МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка веб-приложения для учета обращений клиентов автотранспортного предприятия»

Обучающийся Ш. Р. Алиуллов
(Инициалы Фамилия) (личная подпись)

Руководитель В.Ф. Глазова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема выпускной квалификационной работы (ВКР) – разработка веб-приложения для учета обращений клиентов автотранспортного предприятия (АТП). Один из способов выявления качества предоставляемых услуг – мониторинг отзывов клиентов [2]. В региональных АТП нет единой системы сбора отзывов или даже простого сайта, который позволил бы мониторить качество предоставляемых услуг. Этим объясняется актуальность темы и разработки приложения, которое бы позволило пассажиру за 1 минуту оставить отзыв или пожаловаться на ненадлежащее оказание услуг. Цель ВКР – разработка веб-приложения для учета обращений клиентов автотранспортного предприятия. Достижение поставленной цели предполагает постановку и решение следующих задач:

- рассмотреть предметную область и ее бизнес-процессы;
- сделать концептуальную модель предметной области;
- определить требования к веб-приложению;
- спроектировать программный продукт;
- разработать приложение;
- произвести тестовый запуск приложения на базе практики.

Практическая значимость, предмет и объект выпускной квалификационной работы указаны во введении. Основное содержание работы делится на 3 главы: в первой приводятся результаты анализа предметной области, описание основных бизнес-процессов, выполняется концептуальное моделирование процесса обработки жалоб и предложений. Вторая глава посвящена моделированию приложения, описывает концепцию, желаемый результат. В третьей главе описана реализация программного продукта, этапы создания и внедрения приложения. В приложениях к ВКР размещено техническое задание, скриншоты и листинги приложения. Объем ВКР – 70 страниц с приложениями. Материал состоит из 6 основных разделов: введения, 3 глав, заключения, списка источников информации.

Оглавление

Введение

Пассажирское автотранспортное предприятие (АТП, автобусный парк, автоколонна) владеет автобусным парком и контролирует движение пассажирских экипажей в черте города и между муниципальными образованиями. АТП работает на основании 40-й главы второй части Гражданского кодекса Российской Федерации «Перевозка» и руководствуется Уставом городского автотранспорта [15]. Задача АТП не отличается от задачи любого предприятия – получение прибыли [5]. В рамках своей деятельности для получения прибыли автоколонне нужно выполнять следующие функции:

- вовремя, безопасно и комфортно перевозить пассажиров по установленным маршрутам;
- регулярно обслуживать автопарк, чтобы повысить срок его службы и безопасность перевозок;
- обеспечивать рабочие места в соответствии с трудовым кодексом.

В итоге качественное решение поставленных перед организацией задач наделяют АТП важным социальным значением: маршрутный транспорт каждый день доставляет на работу, в учебные заведения и больницы тысячи людей. С учетом вышесказанного можно выделить объект и предмет ВКР.

Объект – процесс учета обращений клиентов автотранспортного предприятия, предмет – автоматизация процесса учета обращений клиентов автотранспортного предприятия.

Целью является разработка веб-приложения для учета обращений клиентов автотранспортного предприятия, с помощью которого пассажиры смогут оставлять пожелания и предложения руководству АТП.

Во время написания выпускной квалификационной работы были применены следующие методы:

- сбор данных: опрос, наблюдение, анализ, синтез, изучение литературы;

– структурное проектирование;

– объектно-ориентированный анализ и проектирование;

– реинжиниринг.

Практическая значимость работы выражается в возможности дальнейшего применения результатов ВКР для повышения качества предоставляемых услуг, увеличения прибыли и конкурентоспособности АТП.

Основное содержание работы разбито на 3 главы: в первой уточняется предметная область, описывается важность автоматизации учета обратной связи, анализируются существующие системы автоматизации сбора отзывов клиентов, моделируются бизнес-процессы, дается характеристика организации, на примере которой будет производиться тестовое внедрения программного продукта (описывается персонал, материально-техническая база, процесс работы).

Во второй главе моделируется веб-приложение (создаются диаграммы классов и вариантов использования), дается характеристика потокам информации организации, предъявляются требования к системе, описывается концепция приложения и подбираются подходящие технические решения.

Третья глава описывает реализацию приложения с помощью выбранных средств разработки, внедрение приложения на базу практики, результаты внедрения и их анализ с точки зрения повышения эффективности работы организации. Также в третьей главе размещена ссылка на работающее приложение, даны скриншоты интерфейса и процесса разработки.

Глава 1 Концептуальное моделирование системы обратной связи в пассажирских грузоперевозках

1.1 Характеристика предметной области

Для повышения наглядности процесса разработки веб-приложения в качестве целевой организации будет использоваться база преддипломной практики – ООО «Новомосковский Автоплюс». Перед началом разработки необходимо дать характеристику организации. Так удастся эффективнее спроектировать программное обеспечение (ПО) и нагляднее отобразить бизнес-процессы, которые приложение сможет заменить или улучшить.

ООО «Новомосковский Автоплюс» – это АТП, целью которого является получение прибыли путем перевозки пассажиров между муниципальными образованиями Тульской области: Северо-Задонск, Новомосковск, Сокольники, Иваньково, Хмелевка, Придонье, Спасское, Мирный, Шахты.

Назначение компании – ежедневное перемещение населения вышеуказанных населенных пунктов на места работы/учебы/досуга. Для Спасского, Сокольников, Шахт, Мирного ООО «Новомосковский Автоплюс» является практически единственных перевозчиком (захватывает до 90% пассажиропотока), поэтому работа организации имеет важно социальное и деловое значение для Новомосковского района.

«Новомосковский Автоплюс» учрежден Лазаревым Игорем Витальевичевым в форме ООО 06.07.2006 с уставным капиталом в 10000 рублей. С тех пор компания выросла с владельца нескольких «Газелей» до автоколонны с 18 автобусами марки ГАЗ-A64R42 (Некст-версия «Газели»). Ранее компания перевозила пассажиров на автобусах Ikarus-260.00, Ikarus-280.00 70-80-х годов и «Газелях» 3221 1-го поколения. Выручка компании составляет 28 млн р./год.

Основные конкуренты ООО «Новомосковский Автоплюс» – «Автоколонна 1411» и «Тульская ТК». Компании тоже работают в пределах Новомосковского района и обслуживают городские маршруты.

В штате компании служат 10 человек административного состава и 18 водителей.

Полный перечень сотрудников:

- генеральный директор и его заместитель;
- контроллер выхода водителей на рейс;
- системный администратор;
- инспектор отдела кадров;
- 18 водителей;
- медсестра;
- диспетчер;
- бухгалтер;
- кассир;
- специалист по обслуживания автотранспортных средств.

Схематичное отображение взаимодействия сотрудников показаны на рисунке 1 (линейно-функциональная организационная структура).

Рисунок 1 – Организационная структура (линейно-функциональная) ООО «Новомосковский Автоплюс»

На рисунке 1 видно, что линейные элементы выполняют управленческие функции. Водители подчиняются диспетчеру, кассир – бухгалтеру, бухгалтер и диспетчер заместителю, а он – генеральному директору. Для того чтобы глубже рассмотреть взаимодействие сотрудников в контексте выполнения целей функционирования организации, необходимо описать бизнес процессы (БП) ООО «Новомосковский Автоплюс» и составить концептуальную модель предприятия. Но перед этим следует рассмотреть аппаратное и программное обеспечение компании, смысл учета обратной связи от клиентов и текущие способы ее получения на предприятии.

1.1.1 Аппаратное и программное обеспечение организации

Весь административный состав организации находится в одном офисе, который укомплектован 4 персональными компьютерами для системного администратора, заместителя директора, генерального директора и бухгалтера. Характеристики компьютеров:

- ПО для бухучета «Своя технология» (только компьютер бухгалтера);
- оперативная память Kingston KVR 800S4N4 4 Гб;
- жесткий диск Seagate S300HD3200 300 Гб;
- процессор AMD Phenom X3 975 3.4 Ггц;
- видеокарта AMD Radeon 4677 1 Гб;
- операционная система Windows 7;
- пакет Microsoft Office 2007;
- браузер Google Chrome;
- монитор HP 17A.

Также в офисе развернута Wi-Fi сеть от местного провайдера «Ростелеком». В качестве роутера используется Asus AC55U совместно с оптоволоконной сетью и терминалом GPON RT-GM-1.

1.2. Важность автоматизации учета обратной связи

Если клиент не доволен, то он рано или поздно уходит к конкуренту. Смысл обратной связи – исправить ошибки сервиса, которые делают клиента недовольным, а значит, склонным отказаться от услуг компании [17].

В Новомосковском районе только автобусы ООО «Новомосковский Автоплюс» ходят с интервалом в 20 минут, транспорт конкурентов курсирует по маршрутам в 4 раза реже. Если до 7 часов утра потенциальный клиент может выбрать перевозчика, то после – нет.

Поэтому учету обратной связи в компании уделяется мало внимания. Пассажир может только позвонить контроллеру автоколонны, если найдет номер телефона в салоне автобуса на информационном стенде.

Несмотря на то, что клиенты по большей части не могут уйти к конкурентам, уделять внимание обратной связи необходимо по следующим причинам:

- может появиться конкурент с такой же частотой выездов;
- жалобы помогают выявлять недобросовестных водителей, которые не берегут технику и в дальнейшем могут стать источником серьезных проблем (в случае ДТП, драк), если не санкционировать неподобающее поведение водителей [8];
- клиенты используют соцсети как публичную жалобную книгу.

Если раньше можно было забыть об учете мнения клиента без влияния для бизнеса, то сейчас любой просчет может стать достоянием общественности и испортить репутацию компании. Репутация для ООО «Новомосковский Автоплюс» – это конкурентоспособность, гранты и субсидии от администрации [16]. Если не «разрядить» гнев клиента с помощью инструмента быстрой отправки жалобы, то он обратится за поддержкой в соцсети.

С точки зрения общества, жалобы в соцсетях – это позитивное явление. Но также с точки зрения общества и бизнеса было бы лучше, если бы

водителя-нарушителя руководство заметило ранее через многочисленные жалобы на менее серьезные проступки, чем, например, на открытый конфликт водителя и пассажира. Если бы руководство АТП смогло получить своевременную обратную связь, то штрафы или даже увольнения помогли бы избежать инцидентов: нивелировать репутационные издержки, сберечь пассажиров от морального ущерба, избавиться от необходимости привлечения администрации города, прокуратуры или полиции [10].

Если водитель замечен за серьезным нарушением (брань, проезд на красный сигнал светофора, драка), то, очевидно, ранее он совершал нарушения более мелкого масштаба. Но количество жалоб на водителей всегда было на уровне 1-2 в 1-2 недели. Это значит, что доступ к отправке обратной связи затруднен. В ином случае инциденты можно было пресечь еще на уровне зарождения «безнаказанности» сотрудника, когда он допускал менее значимые проступки: грубость, нарушение дорожной разметки и иные негативные допущения.

1.3 Моделирование бизнес-процессов ООО «Новомосковский Автоплюс» «Как есть»

Бизнес-процессы (БП) АТП можно рассмотреть как совокупность действий персонала, которая приводит к созданию главной ценности (процесса) предприятия – транспортировке пассажиров [4]. Моделировать БП можно различными инструментами. Они объединены в нескольких методологиях: SADT, EPC, BPMN. Именно SADT (методология структурного анализа и проектирования) посредством нотации IDEF0 позволяет отобразить модель максимально детализировано и наглядно благодаря гибкой модели описания как бизнес-процессов, так и общей структуры действий [3].

В контексте разработки программного продукта моделирование БП поможет обнаружить сотрудников, функции которых возьмет на себя ПО.

Или наоборот, моделирование может показать, какие сотрудники понадобятся.

Также моделирование покажет нуждающиеся в оптимизации БП – это может вылиться в новые требования в техническом задании (ТЗ).

Начать моделирование легче всего с выявления основного полезного действия АТП и функций, которые ответственны за его выполнение. Для этого нужно рассмотреть работу организации в формате контекстной диаграммы в нотации IDEF0 в виде «Как есть» (рисунок 2) [20].

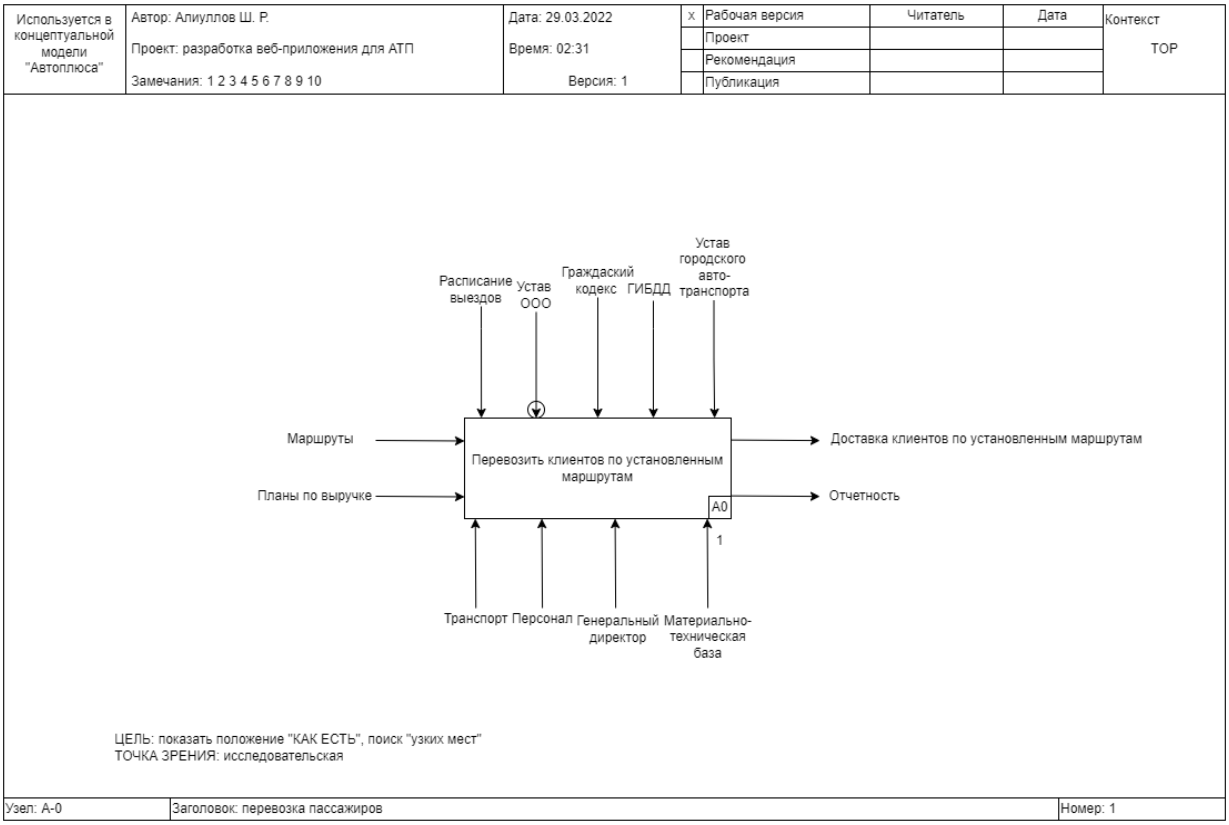

Рисунок 2 – Контекстная диаграмма основного бизнес-процесса для ООО «Новомосковский Автоплюс» («Как есть»)

На рисунке 2 видна основная задача АТП – перевозить клиентов по установленным маршрутам, а также на диаграмме отображены механизмы управления, входы, выходы процесса и механизмы исполнения.

К входам относятся маршруты и планы по выручке. Управляют процессом, прежде всего, законы и службы: ГИБДД, устав городского автотранспорта, Гражданский кодекс России. В рамках организации процессом управляют Устав ООО и элемент «Расписание выездов».

В качестве механизмов выступает транспорт организации в виде 18 автобусов ГАЗ, персонал в количестве 28 человек, генеральный директор и материально-техническая база автоколонны.

Стрелка управления процессом уставом ООО на диаграмме отмечена тоннелем (О), потому что его отображение на следующей диаграмме при декомпозиции нецелесообразно. Выходные стрелки обозначены отчетностью и полезным действием АТП – доставкой клиентов по установленным маршрутам. Чтобы лучше понять работу организации, необходимо декомпозировать процесс А-0 на наиболее значимые элементы в виде IDEF0-диаграммы на рисунке 3.

