Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Факультет Инфокоммуникационных технологий

Образовательная программа Интеллектуальные системы в гуманитарной сфере (Академический бакалавр, Очная ф/о)

Направление подготовки (специальность) 45.03.04 – Интеллектуальные системы в гуманитарной сфере

О Т Ч Е Т

по курсовой работе по дисциплине «Основы Web-программирования»

Тема задания: Разработка вебсайта для диспетчера автобусного парка.

Обучающийся Кузнецов Дмитрий Евгеньевич, К3343

Руководитель курсовой работы: Говоров Антон Игоревич, ассистент

Оценка по курсовой работе ____

Подписи членов комиссии:

_____ Говоров А.И.
(подпись)

_____ Чунаев А.В.
(подпись)

_____ Антонов М.Б.
(подпись)

Дата ____

Санкт-Петербург
2020

Оглавление

ВВЕДЕНИЕ

Работа автобусной системы зависит не только от внутреннего устройства, но и от человеческого фактора, внешних обстоятельств. Для всех маршрутов составляется расписание, в соответствии с которым формируется штат водителей. Деятельность парка необходимо контролировать и корректировать, в связи с чем используются различные схемы и таблицы

При постоянной корректировке базы сотрудников и транспорта возникает потребность поддерживать качество оказываемых услуг. Это возможно достичь при помощи удобной программы системы, которая позволит контролировать все рабочие процессы и изменения.

Цель работы – создать вебсайт для диспетчера автобусного парка частной транспортной фирмы.

Задачи:

- Обзор предметной области
- Выявление функциональных требований
- Проектирование серверной части
- Проектирование клиентской части
- Реализация клиент-серверной архитектуры

В первой главе описана предметная область. Во второй – проведен анализ требований к выполнению работы и рассмотрен процесс проектирования базы данных, клиентской и серверной частей сервиса. В третьей главе описаны средства разработки и сам процесс реализации веб-сервиса: база данных, клиент и сервер.

1. ПРЕДМЕТНАЯ ОБЛАСТЬ

Предметной областью данной работы является автобусный парк.

Диспетчер подобного парка, зачастую, отвечает за штат сотрудников, а именно за добавление их в базу данных или удаления из нее, то есть обладает сведениями о водителях.

Во-вторых, он имеет возможность составления графика работы, следовательно, к данным водителей, на которых он назначает расписание, добавляется информация об автобусах, на которых будет осуществляться работа; маршрутах, где курсирует выбранный сотрудник.

Еще одно частое действие диспетчера – контроль за результатами техобслуживания, проверка состояния подвижного состава. Из этого вытекает возможность работы с данными о поломках автобусов и корректировки расписания, в зависимости от результатов ремонта.

Следующая задача, выполняемая диспетчером – анализ состояния автобусного парка и его компонентов. Для эффективного процесса диспетчер либо получает сведения об отдельных интересующих его элементах, либо получает общую сводку информации обо всем автопарке.

2. ПРОЕКТИРОВАНИЕ

2.1. Функциональные требования

В системе должно быть организовано хранение сведений о водителях, маршрутах, автобусах, графике работы где:

- Каждый водитель характеризуется паспортными данными, классом, стажем работы и окладом, который зависит от класса и стажа; каждый водитель закреплен за конкретным автобусом и маршрутом, но может пересесть в случае необходимости.

- Каждый маршрут описывается номером, названиями начального и конечного пунктов движения, временем начала и конца движения, интервалом движения и протяженностью.

- У каждого автобуса есть информация по его номеру гос. регистрации, типе и вместимости, которая зависит от типа.

Диспетчер автобусного парка должен иметь возможность получить следующие сведения:

- Список водителей, работающих на указанном маршруте, и их график работы.
- Время начала и конца движения автобусов на каждом маршруте.
- Общая протяженность всех маршрутов.
- Список автобусов, отсутствовавших в указанный день работы, и причина данной ситуации
- Количество водителей каждого класса.

Кроме этого, по требованию диспетчера должен выдаваться отчет со следующей информацией:

- Группирование автобусов по типам; вывод для каждого типа обслуживаемых маршрутов с характеристиками, списком автобусов и водителей, работающих на данном маршруте.

- Общая протяженность обслуживаемых автопарком маршрутов.

- Стаж водителей.

2.2. База данных

Следуя функциональным требованиям, в базе данных должны присутствовать таблицы Водителей, Автобусов, Типов автобуса, Журнала, Расписания и Маршрута. При построении схемы и настройке связей между таблицами, получается следующее:

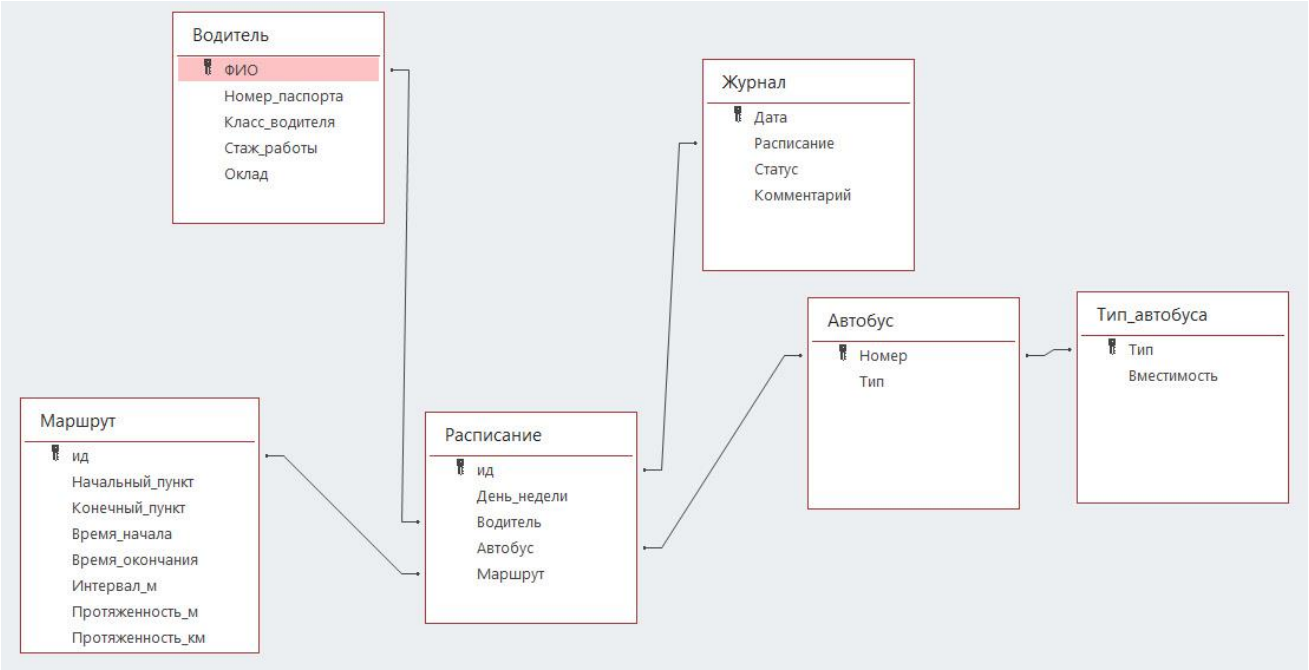

Рисунок 1 – Схема базы данных Таблица

Водитель.

Здесь будут храниться данные по водителям автопарка. Связана с таблицей Расписания связью один ко многим. Имеет характеристики:

- ФИО
- Номер паспорта
- Класс водителя
- Стаж работы
- Оклад

Автобус.

Хранит данные об автобусах, находящихся в автопарке. Связана с таблицами Тип Автобуса и Расписанием. Имеет характеристики:

- Номер
- Тип

Тип Автобуса.

Хранит информацию о типе автобуса. Связана с таблицей Автобуса. Имеет характеристики:

- Тип
- Вместимость

Маршрут.

Хранятся сведения о маршрутах, обслуживаемых автопарком. Связана с таблицей Расписания. Имеет характеристики:

- ид
- Начальный пункт
- Конечный пункт
- Время начала
- Время окончания
- Интервал
- Протяженность (в минутах)
- Протяженность (в километрах)

Расписание.

Хранит данные о записи водителей на рабочие дни. Имеет внешние ключи от таблиц Водителя, Автобуса и Маршрута. Связана с таблицей Журнала. Имеет характеристики:

- ид
- День недели
- Водитель
- Автобус
- Маршрут

Журнал.

Хранятся сведения о данных маршрута, о выходе или не выходе на линию и комментарии. Связана с таблицей расписания. Имеет характеристики:

- Дата
- Расписание
- Статус
- Комментарий

2.3. Серверная часть сервиса

Будет необходимо подробнее настроить следующие файлы:

- urls.py

Здесь будет храниться сопоставитель URL-адресов: определяет список соответствия между прописанными URL-шаблонами и соответствующими функциями отображения. Если полученный HTTP запрос подходит под определенный шаблон, то будет вызвана ассоциированная функция отображения и передана в запрос.

- views.py

Здесь находится обработчик запросов: получает HTTP запросы от веб-клиентов и возвращает HTTP-ответы.

Здесь будут также определяться функции (чтение, обновление, создание, удаление).

- serializers.py

В этом файле осуществляется сериализация – процесс перевода структуры данных в последовательность битов. Цель – преобразовать информацию, определенную с помощью моделей, из базы данных в определенный формат для легкой и эффективной передачи.

```python
class BusCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для создания новой записи в модели Автобус"""

    class Meta:
        model = Bus
        fields = "__all__"


class BusTypeSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Тип автобуса"""
    buses = BusCreateSerializer(many=True)

    class Meta:
        model = BusType
        fields = "__all__"
```

Рис. 2 – пример кода в файле serializers.py

2.4. Клиентская часть сервиса

Вследствие того, что важной концепцией Vue являются компоненты – абстракция, которая позволяет собирать большие приложения из составных деталей и которая представляет из себя пригодные к повторному использованию объекты, продумаем основные детали для каждого компонента:

- Компонент домашней страницы

Здесь пользователю должна быть предоставлена возможность входа в учетную запись, если он не был авторизован.

При входе под своим логином диспетчер должен сразу увидеть график работы водителей на главной странице. Должна быть осуществлена возможность добавления или удаления записей расписания.

- Компонент страницы входа

Здесь у пользователя запрашиваются его логин и пароль.

- Компонент страницы списка водителей

Диспетчер должен видеть всех водителей. На странице пользователю должна быть дана возможность добавления или удаления водителя.

- Компонент страницы запросов

Диспетчер имеет возможность выбрать запрос, на который тот хочет получить отчет; ввести интересующие данные и получить результат.

- Компонент страницы автобусов

Здесь диспетчер может узнать информацию об автобусах всех типов, их номер. Он должен иметь доступ к удалению того или иного автобуса, а также к добавлению нового.

- Компонент страницы маршрутов

Диспетчер должен видеть информация обо всех существующих маршрутах. В случае чего имеет доступ к изменению какой-либо информации о маршруте, либо же удалению или добавлению.

- Компонент страницы графика работы водителей

Здесь диспетчер должен иметь доступ к информации о том, какой водитель, на каком автобусе, в какой день недели и куда направляется. Также должна быть возможность фильтрации по маршруту, автобусу или водителю.

Выводы

В результате проведенного анализа предметной области были выделены функциональные требования, основываясь на которых был осуществлен процесс проектирования:

- базы данных: содержание таблиц, их структура и связи между ними;
- серверной части: содержание urls.py, views.py, serializers.py;
- клиентской части: количество компонентов vue и их содержание.

3. РЕАЛИЗАЦИЯ

3.1. Описание средств разработки

Разработка веб-сервиса будет осуществляться средствами:

- Django

Высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Django оказывает помощь в написании программного обеспечения, которое будет полным, разносторонним, безопасным, масштабируемым, удобным в сопровождении и переносным.

- Django REST framework

Библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта. REST – стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов. Сетевой протокол должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ».

- Vue.js

Фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления, что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений, если использовать его совместно с современными инструментами и дополнительными библиотеками.

