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 Кошкарева Мария Павловна, К3342

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
20 20

Оглавление

ВВЕДЕНИЕ 3

1. ПРЕДМЕТНАЯ ОБЛАСТЬ 4

2. ПРОЕКТИРОВАНИЕ 5

2.1. Функциональные требования 5
2.2. База данных 6
2.3. Серверная часть сервиса 9
2.4. Клиентская часть сервиса 10
Выводы 11

3. РЕАЛИЗАЦИЯ 12

3.1. Описание средств разработки 12
3.2. База данных 13
3.3. Серверная часть сервиса 15
3.3.1. Эндпоинты для базовых операций преимущественно с одной таблицей 15
3.3.2. Эндпоинты, используемые в интерфейсе запросов 18
3.3.3. Эндпоинты, используемые в интерфейсе отчета 19
3.4. Клиентская часть сервиса 20
3.4.1. До входа пользователя 20
3.4.2. Для добавления/удаления основных данных 21
3.4.3. Для контроля 25
3.4.4. Для получения данных 27
Выводы 29

ЗАКЛЮЧЕНИЕ 30

СПИСОК ЛИТЕРАТУРЫ 31

ВВЕДЕНИЕ

В большинстве случаев автобусная система развитого города идет совместно с мероприятиями по техническому обслуживанию и контролю парка подвижного состава, составлению расписания для выпуска автобусов на маршрут, комплектации штата водителей. [1]

При регулярном обновлении базы сотрудников и обслуживаемого транспорта общего пользования необходимо поддерживать качество транспортных услуг автобусного парка на должном уровне, что достигается путем предоставления диспетчеру данного парка доступа к специализированной программной системе, предназначенной для контроля за вышеописанными процессами.

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 о 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.

– Средний возраст и стаж водителей.

2.2. База данных

Следуя функциональным требованиям, в базе данных должны присутствовать таблицы Водителей, Автобусов, Маршрутов, Графика работы, Поломок и Отчетов по работе. При построении схемы и настройке связей между таблицами, получается следующее:

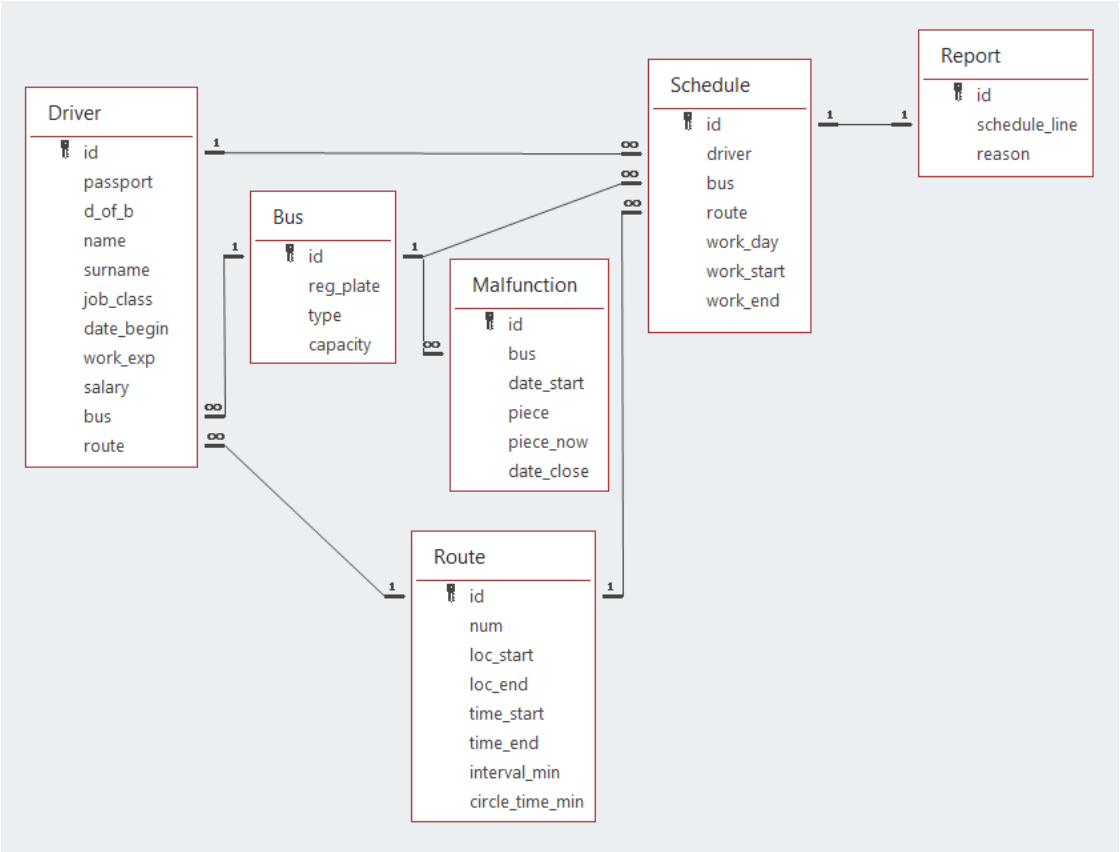


Рисунок 1 – Схема базы данных

Таблица Driver (Водитель).

Здесь будут храниться данные по водителям автопарка. Связана с таблицей Графика работы связью один ко многим (так как в Графике работы должна быть возможность выбора водителя, на которого будет назначен рабочий день); имеет внешние ключи от таблиц Автобуса и Маршрута (по причине того, что у водителя должны быть закрепленные Автобус и Маршрут; но это не подразумевает, что больше никто не может закрепиться за этими же записями таблиц, поэтому связь не один к одному). Имеет следующие характеристики водителя:

– id (Идентификатор)
– passport (Номер и серия паспорта)
– d_of_b (Дата рождения в формате YYYY-MM-DD [2])

– name (Имя водителя)
– surname (Фамилия)
– job_class (Класс водителя (1, 2, 3))
– date_begin (Дата начала работы в формате YYYY-MM-DD)
– work_exp (Опыт работы в годах)
– salary (Оклад).
– bus (Закрепленный за водителем автобус)
– route (Закрепленный за водителем маршрут)

Таблица Bus (Автобус).

Хранит данные об автобусах, находящихся в автопарке. Связана с таблицами Водителя (чтобы осуществлять закрепление транспорта за водителем), Поломок (для записи случаев поломок), Графика работы (нужно предоставить выбор транспорта, на котором пройдет рабочий день) связью один ко многим. Имеет поля:

– id (Идентификатор)
– reg_plate (Номер государственной регистрации)
– type (Тип (особо малый, малый, средний, большой, сочлененный))
– capacity (Вместимость в людях)

Таблица Route (Маршрут).

Хранятся сведения о маршрутах, обслуживаемых автопарком. Связана с таблицами Водителя (для закрепления Водителя к определенному маршруту) и Графика работами (чтобы осуществить выбор маршрута при добавлении рабочего дня) связью один ко многим. Находятся следующие поля:

– id (Идентификатор)
– num (Номер маршрута)
– loc_start (Название начального пункта движения)
– loc_end (Название конечного пункта движения)
– time_start (Время начала движения в формате hh:mm:ss)
– time_end (Время конца движения в формате hh:mm:ss)
– interval_min (Интервал движения в минутах)
– circle_time_min (Протяженность (время движения от кольца до кольца) в минутах)

Таблица Malfunction (Поломка).