На рисунке 3 видно, что контекстную диаграмму удалось декомпозировать на 6 функций со своими входами, выходами, управлением и механизмами:

- осуществить медосмотр водителя;
- осуществить техосмотр автомобиля;
- сформировать путевой лист;
- выехать в рейс;
- доставить пассажиров по маршруту;
- сдать машину и выручку.

К управляющим механизмам добавились (в сравнении с контекстной диаграммой) Приказ Минздрава №845н, Минтранса №9, а управляющий механизм Гражданского кодекса разбит на 40-ю главу второй части кодекса и №259-ФЗ [11][12][13]. Они отмечены тоннелем на диаграмме, т. к. на контекстной диаграмме (рис. 2) их не было. По правилам стандартизации нотации IDEF0 тоннели пришлось добавить к «Диспетчеру» и «Кассиру»,

которые выступают механизмами исполнения (их отображение на контекстной диаграмме было нецелесообразно). Также тоннели были добавлены к стрелкам «Прибыль» и «Сменное задание». Механизм выполнения (сотрудники) также отмечены тоннелем, т. к. на контекстной диаграмме они были включены в один поток «Сотрудники» (за исключением генерального директора).

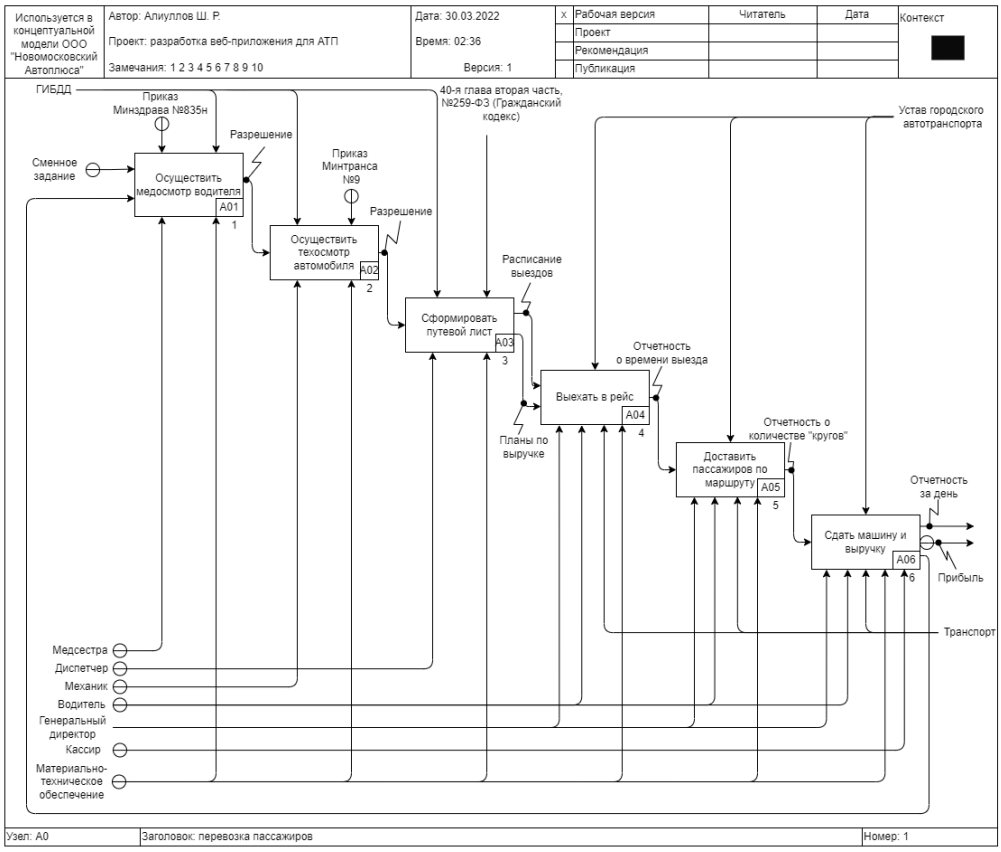

Рисунок 3 – Декомпозиция IDEF0-диаграммы основного бизнес-процесса для ООО «Новомосковский Автоплюс» («Как есть»)

Описать показанные на диаграмме функции в контексте рабочего дня можно описать следующим образом.

Шаг 1. Медсестра на основе сменного задания проверяет здоровье и состояния водителя на соответствие приказу Минздрава №835н.

Шаг 2. Специалист по обслуживанию транспорта (механик) проверяет техническое состояние автомобиля на соответствие приказу Минтранса №9.

Шаг 3. Диспетчер формирует путевой лист в соответствии с №259-ФЗ (ГК РФ).

Шаг 4. Водитель выезжает в рейс.

Шаг 5. Водитель осуществляет главное полезное действие АТП ООО «Новомосковский Автоплюс» – доставляет пассажиров по установленным маршрутам.

Шаг 6. Водитель проходит проверку у медсестры, сдает автомобиль механику, выручку – кассиру.

В контексте разработки веб-приложения наибольшую ценность представляют сведения об обработке жалоб клиентов. Для водителя процесс отправки жалобы на него незаметен даже во время рейса. Поэтому отзыв не влияет на его действия, и, соответственно, на соответствующий БП (даже если учесть дисциплинарные взыскания, потому что они наступают после проверки жалоб, а это 3-7 дней).

В результате вышесказанного на диаграмме декомпозиции контекстной диаграммы невозможно отобразить процесс обратной связи «Как есть» с точки зрения водителя.

Поэтому для описания этого процесса необходимо составить новую диаграмму, которая будет связана с блоком A05 «Доставить пассажиров по маршруту», т. к. именно в момент работы водителя контроллер может получить жалобу на некачественные услуги. Процесс с точки зрения контролера (он обрабатывает жалобы) показан на рисунке 4.

На рисунке 4 видно, что принятие жалобы от пассажиров можно разбить на 6 функций, которые выполняются контролером: принять звонок от пассажира с рейса с водителем, разобраться в предмете жалобы, внести жалобу в журнал, доложить заместителю директора, выбрать меру дисциплинарной ответственности, применить дисциплинарное взыскание.

Заместитель директора в дальнейшем должен выбрать меру дисциплинарной ответственности для водителя: выговор, строгий выговор, увольнение [7].

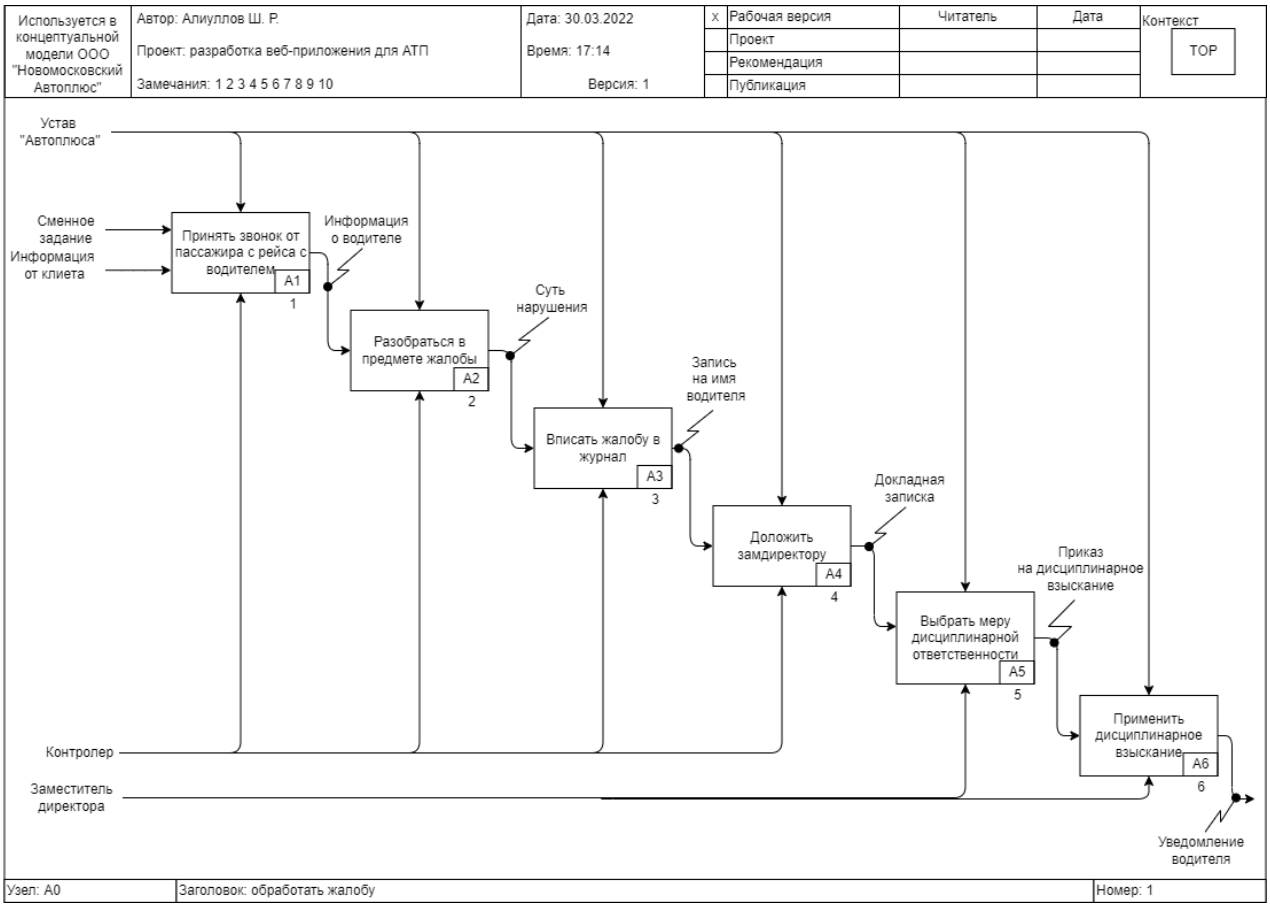

Рисунок. 4 – IDEF0-диаграмма обработки жалоб («Как есть»)

На основе модели «Как есть», можно построить модель «Как должно быть». Для этого нужно выяснить, какие процессы нуждаются в оптимизации.

1.4 Варианты оптимизации процесса получения обратной связи

Основная сложность в процессе обработки жалоб – наличие бумажного журнала: контролер записывает жалобу в журнал, а заместитель каждый день должен проверять его на наличие обновлений. Заместитель директора 90% времени в разъездах, поэтому он не всегда успевает проверить журнал. И из-

за того, что оставить отзыв для пассажира довольно сложно, – нужно найти номер и позвонить, журнал заполняется 1-2 раза в неделю. Поэтому заместитель его практически не проверяет и даже те инциденты, которые дошли до журнала, остаются незамеченными.

Также негативным фактором является то, что контроллер участвует в принятии жалобы. В результате можно выделить главные требования к веб-приложению: оно должно изъять из процесса подачи жалобы бумажный журнал и облегчить работу контролеру, убрав его из цепочки отзыв-заместитель-водитель. На основе этих данных можно построить диаграмму «Как должно быть» с учетом веб-приложения.

1.5 Модель «Как должно быть»

Диаграмма обработки жалоб с включением ПО по получению обратной связи будет выглядеть так, как показано на рисунке 5. Мы видим, что приложение облегчает прием жалоб контролеру и заместителю директора: первого оно исключает, а второму предоставляет простой интерфейс просмотра жалоб, к которому можно получить доступ из любого места через интернет без бумажного журнала. В итоге количество действующих лиц сокращается до заместителя, управляющий механизм остается один – Устав ООО «Новомосковский Автоплюс», а количество функций сокращается с 6 до 3:

- проверить поступление жалоб в приложении;
- выбрать меру дисциплинарной ответственности водителя;
- применить дисциплинарное взыскание.

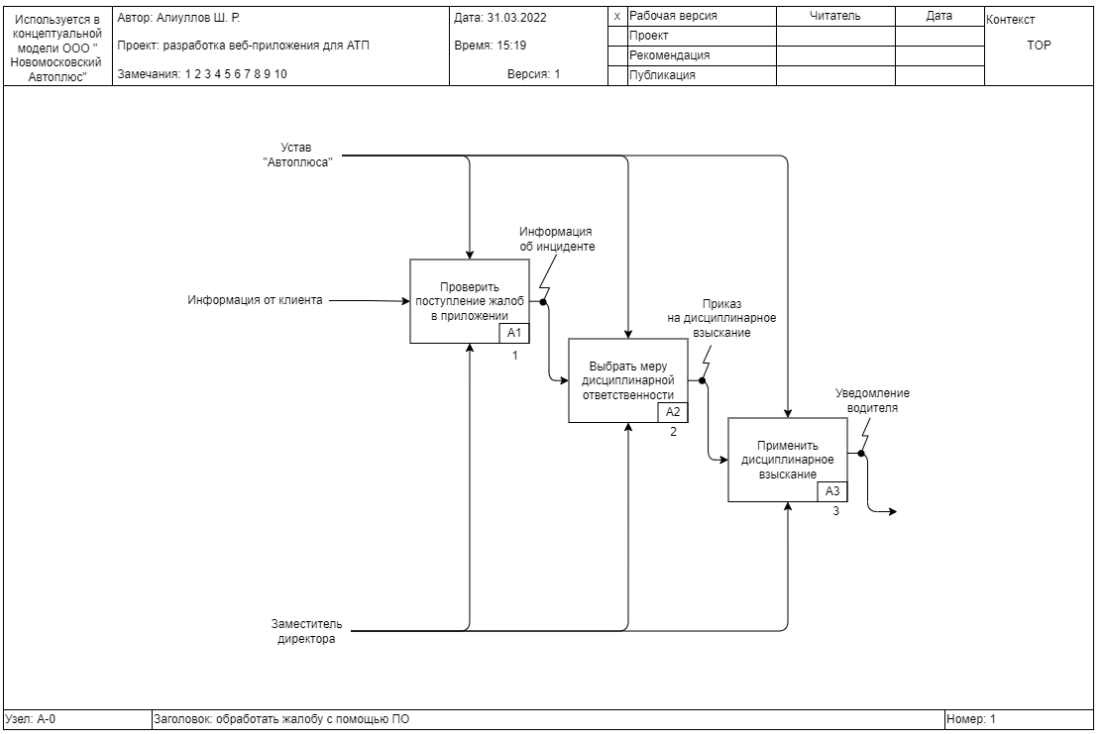

Рисунок 5 – IDEF0-диаграмма обработки жалоб с помощью приложения «Как должно быть»

Еще одним преимуществом является то, что заместителю нет необходимости быть в офисе АТП для получения доступа к журналу. С учетом разъездного характера работы, это повысит эффективность работы (получить доступ к админ-панели приложения можно даже со смартфона).

1.6 Цель внедрения автоматизированного приема обращений клиентов ООО «Новомосковский Автоплюс»

Основные цели разработки и внедрения веб-приложения для приема и учета обратной связи от клиентов АТП – своевременное реагирование на недостатки обслуживания и их устранение; выявление недобросовестных водителей, которые наносят репутационный и экономический ущерб компании; повышение эффективности учета обращений клиентов, снижение «утечек» инцидентов в соцсети.

Из вышесказанного определим назначение информационной системы:

– повышение доступности отправки обратной связи для пассажиров: от появления недовольства до оставления отзыва должно пройти не более 60 секунд;
– удобное предоставление информации о жалобах заместителю директора: он должен получать доступ к базе через интернет.

Из вышесказанного можно сделать вывод, что необходима информационная система, которая улучшит доставку и повысит объем обратной связи, на основе которой руководство АТП сможет принимать более эффективные и своевременные управленческие решения.

Исходя из назначения ИС и решаемых ей задач, можно представить требования к системе в виде технического задания (ТЗ) по ГОСТ 19.201-78 [14]. Приложение А содержит полный текст ТЗ. Оно раскрывает основные функции программного продукта: быстрый доступ к приложению через QR-код в салоне автобуса и оперативный контроль новых жалоб через админ-панель из любого браузера.

1.7 Анализ известных решений сбора отзывов

В результате того, что АТП не так много, как, например, интернет-магазинов, нишевых решений сбора отзывов под конкретные задачи ТЗ не нашлось.

Чаще встречаются платформы-API для встройки на сайт и организации интерфейса для ответа на отзывы: Shoppilot, Goodvice, Cackle Reviews. Также есть более продвинутые системы (Mneniya.Pro), которые позволяют настроить отправку клиентам сообщений с предложением оставить отзыв, а также способны подключать интерфейс получения обратной связи к сайтам и агрегировать мнения с сайтов-отзовиков. Для задач АТП это слишком продвинутые системы. Их специализация под электронную торговлю не дает настроить систему под задачи АТП: мгновенная отправка отзыва, доступ по QR-коду, 18 страниц (по одной на водителя).