- Muse-UI

Фреймворк для создания пользовательских интерфейсов для Vue 2.0, основанный на библиотеке Material UI для ReactJS. Имеет более 40 компонентов, поэтому может приспособиться практически ко всему.

- PostrgreSQL

Объектно-реляционная система управления базами данных с открытым исходным кодом, которая поддерживает большую часть стандарта SQL и предлагает множество современных функций: сложные запросы, внешние ключи, триггеры, изменяемые

представления, транзакционная целостность, многоверсионность

3.2. База данных

```python
class Driver(models.Model):
    fio = models.CharField("ФИО", max_length=300)
    passport_number = models.CharField("Номер паспорта", max_length=20)
    class_types = models.TextChoices('class_types', '1_класс 2_класс 3_класс')
    driver_class = models.CharField("Водительский класс", blank=True, choices=class_types.choices, max_length=20)
    work_exp = models.IntegerField('Стаж работы')
    salary = models.IntegerField('Оклад')

    class Meta:
        verbose_name = 'Водитель'
        verbose_name_plural = 'Водители'

    def __str__(self):
        return self.fio
```

Рис. 3 – Модель Водителя

```python
class Bus(models.Model):
    number = models.CharField('Номер государственной регистрации автобуса', max_length=10)
    bus_type = models.ForeignKey(BusType, on_delete=models.CASCADE, related_name='buses')

    class Meta:
        verbose_name = 'Автобус'
        verbose_name_plural = 'Автобусы'

    def __str__(self):
        return self.number
```

Рис. 4 – Модель Автобуса

```python
class Route(models.Model):
    starting_point = models.CharField("Начальный пункт движения", max_length=300)
    final_destination = models.CharField("Конечный пункт движения", max_length=300)
    time_start = models.TimeField('Время начала движения')
    time_end = models.TimeField('Время окончания движения')
    interval_of_movement = models.IntegerField('Интервал движения в минутах')
    length_in_min = models.IntegerField('Протяженность в минутах')
    length_in_km = models.IntegerField('Протяженность в километрах')

    class Meta:
        verbose_name = 'Маршрут'
        verbose_name_plural = 'Маршруты'

    def __str__(self):
        return self.starting_point+' '+self.final_destination
```

Рис. 5 – Модель Маршрута

```python
class BusType(models.Model):
    bus_type = models.CharField("Тип автобуса по габариам", max_length=100)
    capacity = models.IntegerField('Вместимость')

    class Meta:
        verbose_name = 'Тип автобуса'
        verbose_name_plural = 'Типы автобусов'

    def __str__(self):
        return self.bus_type


class Bus(models.Model):
    number = models.CharField('Номер государственной регистрации автобуса', max_length=10)
    bus_type = models.ForeignKey(BusType, on_delete=models.CASCADE, related_name='buses')

    class Meta:
        verbose_name = 'Автобус'
        verbose_name_plural = 'Автобусы'

    def __str__(self):
        return self.number
```

Рис. 6 – модель автобуса и типа автобуса

```python
class Schedule(models.Model):
    week_day = models.TextChoices('week_day', 'Понедельник Вторник Среда Четверг Пятница Суббота Воскресенье')
    day = models.CharField("День недели", blank=True, choices=week_day.choices, max_length=20)
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE, )
    bus = models.ForeignKey(Bus, on_delete=models.CASCADE)
    route = models.ForeignKey(Route, on_delete=models.CASCADE)

    class Meta:
        verbose_name = 'График работы'
        verbose_name_plural = 'Графики работы'
```

Рис.7 – модель расписания

```python
class Journal(models.Model):
    date = models.DateField('Дата', default=date.today)
    schedule = models.ForeignKey(Schedule, on_delete=models.CASCADE)
    status_type = models.TextChoices('status_type', 'Вышел_на_линию Не_вышел_на_линию')
    status = models.CharField("Статус", blank=True, choices=status_type.choices, max_length=20)
    comment = models.TextField("Комментарий", blank=True)

    class Meta:
        verbose_name = 'Журнал'
        verbose_name_plural = 'Журнал'
```

Рис. 8 – Модель Журнала

3.3. Серверная часть сервиса

Примеры некоторых используемых эндпоинтов:

```python
path("schedule/", views.ScheduleViewSet.as_view({'get': 'list'})),
path("schedule/<int:pk>/", views.ScheduleViewSet.as_view({'get': 'retrieve'})),
path("schedule/create", views.ScheduleViewSet.as_view({'post': 'create'})),
path("schedule/<int:pk>/delete", views.ScheduleViewSet.as_view({'delete': 'destroy'})),
path("schedule/<int:pk>/update", views.ScheduleViewSet.as_view({'post': 'update'})),
```

Рис. 9 – шаблоны адресов для запросов к таблице Расписания

```
path("driver/", views.DriverViewSet.as_view({'get': 'list'})),
path("driver/<int:pk>/", views.DriverViewSet.as_view({'get': 'retrieve'})),
path("driver/create", views.DriverViewSet.as_view({'post': 'create'})),
path("driver/<int:pk>/delete", views.DriverViewSet.as_view({'delete': 'destroy'})),
path("driver/<int:pk>/update", views.DriverViewSet.as_view({'post': 'update'})),
```

Рис. 10 – шаблоны адресов для запросов к таблице Водителей

```
path("route/", views.RouteViewSet.as_view({'get': 'list'})),
path("route/<int:pk>/", views.RouteViewSet.as_view({'get': 'retrieve'})),
path("route/create", views.RouteViewSet.as_view({'post': 'create'})),
path("route/<int:pk>/delete", views.RouteViewSet.as_view({'delete': 'destroy'})),
```

Рис. 11 – шаблоны адресов для запросов к таблице Маршрутов

```
path("journal/", views.JournalViewSet.as_view({'get': 'list'})),
path("journal/<int:pk>/", views.JournalViewSet.as_view({'get': 'retrieve'})),
path("journal/<int:pk>/delete", views.JournalViewSet.as_view({'delete': 'destroy'})),
path("journal/create", views.JournalViewSet.as_view({'post': 'create'})),
```

Рис. 12 – шаблоны адресов для запросов к таблице Журнала

3.4. Клиентская часть сервиса

3.4.1. До входа пользователя

Так как разрабатываемая система предназначена только для диспетчера автопарка, то случайно зашедшим людям не должны быть представлены возможности для работы с базой данных, и с конфиденциальной информацией, при входе на сайт неавторизованного пользователя встречает элемент с просьбой войти под своим логином.

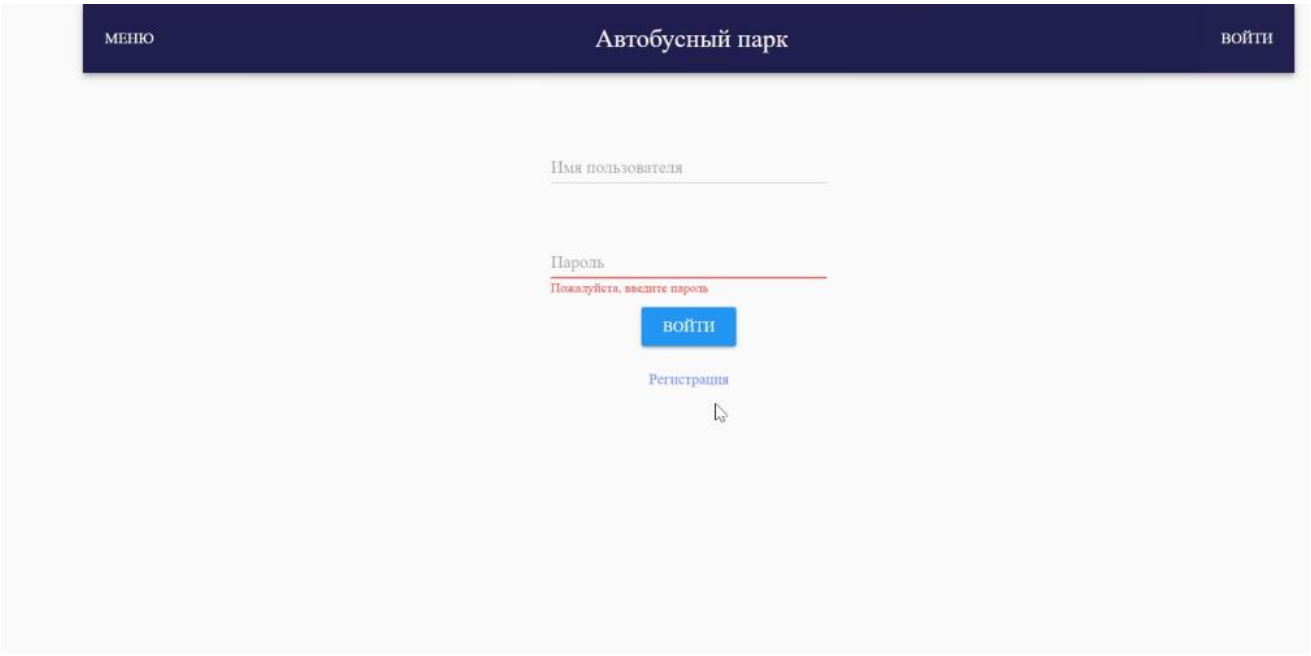

Рис. 13 – страница для случайного пользователя

В самой шапке сайта есть название, кнопка авторизации, а также меню, которое доступно диспетчеру

При нажатии на кнопку «Войти» пользователь увидит перед собой окошко входа с полями ввода логина и пароля. При успешном вводе данных произойдет перенаправление на домашнюю страницу.

А так выглядит поле регистрации. Новый пользователь должен ввести никнейм, пароль и повторить введенный пароль

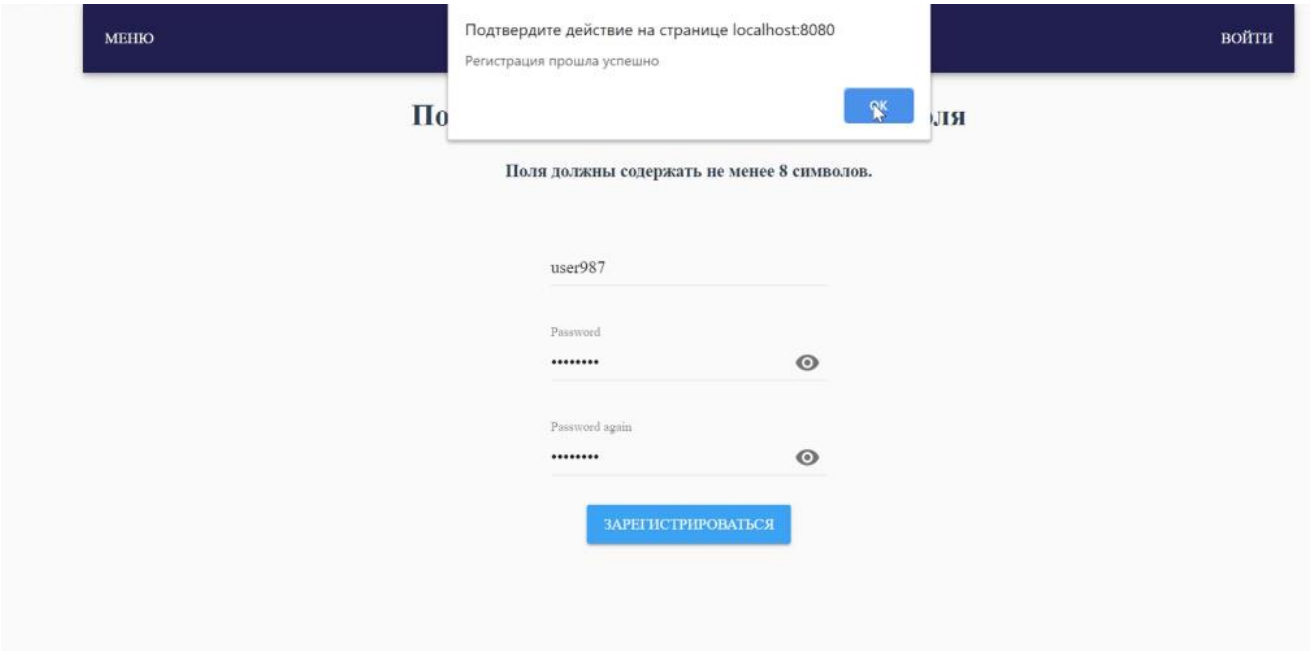

Рис. 14 – Успешная регистрация нового пользователя.