Здесь находятся сведения о поломках автобусов автопарка. Имеет внешний ключ от таблица Автобуса, чтобы можно было создавать карточку поломки на определенный автобус (из-за того, что на один автобус может быть заведено много карточек, чтобы они потом хранились в «истории поломок», связь не один к одному). Здесь присутствуют поля:

– id (Идентификатор)
– bus (Сломанный автобус)
– date_start (Дата начала поломок в формате YYYY-MM-DD)
– piece (Сломанные детали (двигатель, шины, масляный фильтр, фары, поворотники, руль, зеркала заднего вида, другое))
– piece_now (Сломанные детали на данный момент)
– date_close (Дата окончательной починки автобуса в формате YYYY-MM-DD)

Таблица Schedule (График работы).

Хранит данные о записи водителей на рабочие дни. Имеет внешние ключи от таблиц Водителя (выбор водителя на работу), Автобуса (выбор транспорта, на котором пройдет рабочий день) и Маршрута (выбор, где будет курсировать водитель). Связана с таблицей Отчетов по работе связью один к одному (на каждую отдельную запись в расписании приходится одна запись отчета). Характеристики графика работы:

– id (Идентификатор)
– driver (Водитель)
– bus (Автобус)
– route (Маршрут)
– work_day (День работы в формате YYYY-MM-DD)
– work_start (Время начала работы в формате hh:mm:ss)
– work_end (Время конца работы в формате hh:mm:ss)

Таблица Report (Отчет по работе).

Хранятся сведения об успешном или нет прохождении рабочего дня в соответствии с расписанием, поэтому связана с таблицей Графика работы связью один к одному. Поля для описания отчета по работе:

– id (Идентификатор)
– schedule_line (Строка из графика работы)
– reason (Причина (ОК, сломанный автобус, больной водитель, водитель отсутствует, другое))

2.3. Серверная часть сервиса

Так как используется Django REST framework [3], то, помимо прочих, 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. Более того, сопоставитель может извлекать данные из адреса и передавать их в соответствующую функцию в виде аргументов.

Принимая во внимание функциональные требования, необходимо определить следующие шаблоны:

1. Шаблоны адресов для работы с отдельными таблицами (чтение всех экземпляров, чтение отдельной записи по id, удаление, создание, удаление)
2. Шаблоны адресов для работы с блоком запросов для осуществления выборки из нескольких таблиц, фильтрации, сортировки, группировки, применения агрегатных функций и так далее (чтение всех записей, чтение отдельной записи по id).
3. Шаблоны адресов для работы с блоком отчета. Сложные операции аналогичные описанным выше (чтение всех записей).

– views.py

Здесь находится обработчик запросов: получает HTTP запросы от веб-клиентов и возвращает HTTP-ответы. Имеет доступ к данным через модели для удовлетворения запросов. Здесь будут также определяться функции (чтение, обновление, создание, удаление).

Нужно будет проработать следующие детали в функциях отображения:

1. Для отдельных функций отображения определить функции чтения как всех записей, так и отдельных по переданному в запросе id; добавления записи в таблицу с валидацией; удаления записи по переданному id, обновления записи (отдельных полей) по переданному id.
2. Для отдельных функций прописать выборку с условиями, описанными в п.2 про urls.py.

3. Для отдельных функций настроить отправление сразу нескольких данных по результатам запроса для получения всех необходимых сведений с помощью одного запроса.

– 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.

Настроить следующие сериализаторы:

1. Для чтения всех записей или отдельной: задействовать все поля модели.
2. Для добавления новой записи: прописать все поля, за исключением идентификатора.
3. Для обновления записи: выбрать все поля (если присутствуют вычисляемые или подставляемые поля в модели, то оставить только те поля, которые можно заполнить вручную).

2.4. Клиентская часть сервиса

Вследствие того, что важной концепцией Vue [4] 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. К тому же стоит обезопасить веб-сервис от использования кем-то, помимо диспетчера.

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. При успешном вводе открывается домашняя страница, при неуспешном – выводится информация о несоответствии введенных данных с присутствующими в базе.

– Компонент страницы списка водителей

Диспетчер должен видеть всех водителей с их характеристиками и суметь быстро найти нужного. Также на странице пользователю должна быть дана возможность добавления или удаления водителя.

– Компонент страницы поломок

На данной странице должны присутствовать активные поломки вместе с историей поломок. При обнаружении записи в графике работу на поломанный автобус, диспетчер должен иметь возможность разрешить проблему. К тому же должны быть кнопки добавления и удаления карточек поломок.

– Компонент страницы отчетов по работе

Пользователю показываются прошедшие записи графика работы, у которых можно изменить причину в отчете по работе.

– Компонент страницы запросов

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

– Компонент страницы отчета

Здесь диспетчер может только просматривать уже сформированный отчет по автопарку; должна быть предоставлена возможность выбора части отчета для ознакомления.

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 [5]

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 [6]

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 [7]

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

Модель Driver (Водитель)

```
class Driver(models.Model):
    CLASSES = (
        (3, 3), (2, 2), (1, 1)
    )

    passport = models.CharField(max_length=10, unique=True)
    d_of_b = models.DateField(null=True)
    name = models.CharField(max_length=30)
    surname = models.CharField(max_length=30)
    job_class = models.IntegerField(choices=CLASSES)
    date_begin = models.DateField()
    work_exp = models.IntegerField(default=0, help_text='fills in based on '
                                               '"Date begin"')
    salary = models.IntegerField(default=0, help_text='fills in based on '
                                             '"Work exp" and "Job class"')
    bus = models.ForeignKey(Bus, on_delete=models.CASCADE)
    route = models.ForeignKey(Route, on_delete=models.CASCADE)
```

Рисунок 2 – Модель Водителя

В данной модели есть два вычисляемых поля:

1. Опыт работы: текущая дата минус дата начала работы.
2. Оклад: минимальный – 19 000; за каждые два года опыта доплачиваются 3 000; при опыте работы свыше 30 лет надбавки не начисляются, устанавливается пороговая сумма в 65 000; в зависимости от класса водителя оклад также умножается на коэффициент (1 класс – 1.2, второй – 1.1, третий – 1.07).

Модель Bus (Автобус).

```
class Bus(models.Model):
    TYPES = (
        (1,'very little'),(2,'little'),
        (3,'average'), (4,'big'),
        (5,'articulated')
    )

    reg_plate = models.CharField(max_length=6, unique=True)
    type = models.IntegerField(choices=TYPES)
    capacity = models.IntegerField(default=0,
                                   help_text='fills in based on "Type"')
```

Рисунок 3 – Модель Автобуса

Здесь одно вычисляемое поле:

1. Вместимость. В зависимости от типа автобуса устанавливаются следующие значения: особо малый – 5, малый – 40, средний – 65, большой – 90, сочлененный – 110.