Единственным решением, которое почти подходит под ТЗ, является сервис QReview. Иллюстрация его работы можно посмотреть на рисунке 6.

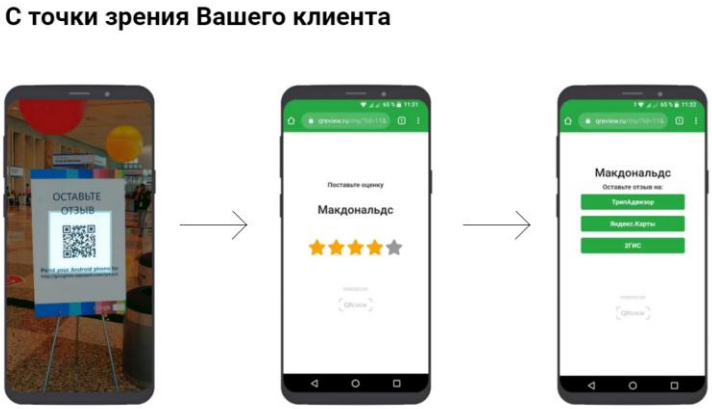

Рисунок 6 – Сервис QReview

На рисунке 6 видно, что сервис предлагает доступ к отзывам по QR-коду, но перенаправляет клиента на общеизвестные сайты-агрегаторы отзывов. Поэтому QReview не позволяет настроить уникальный код на каждого водителя и собрать форму для быстрого отзыва с простым выбором ответов.

В сети также есть еще несколько подобных проектов: Repometr, ME-QR-REVIEW. Но они обладают теми же недостатками, что и QReview.

Выводы по главе 1

В первой главе была дана краткая характеристика предметной области с разбором роли отзывов в работе АТП. Далее были рассмотрены последствия отсутствия платформы по приему обратной связи и проанализированы бизнес-процессы организации в модели «Как есть» и «Как должно быть».

По результатам моделирования сформулированы требования к системе. Они были оформлены в приложении А. В конце главы был проведен обзор текущих программных решений, которые могли бы решить поставленные задачи.

Глава 2 Логическое проектирование веб-приложения «Контролер»

2.1 Концепция веб-приложения

Веб-приложение «Контролер» призвано наладить коммуникацию между руководством ООО «Новомосковский Автоплюс»: обеспечить быстрый бесплатный способ получения обратной связи руководством компании. ПО позволит исправить текущую ситуацию с получением обратной связи: клиентам приходится звонить на номер диспетчера, чтобы пожаловаться на поведение или вождение водителя. После внедрения приложения клиент сможет оставить отзыв за 60 секунд без заполнения форм и звонков по телефону. Веб-приложение будет открываться на телефоне клиента через ссылку по QR-коду в автобусе.

Ссылка должна вести на страницу водителя, в автобусе которого едет пользователь. Приложение должно работать быстро, быть расширяемым, иметь админ-панель, где можно посмотреть все оставленные отзывы. Перечисленные требования можно реализовать с помощью нескольких средств разработки: Flusk, Django, HTML в связке с JS.

HTML и JS позволяют разработать веб-приложение любой сложности. Основной минус применения этой технологии – трудоемкость, которая заключается в отсутствии готовой архитектуры будущего приложения. Группы разработчиков для облегчения процесса программирования выпустили «модули» (веб-фреймворки), которые предоставляют готовую архитектуру приложения, чтобы во время разработки не приходилось писать базовую логику с нуля: админ-панель, движок работы с формами, базой данных и другими элементами [9].

Среди наиболее популярных и удобных фреймворков выделяется Flask и Django (на них построен Netflix, Google, Reddit, Pinterest и тысячи других сайтов) [6]. Фреймворки написаны на языке программирования Python и

позволяют сократить время разработки, предоставляя программисту готовую архитектуру веб-приложения.

Django и Flask схожи, но первый лучше подходит для веб-приложения сбора обратной связи, потому что содержит:

- объектно-реляционное отображение (ORM). Оно позволяет связывать базы данных с объектами во время программирования, что облегчает процесс разработки;
- инструменты работы с базами данных. Встроенная система миграции позволяет перенести объектное отображение баз данных в реальную базу данных на сервере;
- админ-панель. Django позволяет гибко настроить панель администратора без написания ТЗ и выделения времени на разработку интерфейса доступа к базам данных [19].

По вышеперечисленным причинам наиболее целесообразно начать разработку веб-приложения для учета отзывов на основе веб-фреймворка Django.

Django использует шаблон проектирования MVC (Model-View-Controller) – это архитектура, которая разделяет данные разрабатываемого приложения и его логику работы на модель/представление/контроллер.

- модель организует данные в БД, содержит описания таблиц и выходных данных;
- представление содержит логику работы, которая ответственна за отображение и запись информации веб-приложения на основе его модели;
- контроллер интерпретирует активность пользователя и обновляет модель приложения.

Основное преимущество MVC – легкое расширение функциональности приложения и возможность повторного использования кода. Например, одна «модель» может служить отображением для нескольких «представлений».

Также, не меняя «представления», разработчик может поменять реакцию системы на действия пользователя, переписав «контроллер». Разделение бизнес-логики и интерфейса также помогает ускорить разработку, разделив задачи на несколько команд программистов.

В Django «моделью» является файл models.py, в нем описывается структура базы данных приложения. Фреймворк сам подберет соответствующие команды SQL для формирования БД на основе исходного кода.

«Представлением» в Django является файл views.py. В нем пишется код, на основе которого собирается интерфейс приложения: запускаются сохраненные HTML-файлы, вызываются объекты из базы данных для представления в системе. Также в качестве представления можно рассматривать HTML-шаблоны, файлы HTML с разметкой страниц и CSS-файлы, с помощью которых строится интерфейс приложения.

«Контроллером» в Django выступает URL-маршрутизатор urls.py. Он запускает «представление» соответствующее какому-либо URL.

2.2 Логическое моделирование веб-приложения «Контролер»

Для последующей программной реализации системы, на основе представленной в первой главе модели «Как должно быть», необходимо построить логическую модель программного продукта: диаграмму вариантов использования, диаграмму классов [1].

Диаграмма вариантов использования формализует и наглядно описывает функциональные требования к ПО, на ее основе строится диаграмма классов. Диаграммы позволяют более детально представить объекты информационной системы, информацию, которая будет в нее поступать, и как она будет храниться [18].

На рисунке 7 представлена UML-диаграмма вариантов использования на основе модели «Как должно быть».

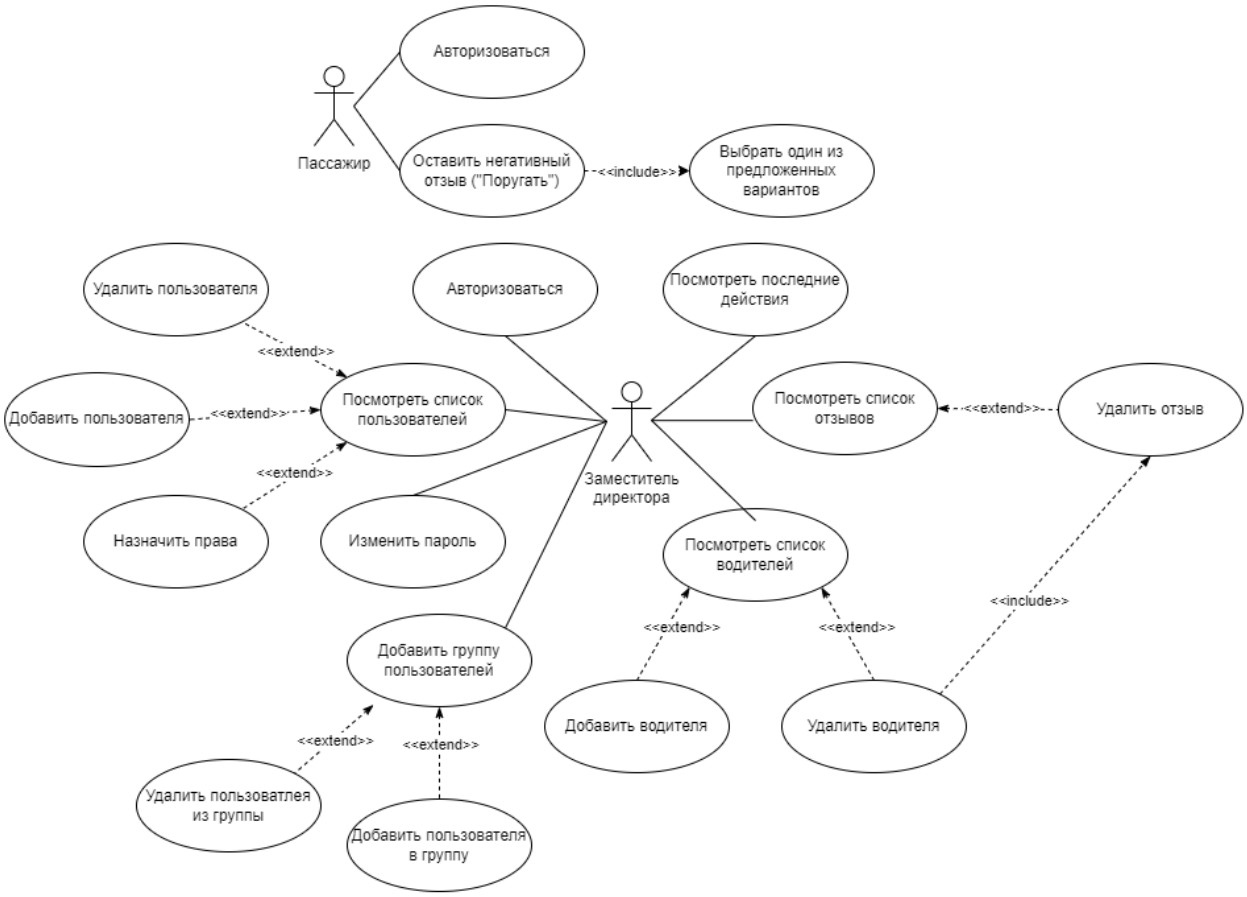

Рисунок 7 – Диаграмма вариантов использования веб-приложения «Контролер»

На рисунке 7 видно, что диаграмма вариантов использования содержит двух актеров (заместитель директора и пассажир) и функции веб-приложения.

Пассажир должен иметь возможность оставить положительный или негативный отзыв без форм ввода (выбором вариантов), а также возможность предложить улучшение сервиса в форме ввода. Негативный отзыв должен включать в себя варианты про нарушение ПДД, грубость, отсутствие маски. Положительный включает только 2 варианта: про вежливое обращение и аккуратное вождение.

Заместитель директора должен иметь возможности:

– авторизоваться;

– посмотреть список пользователей;

– изменить пароль входа, добавить группы пользователей;

– посмотреть последние действия с базой данных;

– посмотреть список отзывов и водителей.

Пункт «Посмотреть список всех водителей» расширяется за счет действий, направленных на удаление и добавление пользователей, а также на назначение прав пользователям по удалению записей. Пункт «Добавить группу пользователей» расширяется на пункты «Удалить пользователя из группы» и «Добавить пользователя в группу».

Также расширяется действие «Посмотреть список водителей» (на действие добавления и удаления водителей). Пункт «Удалить водителя» включает действие «Удалить отзыв» по причине того, что после удаления какого-либо водителя, все отзывы о нем должны пропасть. Пункт «Посмотреть список всех отзывов» расширяется за счет действия «Удалить отзыв», который позволяет стереть отзыв о любом водителе.

На основе диаграммы вариантов использования и модели «Как есть» можно построить диаграмму классов (рисунок 8).

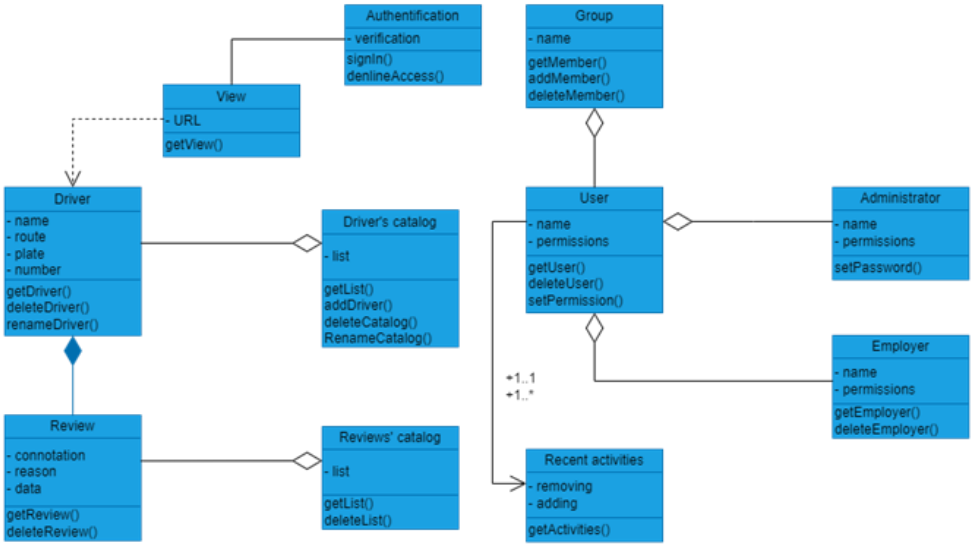

Рисунок 8 – Диаграмма классов

На рисунке 8 видно, что класс Authentification позволяет администратору войти в систему, он содержит атрибут verification, методы входа (signIn()) и отказа (declineAccess()).

Класс View отображает страницу в зависимости от ссылки, которую в него передали (передается QR-кодом), поэтому класс связан с Driver отношением зависимости. View содержит метод getView(), который позволяет отобразить страницу веб-приложения.

Класс Driver относится к сущности водителя, он имеет атрибуты: имя (name), маршрут (route), номер машины (plate), табельный номер (number). Конструктор класса создает соответствующий объект в базе данных, методы – вызывают имя водителя (getDriver()) (это нужно для отображения на главном экране приложения), удаляют водителя (deleteDiver()) и переименовывают его (renameDiver()).

Класс Review относится к сущности отзыва. Он имеет атрибуты: коннотация (connotation), причина отзыва (reason), дата публикации (data). Методы включают в себя получение (getReview()) и удаление (deleteReview()) отзыва. Review связан с Driver отношением композиции: если удалить водителя из системы, то все отзывы, которые относились к этому водителю, удалятся автоматически.

Классы Driver и Review связаны отношением агрегации с классами Driver’s catalog и Reviews’ catalog, которые относятся к сущностям списков внесенных в систему водителей и относящимся к ним отзывам, которые содержат методы, позволяющие получить, удалить или добавить объект.

Класс User относится к сущности пользователя, он содержит атрибуты имени (name) и разрешений (permissions), в числе методов – получение пользователя (getUser()), удаление пользователя (deleteUser()), установка прав пользователя (setPermission()).

Класс Administrator имеет схожие с User поля, но отличается методом, который устанавливает пароль администратора. Administrator связан с User отношением агрегации. Также отношением агрегации с User связан класс

Employer (сотрудники), который тоже имеет атрибуты name и permissions, методы – получения (getEmployer()) и удалениях (deleteEmployer()) пользователя.

Класс Group относится к сущности группы пользователей и содержит поле name (имя пользователя), а также методы addMember(), getMember(), deleteMember(), которые добавляют, позволяют получить и удалить участника группы.

Класс Recent activities относится к сущности недавних действий пользователя. Класс содержит два атрибута: история о добавлении пользователей (adding) и их удалении (removing), метод класса (getActivities()) позволяет получить последнюю активность в базе. Recent activities связан с User N-арной связью с мощностью связи один-ко-многим (один пользователь на множество действий).

2.3 Характеристика потоков информации

Сейчас для отметки о поступлении жалобы используется журнал. После внедрения системы необходимости в журнале не будет, на его место придет админ-панель веб-приложения, что отражено в диаграмме «Как должно быть». Это единственное изменение документопотока организации.

В качестве входной информации используются данные, введенные пользователем. В случае если пользователь хочет оставить положительный или отрицательный отзыв, ему предлагается выбор вариантов ответов (в чем именно причина отзыва). Полученные данные вносятся в базу данных (SQL) автоматически фреймворком Jango. Также в БД вносятся данные о водителях. Поля БД отзывов со всеми возможными вариантами отображены в таблице 1.