Так выглядит основное меню после прохождения авторизации. Кнопка “Войти” сменяется кнопкой “Выйти”. Есть выпадающее меню, где, например, есть вкладка запросов

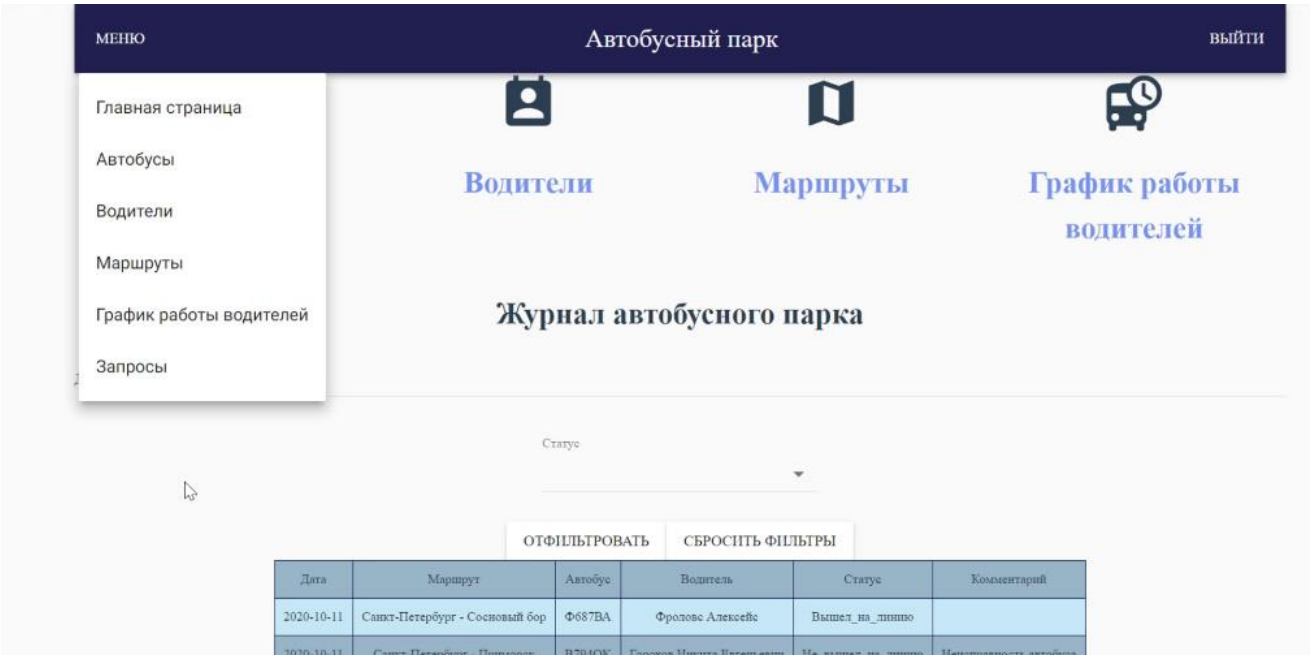

Рис. 15 – Основное меню и выпадающее меню

3.4.2. Для добавления/удаления/изменения основных данных

Диспетчер может видоизменять различную информацию о маршрутах, водителях, автобусах. Может редактировать расписание.

Рис. 16 – Журнал автобусного парка.

Здесь у диспетчера есть возможность добавления/удаления записи из журнала. Помимо этого диспетчер может отфильтровать его и написать комментарий.

Рис. 17 – Появление новой записи в журнале

Рис. 18 – Удаление выбранной записи.

У диспетчера есть доступ к информации обо всех автобусах в парке. Он может удалить или добавить автобус, а также посмотреть информацию о конкретном автобусе.

Рис. 19 – Весь список автобусов в парке

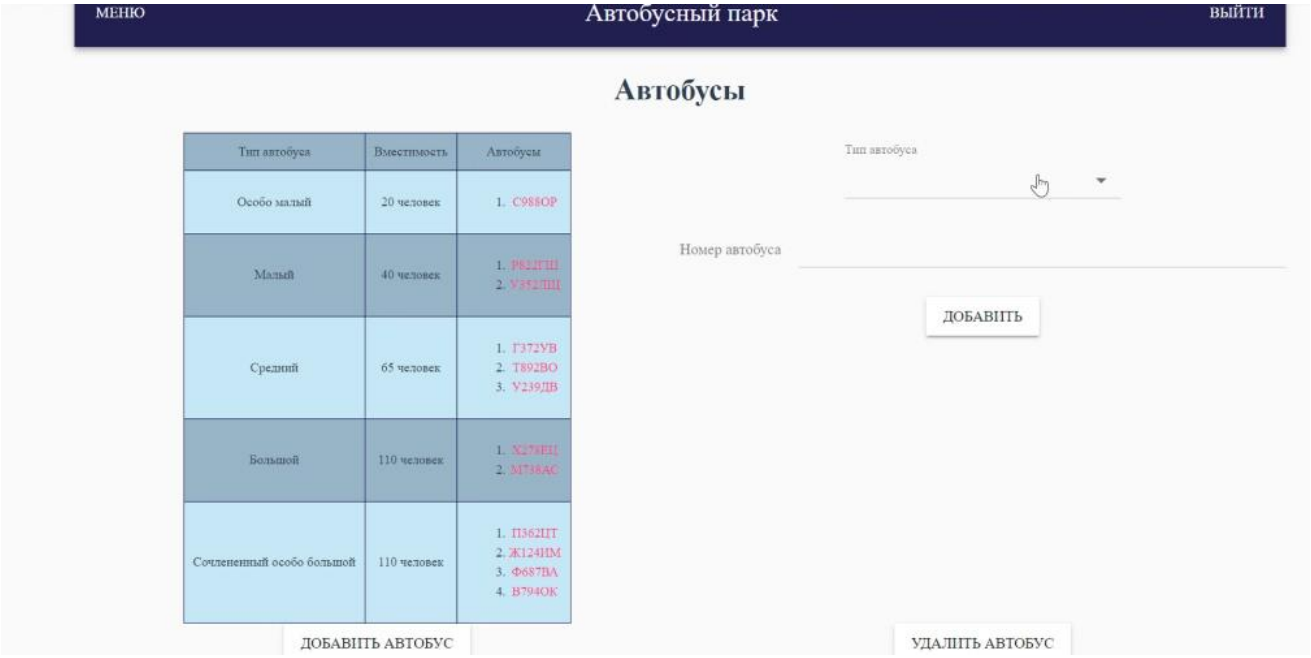

Рис. 20 – Меню добавления нового автобуса в парк

Рис. 21 – Обновленное меню автобусного парка

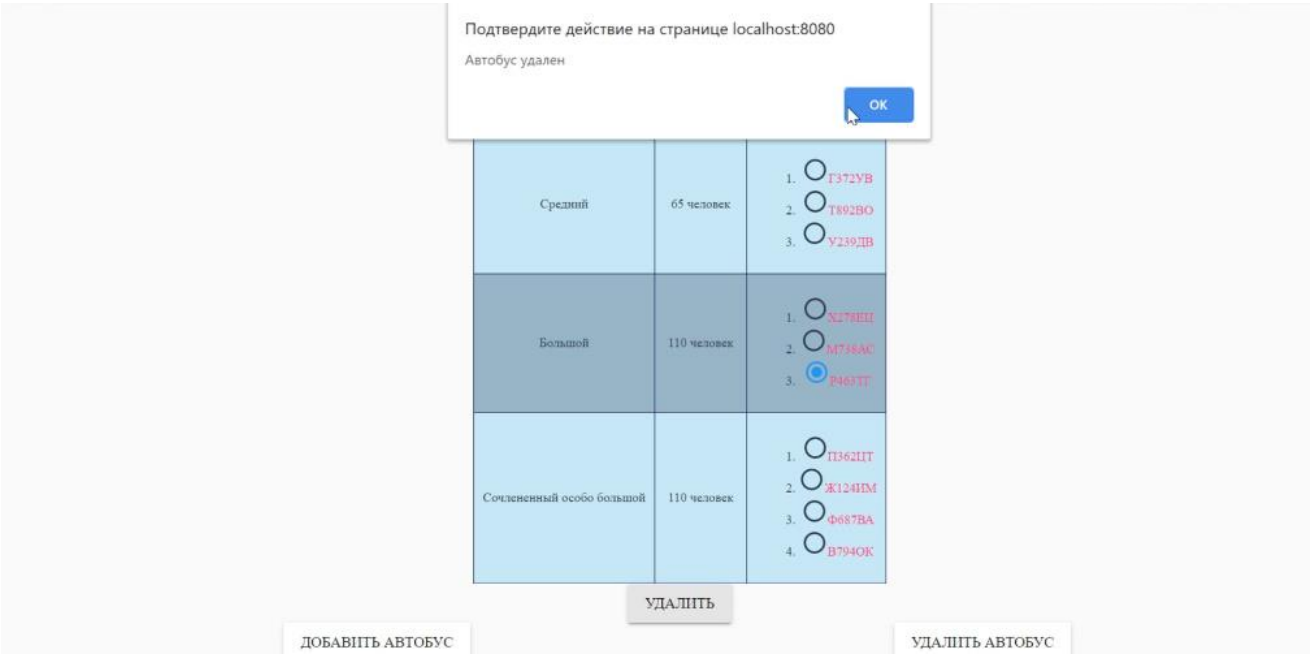

Рис. 22 – Удаление записи

Рис. 23 – Информация об отдельном автобусе

У диспетчера есть доступ к информации о водителях. Он может добавлять/удалять водителя, фильтровать их по классу, изменять информацию о конкретном водителе.