Модель Route (Маршрут)

```
class Route(models.Model):
    num = models.IntegerField(unique=True)
    loc_start = models.CharField(max_length=100)
    loc_end = models.CharField(max_length=100)
    time_start = models.TimeField()
    time_end = models.TimeField()
    interval_min = models.IntegerField()
    circle_time_min = models.IntegerField()
```

Рисунок 4 – Модель Маршрута

Модель Malfunction (Поломка)

```
class Malfunction(models.Model):
    # piece: engine, tyres, oil filter, lights, turn signals,
    #            steering wheel, rear-view mirrors, other
    bus = models.ForeignKey(Bus, on_delete=models.CASCADE)
    date_start = models.DateField()
    piece = models.CharField(max_length=100)
    piece_now = models.CharField(null=True, blank=True, max_length=100)
    date_close = models.DateField(null=True, blank=True)
```

Рисунок 5 – Модель Поломки

В модели два подставляемых поля:

1. Сломанные детали на данный момент: если запись поломки создается, а не обновляется, то в это поле подставляется значение «Сломанные детали».
2. Дата окончательной починки: если в поле «Сломанные детали на данный момент» пусто, то ставится текущая дата.

Модель Schedule (График работы)

```
class Schedule(models.Model):
    driver = models.ForeignKey(Driver, on_delete=models.CASCADE)
    bus = models.ForeignKey(Bus, on_delete=models.CASCADE, default=4,
                            help_text="Don't change to get the driver's bus")
    route = models.ForeignKey(Route, on_delete=models.CASCADE, default=3,
                              help_text="Don't change to get the driver's route")
    work_day = models.DateField(null=True)
    work_start = models.TimeField(null=True)
    work_end = models.TimeField(null=True)
```

Рисунок 6 – Модель Графика работы

В модели два подставляемых поля:

1. Автобус: если не был убран автобус по умолчанию с номером «000000», то ставится закрепленный за выбранным водителем автобус.
2. Маршрут: если не был убран маршрут по умолчанию с номером «0», то ставится закрепленный за выбранный водителем маршрут.

Модель Report (Отчет по работе)

```
class Report(models.Model):
    REASON_LIST = (
        (0,"OK"), (1,'Broken bus'),
        (2,'Sick driver'), (3,'No driver'),
        (4,'Other'),
    )
    schedule_line = models.OneToOneField(Schedule, on_delete=models.CASCADE)
    reason = models.IntegerField(choices=REASON_LIST,default=0)
```

Рисунок 7 – Модель Отчета по работе

При создании новой записи в таблице «График работы» автоматически создается соответствующая запись в таблице «Отчет по работе» с причиной ОК.

3.3. Серверная часть сервиса

3.3.1. Эндпоинты для базовых операций преимущественно с одной таблицей

Используются для операций с таблицей Поломок:

```
path('malfunction/', Malfunctions.as_view()),
path('malfunction/<int:pk>/', Malfunctions_detail.as_view()),
```

Рисунок 8 – шаблоны адресов для запросов к таблице Поломок

1. Принимается GET-запрос; в таблице Поломок выбираются значения с датой и без даты окончательного ремонта; отправляются активные и неактивные карточки поломок;

   Принимается POST-запрос; также принимаются все поля; создается карточка поломок.
2. Принимается PUT-запрос с id карточки поломки; также принимаются отдельные поля (не обязательно все); обновляется карточка.

Используются для операций с таблицей Отчета по работе:

```
path('schedule_reports/', SchedReport.as_view()),
path('schedule_report/<int:pk>/', SchedReport_detail.as_view()),
```

Рисунок 9 – шаблоны адресов для запросов к таблице Отчетов по работе

1. Принимается GET-запрос; в таблице поломок выбираются значения до и после текущей даты; сортировка по дню работы и времени окончания; отправляются доступные и недоступные отчеты по работе.
2. Принимается PUT-запрос с id отчетом по работе; принимаются отдельные поля; отчет обновляется.

Используются для операций с таблицей Маршрутов:

```
path('route/', Routes.as_view()),
path('one_route/', Routes_detail.as_view()),
```

Рисунок 10 – шаблоны адресов для запросов к таблице Маршрутов

1. Принимается GET-запрос; в таблице Маршрутов берутся все записи кроме номера маршрута «0» (используется для подставления в таблице Графика работы); отправляются данные.
2. Принимается GET-запрос с id маршрута; отправляется запись.

Используются для операций с таблицей Автобусов и Поломок:

```
path('bus/', Buses.as_view()),
path('buses_status/', BusesStatus.as_view()),
```

Рисунок 11 – шаблоны адресов для запросов к таблице Автобусов/Поломок

1. Принимается GET-запрос; в таблице Автобусов берутся все записи кроме номера автобуса «000000» (используется для подставления в таблице Графика работы); отправляются данные.

2. Принимается GET-запрос; в таблице Автобусов берутся все записи кроме номера «000000»;

   Из таблицы Поломов берутся все активные карточки; сортируются по дате поломки; берутся только id автобусов;

   Из всех автобусов выбираются поломанные и работающие автобусы; возвращаются поломанные и работающие автобусы.

Используются для операций с таблицей Водителей:

```
path('driver/', Drivers.as_view()),
path('driver/<int:pk>/', Drivers_detail.as_view()),
path('all_drivers/', AllDrivers.as_view()),
```

Рисунок 12 – шаблоны адресов для запросов к таблице Водителей

1. Принимается GET-запрос с id водителя; отправляется запись.
2. Принимается DELETE-запрос с id водителя; удаляется запись.
3. Принимается GET-запрос; в таблице Водителей берутся все записи; сортировка по фамилии; отправляются данные.

Используются для операций с таблицей Графика работы:

```
path('schedule/', Schedules.as_view()),
path('schedule/<int:pk>/', Schedules_detail.as_view()),
path('schedule_r/', Schedules_R.as_view()),
```

Рисунок 13 – шаблоны адресов для запросов к таблице Графика работы

1. Принимается GET-запрос; в таблице Графика работы берутся все записи; сортировка по дню работы; отправляются данные.

   Принимается POST-запрос; принимаются все поля; создается запись в таблице Графика работ.

2. Принимается DELETE-запрос с id записи в таблице Графика работы; удаляется запись.

   Принимается PUT-запрос с id записи в таблице Графика работы; принимаются некоторые поля (не обязательно все); запись обновляется.

3. Принимается GET-запрос со id маршрута; в таблице выбираются записи с данным маршрутом; отправляются данные.

3.3.2. Эндпоинты, используемые в интерфейсе запросов

Используется для получения списка водителей, работающих на определенном маршруте, с их графиком работы:

```
path('dr_on_r/', DriverOnRoute.as_view()),
```

Рисунок 14 – шаблон адреса для запроса про водителей на маршрутах

1. Принимается GET-запрос с номером маршрута.
2. Выбираются водители из таблицы Водитель, прикрепленные к данному маршруту.
3. Выбираются записи из таблицы График работы, содержащие выбранных водителей.
4. Отправляются водители и графики работы.