Поле id содержит идентификационный номер отзыва, comment_connotation – полярность отзыва (позитивный, негативный, предложение), поле comment_date – дату отзыва, поле driver_id –

идентификационный номер водителя, который присваивается ему в таблице водителей (таблица 2).

Таблица 1 – Поля таблицы отзывов

id	comment_connotation	comment_reason	comment_date	driver_id
1	Позитивная	Вежливый	2000-01-01	1
2	Позитивная	Аккуратно водит	2000-01-01	2
3	Предложение	«Текст предложения»	2000-01-01	3
4	Негативная	Нагрубил	2000-01-01	4
5	Негативная	Нарушил ПДД	2000-01-01	5
6	Негативная	Не надел маску	2000-01-01	6

Таблица 2 – Поля таблицы данных о водителях

id	driver_name	driver_number	car_plate	car_route
1	Феофилактов Андрей Анатольевич	947083	ВВ32371	140

Поле id содержит номер водителя в таблице, через это значение ФИО и другие данные водителя отображаются в системе. Поле driver_name содержит ФИО водителя, driver_number – табельный номер, car_plate – автомобильный номер, car_route – номер маршрута.

Выводы по главе 2

Во второй главе рассмотрена концепция веб-приложения, описаны цели его создания, механизм работы и технология разработки. В качестве технологии разработки используется веб-фреймворк Django.

Построена логическая модель программного продукта, которая включает в себя диаграмму вариантов использования и диаграмму классов. В диаграмме вариантов использования рассмотрены сценарии взаимодействия различных групп пользователей с системой.

Глава 3 Разработка программного продукта

3.1 Выбор архитектуры информационной системы

Заместитель директора, как один из основных пользователей системы, не всегда находится в офисе, а пассажиры должны быстро добавлять отзывы с любой точки маршрута и с любого смартфона (iOS, Android, Windows Phone). Поэтому объединение всех пользователей в одну компьютерную сеть невозможно. По этой причине лучший вариант архитектуры должен соответствовать веб-приложению: БД будет храниться на сервере, а на устройстве пассажира и заместителя директора должен быть только браузер (ничего не нужно устанавливать). Python-код на сервере (бэкенд) отвечает за HTTP-запросы и логику, а интерфейс отрисовывается за счет HTML (фронтэнд).

3.2 Инструменты разработки

Согласно ТЗ, ИС должна работать как в мобильных телефонах пассажиров, так и на ПК заместителя директора, не требовать установки или загрузки. Остальные требования для выбора инструментов разработки:

- наличие библиотек работы с БД;
- скорость разработки;
- бесплатность;
- надежность.

Упомянутый во второй главе веб-фрейморк Django был выбран именно по этим критериям. Разработка с его помощью ведется на языке программирования Python. Это высокоуровневый скриптовый язык, который ориентирован на повышение читаемости кода, скорости разработки и использование библиотек. Django включает в себя ORM-систему (Object-Relational Mapping – объектно-реляционное отображение), которая позволяет

переводить логику работы с БД, написанную на Python, в команды SQL автоматически. Таким образом можно опосредованно управлять базой данных.

Для разработки ПО используется командная строка Windows 10, среда разработки Geany, интерпретатор Python 3, программа SQLiteStudio, браузер на основе движка Chromium, Django 3. С учетом разработанных во второй главе моделей, разработку программного продукта с использованием Django можно разделить на несколько этапов по созданию:

- проекта;
- модуля приложения;
- модуля View из концепции MVC;
- URL-привязок для view;
- модели для сборки БД (Model из MVC);
- админ-панели;
- логики приема отзывов и страниц водителей;
- шаблона для рендеринга страниц приложения;
- URL-привязки для каждого водителя.

Далее будет описан каждый из перечисленных этапов.

3.3 Обзор файлов проекта

Django запускается и настраивается через командную строку. Для старта проекта необходимо создать директорию проекта в памяти компьютера, запустить в этой директории командную строку и выполнить команду django-admin startproject. Эта строчка вызывает встроенную в фреймворк утилиту django-admin с командой startproject и названием проекта в качестве аргумента (рисунок 9).

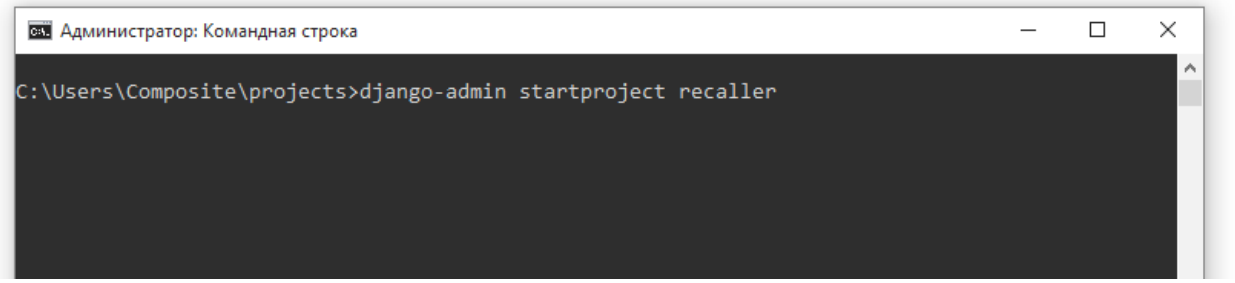

Рисунок 9 – Запуск проекта Django

На рисунке 9 видно, что в качестве рабочего названия в Django «Контролер» получил название recaller. После выполнения команды, в директории projects появляется папка recaller, в которой создается файл manage.py. Это утилита, через которую будет происходить управление проектом: создание модулей, запуск сервера, миграции БД и другие операции. Также в директории появляется вложенная директория recaller. Ее содержимое показано на рисунке 10.

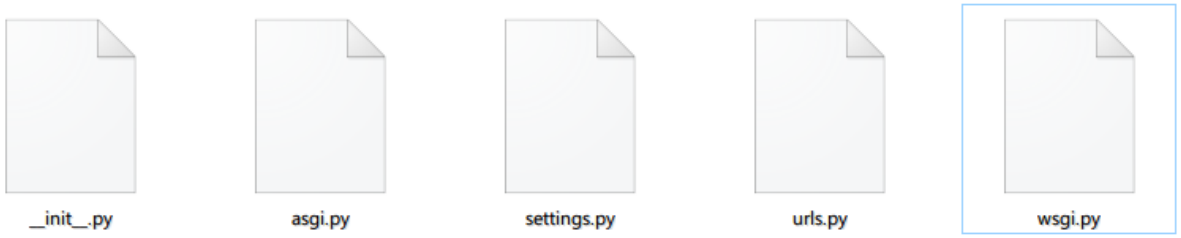

Рисунок 10 – Содержимое директории recaller

На рисунке 10 видно, что проект содержит 5 файлов. __init__.py инициализирует директорию как проектную (как «пакет»). Asgi.py и wsgi.py позволяют развернуть приложение на сервере. Settings.py хранит настройки проекта, которые ответственны за расположение БД, временную зону UTC, язык приложения и другие параметры. Urls.py содержит URL-адрес для каждого приложения проекта (URL-привязка). Как видно по форматам

файлов, все настройки и файлы проекта написаны на Python. На рисунке 11 показан фрагмент файла settings.py.

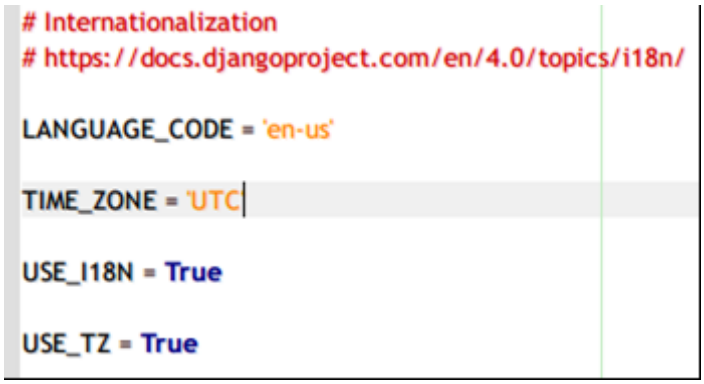

Рисунок 11 – Фрагмент кода settings.py

На рисунке 11 видны некоторые из настроек проекта: временная зона будущего приложения и настройки, отвечающие за язык интерфейса.

3.4 Начало работы над проектом

Как было сказано в главе 2, проекты на Django разбиты на модули (приложения), которые отвечают за различные функции. Чтобы начать работу над проектом, нужно создать хотя бы одно приложение. В случае «контролера» модуль будет называться drivers по причине того, что он ответственен за представление водителей в системе (рисунок 12).

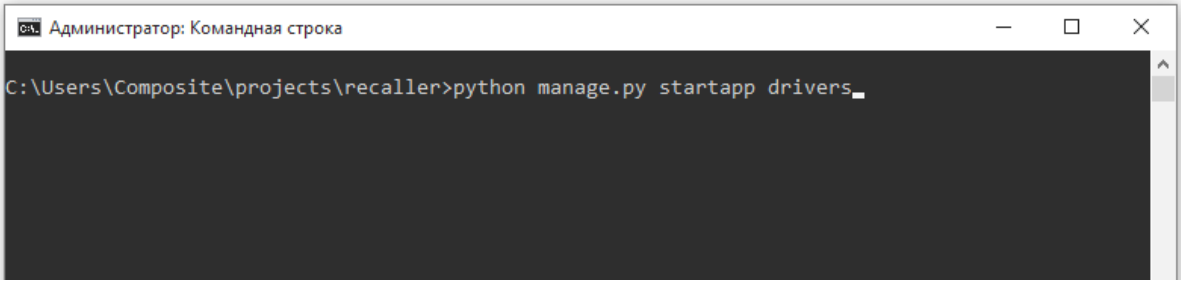

Рисунок 12 – Команда для создания модуля (приложения) drivers

Как видно на рисунке 12, работа в Django ведется через утилиту manage.py. Она запускает команду startapp, которая в качестве аргумента принимает название нового модуля. После обработки команды в проекте появится новая директория drivers (рисунок 13).

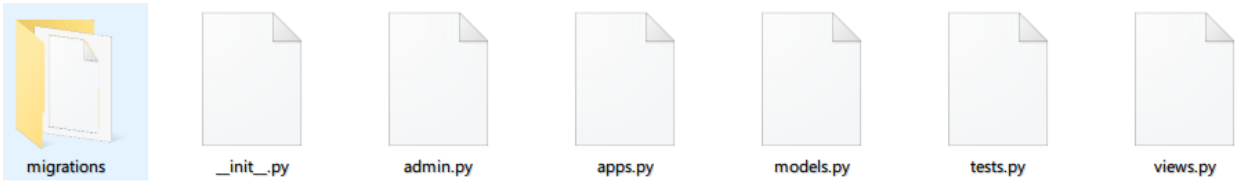

Рисунок 13 – Содержимое директории drivers

Как видно на рисунке 13, drivers содержит 6 файлов и 1 каталог. Migrations содержит файлы миграций БД, которые образуются после трансляции логики БД в SQL. Файл init.py инициализирует директорию как папку проекта, файл admin.py содержит настройки админ-панели, apps.py ответственен за конфигурацию модуля, models.py строит модель БД, по которой будет создана база данных после миграции, файл views.py отвечает за представление модуля (в нем пишется логика работы приложения). Все файлы изначально пустые, именно в них будет реализован функционал приложения «Контролер».

3.5 Создание URL-привязок приложения

Продолжить работу над проектом необходимо с создания файла urls.py в директории приложения drivers. Файл будет содержать локальную URL-привязку, которая ответственна за доступ к модулю из строки браузера (рисунок 14).

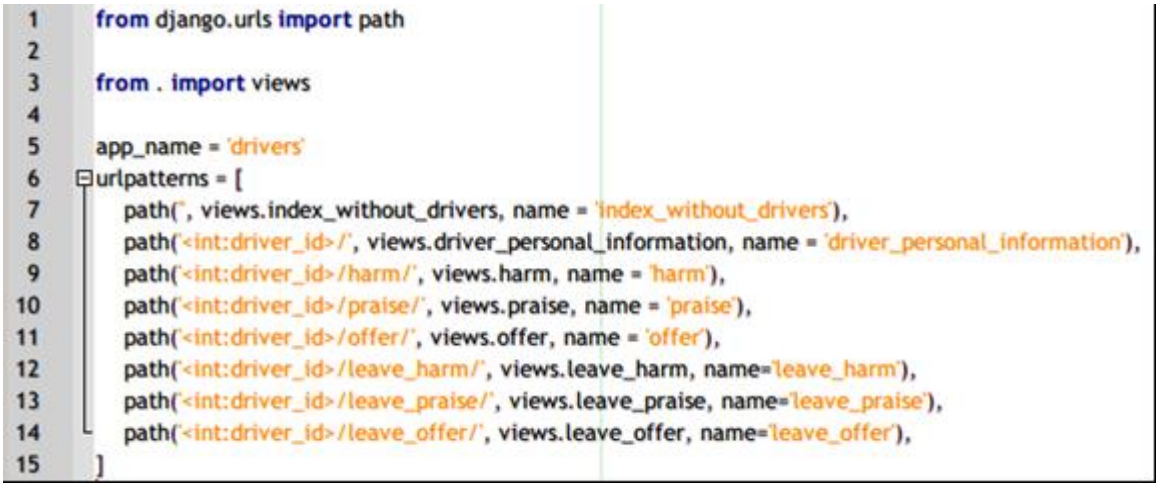

Рисунок 14 – Содержимое файла urls.py

На рисунке 14 видно, что файл содержит всего 15 строк, которые импортируют модули Django и создают пути для входа в приложение через адресную строку браузера после разворачивания кода на сервере. Например, седьмая строчка задает пустой путь ‘’ (две кавычки без кода). Это значит, что просто по адресу приложения без ID водителя (https://controler.pythonanywhere.com/drivers/) запустится функция index_without_drivers в файле views, которая запустит соответствующий HTML-файл.

В следующих строчках определяется уникальная ссылка для каждого водителя (<int:driver_id>/). Именно эта ссылка будет зашифрована в QR-коде и будет вести на страницу отзыва конкретного водителя. В качестве ID водителя в системе используется его номер в БД. На восьмой строке после ID подключается функция driver_personal_information из файла views.py, эта функция открывает домашнюю страницу с выбором тематики отзыва. На следующих строчках URL-привязка ведет к представлениям harm, praise и offer. Каждое представление ответственно за открытие страницы для отправки негативного или позитивного отзыва, а также предложения (offer). Строчки с 12 по 14 ответственны за сохранение отзывов в базе данных.

До прописанных путей локальной привязки путь в строке браузера задает корневая привязка, та привязка, которую можно найти не в директории модуля, а в корневой папке проекта (рисунок 15).

```
1  from django.contrib import admin
2  from django.urls import path, include
3
4  urlpatterns = [
5      path('drivers/', include('drivers.urls')),
6      path('admin/', admin.site.urls),
7      path('', include('drivers.urls')),
8  ]
```

Рисунок 15 – Корневая URL-привязка проекта

На рисунке 15 видно, что строка с номером 5 задает путь drivers/, шестая – admin/. В итоге путь к водителю будет выглядеть так: http://controler.pythonanywhere.com/drivers/10. Число в URL отвечает за ID водителя.

Код include(‘drivers.urls‘) на пятой строчке подключает к корневой привязке локальную привязку, таким образом, давая Django возможность продолжить путь в строке браузера за счет адресов из локальной привязки. За доступ к самому приложению drivers из строки браузера отвечает корневая привязка, а за то, что после слова drivers – локальная привязка.

Путь http://controler.pythonanywhere.com/admin запускает админ-панель сайта, поэтому в локальной привязке drivers нет упоминания админ-панели (это другой модуль).

3.6 Содержание файла представления (Views) системы

В локальной URL-привязке пути всегда ведут к представлению системы (привязка открывает доступ к представлению). Представление прописано в файле views.py, который расположен в директории drivers по

причине того, что views.py относится только к модулю drivers. У каждого модуля свой файл views.py. Например, для отображения админ-панели сайта используется другое представление. Фрагмент содержимого файла views.py показан на рисунке 16.

```
def driver_personal_information(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Неправильная ссылка или водителя нет в базе данных")

    return render(request, 'index.html', {'actual_driver_obj': actual_driver_obj})

def harm(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id) # здесь второй вызов объекта водителя, потому что он передается в harm.html. dirvert_id взяли из адресной стр
    except:
        raise Http404("Не удалось получить объект водителя")

    return render(request, 'actions/harm.html', {'actual_driver_obj': actual_driver_obj}) # передал объект водителя с предыдущей страницы на страницу harm

def praise(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    return render(request, 'actions/praise.html', {'actual_driver_obj': actual_driver_obj})
```

Рисунок 16 – Фрагмент представления модуля drivers

На рисунке 16 можно распознать представления, ссылки на которые видны на рисунке 14. Например, 12-я строка создает переменную с данными водителя, которые подгружаются из БД. На БД в данном случае ссылается объект Driver.