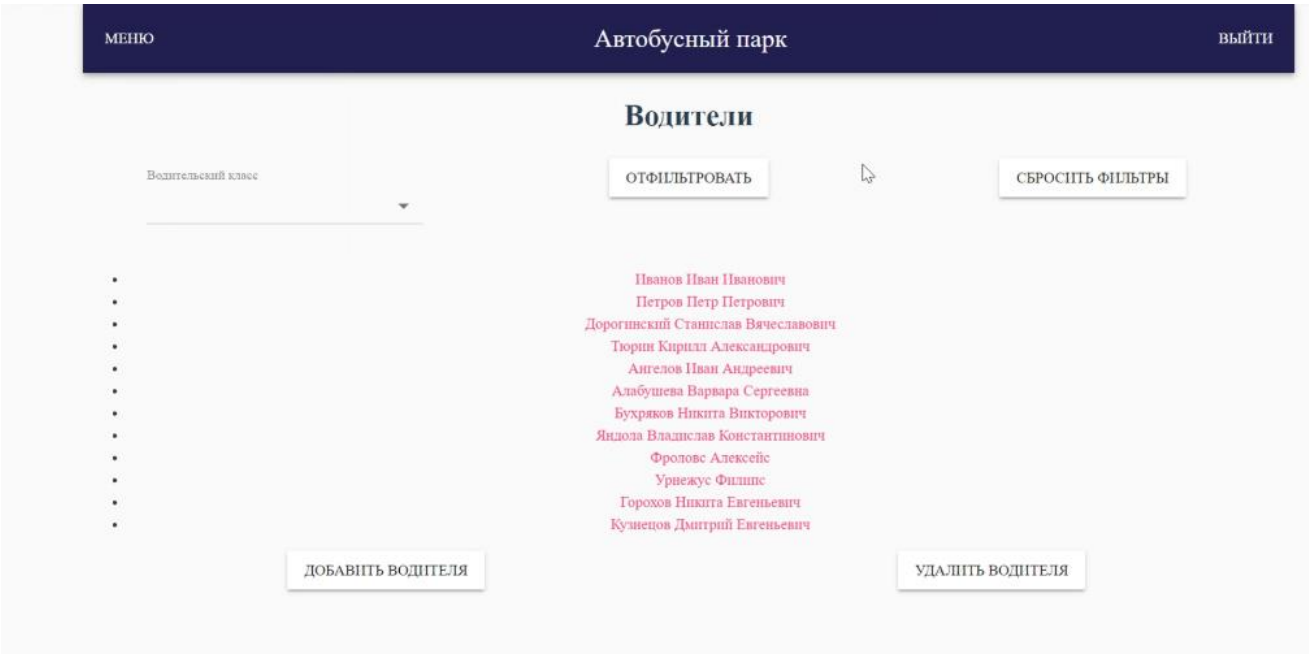

Рис. 24 – Полный список водителей

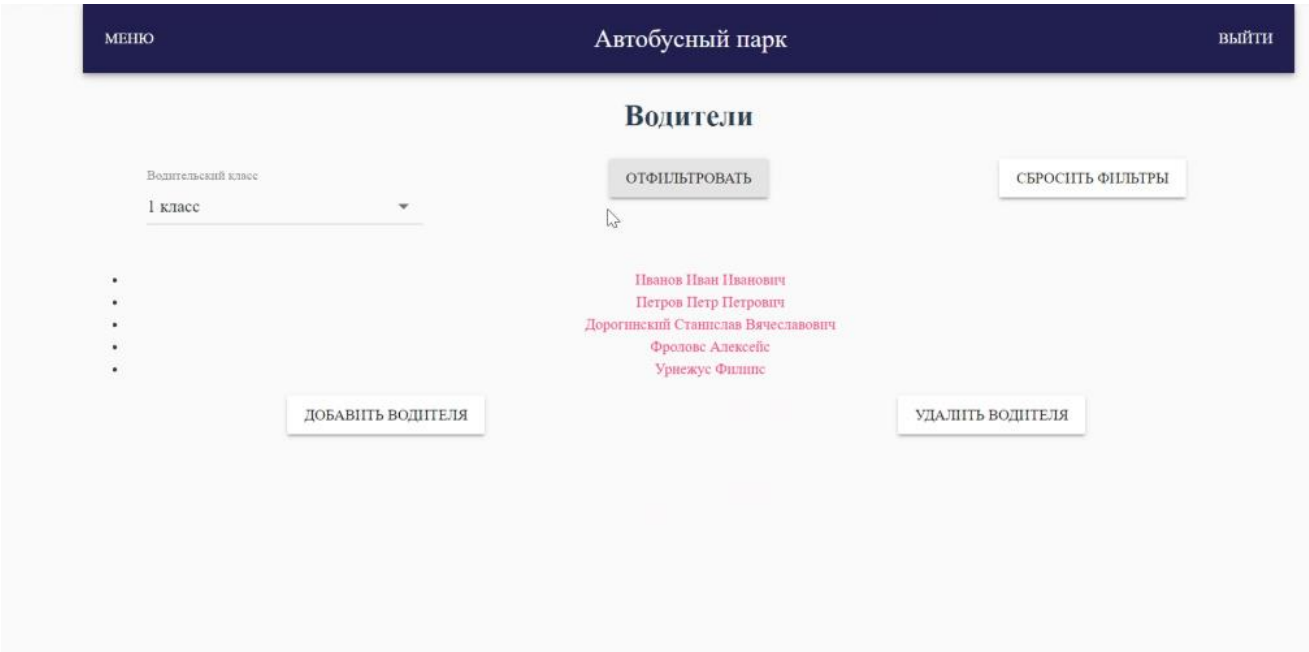

Рис. 25 – Список водителей отфильтрованный по параметру 1-ый класс

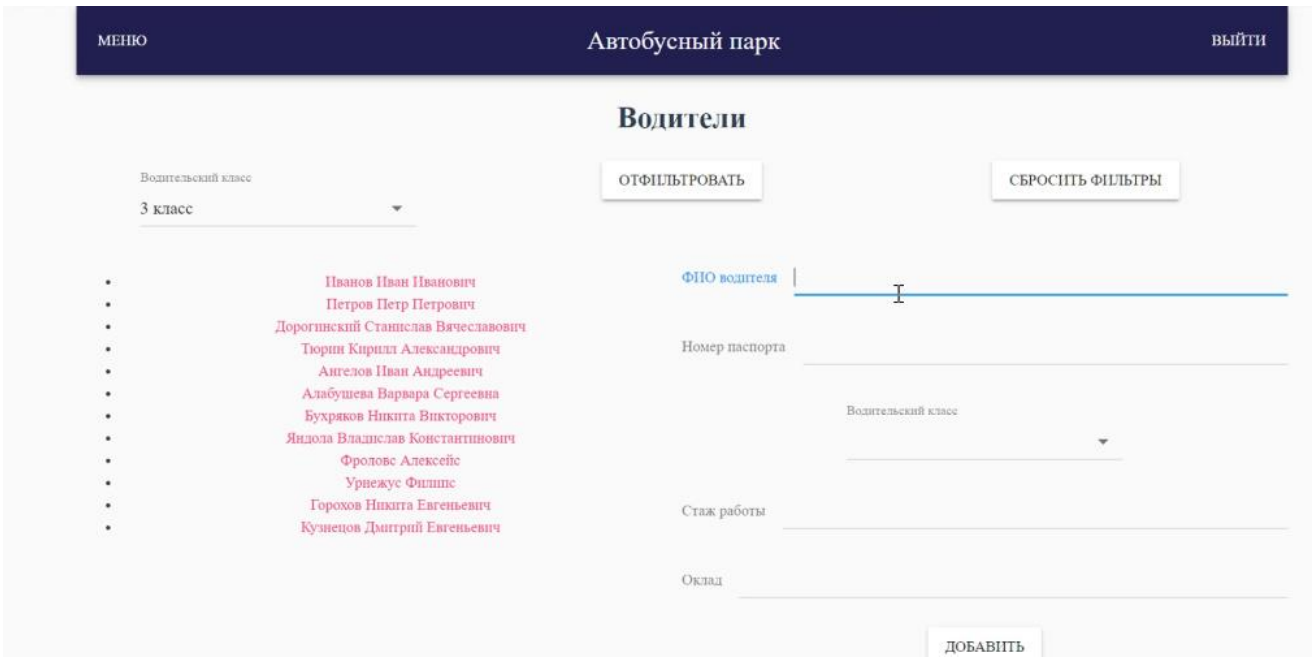

Рис. 26 – Меню добавления нового водителя

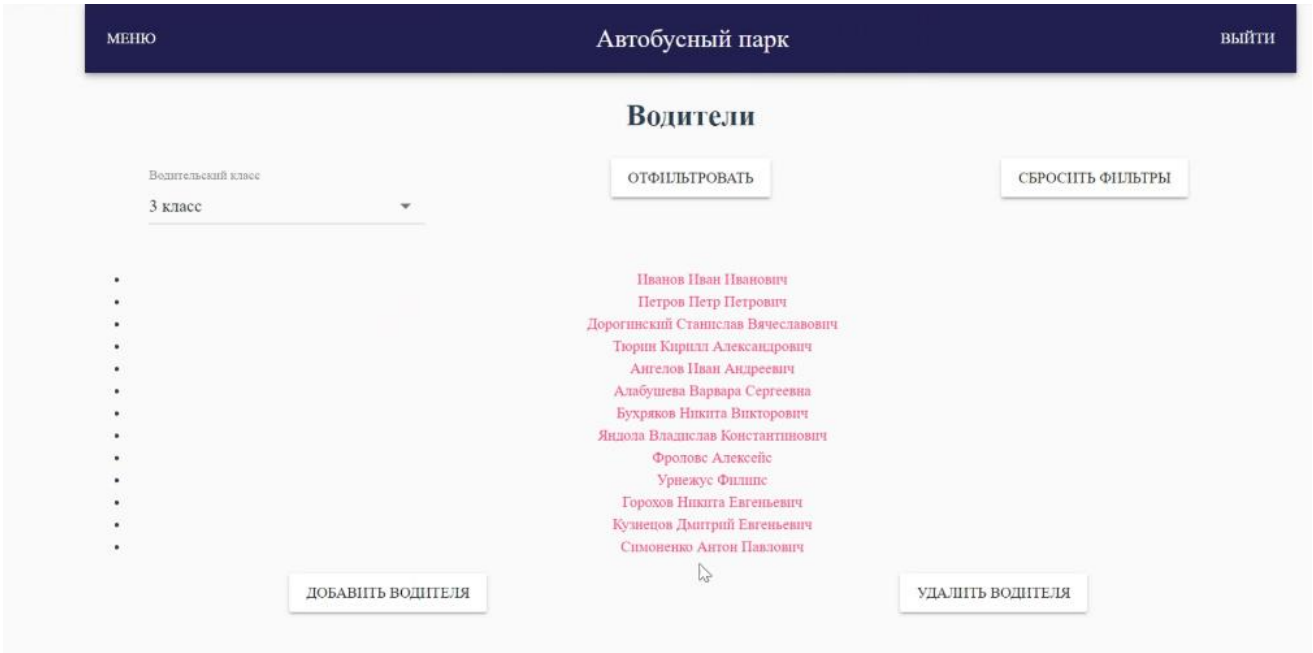

Рис. 27 – Обновленный список водителей

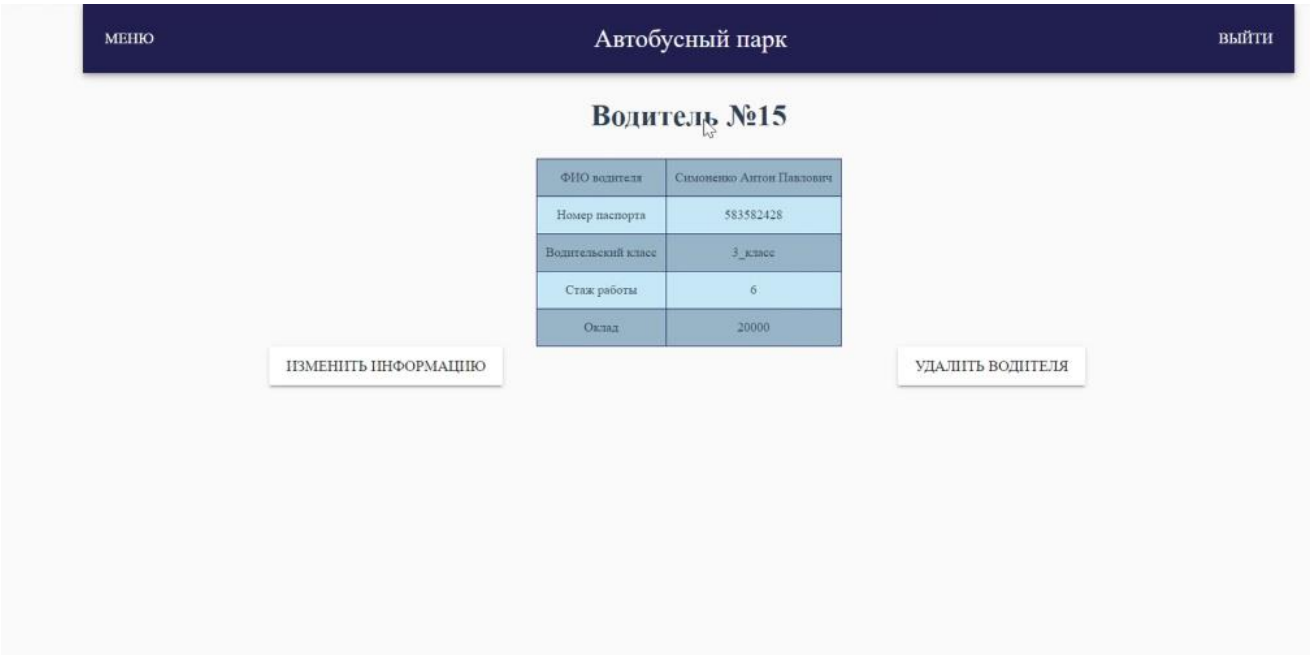

Рис. 28 – Информация о конкретном водителе

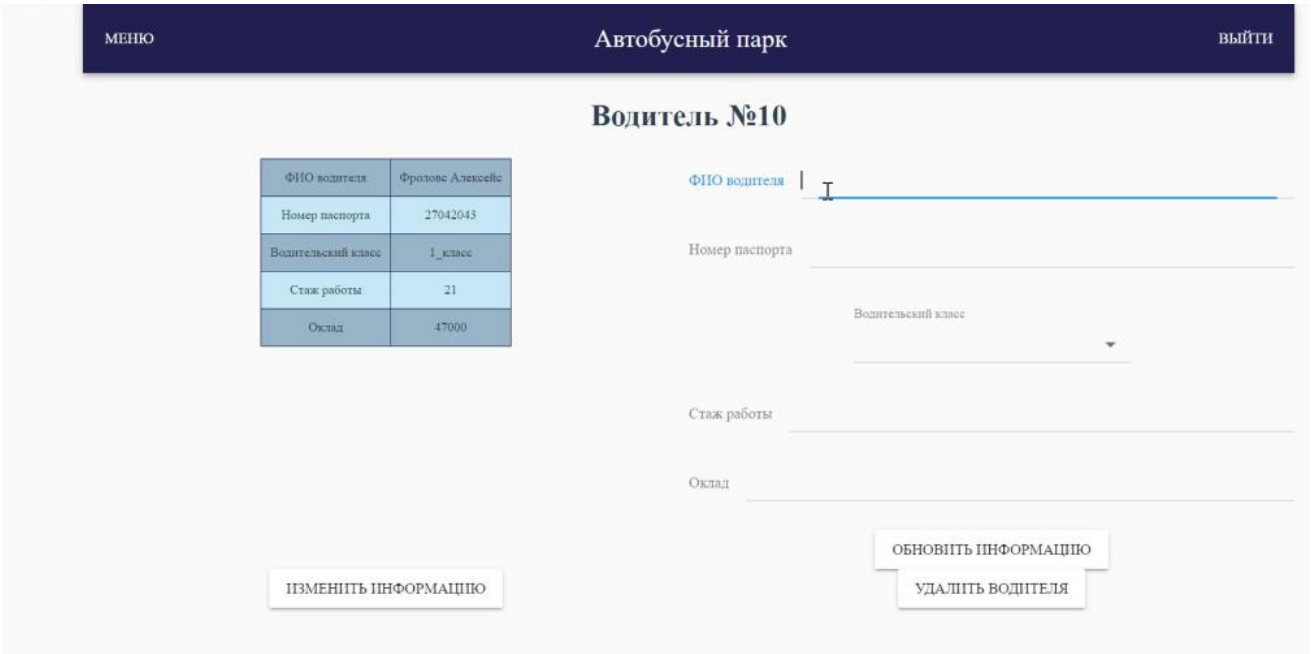

Рис. 29 – Изменение информации о водителе

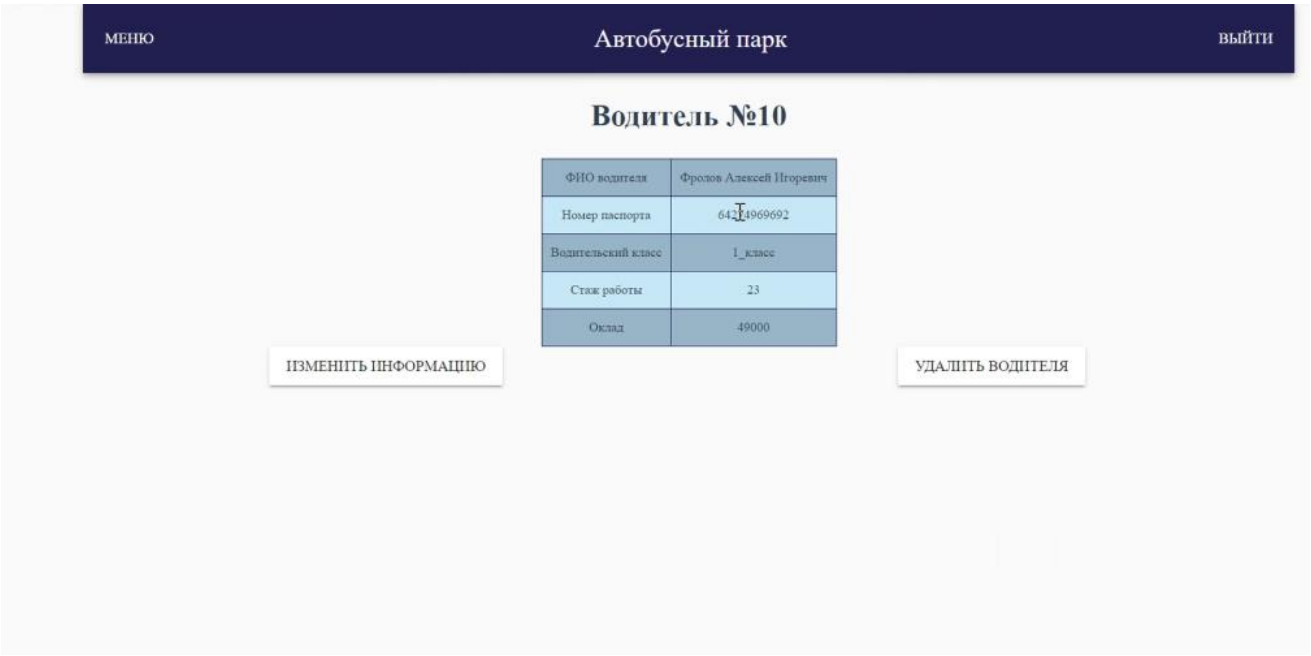

