
        type: "GET",
        success: (response) => {
            this.clients      =      re-
sponse.data.data
        }
    })
},
loadManifests() {
    $.ajax({
        url:      "http://127.0.0.1:8000/au-
topark/manifests/",
        type: "GET",
        success: (response) => {
            this.manifests    =      re-
sponse.data.manifest
        }
    })
},
loadInformation(id) {
    $.ajax({

```

```

        url:      "http://127.0.0.1:8000/au-
topark/orders/" + id,
        type: "GET",
        success: (response) => {
            this.composition      =      re-
sponse.data.items
            this.client = response.data.cli-
ent
        }
    })
},
goHome() {
    this.$router.push({name: "Autopark"})
},
handleSortChange({name, order}) {
    if (name === 'direction') {
        this.orders = this.orders.sort(func-
tion (a, b) {
            var nameA = a[name].toLower-
Case(), nameB = b[name].toLowerCase()
            if (order === 'asc') {
                if (nameA < nameB) {
                    return 1
                }
                if (nameA > nameB) {
                    return -1
                }
                return 0
            } else {
                if (nameA < nameB)

```

```

        return -1
    if (nameA > nameB)
        return 1
    return 0
    }
    }
    )
    } else {
        this.orders = this.orders.sort((a,
b) => order === 'asc' ? b[name] - a[name] : a[name] -
b[name]);
    }
},
openAddDialog() {
    this.openAdd = true
},
closeAddDialog() {
    this.openAdd = false;
},
openEditDialog(client,    direction,    sec-
ond_point, kilometrage, weight, price, manifest) {
    this.editForm.client = client
    this.editForm.direction = direction
    this.editForm.second_point    =    sec-
ond_point

    this.editForm.kilometrage = kilometrage
    this.editForm.weight = weight
    this.editForm.price = price
    this.editForm.manifest = manifest
    this.openEdit = true

```

```

        },
        closeEditDialog() {
            this.openEdit = false;
        },
    }
}
</script>

```

```

<style scoped>

```

```

</style>

```

Листинг Б.15 – Страница с путевыми листами

```

from django.shortcuts import render
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import permissions
from rest_framework.generics import get_object_or_404

from autopark.serializers import *

class DriversView(APIView):
    """Водители"""

    permission_classes = [permissions.AllowAny, ]

    def get(self, request):
        drivers = Drivers.objects.all()
        serializer = DriversSerializer(drivers,
many=True)

```

```

        return Response({"data": serializer.data})

class CarsView(APIView):
    """Машины"""

    permission_classes = [permissions.AllowAny, ]

    def get(self, request):
        cars = Cars.objects.all()
        serializer = CarsSerializer(cars, many=True)
        return Response({"data": serializer.data})

class ClientsView(APIView):
    """КЛИЕНТЫ"""

    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request, pk=None):
        clients = Clients.objects.all()
        if pk is not None:
            clients = Clients.objects.filter(pk=pk)
        serializer = ClientsSerializer(clients,
many=True)
        return Response({"data": serializer.data})

    def post(self, request):
        client = ClientsSerializer(data=request.data)
        if client.is_valid():
            client.save()

```



```

        return Response(status=201)
    else:
        return Response(status=400)

    def put(self, request, pk):
        saved_client = get_object_or_404(Clients.objects.all(), pk=pk)
        data = request.data
        serializer = ClientsSerializer(instance=saved_client, data=data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(status=201)
        else:
            return Response(status=400)

    def delete(self, request, pk):
        client = get_object_or_404(Clients.objects.all(), pk=pk)
        client.delete()
        return Response(status=204)

class ManifestsView(APIView):
    """Путевой лист и номера заказов для него"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request, pk=None):
        if pk is not None:
            manifest = Manifests.objects.filter(pk=pk)

```