Driver.objects.get(id = driver_id) на 20-й строке подключается к БД SQL, в которую внесены данные водителей, и вызывает из базы имя и другие данные водителя. Вызов происходит по ID, который передается в качестве аргумента (он берется из адресной строки, а там адрес появляется по QR-коду).

Render() в каждой функции ответственен за запуск соответствующего функции HTML-шаблона, в который прописываются данные водителя. Например, функция driver_personal_information() «отрендерит» файл домашней страницы. Результат запуска этой функции с взятием персональных данных из БД по адресу из QR-кода показаны на рисунке 17.

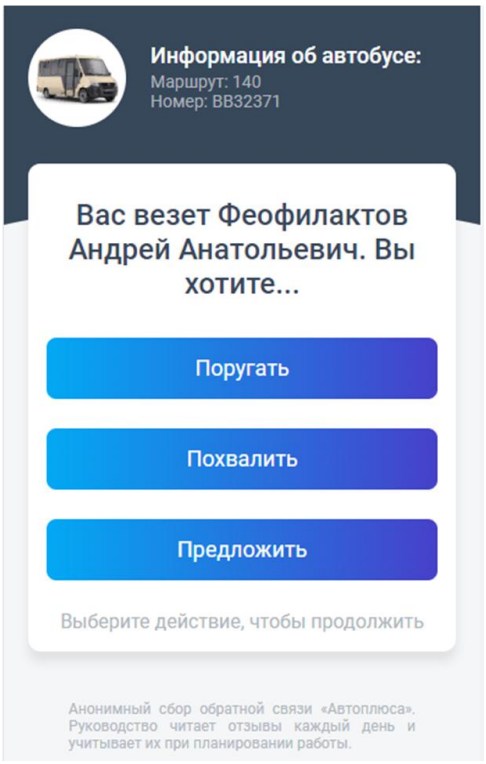

Рисунок 17 – Страница, которая открывается по QR-коду водителя

На рисунке 17 изображен результат работы функции driver_personal_information(). ФИО водителя и информация об автобусе (маршрут, номер) подгружаются из БД с помощью переменной actual_driver_obj.

Переменная подставляется в HTML-шаблон с помощью функции render(), которая принимает объект с данными водителя и название HTML-файла (строчка 16 рисунка 15). HTML-шаблоны были написаны заранее, о них будет рассказано в параграфе 3.7. Строчки «Поругать», «Похвалить», «Предложить» запускают соответствующие представления на рисунке 15 (18 и 26 строки). Они также запускают функцию render(). Результат работы функции harm() («Поругать») показан на рисунке 18.

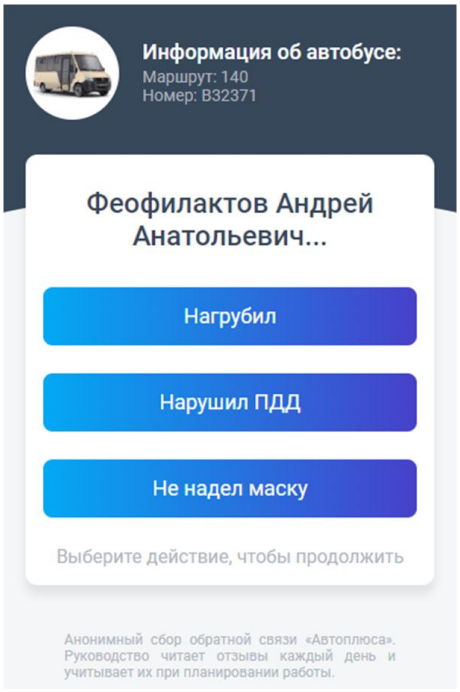

Рисунок 18 – Пункт «Поругать», результат работы функции harm()

После нажатия какой-либо кнопки на рисунке 18 данные запишутся в БД, а пассажир увидит соответствующую выбору страницу (рисунок 19).

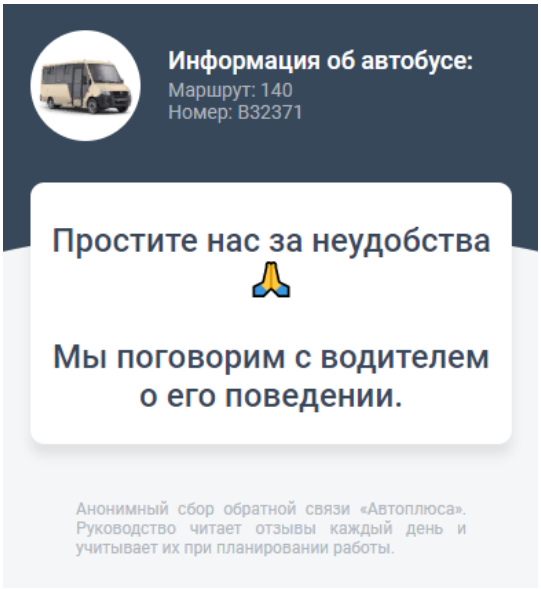

Рисунок 19 – Извинение после нажатия пассажиром кнопки «Нагрубил»

На рисунке 19 виден результат работы функции, которая ответственна за рендеринг страницы с извинениями и за запись негативного отзыва в БД (рисунок 20).

Рисунок 20 – Функция leave_harm

Как видно из рисунка 20, функция leave_harm() сохраняет в БД коннотацию («Негативная»), причину («Нагрубил») и дату отзыва с помощью модели Comment. Коннотация и причина записаны в HTML-шаблоне и передаются с помощью метода POST, который прописан в HTML на соответствующей странице.

Далее через ORM Django трансформирует код в команды SQL и сохраняет данные в БД функцией save(). После записи функция показывает пользователю страницу с извинением, передавая в HTML-шаблон данные о маршруте и номере машины (на рисунке 18 они изображены в шапке страницы).

Таким же образом написаны обработчики других негативных причин («Нарушил ПДД», «Не надел маску») и позитивной коннотации (рисунок 21).

На рисунке 21 видны положительные коннотации отзывов, при их выборе сработает функция leave_praise() и render(). Первая сохранит объект в БД на имя водителя, вторая – запустит рендер страницы благодарности за отзыв (рисунок 22).

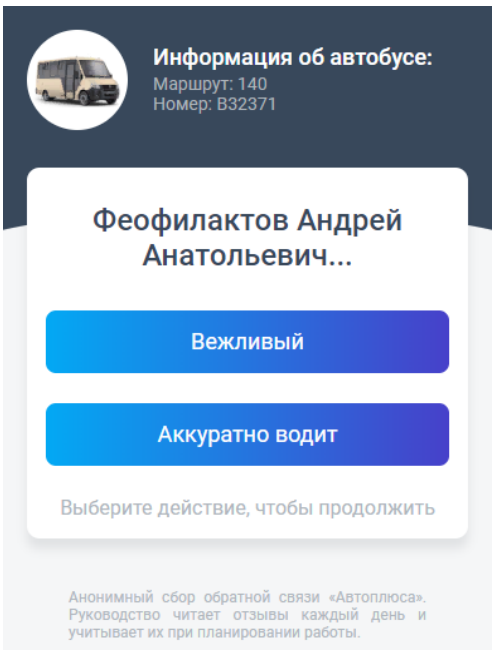

Рисунок 21 – Меню при нажатии кнопки «Похвалить»

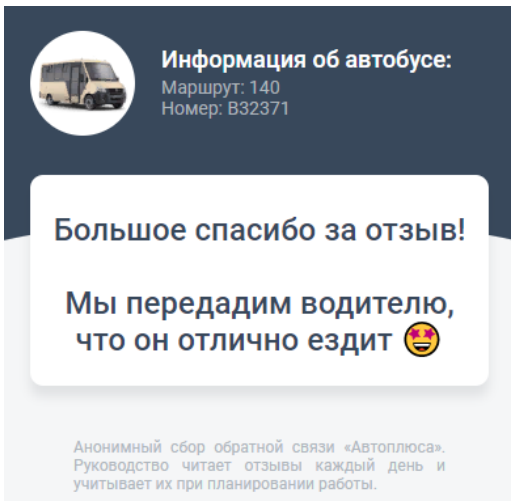

Рисунок 22 – Благодарность за положительный отзыв

На рисунке 22 видно, что для каждой коннотации последняя страница отличается. Также от других коннотаций отличается меню предложения идеи (рисунок 23).

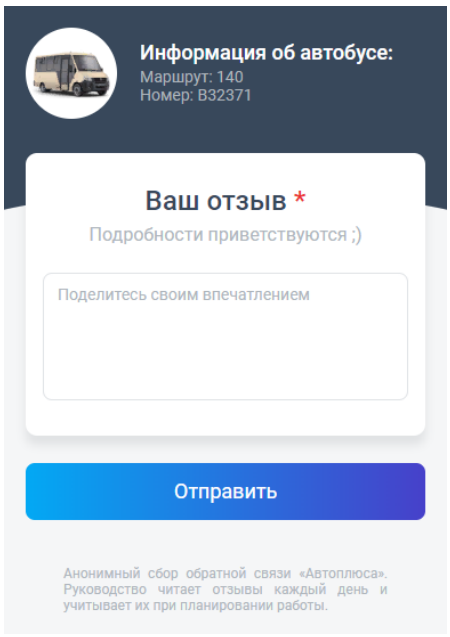

Рисунок 23 – Форма отправки предложения

На рисунке 23 видно, что при выборе кнопки «Предложить» нет кнопок на выбор, а есть поле ввода. Текст из поля ввода записывается в таблицу в качестве коннотации отзыва. Благодарность за предложение изображено на рисунке 24.

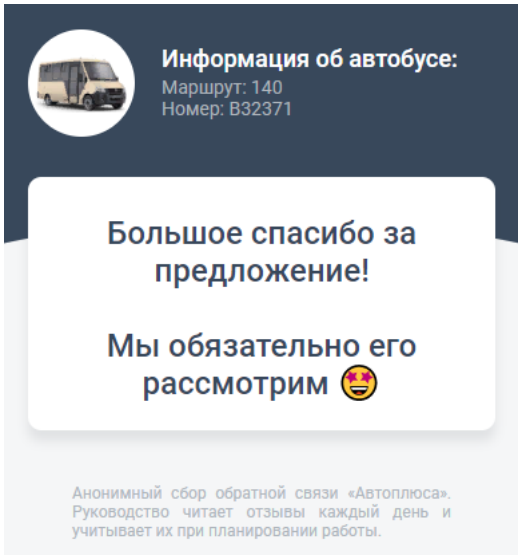

Рисунок 24 – Благодарность за предложение

На рисунке 24 видно, что текст благодарности за предложение тоже отличается. Как и другие тексты «Контролера» он лишен «канцеляризма» и направлен на то, чтобы успокоить расстроенного плохим обслуживанием пассажира и встать на его сторону. В следующих разделах будет описаны модели, по которым формируется HTML-шаблонизатор и база данных.

3.7 Использование шаблонизатора для отображения страниц

Чтобы для каждого представления и страницы не писать новый HTML-код, Django предусматривает метод шаблонизации – внедрение изменяющихся данных в статичные HTML-файлы. Например, страница водителя с предложением «похвалить» или «поругать» не пишется отдельно для каждого водителя. Это один HTML-файл, который предусматривает встраивание данных. Динамика реализуется с помощью функции render(). Первым аргументом она принимает запрос (request), вторым – файл-шаблон, в который будут встроены данные, третьим – объект, который будет встроен в шаблон. Django «ищет» файл с шаблонами в директории templates в корневой папке проекта (рисунок 25).

Рисунок 25 – Содержимое папки templates

На рисунке 25 видно, что директория templates содержит один файл и 2 каталога. В папке actions находятся шаблоны для рендеринга страниц с вариантами ответов, в папке base – шапка и подвал страницы, которые подгружаются в каждый файл выбора ответов (рисунок 26).

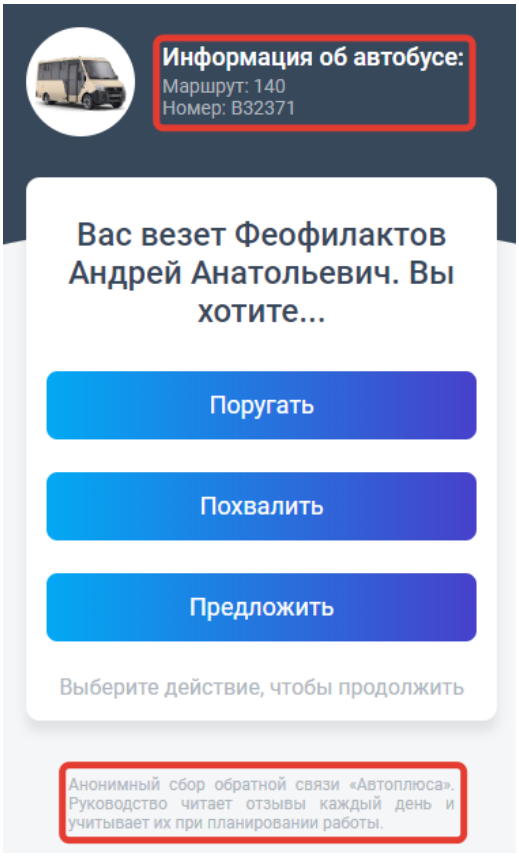

Рисунок 26 – Встраиваемый текст из файла base.html

На рисунке 26 красным отмечены объекты, которые написаны 1 раз в файле base и далее загружаются в 18 страниц водителей, в две страницы с вариантами ответов и в 5 страниц с благодарностями. ФИО водителя встраивается представлением по уникальной ссылке водителя. Фрагмент кода файла base показан на рисунке 27.

На рисунке 27 виден HTML-код шапки и подвала, который загружается на каждую страницу приложения. На 25-й и 26-й строке можно заметить встраивание объекта водителя, который передается из файла views.py. Из этого объекта вызывается маршрут через свойство объекта водителя (actual_driver_obj.car_route) и номер автобуса.

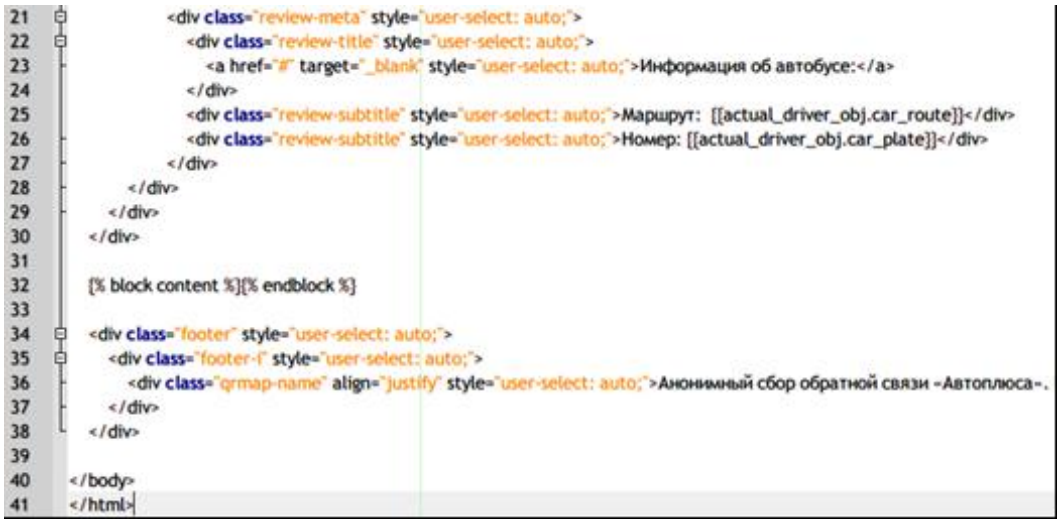

Рисунок 27 – Фрагмент файла base.html

Код на 32-й строчке сообщает Django, что между блоками {% block content %} и {% endblock %} нужно встроить код того файла, который будет вызван вместе с шаблоном (HTML-файл, который передается в функцию render()). На стороне таких встраиваемых файлов шаблонизация выглядит иначе (рисунок 28).