Рис. 30 – Обновленная информация о водителе

В меню маршрутов диспетчеру должна выводиться обо всех маршрутах, а также должна быть возможность добавлять/удалять/изменять информацию о маршруте.

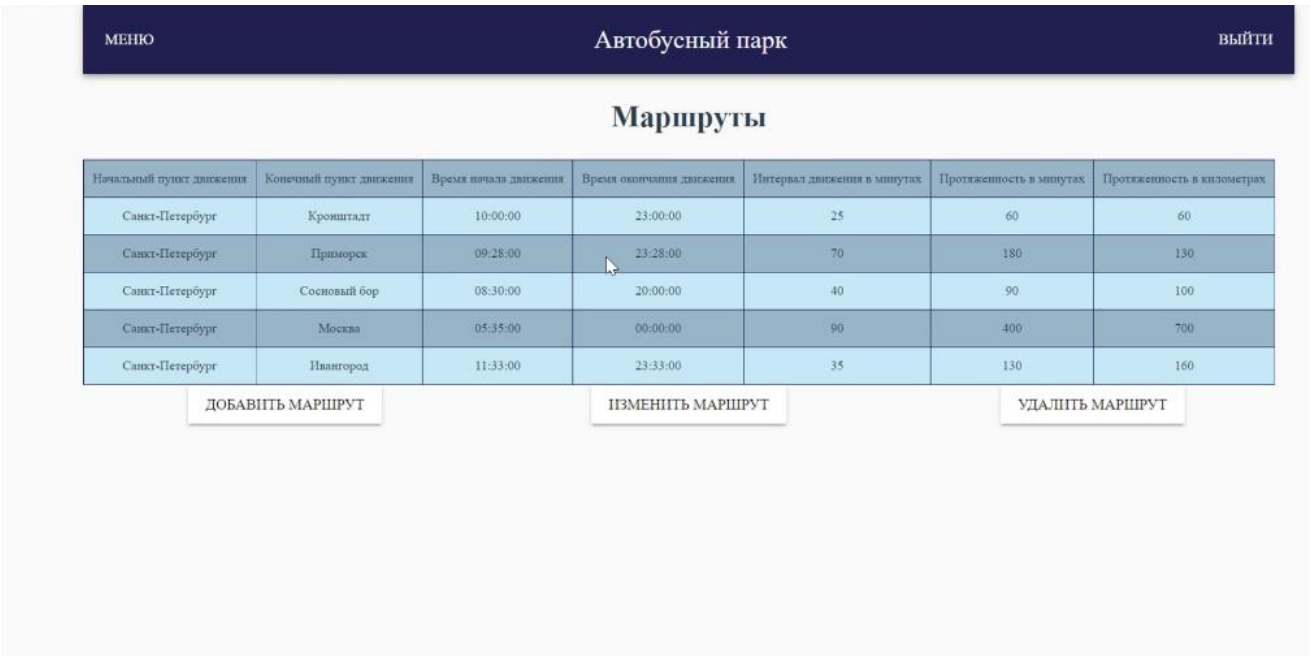

Рис. 31 – Информация обо всех маршрутах

Рис. 32 – Добавление нового маршрута

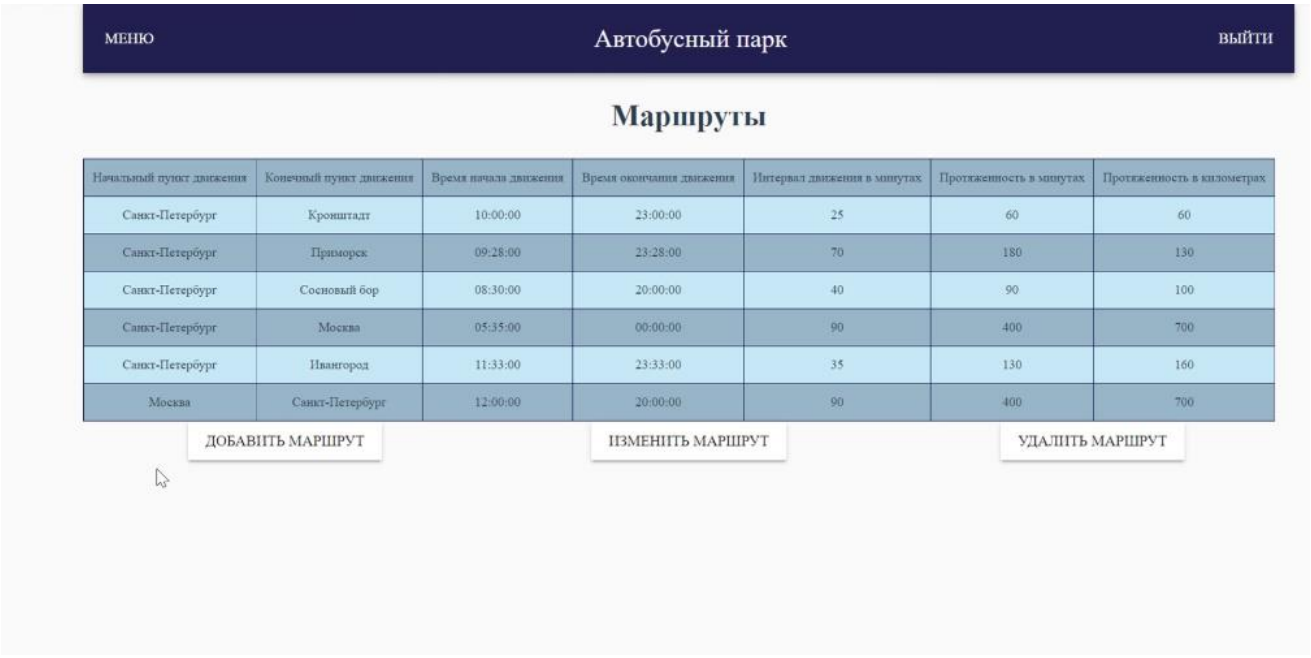

Рис. 33 – Обновленная информация о маршрутах

Рис. 34 – Изменение информации о маршруте

Во вкладке графика работы водителей у диспетчера должен быть доступ к информации обо всех водителях, их маршрутах и автобусах, на которых водители совершают рейс.