Используется для получения времени начала и конца движения автобусов на каждом маршруте:

```
path('bus_on_r/', BusOnRoute.as_view()),
```

Рисунок 15 – шаблон адреса для запроса про автобусы на маршрутах

1. Принимается GET-запрос.
2. В таблице Графика работы значения группируются по номеру маршрута.
3. Для каждой группы вычисляются минимальное время начала и максимальное время окончания работы.
4. Отправляются сгруппированные данные с временем начала и конца движения.

Используется для получения общей протяженности обслуживаемых автопарком маршрутов:

```
path('routes_time/', RouteTime.as_view()),
```

Рисунок 16 – шаблон адреса для запроса про протяженность маршрутов

1. Принимается GET-запрос.
2. В таблице Маршрутов считается сумма всех полей Протяженности.
3. Отправляется суммарная протяженность.

Используется для получения списка автобусов, не вышедших по расписанию в определенный день, с причиной:

```
path('bad_report/', BusReport.as_view()),
```

Рисунок 17 – шаблон адреса для запроса про пропущенные дни

1. Принимается GET-запрос с датой.
2. Выбираются записи из таблицы Графика работы на данный день.
3. Из таблицы Отчетов по работе выбираются отчеты для выше определенных записей Графика работы; убирается причина «ОК»; сортируются по автобусам.
4. Из таблицы Отчетов по работе выбираются отчеты для выше определенных записей Графика работы; оставляются уникальные значения автобусов.
5. Отправляются автобусы и невыполненные записи Графика работы.

Используется для получения количества водителей каждого класса:

```
path('dr_class/', DriverClass.as_view()),
```

Рисунок 18 – шаблон адреса для запроса про классы водителей

1. Принимается GET-запрос.
2. В таблице Водителей, сортированной по классу, группируются значения по классу; для каждой группы считается число водителей.

Так как в результате пропадают характеристики водителей, то был добавлен следующий шаг.

3. Из таблицы Водителей выбираются записи с классом 1; берется номер паспорта, имя, фамилия; то же со 2 и 3 классами.

И только теперь отправление данных.

4. Отправляются сгруппированные данные с количеством водителей каждого класса, характеристики водителей 1 класса, 2 класса, 3 класса.

3.3.3. Эндпоинты, используемые в интерфейсе отчета

Используется для получения отчета по автопарку:

```
path('report_buses/', Report_buses.as_view()),
```

Рисунок 19 – шаблон адреса для запроса на получение отчета

1. Из таблицы Автобусов берутся все записи, кроме автобуса «000000» (используется для подставления значений в модели Графика работы); группируются значения по типу автобуса; для каждой группы считается количество автобусов; сортируются по типам автобусов.
2. Из таблицы Водителей берутся все записи; добавляется поле Возраст (текущая дата минус дата рождения); добавляется поле Опыт работы в днях (текущая дата минус дата начала работы); берутся отдельные поля.

3. В таблицу Водителей так же добавляются поля Возраст и Опыт работы в днях; берутся поля Возраст, Опыт работы в днях; считаются средние значения для этих двух полей.
4. Из таблицы Водителей берутся значения с малым типом автобуса; оставляются только уникальные значения Маршрутов; то же для остальных типов автобуса.
5. Отправляются средние значения Возраста и Опыта работы всех водителей; значения Возраста и Опыта работы с характеристиками водителей; маршруты для малого типа автобуса; то же с остальными типами; типы автобусов в автопарке.

3.4. Клиентская часть сервиса

3.4.1. До входа пользователя

Учитывая тот факт, что разрабатываемая система предназначена только для диспетчера автопарка и что случайно зашедшим людям не должны быть представлены возможности для работы с базой данных, и вообще с информацией часто конфиденциальной, при входе на сайт (компонент Home) неавторизованного пользователя встречает элемент с просьбой диспетчера войти под своим логином.

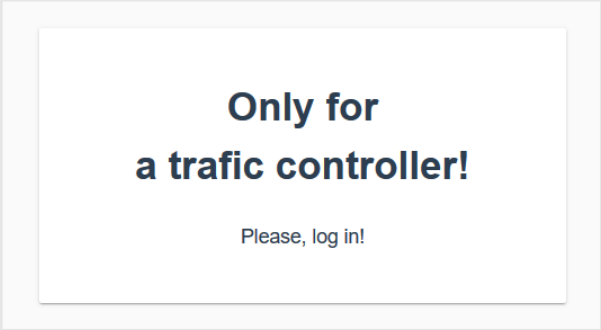


Рисунок 20 – элемент страницы для неавторизованного пользователя

Помимо этого, видна шапка сайта:

– Слева расположена кнопка, открывающая боковое меню, которое недоступно, пока пользователь не зайдет в свою учетную запись. Пример работающего меню на рис. 23

– Посередине – название компании с ссылкой на домашнюю страницу.

– Справа находится кнопка авторизации.

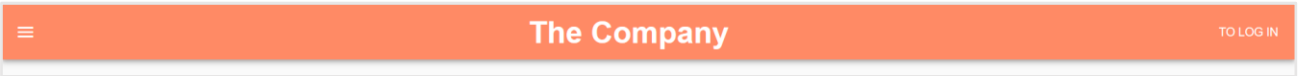


Рисунок 21 – Шапка сайта

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

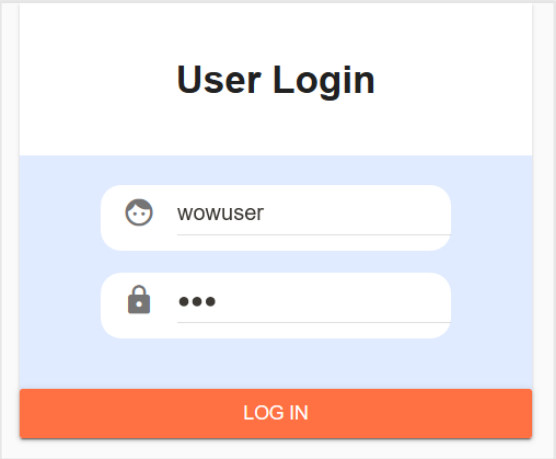


Рисунок 22 – Форма входа пользователя

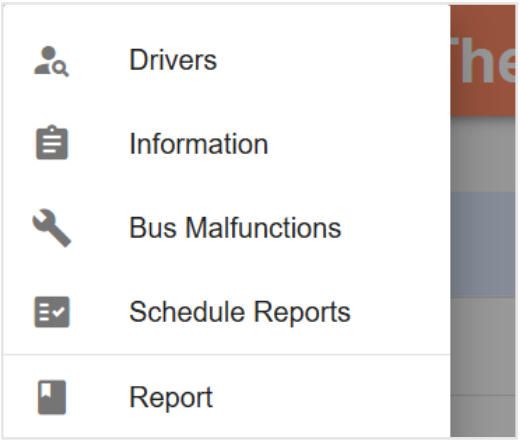


Рисунок 23 – Элементы в боковом меню после авторизации пользователя

3.4.2. Для добавления/удаления основных данных

Диспетчер следит за расписанием, добавляя или удаляя необходимые записи, и за штатом водителей, также расширяя базу данных или уменьшая общее количество работников.