```

        serializerManifest = ManifestsSerializer(
            manifest, many=True)
        order = Orders.objects.filter(
            manifest_id=pk)
        serializerOrder = OrdersSerializer(
            order, many=True)
        return Response({"manifest":
            serializerManifest.data, "orders":
            serializerOrder.data})
    manifests = Manifests.objects.all()
    serializerManifest = ManifestsSerializer(
        manifests, many=True)
    return Response({"manifest":
        serializerManifest.data})

    def post(self, request):
        id = User.objects.values_list('id',
            flat=True).get(username=request.user)
        driver = Drivers.objects.values_list('id',
            flat=True).get(user=id)
        manifest = ManifestsPostSerializer(
            data=request.data)
        if manifest.is_valid():
            manifest.save(driver_id=driver)
            return Response(status=201)
        else:
            return Response(status=400)

    def put(self, request, pk):
        saved_manifest = get_object_or_404(
            Manifests.objects.all(), pk=pk)

```

```

        data = request.data
        serializer = ManifestsSerializer(in-
stance=saved_manifest, data=data, partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(status=201)
        else:
            return Response(status=400)

    def delete(self, request, pk):
        manifest = get_object_or_404(Manifests.ob-
jects.all(), pk=pk)
        manifest.delete()
        return Response(status=204)

class OrdersView(APIView):
    """Заказы и их состав"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request, pk=None):
        orders = Orders.objects.all()
        if pk is not None:
            orders = Orders.objects.filter(pk=pk)
            serializerOrder = OrdersSerializer(orders,
many=True)
            client = Clients.objects.filter(id=Or-
ders.objects.values_list('client',
flat=True).get(pk=pk))

```

```

        serializerClient = ClientsSerializer(client,
many=True)

        composition = Order_composition.objects.filter(
order_id=pk)

        serializerComposition = OrderCompositionSe-
rializer(composition, many=True)

        return Response(

            {"orders":          serializerOrder.data,
"items": serializerComposition.data, "client": serializ-
erClient.data})

        serializerOrder      =      OrdersSerializer(orders,
many=True)

        return      Response({"orders":      serializerOr-
der.data})

```

```

def post(self, request):
    order = OrdersSerializer(data=request.data)
    if order.is_valid():
        order.save()
        return Response(status=201)
    else:
        return Response(status=400)

```

```

def put(self, request, pk):
    saved_order      =      get_object_or_404(Orders.ob-
jects.all(), pk=pk)
    data = request.data
    serializer = OrdersSerializer(instance=saved_or-
der, data=data, partial=True)
    if serializer.is_valid():

```

```
        serializer.save()
        return Response(status=201)
    else:
        return Response(status=400)