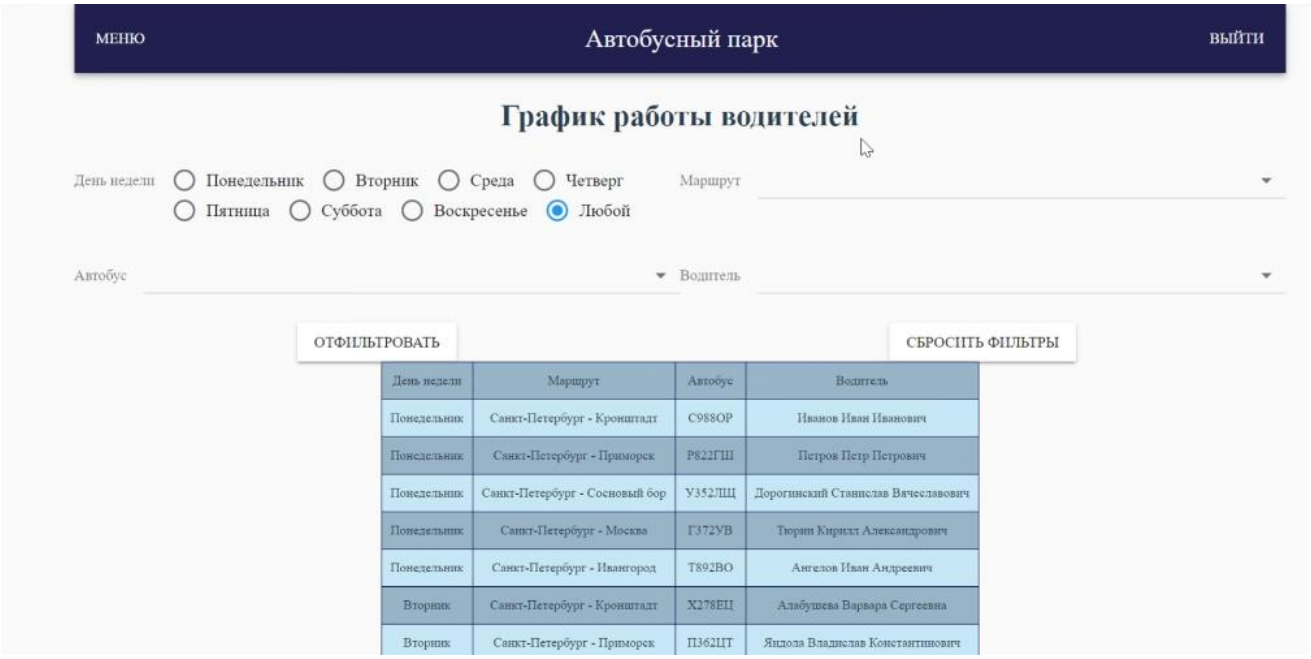

Рис. 35 – Меню графика работы водителей

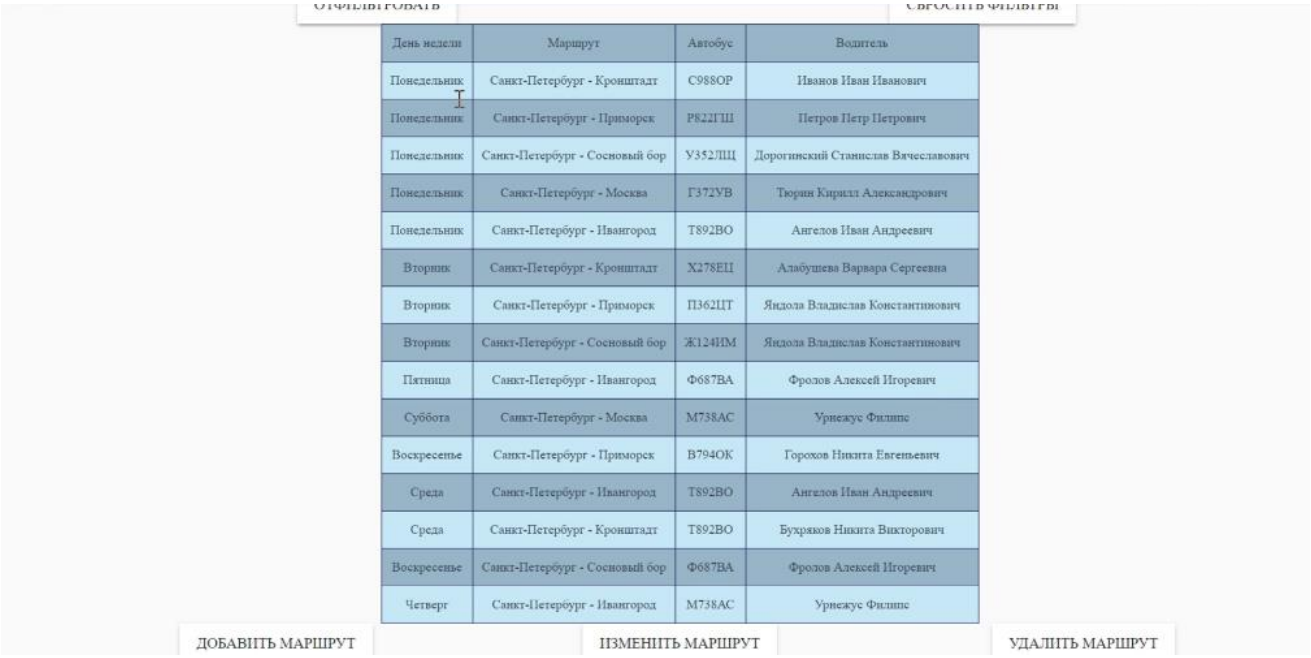

Рис. 36 – Полная информация о графике работы водителей

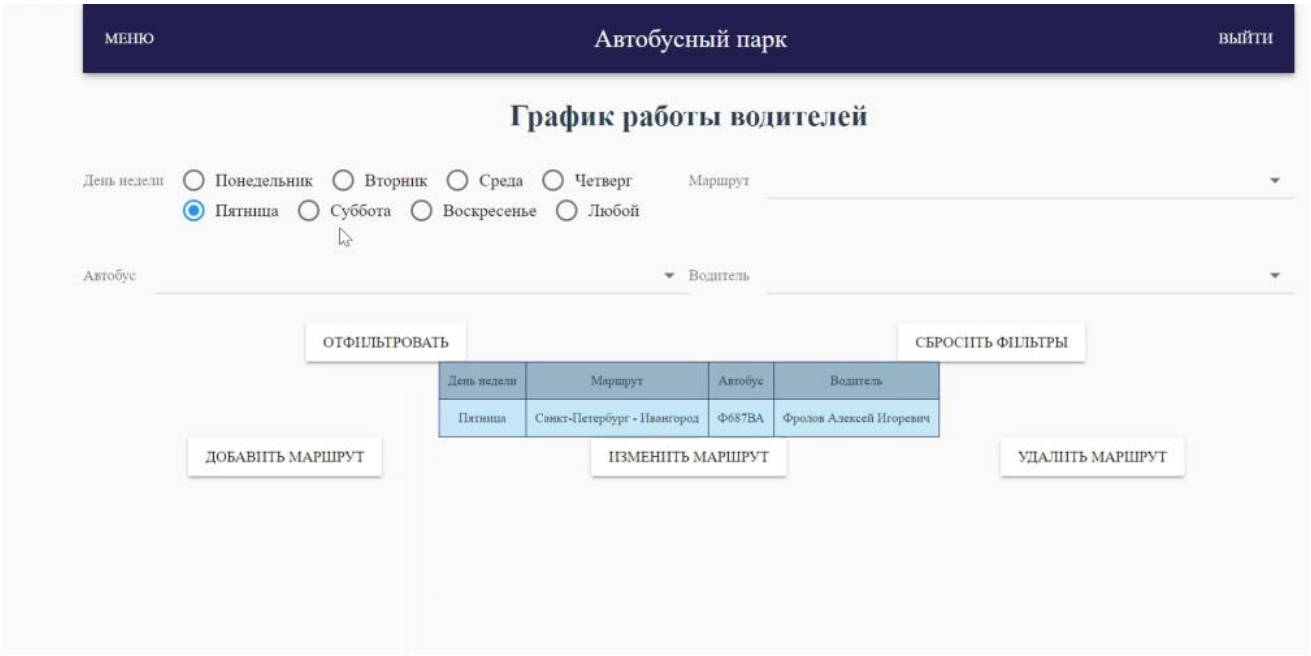

Рис. 37 – Информация о графике работы с фильтром по дню недели (пятница)

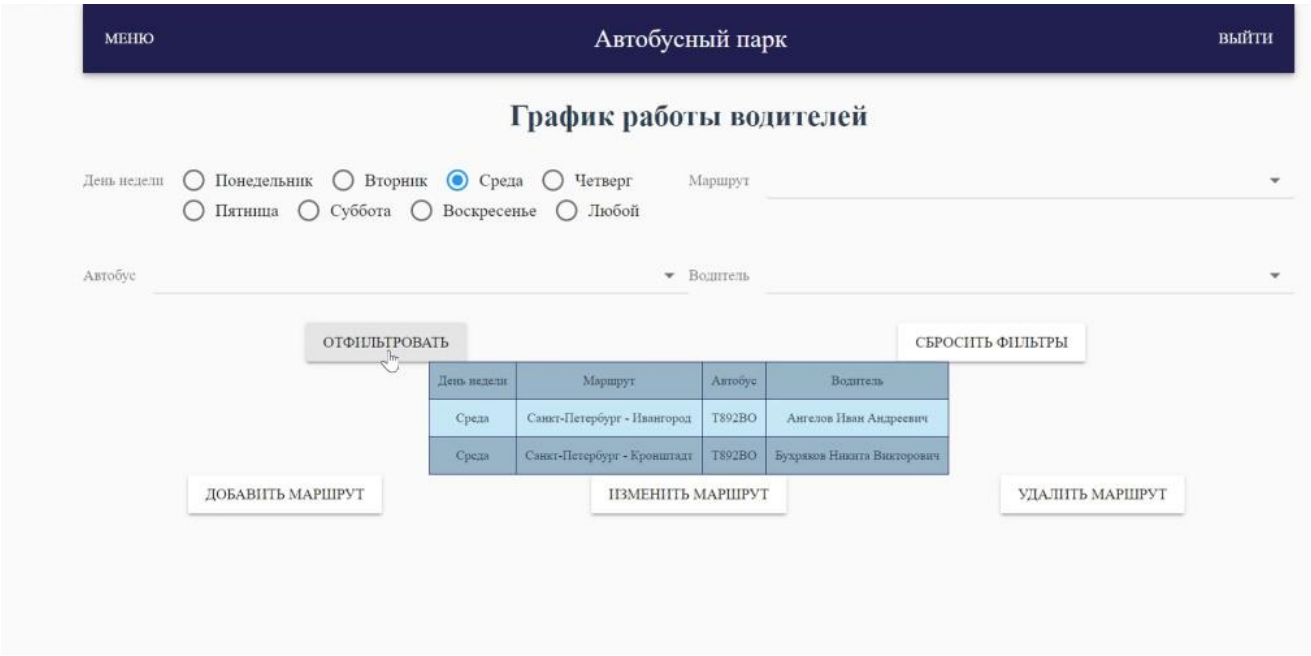

Рис. 38 - Информация о графике работы с фильтром по дню недели (среда)

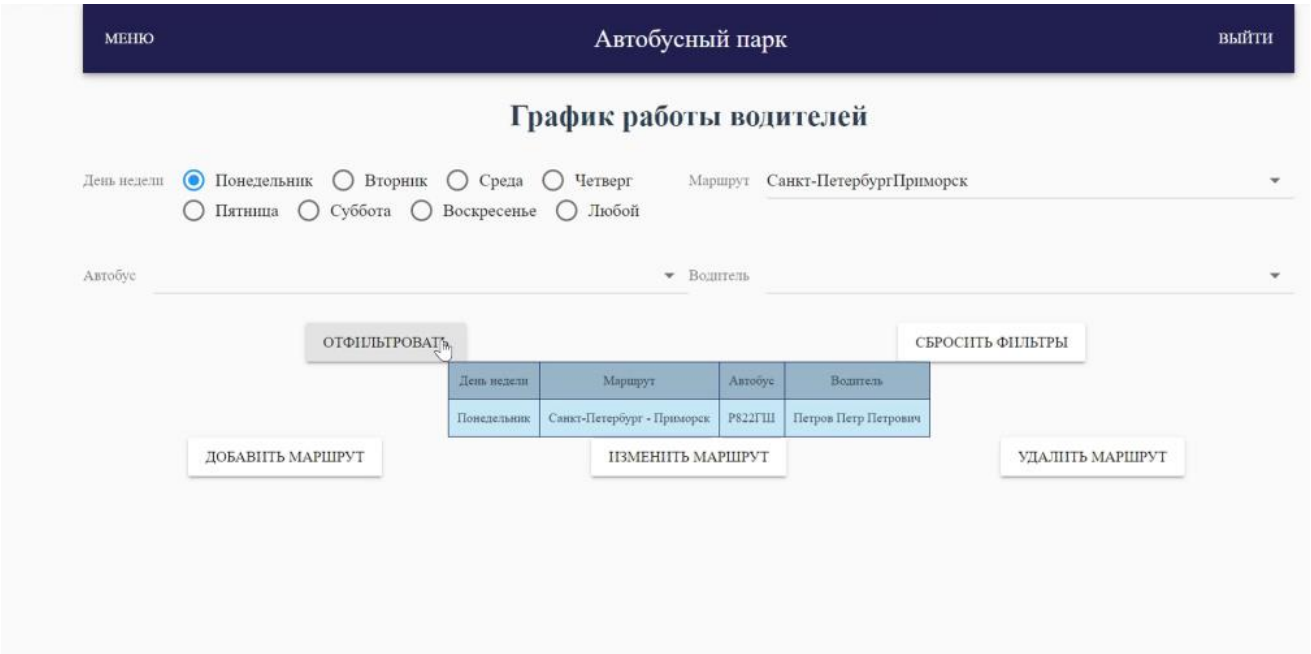

Рис. 39 - Информация о графике работы с двумя фильтрами. По дню недели и маршруту