Рисунок 28 – код шаблона со стороны вставляемого HTML-файла

На рисунке 28 представлен файл главной страницы, которая открывается после сканирования QR-кода (она предлагает «Поругать», «Похвалить», «Предложить»). Начинается файл с объявления того, что файл расширяется в файл base.html. Для этого используется слово extends на первой строке.

Далее блоки {% block title %} и {% block content %} отмечают фрагменты кода, которые Django встроит в base на место одноименных блоков (title и content) в base.html. Соответственно, код из {% block content %} с рисунка 28 Django вставит в {% block content %} на рисунке 27. Таким образом, можно делать множество блоков и встраивать их в различные страницы веб-приложения.

Переход по страницам приложения осуществляется через переходы по представлениям (рисунок 29)

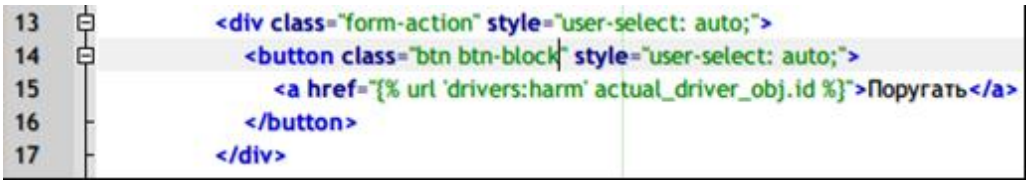

Рисунок 29 – Внутренняя ссылка на страницу с негативным отзывом

На рисунке 29 видно (15-я строка), что кнопка «Поругать» ведет на представление harm из модуля drivers (drivers:harm), а объект actual_driver_obj передает в представление harm ID водителя, которое было получено из объекта водителя, ссылка которого открылась по QR-коду. Представление harm показывает варианты негативных отзывов («Нагрубил» и другие). Код представления harm показан на рисунке 30.

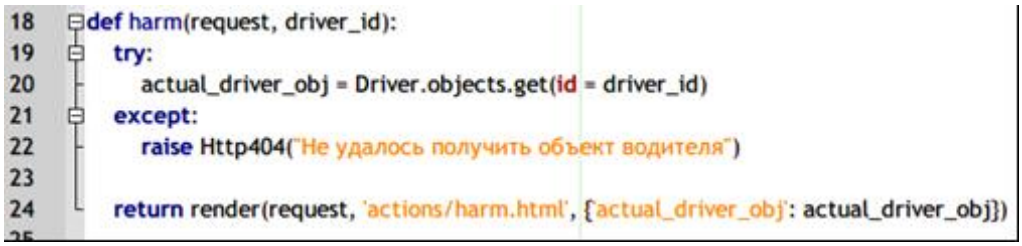

Рисунок 30 – Код представления (view) harm

На рисунке 30 видно, что функция harm() принимает request и driver_id. Request необходим для передачи HTTP-запроса, а по driver_id в блоке try код на 20-й строчке пытается получить объект водителя из БД. Если это получается, то функция render открывает файл harm.html (который тоже расширяется из шаблона) и передает в него объект водителя под тем же именем actual_driver_obj (просто передать объект без обращения в БД по ID не получится). На рисунке 31 показан фрагмент кода файла harm.html (он открывается после нажатия кнопки «Грубость»).

```
<form action="{% url 'drivers:leave_harm' actual_driver_obj.id %}" method="POST">
    <div class="form-action" style="user-select: auto;">

    {% csrf_token %}

    <input type="hidden" name="connotation" value="Негативная">
    <button type="submit" name="reason" value="Грубость" class="btn btn-block btn-qrmap" style="user-select: auto;">Нагрубил</button>
    </div>
</form>
```

Рисунок 31 – Фрагмент кода файла harm.html

Как видно из рисунка 31, кнопка причины негативного отзыва «Грубость» отправляет POST-запрос в представление leave_harm и направляет в leave_harm объект водителя. Представление leave_harm показано на рисунке 32.

```
def leave_harm(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    data_to_DB = actual_driver_obj.comment_set.create(comment_connotation = request.POST['connotation'], comment_reason = request.POST['reason'], comment_date = timezone.now())
    data_to_DB.save()

    return render(request, 'actions/thanks/thanks_for_harm.html', {'actual_driver_obj': actual_driver_obj})
```

Рисунок 32 – Фрагмент кода представления leave_harm

Как видно из рисунка 32, принятый POST-запрос передает request, и из этого объекта в дальнейшем извлекается коннотация (connotation) и причина

(reason) отзыва. Дальше эта информация записывается в БД на водителя, ID которого было передано в качестве второго аргумента. В итоге в БД проявляется запись на рисунке 33.

	id	comment_connotation	comment_reason	comment_date	driver_id
1	134	Негативная	Грубость	2022-04-12 23:02:50.491999	7

Рисунок 33 – запись в базе данных

Как и на рисунке 33 построены отзывы других коннотаций и предложение, при оставлении предложения коннотацией является текст, введенный пассажиром.

3.8 Механизм добавления отзыва в базу данных

База данных формируется с помощью файла models.py. В нем содержатся поля таблиц и их характеристики. Физическая модель базы данных файла models.py отражена на рисунке 34.

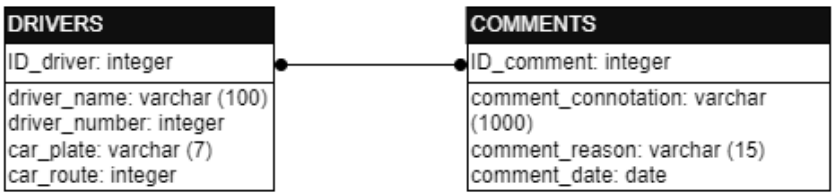

Рисунок 34 – схематичное отображение БД в models.py

На рисунке 34 видно, что файл models.py содержит 2 модели БД: водителей (Drivers) и отзывов (Comments). Содержимое файла models.py показано на рисунке 35.

```
from django.db import models

from django.utils import timezone

class Driver(models.Model):
    driver_name = models.CharField('Имя водителя', max_length = 100)
    driver_number = models.CharField('Табельный номер', max_length = 7)
    car_plate = models.CharField('Номер машины', max_length = 7)
    car_route = models.CharField('Маршрут', max_length = 3, default = 140)

    def __str__(self):
        return self.driver_name

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

class Comment(models.Model):
    driver = models.ForeignKey(Driver, on_delete = models.CASCADE)
    comment_connotation = models.CharField('Полярность отзыва', max_length = 15)
    comment_reason = models.CharField('Причина', max_length = 15)
    comment_date = models.DateTimeField('Дата отзыва')

    def __str__(self):
        return self.comment_connotation

    class Meta:
        verbose_name = 'Отзыв'
        verbose_name_plural = 'Отзывы'
```

Рисунок 35 – Содержимое файла models.py

На рисунке 35 видно, что models.py содержит две модели: Driver и Comment. Первая представляет собой основу для реляционной БД водителей, вторая – для БД отзывов. Функции __str__(self) представляют собой геттеры, а классы Meta определяют множественное и единственное число названия моделей для админ-панели.

Driver создает 4 поля таблицы:

– driver_name;

– driver_number;

– car_plate;

– car_route.

Методы CharField() определяют содержание полей таблицы как поле для символов, а max_length определяет максимальный размер. При миграции Django переведет этот код в SQL и создаст соответствующие БД. Весь остальной код приложения «Контролер» размещен в приложении Б (рисунки Б.1 – Б.8).

Класс Comment схож с Driver, но отличается наличием внешнего ключа ForeignKey, который заставляет любой отзыв удалиться после удаления водителя, на которого этот отзыв был оставлен.

Миграция модели в БД показаны на рисунке 36.

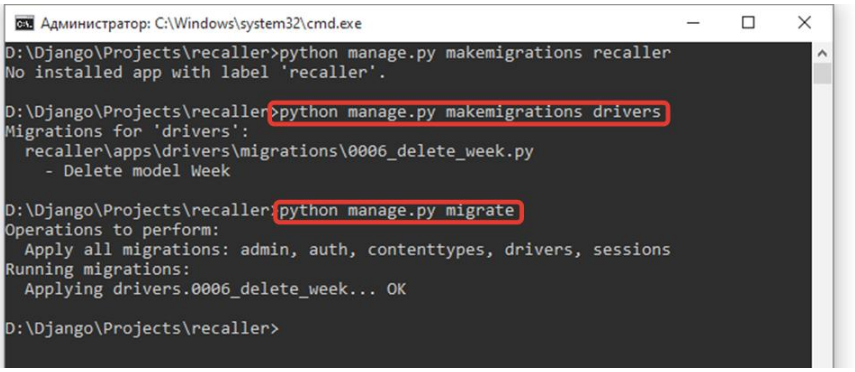

Рисунок 36 – Команды миграции из models.py в БД

На рисунке 36 отображены две команды: для создания миграции и для ее применения. Команда python manage.py makemigrations drivers создает миграцию и перечисляет изменения. В данном случае была удалена модель Week. Команда python manage.py migrate применяет миграцию. Команда python manage.py sqlmigrate drivers 0006 позволяет посмотреть, какой SQL-код применил Django для внесения изменений в таблицы (рисунок 37).

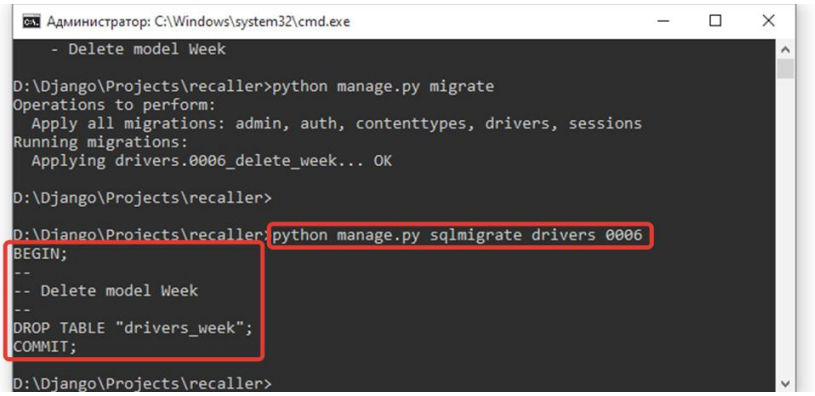

Рисунок 37 – Вывод команды sqlmigrate

На рисунке 37 видно, что после команды нужно ввести название модуля (drivers) и номер миграции (0006), чтобы узнать примененные SQL-команды.

По миграциям можно отследить изменения и восстановить прежнее состояние БД. Миграции хранятся в директории migrations и позволяют отследить изменения БД.

3.9 Админ-панель приложения «Контролер»

Админ-панель предоставляет инструменты для просмотра оставленных отзывов, добавления и удаления водителей. Интерфейс главной страницы показан на рисунке 38.

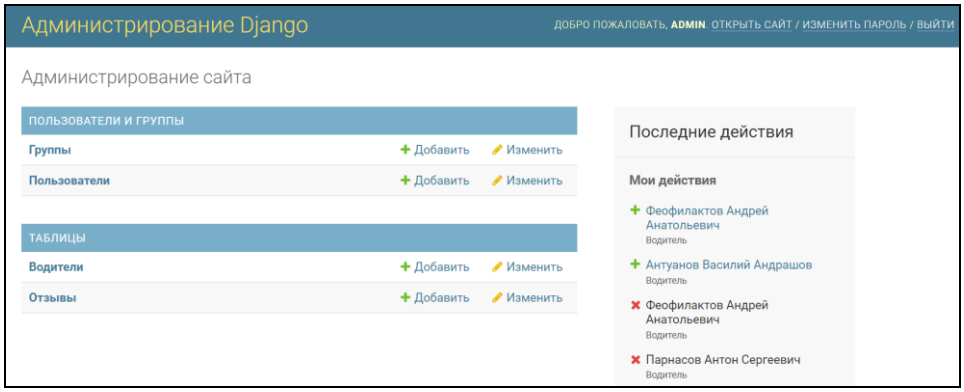

Рисунок 38 – Главная страница админ-панели

На рисунке 38 видно, что на главной странице можно получить доступ к таблицам отзывов и водителей, добавить пользователя или группу пользователей, посмотреть последние действия, открыть сайт, изменить пароль администратора или выйти из системы. Меню добавления пользователя изображено на рисунке 39.

Как видно на рисунке 39, пользователю можно назначить пароль и имя. После добавления пользователя ему можно назначить права и статус (рисунок 40).

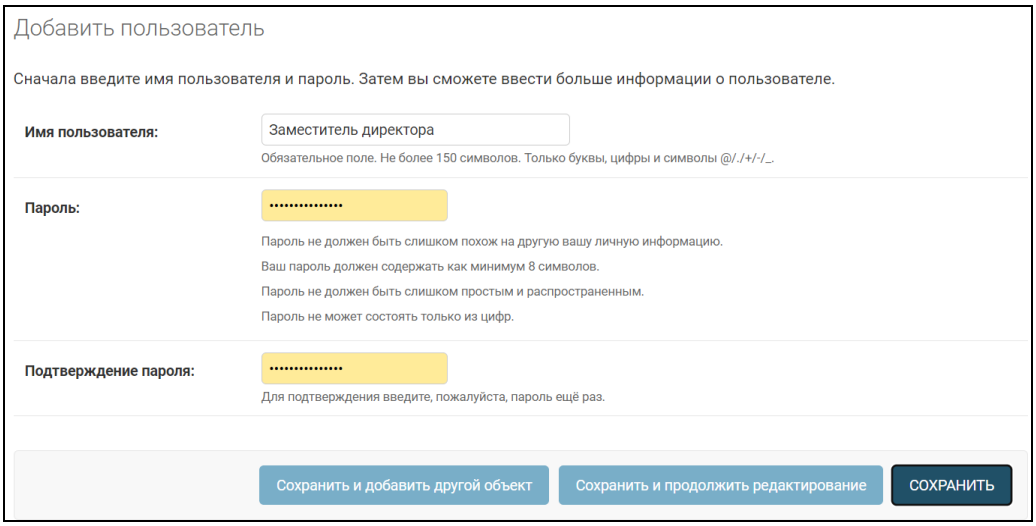

Рисунок 39 – Меню добавления пользователя.

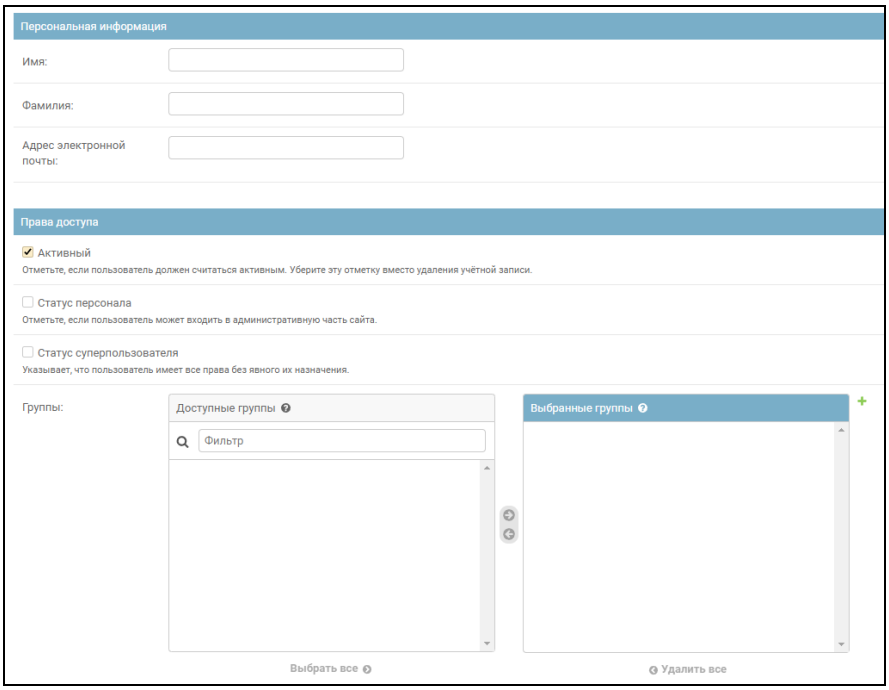

Рисунок 40 – Данные пользователя

На рисунке 40 видно, что пользователю можно назначить статус персонала, активного пользователя или суперпользователя. Также менять права можно сразу нескольким сотрудникам, если объединить их в группу список прав насчитывает 50 команд: can add group (может добавлять группы),

can change group (может менять группу), can delete group (может удалять группу) и другие. (рисунок 41).