Schedule

Driver	Bus	Route	Day	Starts at	Ends at
KITLY (9364991883)	O222KA	49	2020-07-15	06:00:00	08:30:00
NILSON (3614661728)	B028EK	337	2020-07-14	06:00:00	08:30:00
MELL (3242342342)	M765OP	337	2020-07-11	05:40:00	08:50:00
JON (4636356356)	A029EC	337	2020-07-11	20:40:00	22:50:00
YOON (1294791824)	X200AO	49	2020-07-08	16:30:00	18:20:00
NILSON (3614661728)	M765OP	337	2020-06-22	18:00:00	02:00:00
JOHNSON (0813157990)	M765OP	49	2020-06-22	16:20:00	20:10:00
JON (4636356356)	A029EC	337	2020-01-04	05:00:00	13:00:00

DELETE ADD

Рисунок 24 – Часть домашней страницы сайта

В блоке расписания слева находится список водителей в виде кнопок, при нажатии на которые можно просмотреть краткую информацию о сотруднике (рис. 25). Если необходимо удалить запись, то это можно сделать, нажав на копку «DELETE». При добавлении записи (кнопка «ADD») диспетчер заполняет следующие поля (рис. 26):

1. Водитель, который будет выполнять работу в данный день.
2. Автобус и маршрут. По умолчанию подставляются закрепленные за выбранным водителем номера автобуса и маршрута.
3. День работы, время начала и окончания.

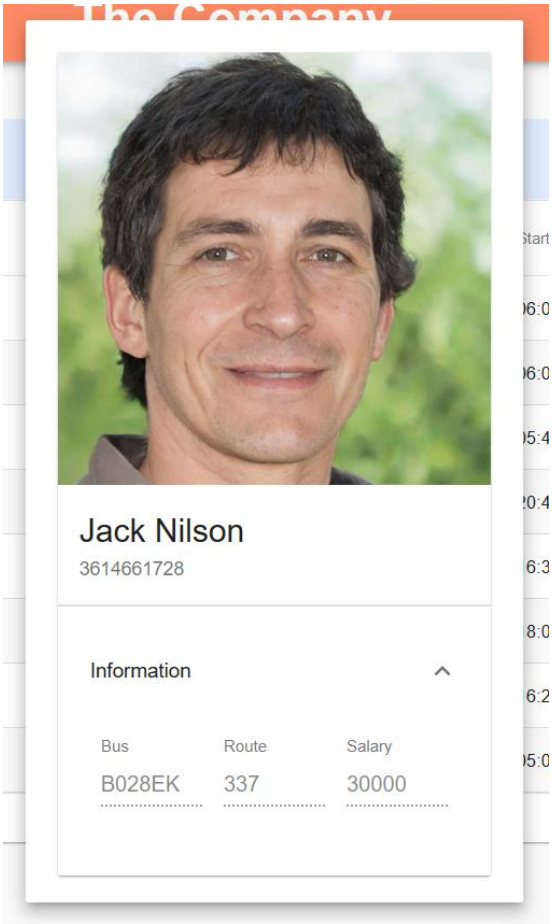


Рисунок 25 – Карточка с краткими данными водителя

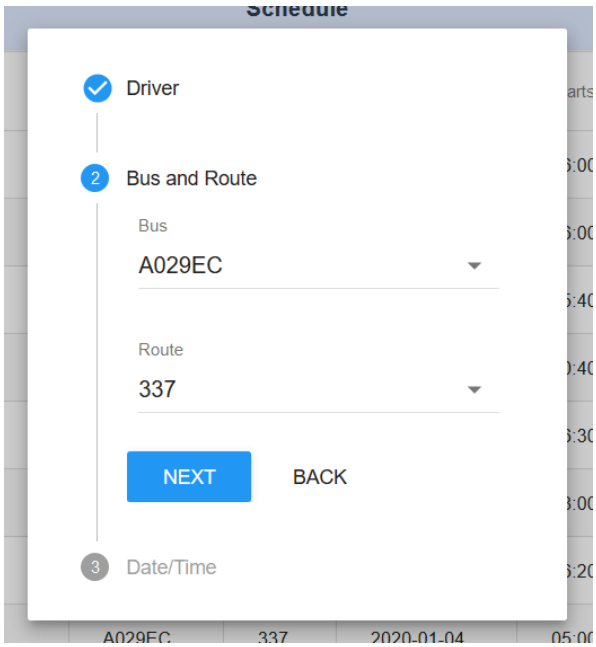


Рисунок 26 – Форма добавления записи в график работы

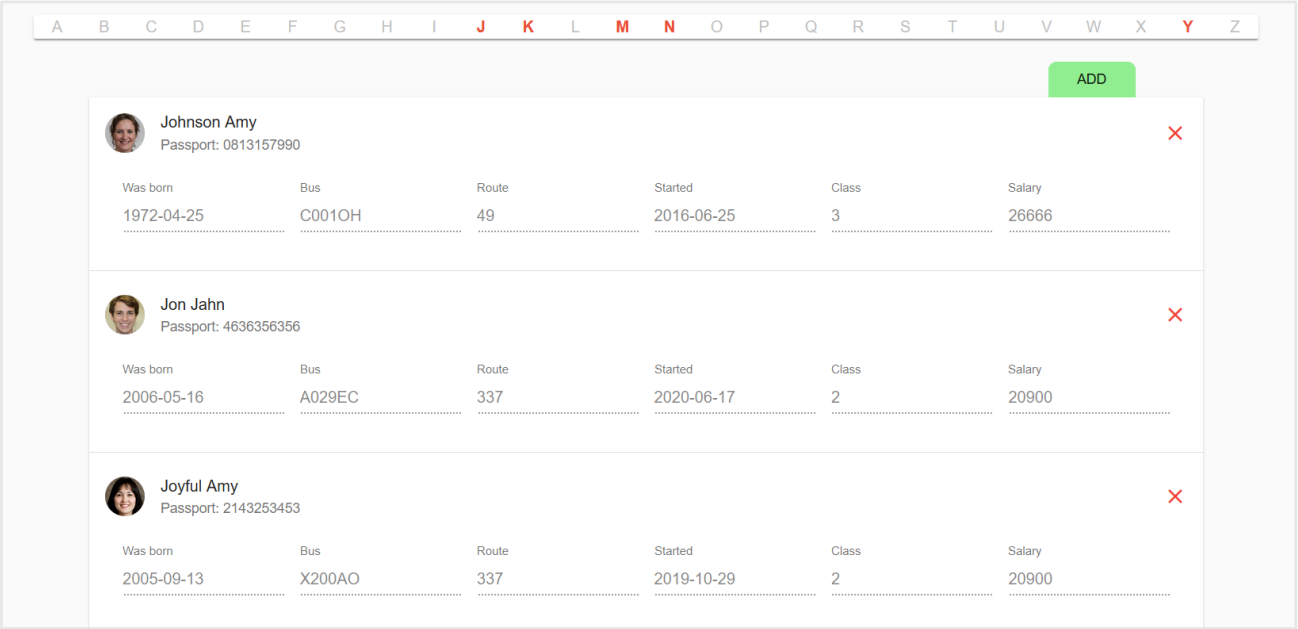


Рисунок 27 – Часть страницы сайта по адресу …/drivers

Основная часть экрана заполнена каточками с данными водителей (компонент Drivers). Сверху находится строка с алфавитом, с помощью которой осуществляется навигация по списку работников.