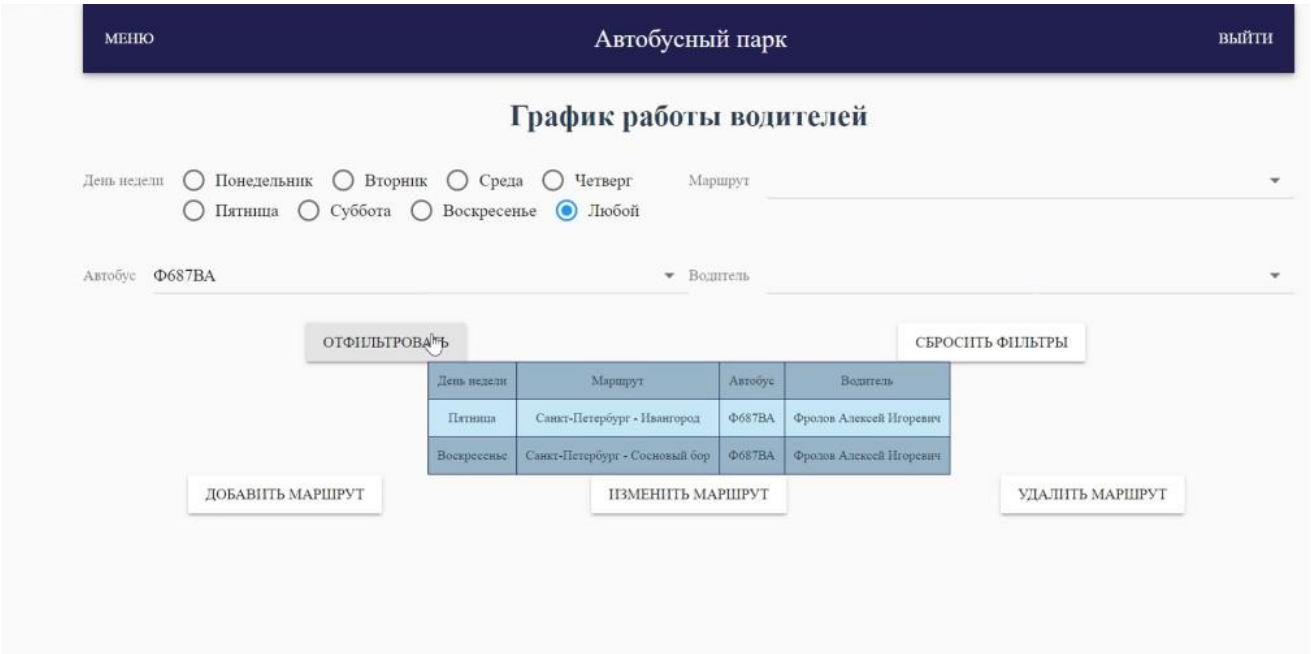

Рис. 40 – Фильтрация по номеру автобуса

Также необходимо было реализовать запросы по курсовой работе. Из выпадающего меню переходим на вкладку запросов.