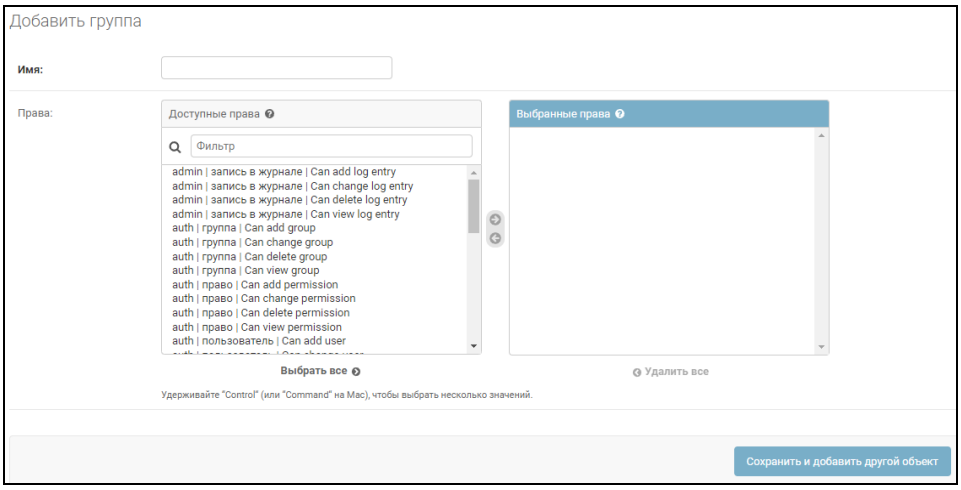

Рисунок 41 – Меню создания группы пользователей

На рисунке 41 изображен список прав, которыми можно наделить группу пользователей. Пользователи с правами администратора смогут вносить изменения в таблицы приложения (рисунок 42).

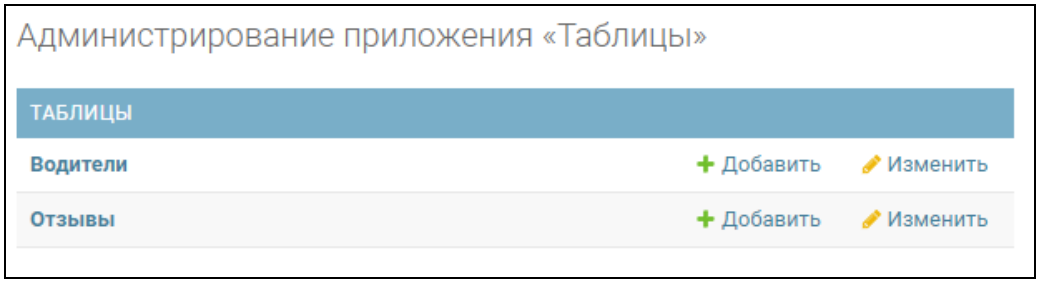

Рисунок 42 – Панель администрирования таблиц

Из панели администрирования на рисунке 42 видно, что в системе две записанные в файле models.py модели «Водители» (Driver) и «Отзывы» (Comment). При добавлении водителя или отзыва, к заполнению предлагаются те поля, которые были определены в модели (рисунок 43).

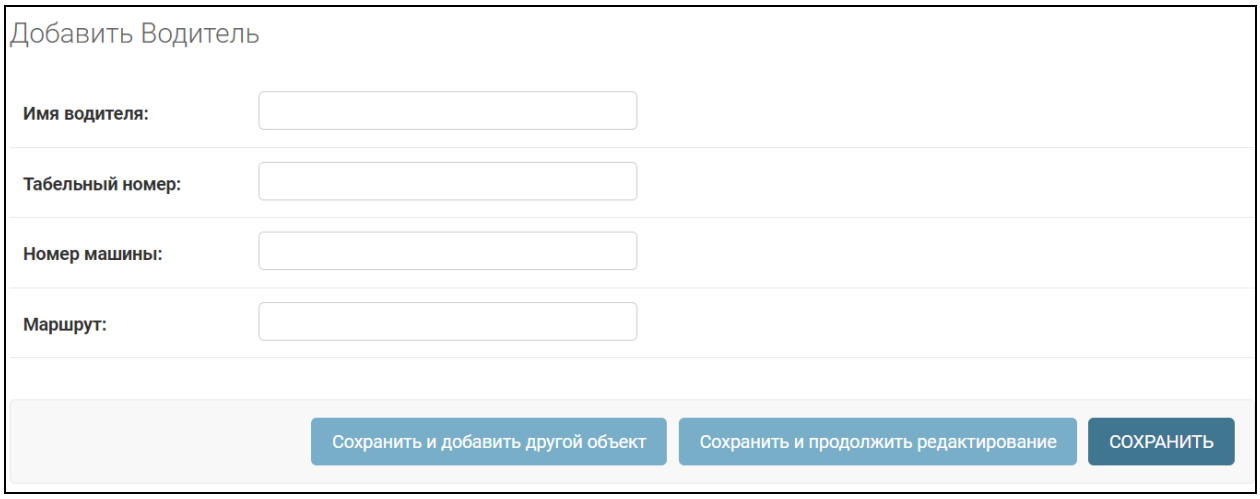

Рисунок 43 – Меню добавления водителя

На рисунке 43 показан интерфейс добавления новых водителей в систему. Также можно посмотреть список водителей (рисунок 44).

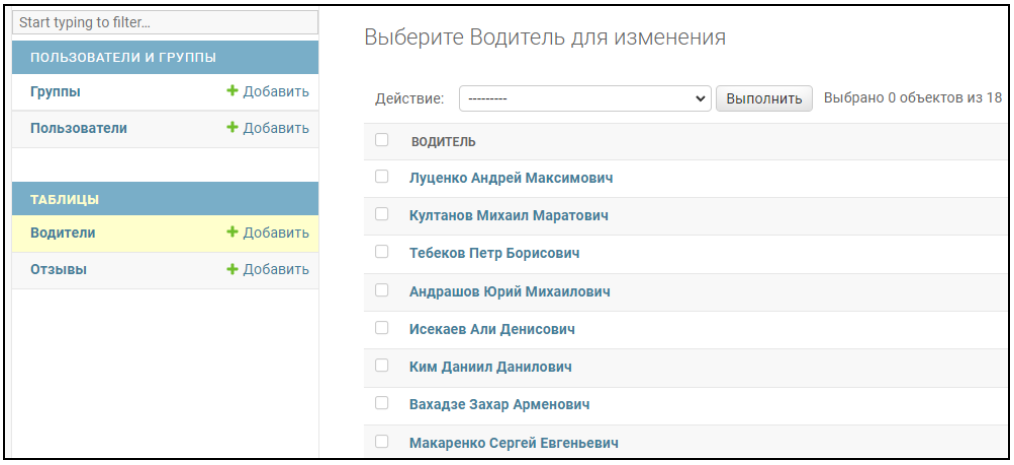

Рисунок 44 – Список водителей

На рисунке 44 отображен список водителей. Каждую позицию можно удалить (рисунок 45).

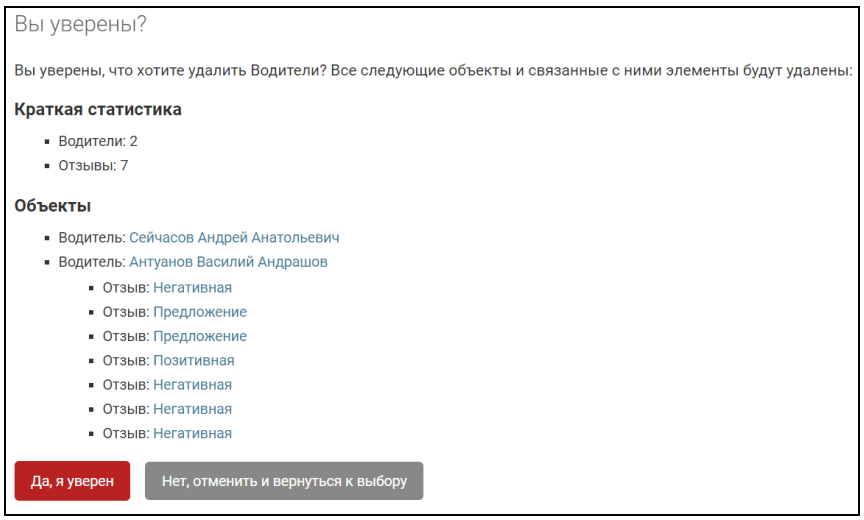

Рисунок 45 – Процесс удаления позиций из таблицы

На рисунке 45 видно, что вместе с водителями удалятся и принадлежащие им отзывы. Это реализовано с помощью внешнего ключа, о котором говорилось в пункте 3.8 в описании рисунка 35. После удаления или добавления данных в админ-панели информация дублируется в БД.

3.10 Оценка эффективности приложения

В период с 01.05.2022 по 07.05.2022 в автобусах ООО «Новомосковский Автоплюс» были развешены QR-коды, просканировав которые, пассажиры могли написать отзыв или мнение об обслуживании (рисунок 46).

На рисунке 46 изображен QR-код, который ведет на страницу водителя Федора К. С. Для перехода по ссылке достаточно просканировать код приложением для сканирования QR-кодов. На некоторых смартфонах даже не нужно устанавливать приложение: системная камера автоматически определяет код и предлагает перейти по ссылке.

Рисунок 46 – QR-код, по которому можно открыть приложение «Контролер»

Коды были развешены в 10 автобусах на инфостенде, которым оснащен каждый ГАЗ-А63R42. За это время система приняла 32 обращения. Из которых – 11 позитивной и 14 негативной коннотации. Также было получено 7 предложений (рисунок 47).

id	comment_connotation	comment_reason	comment_date	driver_id
1	Позитивная	Вежливость	2022-05-01	8
2	Позитивная	Аккуратное вождение	2022-05-01	8
3	Негативная	Отсутствие маски	2022-05-01	3
4	Предложение	Почистите салон	2022-05-02	7
5	Негативная	Нарушение ПДД	2022-05-02	10
6	Негативная	Грубость	2022-05-02	1
7	Негативная	Грубость	2022-05-02	1
8	Негативная	Грубость	2022-05-02	1
9	Предложение	Запретите тормозить не на остановках	2022-05-03	4
10	Негативная	Нарушение ПДД	2022-05-03	2
11	Позитивная	Вежливость	2022-05-03	9
12	Негативная	Отсутствие маски	2022-05-03	3
13	Позитивная	Вежливость	2022-05-04	5
14	Негативная	Нарушение ПДД	2022-05-04	7
15	Предложение	Водитель переписывается	2022-05-04	6
16	Негативная	Грубость	2022-05-05	1
17	Негативная	Нарушение ПДД	2022-05-05	10

Рисунок 47 – Список полученных отзывов за 7 дней тестового внедрения

На рисунке 47 отзывы в базе данных. В админ-панели результаты выглядят похожим образом (рисунок 48).

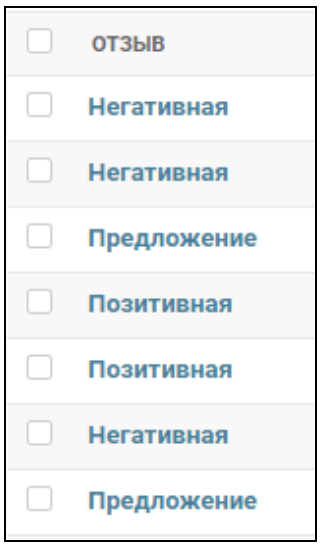

Рисунок 48 – Список отзывов в админ-панели

На рисунке 48 заметны коннотации отзывов. В системе отзывы отображаются именно по коннотациям по причине установки геттера на атрибут comment_connotation. По нажатию на отзыв можно посмотреть данные водителя и детали отзыва (рисунок 49).

На рисунке 49 видно водителя, на которого оставили отзыв, коннотацию, причину и дату отзыва. В качестве коннотации отзыва типа «Предложение» используется формулировка пассажира.

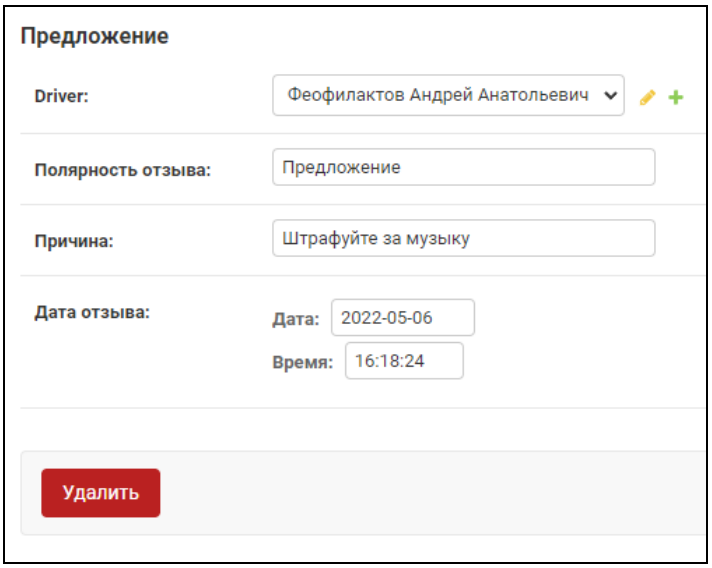

Рисунок 49 – Просмотр конкретного отзыва

За время тестового внедрения было получено 32 отзыва. 14 были негативной окраски, из которых 5 говорят о нарушениях ПДД.

Основываясь на отзывах, руководство предприятия может провести разъяснительную беседу с замеченными за нарушениями водителями, а затем проконтролировать поступление новых отзывов на нарушителя.

На основе полученных данных можно сделать вывод, что «Контролер» помог выявить случаи ненадлежащего сервиса и поведения на дороге, что в дальнейшем могло бы привести к несчастным случаям и репутационным потерям.

Выводы по главе 3

В третьей главе были рассмотрены инструменты разработки «Контролера»: язык программирования Python, веб-фреймворк Django, другое аппаратное и программное обеспечение.

После обзора инструментария был произведен обзор файлов проекта и описано начало работы: создание проекта, модулей, представлений системы, шаблонов, моделей и логики.

После обзора со стороны разработчика и пассажира была рассмотрена админ-панель приложения. В завершении главы даны результаты тестового внедрения программного продукта.

Заключение

В ходе написания выпускной квалификационной работы был реализован программный продукт «Контролер», который облегчает работу автотранспортного предприятия, а именно – автоматизирует получения обратной связи и позволяет клиентам менее чем за минуту оставить отзыв на водителей общественного транспорта быстрым выбором вариантов ответа без заполнения форм, указаний ФИО и остальных подробностей.

Перед началом работы была проанализирована деятельность предприятия, смоделированы основные бизнес-процессы в нотации IDEF0. Далее были составлены модели «Как должно быть», и на их основе начата разработка программного продукта: логическое моделирование и программная реализация веб-приложения.

В качестве технологии был выбран язык программирования Python с веб-фреймворком Django в качестве бэкенда. Полученные данные хранятся с помощью SQLite. Такая технология разработки позволяет быстро модифицировать приложение под нужды руководства (добавлять и удалять водителей или пользователей, менять содержание приложения, расширять функционал).

Итогом работы стало веб-приложение, загруженное на хостинг PythonAnywhere и доступное по адресу https://controler.pythonanywhere.com (откроется инструкция, для отправки отзыва нужно перейти по ссылке конкретного водителя, например, https://controler.pythonanywhere.com/drivers/9). Управление пользователями системы, водителями и полученными отзывами осуществляется через админ-панель приложения. Исходя из реализации программного продукта, можно сделать вывод, что цель «сделать сбор отзывов легким, быстрым и эффективным путем создания мобильной платформы, которая позволит оставлять мнение об услугах за 60 секунд без звонков и заполнения форм» можно считать достигнутой.