Диспетчер удаляет водителя, нажав на красную иконку справа определенной записи, а добавляет нажатием на зеленую кнопку справа сверху.

Сначала нажимается кнопка «ПОКАЗАТЬ», чтобы просмотреть фотографию следующего кандидата на добавление в базу данных, затем вводятся данные водителя: фамилия, имя, серия и номер паспорта, дата рождения, закрепленные за сотрудником автобус и маршрут, дата начала работы, класс водителя.

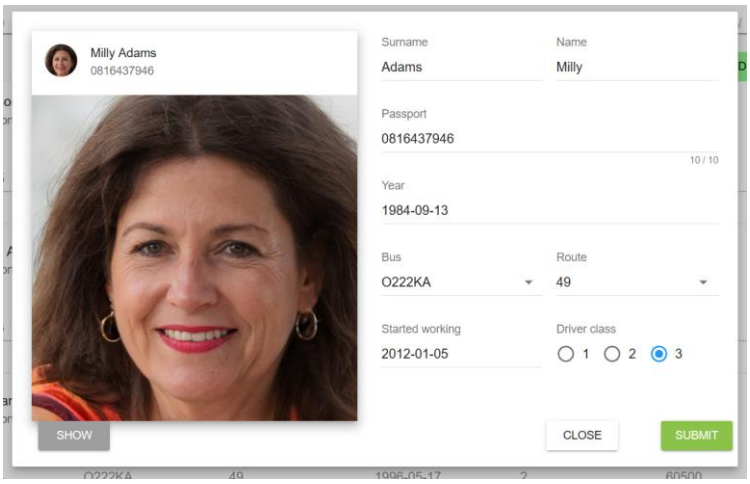


Рисунок 28 – форма добавления водителя

3.4.3. Для контроля

Диспетчер осуществляет контроль за результатами технического обслуживания парка подвижного состава и за выполнением работы персоналом.

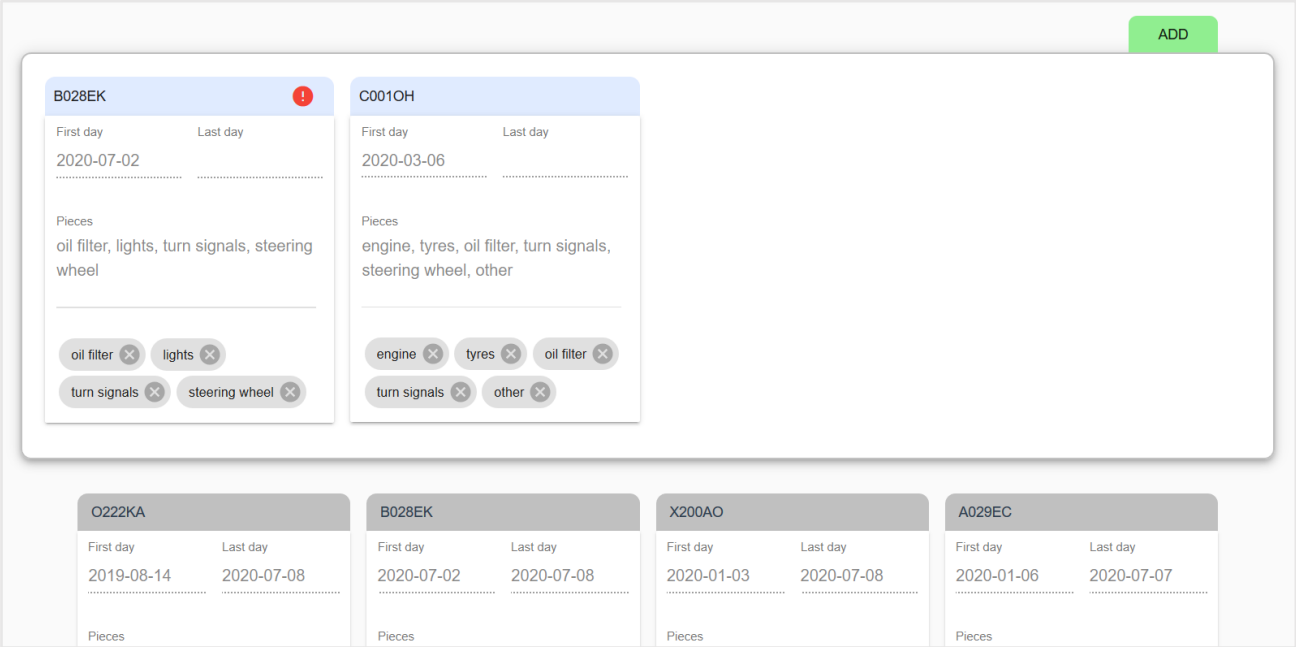


Рисунок 29 – Часть страницы сайта по адресу …/malfunctions

В верхней части экрана находятся активные карточки поломок, тогда как внизу – уже починенных автобусов с заполненной датой окончательного ремонта (компонент BusMals).

В активной карточке диспетчер по результатам технического обслуживания постепенно убирает поломанные детали до того момента, пока автобус не будет считаться починенным. Если в процессе ремонта окажется, что на этот автобус назначена запись в графике работы, то справа появится красная иконка, при нажатии на которую можно будет либо изменить запись расписания (пересадить водителя на другой автобус или изменить день), либо удалить.

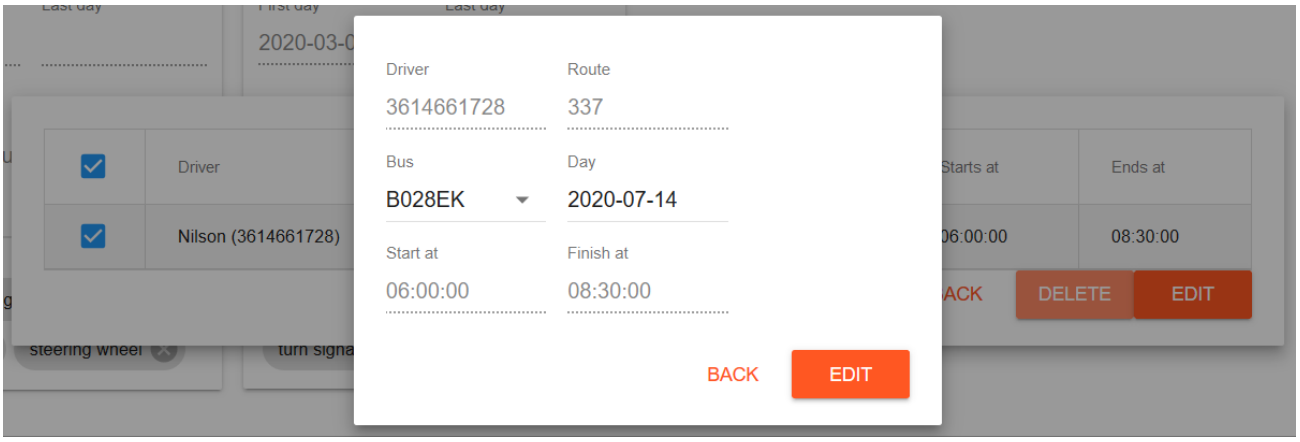


Рисунок 30 – Форма выбора действия с записью графика работы и форма изменения записи

В добавление к вышесказанному диспетчер также может самостоятельно оформить новую карточку поломки автобуса, выбрав транспорт (на который нет активной карточки), дату поломки и указав сломанные детали.