def delete(self, request, pk):
    order = get_object_or_404(Orders.objects.all(),
pk=pk)
    order.delete()
    return Response(status=204)
```

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

ДНЕВНИК ПРАКТИКИ

за период с 02.02.2020 по 02.07.2020

Студент	<u>Багрова П.А.</u> (Фамилия, И.О.)
Факультет	<u>СПО</u>
Группа	<u>У2331</u>
Направление (специальность)	<u>09.02.07 Информационные системы и программирование</u>
Место прохождения практики	<u>Факультет СПО</u>
Руководитель практики	<u>Ефимова Т.Н., факультет СПО, преподаватель Говоров А.И., факультет СПО, преподаватель</u>
Ответственный за проведение практики от университета	<u>Королев В.В., зам. директора факультета</u>

Индивидуальное задание
выполнено полностью

(подпись ответственного
за проведение практики от уни-
верситета)

(дата)

Санкт-Петербург
2020

Период	Краткое содержание работы	Отметка о выполнении
02.02.2020 – 09.02.2020	Вводный инструктаж. Ознакомление с инструкцией по технике безопасности. Ознакомление с целями и задачами практики.	
09.02.2020 – 09.03.2020	Анализ индивидуального задания. Обследование предметной области согласно индивидуальной теме учебной практики.	
10.03.2020 – 31.03.2020	Описание предметной области. Создание диаграммы классов. Создание таблиц.	
01.04.2020 – 07.04.2020	Создание модели Django в соответствии с моделью данных и настройка связи между таблицами	
08.04.2020 – 21.04.2020	Реализация элементов инфраструктуры Django, в соответствии с архитектурным паттерном Model-View-Template или сокращенно MVT. Реализация интерфейсов к системе средствами Django Templates или сторонними средствами.	
17.06.2020 – 23.06.2020	Подготовка отчетных материалов по результатам практики.	
24.06.2020 – 02.07.2020	Защита результатов практики.	

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет среднего профессионального образования

О Т З Ы В

руководителя учебной практики
по профессиональному модулю
ПМ.11 «Разработка, администрирование и защита баз данных»

Обучающийся Багрова П.А.

(Фамилия, И.О.)

Факультет СПО

Группа Y2331

Направление (специальность) 09.02.07 Информационные системы и программирование

Место прохождения практики Факультет СПО

Тема индивидуального задания

Оценка достигнутых результатов

№ п/п	Планируемые результаты обучения (показатели)	Оценка			
		5	4	3	2
1.	Корректность определения структуры базы данных				
2.	Качество реализации компонентов описания модели данных средствами Django ORM				
3.	Качество реализации контроллеров Django				
4.	Качество реализации интерфейсов к системе средствами Django Templates или сторонними средствами.				
Итоговая оценка					

Отмеченные достоинства:

.....

Отмеченные недостатки:

.....

Заключение: Считаю, все задачи, поставленные на практику, выполнены и по результатам практики студент(ка) заслуживает оценки «.....».

Руководитель практики _____

(подпись)

(ФИО)

« ____ » _____ 2020г.

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

АТТЕСТАЦИОННЫЙ ЛИСТ
Характеристика профессиональной деятельности студента во время прохождения учебной практики

Студент ____ Багрова П.А. _____

Группа _Y2331_

Специальность 09.02.07 Информационные системы и программирование

Место проведения практики _____

Сроки прохождения практики 20.01.2020 – 02.07.2020

Наименование профессиональных модулей (видов деятельности)

ПМ.11 «Разработка, администрирование и защита баз данных»

Виды выполняемых работ:

Результаты (освоенные профессиональные компетенции)	Основные показатели оценки результата	Отметка о выполнении ¹
ПМ.11 Разработка, администрирование и защита баз данных		
ПК 11.1. Осуществлять сбор, обработку и анализ информации для проектирования баз данных	- осуществление корректного сбора, обработки и анализа информации для проектирования баз данных	
	- обоснование выявления объектов проектируемой БД и установки отношений между ними на основе анализа предметной области;	
ПК 11.2. Проектировать базу данных на основе анализа предметной области.	- использование CASE-средств автоматизированного проектирования при моделировании базы данных при построении концептуальной, даталогической и физической моделей БД;	
	- соответствие проекта структурной и манипуляционной частей БД заданным критериям функциональности.	
ПК 11.3. Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.	— использование CASE-средств автоматизированного проектирования при моделировании базы данных при построении концептуальной, даталогической и физической моделей БД;	

¹ Указывается «+» или «-». Считается, что программа практики выполнена, если студентом выполнено не менее 70% перечисленных видов работ.

Результаты (освоенные профессиональные компетенции)	Основные показатели оценки результата	Отметка о выполнении ¹
	— обоснование выбранных методов защиты объектов базы данных в соответствии с требованиями задачи.	
ПК 11.4. Реализовывать базу данных в конкретной системе управления базами данных.	— соответствие реализации структурной части БД средствами СУБД даталогической и физической моделям данных;	
	— соответствие реализации манипуляционной части БД средствами СУБД заданным критериям;	
	— соответствие реализации приложения БД заданным критериям функциональности;	
ПК 11.5. Администрировать базы данных.	— обоснованность выбора архитектуры клиент-серверного взаимодействия в соответствии с технологией разработки базы данных;	
	— соответствие заданным требованиям обеспечения целостности данных и контроля доступа к данным;	
	— соответствие заданным требованиям управления привилегиями пользователей базы данных программными средствами;	
	— соответствие конфигурирования сетевых устройств требованиям обеспечения доступа к данным.	
ПК 11.6. Защищать информацию в базе данных с использованием технологии защиты информации.	— соответствие конфигурирования сетевых устройств требованиям защиты данных при передаче данных по сети	
	— соответствие заданным требованиям программных средств защиты информации в базе данных средствами СУБД	
	— соответствие заданным требованиям управления привилегиями пользователей базы данных программными средствами;	

Руководитель практики от факультета СПО: _____

Дата: _____