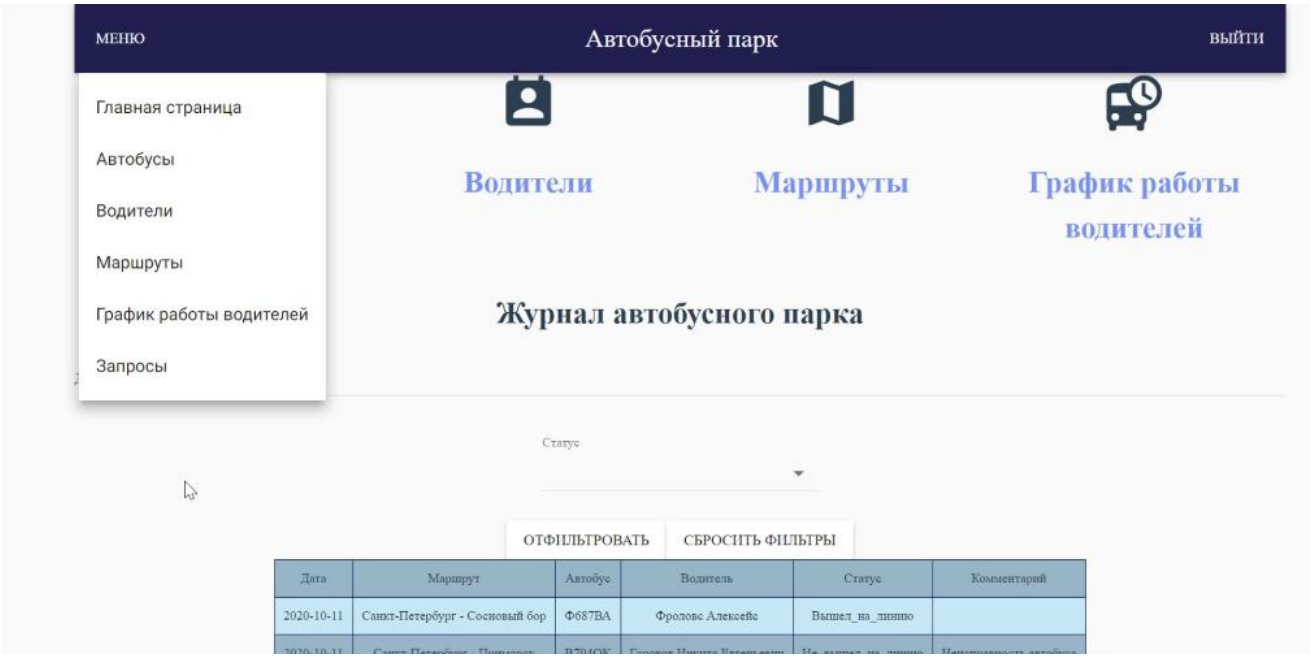

Рис. 41 – Выпадающее меню и вкладка запросов

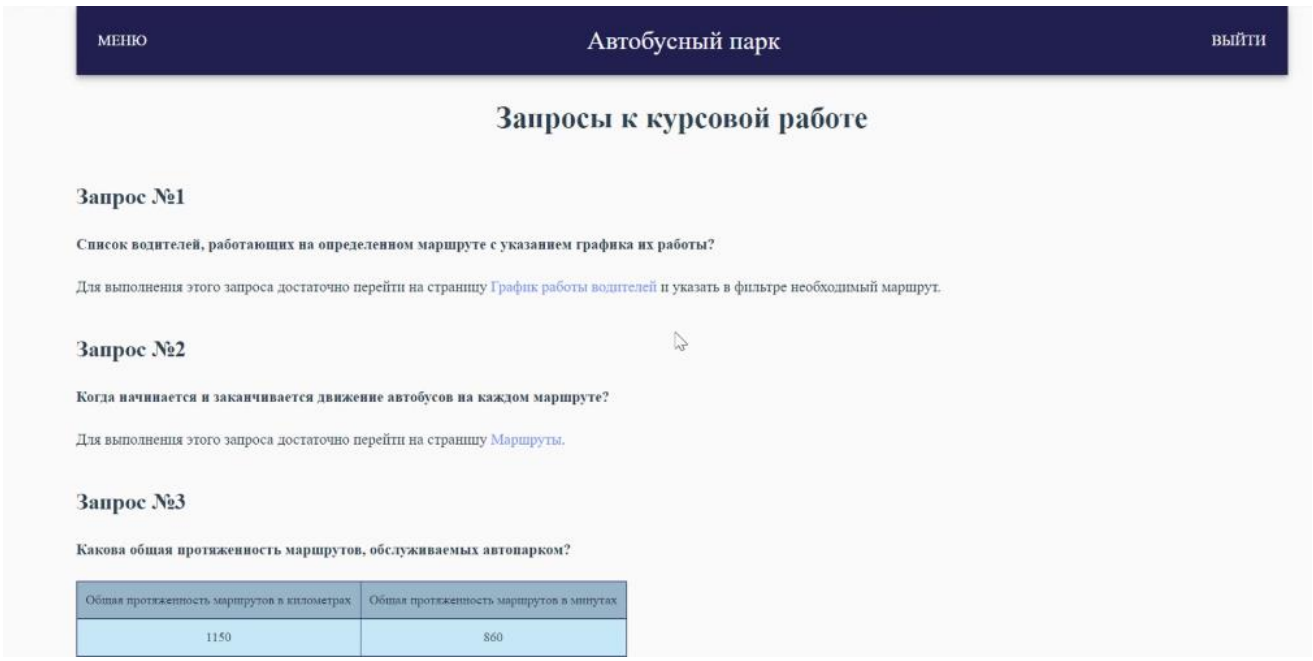

Рис. 42 – Запросы 1-3 по курсовой работе

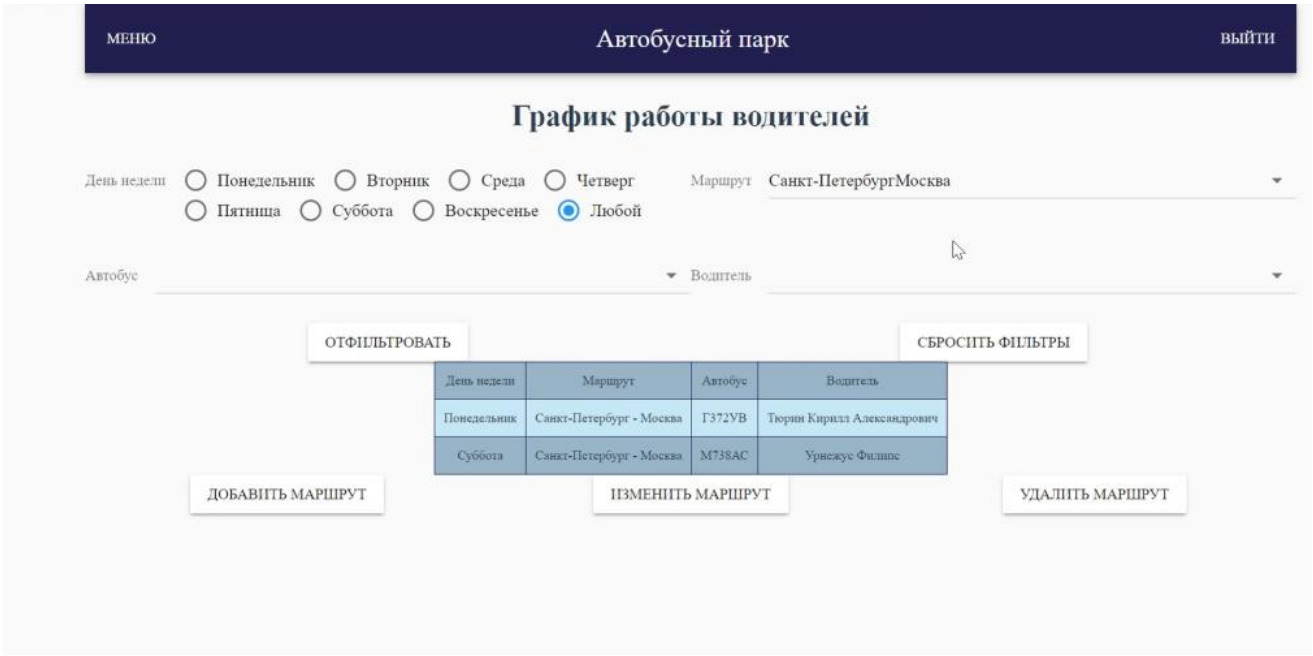

Рис. 43 – Результат первого запроса с фильтрацией по маршруту

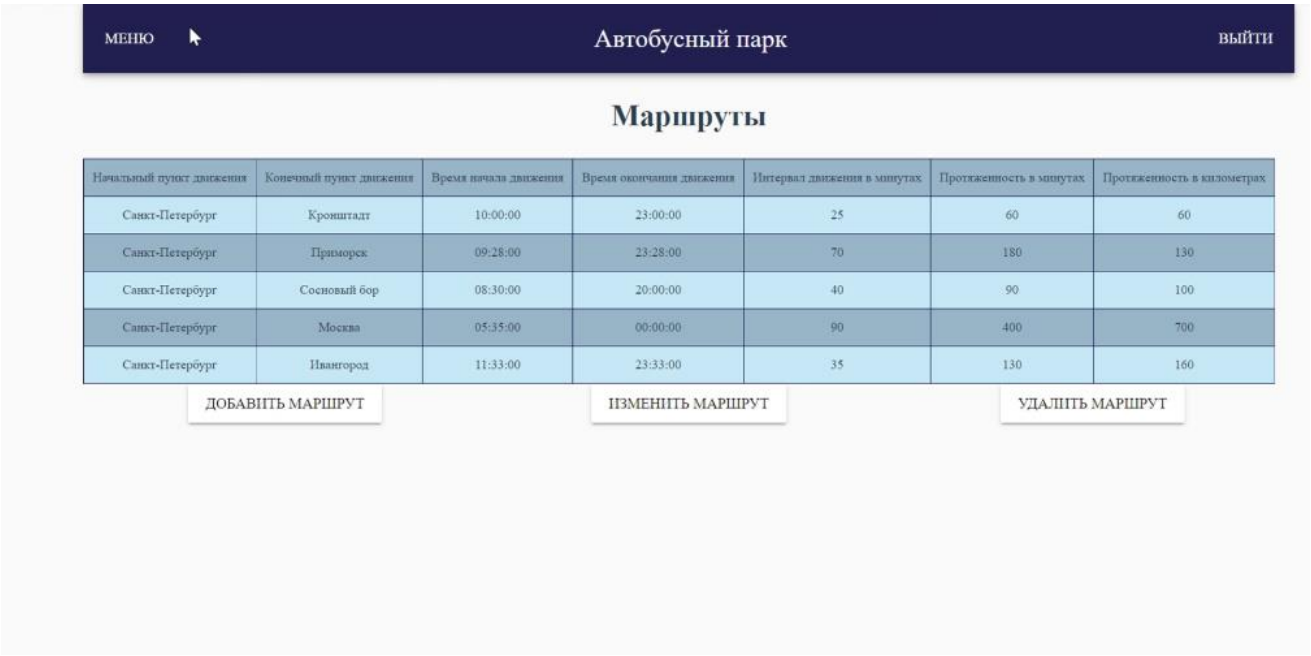

Рис. 44 – Второй запрос с началом и концом движения автобусов

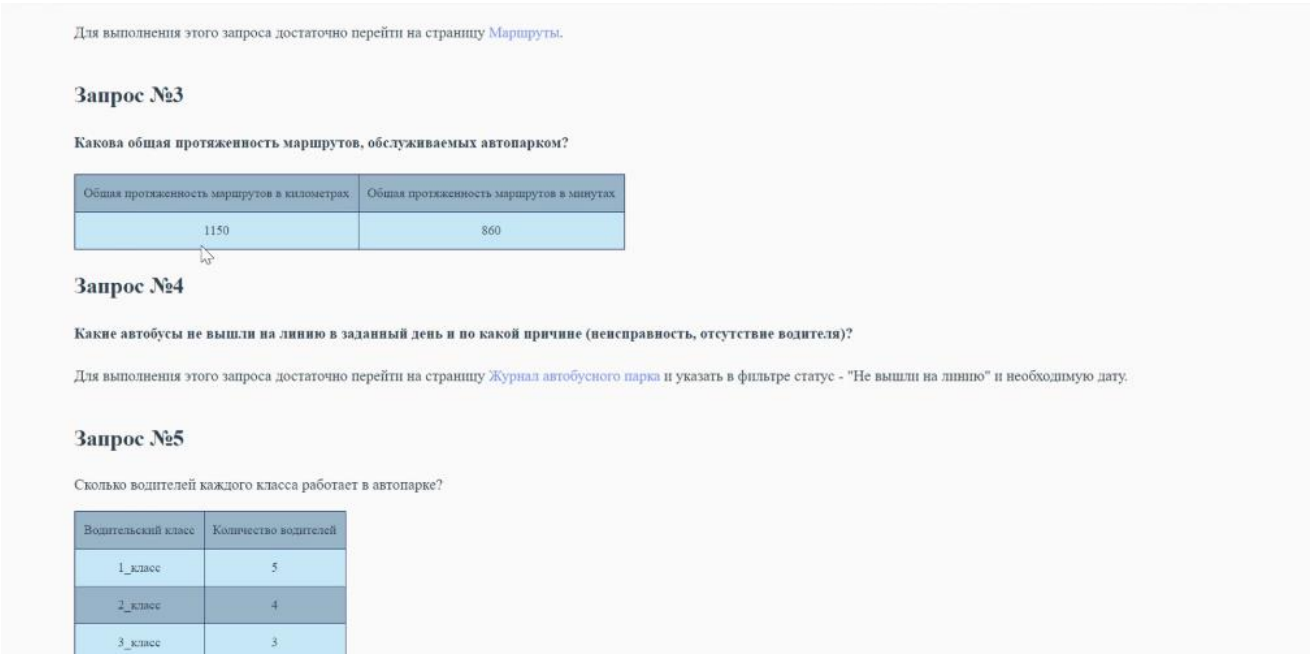

Рис. 45 – Результаты запросов 3-5 по курсовой работе

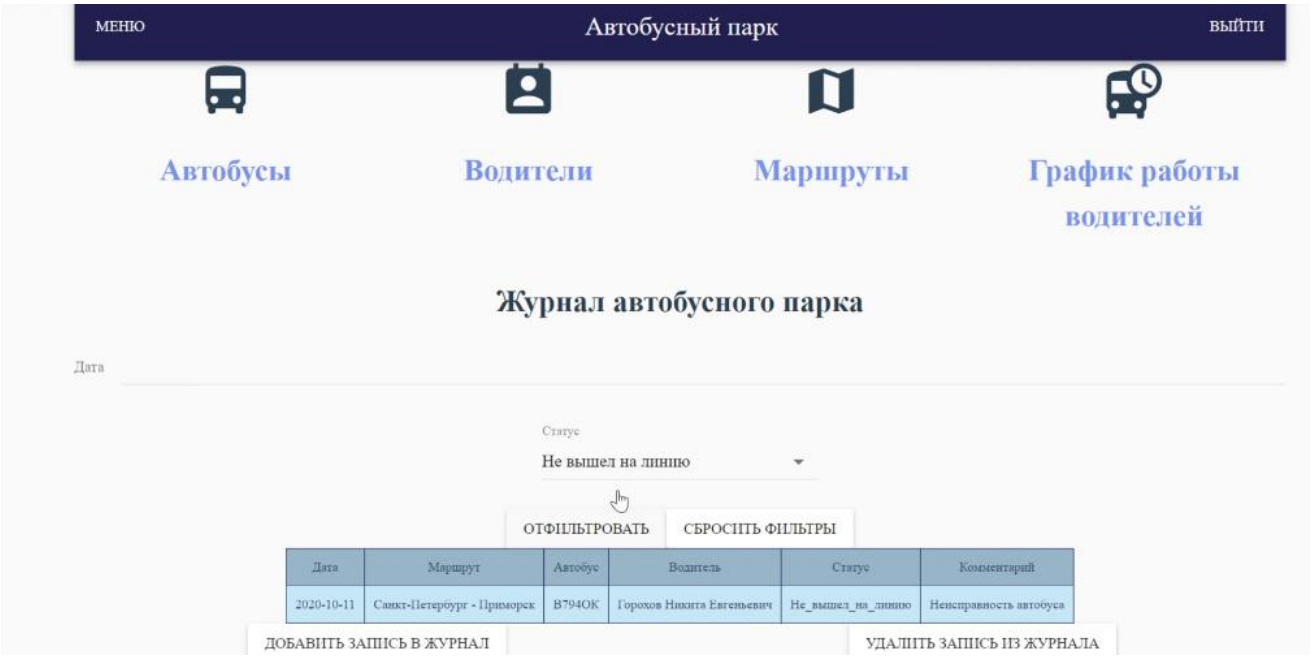

Рис. 46 – Четвертый запрос о не выходе на линию

Запросы 3 и 5 отображаются в самой вкладке запросов. Данные в них могут изменяться в связи с изменением информации в базе данных

Выводы

В результате выбора средств разработки был осуществлен процесс реализации:

− базы данных: созданы модели в Django, расписаны типы данных в полях, логика вычисления или подставления отдельных полей;

− серверная часть: созданы шаблоны адресов в urls.py для всех групп, описанных на этапе проектирования; созданы сериализаторы в serializers.py для моделей в зависимости от типа запроса; созданы представления во views.py для вышеописанных шаблонов адресов;

− клиентская часть: созданы vue компоненты для отдельных страниц, проработано оформление пользовательского интерфейса.

ЗАКЛЮЧЕНИЕ

Был проведен анализ предметной области курсовой работы, на основании чего выдвинуты предполагаемые действия главного действующего лица, состоящие в работе с таблицами базы данных, составлении расписания, карточек поломок, отчетов по работе, получения ответов на запросы.

С помощью этого были выявлены функциональные требования (состав таблиц, логика процессов, наименования запросов, содержание отчета), опираясь на которые, были проведены процессы проектирования и реализации.

В рамках следующих задач были спроектированы: база данных (таблицы с полями, соответствующие характеристикам в функциональных требованиях, и отношения между ними), серверная (состав шаблонов адресов, сериализаторов, представлений) и клиентская части сервиса (количество компонентов, их состав и основная функция).

При решении задачи реализации были выбраны средства разработки Django, Django REST framework, Vue.js, Muse-UI и PostgreSQL, что позволило быстро создать безопасный и поддерживаемый веб-сайт, гибкий и мощный API для проекта, пользовательский интерфейс и базу данных с множеством современных функций. В результате были сформированы модели с характеристиками, шаблоны адресов, функции представлений, компоненты страниц и база данных.

Все поставленные задачи были выполнены в полном объёме. При дальнейшей работы и увеличения эффективности процесса следует добавить более подробные ограничения на ввод значений полей дат, уведомления диспетчеру в личном кабинете при возникновении проблем в карточках поломок, чтобы не было необходимости просматривать каждую карточку отдельно. Тем не менее, реализованная веб-система позволяет обеспечить качественную работу автобусного парка.

СПИСОК ЛИТЕРАТУРЫ

1. Date and Time Formats // World Wide Web Consortium [Электронный ресурс] URl: https://www.w3.org/TR/NOTE-datetime
2. Home // Django REST framework [Электронный ресурс] URL: https://www.django-restframework.org/
3. Introduction // Vue.js [Электронный ресурс] URL: https://vuejs.org/v2/guide/
4. Django documentation // The Web framework for perfectionists with deadlines [Электронный ресурс] URL: https://docs.djangoproject.com/en/3.0/
5. Muse UI [Электронный ресурс] URL: https://muse-ui.org/#/en-US
6. PostgreSQL: Documentation // PostgreSQL: The world’s most advanced open source database [Электронный ресурс] URL: https://www.postgresql.org/docs/