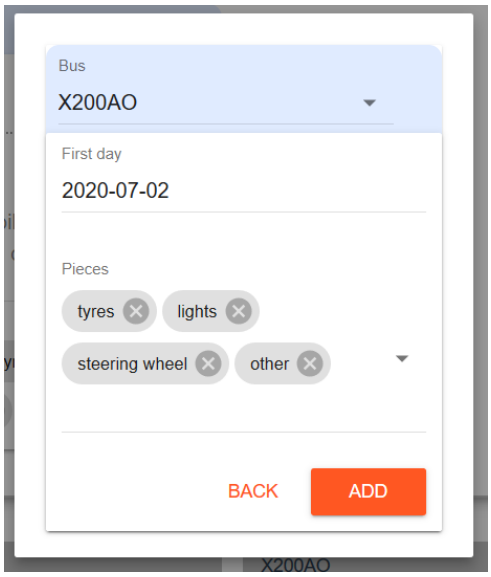
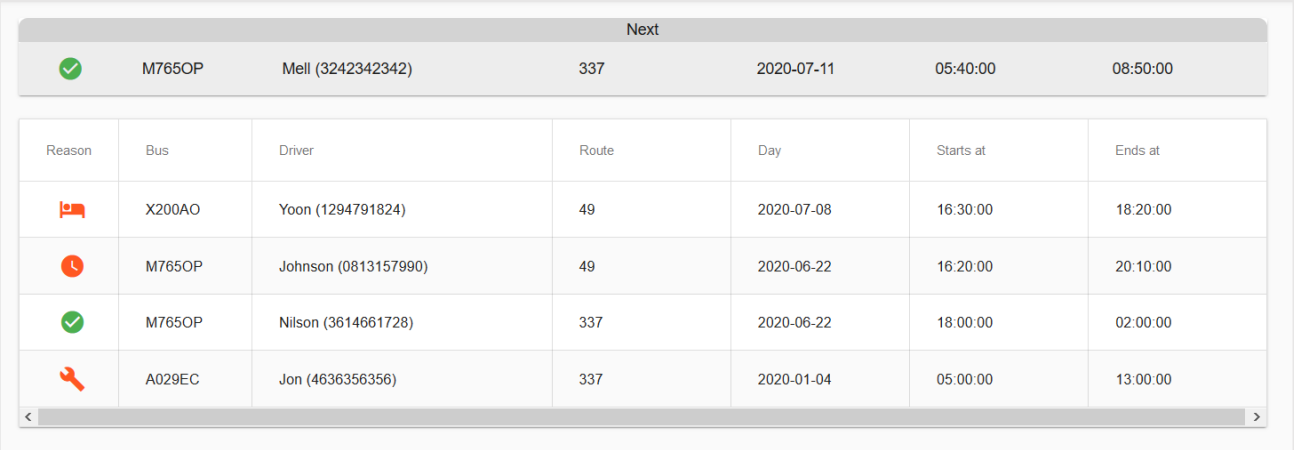


Рисунок 31 – Форма добавления карточки поломки



Next

	M765OP	Mell (3242342342)	337	2020-07-11	05:40:00	08:50:00

Reason	Bus	Driver	Route	Day	Starts at	Ends at
	X200AO	Yoon (1294791824)	49	2020-07-08	16:30:00	18:20:00
	M765OP	Johnson (0813157990)	49	2020-06-22	16:20:00	20:10:00
	M765OP	Nilson (3614661728)	337	2020-06-22	18:00:00	02:00:00
	A029EC	Jon (4636356356)	337	2020-01-04	05:00:00	13:00:00

Рисунок 32 – Часть страницы сайта по адресу …/sched_reports

Основная часть – записи графика работы до текущего дня включительно. В верхней половине находится информация о следующей записи, изменения для которой недоступны до момента наступления указанного дня работы (компонент SchedReport).

Диспетчер может выбрать любую активную запись и изменить причину, если оказалось, что работа началась позже или вообще не была выполнена.

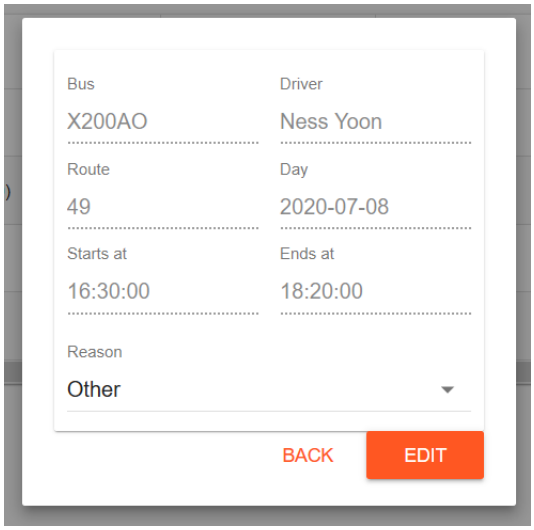


Рисунок 33 – Форма для изменения причины в отчете по работе

3.4.4. Для получения данных

Цель их использования диспетчером – получить информацию по заранее определенным запросам или общие сведения по состоянию автопарка.