Список используемой литературы и используемых источников

1. А. О. Блинов. Реинжиниринг бизнес-процессов : учебное пособие для студентов вузов, обучающихся по специальностям экономики и управления. М. : ЮНИТИ-ДАНА, 2015. 343 с.
2. Барлоу Д, Меллер К. Жалоба – это подарок. Как сохранить лояльность клиентов в сложных ситуациях. М. : Олимп-Бизнес, 2018. 352 с.
3. Бахтизин В. В. Методология функционального проектирования IDEF0 : учеб. пособие для студентов. Минск : БГУ, 2003. 24 с.
4. Боргардт Е. А. Автотранспортное предприятие: экономика и управление : учеб. пособие для студентов специальности 190601. Тольятти : ТГУ, 2011. 155 с.
5. В.П. Поляков. Экономическая информатика. М. : Юрайт, 2019. 49 с.
6. Гринберг М. Разработка веб-приложений с использованием Flask. М. : ДМК Пресс, 2016. 272 с.
7. Дисциплинарные взыскания [Электронный ресурс] : статья 192 Гражданского кодекса Российской Федерации. URL: http://www.consultant.ru/document/cons_doc_LAW_34683/3a3bad3e8cac339021393236fd85d5a46a357735/ (дата обращения 01.02.2022).
8. Климентова, А. А. Влияние стиля вождения на эксплуатацию автомобильного транспорта // Молодой ученый. 2020. № 15. С. 111-113.
9. Кулаков К. А., Димитров В. М. Архитектура и фрейворки веб-приложений : учеб. пособие. СПб : ПетрГУ, 2020. 59 с.
10. Мазур В. В. Управление персоналом. Инновационные технологии : учебное пособие. М. : Дашков и К, 2022. 82 с.
11. Регламент медосмотра [Электронный ресурс] : приказ Минздрава №845н. URL: https://minzdrav.gov.ru/documents/8790-prikaz-ministerstva-zdravoohraneniya-rossiyskoy-federatsii-ot-9-noyabrya-2012-g-845n-ob-utverzhdenii-standarta-spetsializirovannoy-meditsinskoy-pomoschi-muzhchinam-pri-gipogonadizme/ (дата обращения: 30.04.2022).

12. Регламент перевозки пассажиров [Электронный ресурс] : Гражданский кодекс Российской федерации от 26.01.1996 N 14-ФЗ. URL: http://www.consultant.ru/document/cons_doc_LAW_9027/84989d331874be7730e1c99b836fbe639a8efe6d/ (дата обращения: 16.04.2022).

13. Регламент техосмотра [Электронный ресурс] : приказ Минтранса №9. URL: http://www.consultant.ru/document/cons_doc_LAW_385069/ (дата обращения: 30.04.2022).

14. Стандарт заполнения технического задания [Электронный ресурс] : ГОСТ 19.201-78. URL: https://docs.cntd.ru/document/1200007648/ (дата обращения: 01.05.2022).

15. Устав автомобильного транспорта и городского наземного электрического транспорта [Электронный ресурс] : Федеральный закон от 08.11.2007 N 259-ФЗ (последняя редакция). URL: http://www.consultant.ru/document/cons_doc_LAW_72388/.

16. Exploring The Corporate Image [Электронный ресурс]. URL: https://eprints.mdx.ac.uk/18972/1/ExploringTheCorporateImageFormationProcess_1.pdf (дата обращения: 26.05.2022).

17. Python web development [Электронный ресурс]. URL: https://realpython.com/tutorials/web-dev/ (дата обращения: 26.05.2022).

18. What is business process management? The key to enterprise agility [Электронный ресурс]. URL: https://www.cio.com/article/230560/what-is-business-process-management-bpm-the-key-to-enterprise-agility.html.

19. What is customer service: Definition, types, benefits, stats. The key to enterprise agility [Электронный ресурс]. URL: https://www.the-future-of-commerce.com/2021/08/02/what-is-customer-service-definition-examples/ (дата обращения: 26.05.2022).

20. Your Guide to Business Process Reengineering [Электронный ресурс]. URL: https://onlinebusiness.northeastern.edu/blog/what-is-business-process-reengineering/ (дата обращения: 26.05.2022).

Приложение А

Техническое задание на разработку веб-приложения «Контролер»

1 Введение

1.2 Рабочее название

На время разработки и тестирования наименование программы – «Контролер».

1.3 Характеристика области применения

ПО «Контролер» предназначено для клиентов АТП, которые находятся в пути на автобусах компании, и для работников АТП, которые контролируют обработку возражений клиентов.

2. Основание разработки

Основанием разработки является договор на прохождение практики в ООО «Новомосковский Автоплюс» и задание на выполнение ВКР.

2.1 Назначение разработки

Приложение будет применяться клиентами АТП во время поездки на автобусах компании и сотрудниками – во время контроля обратной связи.

3 Функциональное значение

Основная функция для пассажиров – предоставление возможности быстро оставить отзыв, жалобу, предложение относительно оказания услуг.

Основная функция для работников АТП – возможность быстро проверить наличие жалоб.

3.1 Эксплуатационное значение

Приложение должно предоставлять пассажирам возможность оставить отзыв на конкретного водителя и машину после сканирования QR-кода в салоне автобуса. Доступ к базе отзывов должен иметь директор и его заместитель через админ-панель.

4 Требования к ПО

4.1 Требования к функциональности

4.1.1 Требования к исполняемым функциям

После сканирования кода в браузере смартфона показывается главный экран веб-приложения. В каждой машине должен висеть уникальный QR-код, который ведет на уникальную страницу водителя и автомобиля. Отзывы также должны записываться на конкретного сотрудника. Пользователь может:

- пожаловаться;
- написать отзыв;
- выразить положительную оценку.

Интерфейс должен требовать минимум действий от пользователя. Поэтому в большинстве случаев клиенту будут предлагаться пункты выбора, а не поле ввода. Например, на главном экране должны быть три кнопки: «Похвалить», «Поругать», «Предложить». Только после нажатия кнопки «Предложить» должно открываться поле ввода. В остальных случаях оценка должна ставиться простым выбором пунктов:

– «Поругать»: водитель грубит, не носит маску, нарушает ПДД;

– «Похвалить»: в салоне чисто, водитель вежлив, быстро доехали.

После принятия обращения клиент должен увидеть сообщение: благодарность за отзыв или обещание принять меры;

4.1.2 Требования к организации входных и выходных данных

Отзывы хранятся в БД SQL в следующих полях:

– коннотация оценки: положительная, отрицательная, предложение;

– номер машины;

– ФИО водителя;

– дата отзыва.

В админ-панели должна быть возможность добавить или удалить водителя. В поле «Отзывы» должны быть указаны все отзывы с информацией о том, кто, когда и по какой причине его получил.

4.1.3 Временные характеристики работы

Обратная связь поступает в БД сразу после отправки клиентом.

4.2 Надежность

Доля безотказной работы должна составлять не менее 99%.

4.2.1 Требования к обеспечению надежности

Данные об обратной связи должны храниться у хостинг-провайдера. Поэтому необходимо обеспечить резервное копирование данных приложения.

4.2.2 Восстановление после сбоя

Восстановление должно происходить восстановлением резервных копий и не должно превышать времени, требуемого на разворачивание бэкапа.

4.2.3 Сбои из-за действий пользователя

Приложение не должно отказывать из-за любых действий пользователя.

4.3 Условия эксплуатации

Приложение эксплуатируется в двух видах: на компьютере руководителя для просмотра обратной связи и на смартфонах клиентов во время оформления обратной связи.

4.3.1 Климатические условия эксплуатации

Специальных условий не выдвигается.

4.3.2 Обслуживание ПО

Приложение не должно требовать обслуживания.

4.3.3 Требования к персоналу

Для расширения функциональности приложения и процессе резервного копирования требуется 1 системный администратор. В процессе эксплуатации приложения сотрудников, кроме пользователя, требоваться не должно.

4.4 Системные требования

Приложение не должно предъявлять к технических средствам требования, сверх указанных:

- для админ-панели:
- оперативная память: 1 Гб;
- процессор: 32/64 бита, 1 Ггц.

Для клиентской части:

- смартфон под управлением Android 4+ или iOS 8+;
- оперативная память: 512 Мб;

– процессор: ARM v6, 1 Ггц.

4.5 Требования к программной совместимости

Должна быть возможность скачать SQL-файл БД.

4.6 Требования к упаковке

Клиент должен получать доступ к приложению путем сканирования QR-кода в автобусе, администратор – путем ввода адреса в строке браузера.

4.7 Требований к хранению

Требований нет

4.8 Спецтребования

Специальных требований нет.

5 Требования к программной документации

Перечень необходимых документов:

– техзадание.

6 Экономические показатели

ПО позволит улучшить:

– конкурентоспособность;

– лояльность клиентов;

– репутацию АТП.

Репутация улучшится благодаря сохранению информации обо всех инцидентах внутри компании. Сейчас пользователи жалуются на водителей в соцсетях, откуда журналисты берут материал для новостей. Конкурентоспособность повысится в результате выявления водителей-нарушителей. Лояльность клиентов повысится по причине улучшения клиентского сервиса.

7 Стадии разработки

Разработка ПО должна пройти 5 стадий: составление ТЗ, моделирование предметной области, разработка, испытания, внедрение.

8 Порядок приемки

Критерием приемки является прохождение тестового внедрения.

9 Источники информации

- ГОСТ 19.201-78;
- ГОСТ 24.701-86.

Приложение Б

Избранные листинги приложения «Контролер»

```python
from django.http import HttpResponse, Http404, HttpResponseRedirect
from django.shortcuts import render
from django.utils import timezone

from .models import Driver, Comment

def index(request):
    return render(request, 'index_without_drivers.html')

def  driver_personal_information(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Неправильная ссылка или водителя нет в базе данных")

    return render(request, 'index.html', {'actual_driver_obj': actual_driver_obj})

def harm(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    return render(request, 'actions/harm.html', {'actual_driver_obj': actual_driver_obj})

def praise(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    return render(request, 'actions/praise.html', {'actual_driver_obj': actual_driver_obj})

def offer(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    return render(request, 'actions/offer.html', {'actual_driver_obj': actual_driver_obj})
```

Рисунок Б.1 – Листинг файла views.py

```
def leave_harm(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    data_to_DB = actual_driver_obj.comment_set.create(comment_connotation = request.POST['connotation'], comment_reason = request.POST['reason'], comment_date = timezone.now())
    data_to_DB.save()

    return render(request, 'actions/thanks/thanks_for_harm.html', {'actual_driver_obj': actual_driver_obj})

def leave_praise(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    data_to_DB = actual_driver_obj.comment_set.create(comment_connotation = request.POST['connotation'], comment_reason = request.POST['reason'], comment_date = timezone.now())
    data_to_DB.save()

    return render(request, 'actions/thanks/thanks_for_praise.html', {'actual_driver_obj': actual_driver_obj})

def leave_offer(request, driver_id):
    try:
        actual_driver_obj = Driver.objects.get(id = driver_id)
    except:
        raise Http404("Не удалось получить объект водителя")

    data_to_DB = actual_driver_obj.comment_set.create(comment_connotation = request.POST['connotation'], comment_reason = request.POST['offer'], comment_date = timezone.now())
    data_to_DB.save()

    return render(request, 'actions/thanks/thanks_for_offer.html', {'actual_driver_obj': actual_driver_obj})
```

Рисунок Б.2 – Продолжение листинга views.py

```
from django.urls import path

from . import views

app_name = 'drivers'
urlpatterns = [
    path('', views.index_without_drivers, name = 'index_without_drivers'),
    path('<int:driver_id>/', views.driver_personal_information, name = 'driver_personal_information'),
    path('<int:driver_id>/harm/', views.harm, name = 'harm'),
    path('<int:driver_id>/praise/', views.praise, name = 'praise'),
    path('<int:driver_id>/offer/', views.offer, name = 'offer'),
    path('<int:driver_id>/leave_harm/', views.leave_harm, name='leave_harm'),
    path('<int:driver_id>/leave_praise/', views.leave_praise, name='leave_praise'),
    path('<int:driver_id>/leave_offer/', views.leave_offer, name='leave_offer'),
]
```

Рисунок Б.3 – Листинг локальной URL-привязки urls.py

```python
from django.db import models

class Driver(models.Model):
    driver_name = models.CharField('Имя водителя', max_length = 100)
    driver_number = models.CharField('Табельный номер', max_length = 7)
    car_plate = models.CharField('Номер машины', max_length = 7)
    car_route = models.CharField('Маршрут', max_length = 3, default = 140)

    def __str__(self):
        return self.driver_name

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

class Comment(models.Model):
    driver = models.ForeignKey(Driver, on_delete = models.CASCADE)
    comment_connotation = models.CharField('Полярность отзыва', max_length = 250)
    comment_reason = models.CharField('Причина', max_length = 250)
    comment_date = models.DateTimeField('Дата отзыва')

    def __str__(self):
        return self.comment_connotation

    class Meta:
        verbose_name = 'Отзыв'
        verbose_name_plural = 'Отзывы'
```

Рисунок Б.4 – Листинг моделей models.py

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('drivers/', include('drivers.urls')),
    path('admin/', admin.site.urls),
    path('', include('drivers.urls')),
]
```

Рисунок Б.5 – Листинг корневой URL-привязки urls.py

```
{% load static %}

<!DOCTYPE html>
<html lang="ru-RU">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
    <title>{% block title %} Отправить отзыв руководству компании «Автоплюс» {% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/review.css' %}">
    <link type="image/x-icon" rel="shortcut icon" href= "{% static 'media/images/favicon.ico' %}">
</head>
<body id="review">
    <div class="review-header" style="user-select: auto;">
        <div class="review-i" style="user-select: auto;">
            <div class="review-company" style="user-select: auto;">
                <div class="company-logotype" style="user-select: auto;">
                    <div class="company-logotype__i" style="user-select: auto;">
                        <a href="#" target="_blank" style="user-select: auto;"><img src="{% static 'media/images/logo.png' %}" alt="" style="user-select: auto;"></a>
                    </div>
                    </div>
                    <div class="review-meta" style="user-select: auto;">
                        <div class="review-title" style="user-select: auto;">
                            <a href="#" target="_blank" style="user-select: auto;">Информация об автобусе:</a>
                        </div>
                        <div class="review-subtitle" style="user-select: auto;">Маршрут:  {{actual_driver_obj.car_route}}</div>
                        <div class="review-subtitle" style="user-select: auto;">Номер: {{actual_driver_obj.car_plate}}</div>
                    </div>
            </div>
        </div>
    </div>

    {% block content %}{% endblock %}

    <div class="footer" style="user-select: auto;">
        <div class="footer-i" style="user-select: auto;">
            <div class="qrmap-name" align="justify" style="user-select: auto;">Анонимный сбор обратной связи «Автоплюса». Руководство читает отзывы каждый день и учитывает их при планировании работы.</di
        </div>
    </div>

</body>
</html>
```

Рисунок Б.6 – Листинг шаблона base.html

```
1   {% extends '../../base/base.html' %}
2
3   {% block title %}Поругать{% endblock %}
4
5   {% block content %}
6
7   <form id="review-form">
8       <input type="hidden" style="user-select: auto;"><div class="review-content" style="user-select: auto;">
9           <div class="review-i" style="user-select: auto;">
10              <div class="review-box" style="user-select: auto;">
11                  <div class="review-box__title" style="user-select: auto;">Простите нас за неудобства 🙏 </br></br>Мы поговорим с водителем о его поведении.</div>
12              </div>
13          </div>
14  </form>
15
16  {% endblock %}
```

Рисунок Б.7 – Листинг файла извинения за грубость водителя thanks_for_harm.html

```
{% extends 'base/base.html' %}

{% block title %}Написать отзыв{% endblock %}

{% block content %}

<input type="hidden" style="user-select: auto;">
<div class="review-content">
    <div class="review-f" style="user-select: auto;">
        <div class="review-box" style="user-select: auto;">
            <div class="review-box__title" style="user-select: auto;">Вас везет {{actual_driver_obj}}. Вы хотите...</div></br>
            <form action="{% url 'drivers:harm' actual_driver_obj.id %}" method="POST">
                <div class="form-action" style="user-select: auto;">

                    {% csrf_token %}

                    <button type="submit" class="btn btn-block " style="user-select: auto;">Поругать</button>
                </div>
            </form></br>
            <form action="{% url 'drivers:praise' actual_driver_obj.id %}" method="POST">
                <div class="form-action" style="user-select: auto;">

                    {% csrf_token %}

                    <button type="submit" class="btn btn-block" style="user-select: auto;">Похвалить</button>
                </div>
            </form></br>
            <form action="{% url 'drivers:offer' actual_driver_obj.id %}" method="POST">
                <div class="form-action" style="user-select: auto;">

                    {% csrf_token %}

                    <button type="submit" class="btn btn-block" style="user-select: auto;">Предложить</button>
                </div>
            </form></br>
            <div class="review-box__subtitle" style="user-select: auto;">Выберите действие, чтобы продолжить</div>
        </div>
    </div>
</div>

{% endblock %}
```

Рисунок Б.8 – Листинг «домашней» страницы водителя index.html