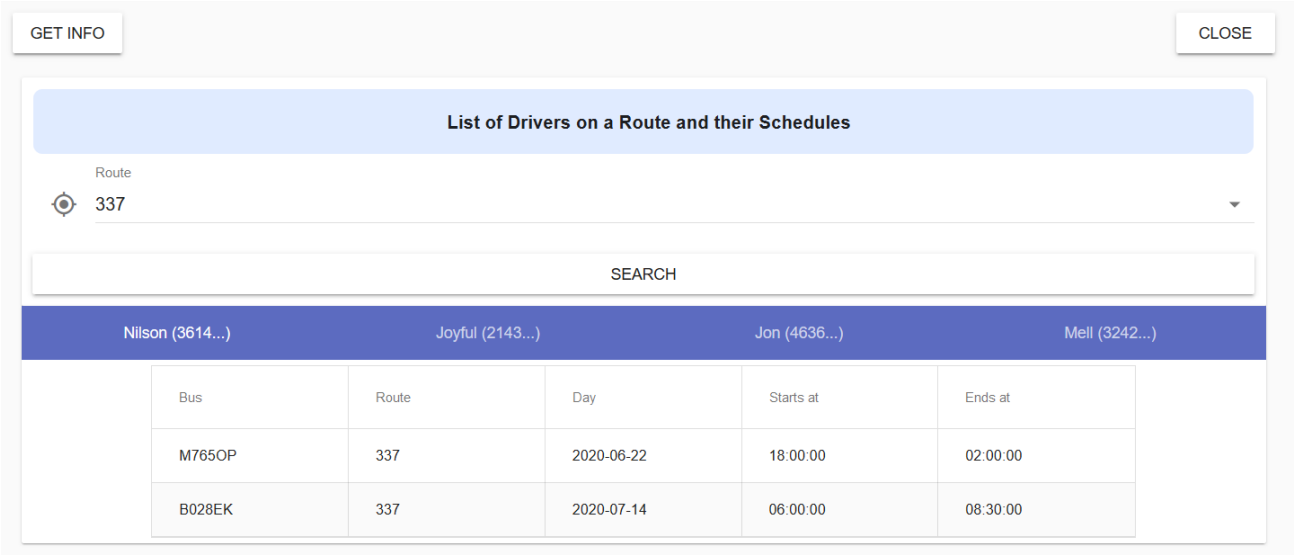


Рисунок 34 – Часть страницы сайта по адресу …/queries

В верхней половине находится блок с запросом данных (если необходимо) для дальнейшего осуществления запроса, в нижней – результаты. Справа сверху находится выпадающее меню (рис. 35), с помощью которого можно выбрать один из пяти запросов (компонент Queries):

1. Список водителей, прикрепленных к определенному маршруту, и график их работы.
2. Время начала и окончания движения автобусов на всех маршрутах.
3. Общая протяженность маршрутов.

4. Автобусы, не вышедшие не линию в заданный день, с причиной.
5. Количество водителей каждого класса.

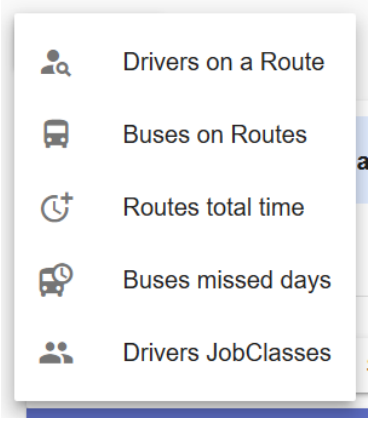


Рисунок 35 – Выпадающее меню с запросами

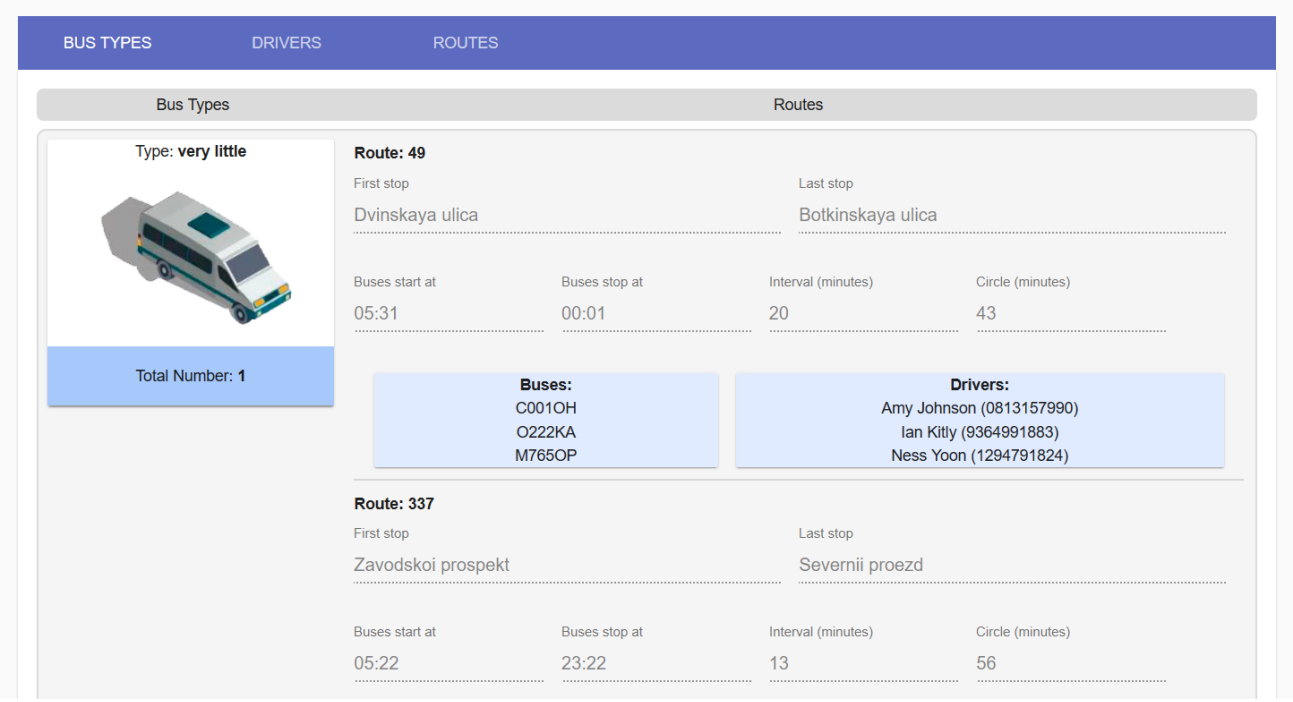


Рисунок 36 – Часть страницы сайта по адресу …/report

Большую часть экрана занимает поле с основной информацией. Сверху находятся вкладки частей отчета, с помощью которых осуществляется переход (компонент Report):

1. Типы автобусов и их количество в автопарке; обслуживаемые ими маршруты с характеристиками и списками водителей и автобусов, закрепленных за данными маршрутами.
2. Средний возраст и стаж работы водителей.
3. Общая протяженность всех маршрутов.

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 обзор предметной области данной 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 результатов вышеописанной задачи 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 полной мере. Для осуществления более эффективного процесса работы диспетчера в будущем можно будет 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 покрыть основные потребности диспетчера автобусного парка.

СПИСОК ЛИТЕРАТУРЫ

1. Автобусные парки // Московский автобус [Электронный ресурс] URL: http://bus.ruz.net/history/article_02/ (дата обращения: 08.07.2020)
2. Date and Time Formats // World Wide Web Consortium [Электронный ресурс] URl: https://www.w3.org/TR/NOTE-datetime (дата обращения: 09.07.2020)
3. Home // Django REST framework [Электронный ресурс] URL: https://www.django-rest-framework.org/ (дата обращения: 07.07.2020)
4. Introduction // Vue.js [Электронный ресурс] URL: https://vuejs.org/v2/guide/ (дата обращения: 07.07.2020)
5. Django documentation // The Web framework for perfectionists with deadlines [Электронный ресурс] URL: https://docs.djangoproject.com/en/3.0/ (дата обращения: 07.07.2020)
6. Muse UI [Электронный ресурс] URL: https://muse-ui.org/#/en-US (дата обращения: 07.07.2020)
7. PostgreSQL: Documentation // PostgreSQL: The world’s most advanced open source database [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения: 07.07.2020)