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 кафедры/департамента/центра полностью)

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

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка автоматизированной системы учета заявок по ремонту автомобилей в СТО»

Обучающийся В.В. Коврижкин

(Инициалы Фамилия) (личная подпись)

Руководитель В.В. Дружинкин

(ученая степень (при наличии), ученое звание (при наличии), (Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема выпускной квалификационной работы: «Разработка автоматизированной системы учета заявок по ремонту автомобилей в СТО».

Целью выпускной квалификационной работы является автоматизация учёта заявок в ООО «Кувалда-центр», что позволит обеспечить лёгкость и быстроту обработки заказов, а также оформлении и получении информации о заказах клиентов.

Объектом исследования является процесс оказания услуг клиентам в ООО «Кувалда-центр».

В первом разделе описана деятельность ООО «Кувалда-центр». Проведён анализ бизнес-процессов предприятия с описанием функционирования системы в целом в виде контекстной диаграммы. Рассмотрены основные функции и задачи менеджера автосервиса. Приведена схема информационной сети предприятия.

Во втором разделе проводится логическое проектирование приложения. Приводятся диаграмма прецедентов и диаграмма классов. Описываются концептуальная и логическая модели, описана структура базы данных. Составлены требования к аппаратному и программному обеспечению информационной системы, обоснован выбор СУБД и средств разработки.

В третьем разделе описан процесс разработки информационной системы, описывается функционал приложения, приводятся элементы интерфейса, отличающиеся, в зависимости от прав пользователей.

Заключение содержит выводы проделанной работы, описываются решеные задачи и возможные пути развития приложения.

Содержание

Введение

В современном мире компьютеры применяются практически во всех сферах деятельности человека. Большинство организаций уже не могут обходится без использования компьютеров в своей работе, которые с успехом заменяют рутинную работу, ранее выполнявшуюся вручную, повышая эффективность работы любой организации.

На сегодняшний день сфера услуг является одной из важнейших отраслей экономики. Сфера услуг как отрасль экономической деятельности представляет собой совокупность организаций, цель которых – оказание разнообразных видов платных услуг населению. При этом сфера услуг решает разнообразные важные социальные задачи, а их значение в жизни общества постоянно растёт. Одним из важных, и востребованных видов услуг является оказание услуг по ремонту и обслуживанию автомобилей.

В автосервисе предоставляются услуги по плановому техническому обслуживанию, компьютерной диагностики неисправностей, устранению текущих неисправностей, дооснащению и капитальному ремонту.

Задачей учёта заказов на обслуживание в автосервисе является не только их учёт, но и контроль их выполнения. Многие заказы на обслуживание выполняются в течении дня, а то и в течении нескольких часов, поэтому при возрастающем количестве клиентов и заказов становиться трудно контролировать работы в автосервисе, их сроки и историю. Для решения проблем управления, контроля, повышения качества обслуживания, и уменьшения времени на обслуживание клиента, требуется использовать современные информационные технологии, основанные на применении автоматизированных информационных систем.

База данных является наиболее распространённым решением для хранения, поиска и систематизации данных о клиентах и заказов на обслуживание.

Разрабатываемое приложение осуществляет информационную поддержку работы автосервиса, и предназначается для организации учёта клиентов, их обращений, произведенных работах. С помощью приложения менеджеры автосервиса смогут хранить сведения о клиентах и их автомобилях, формировать заявки, контролировать их выполнение и формировать различные отчёты.

В данной работе необходимо разработать приложение по автоматизации работы автосервиса. На начальном этапе разработки приложения необходимо изучить, анализировать и моделировать деятельность автосервиса для возможного улучшения или оптимизации методов работы.

Цель выпускной квалификационной работы: разработать приложение по учёту клиентов и заявок на обслуживание клиентов автосервиса.

Для достижения цели потребуется решить следующие задачи:

- изучить функциональную структуру автосервиса «Кувалда-центр»;
- произвести анализ бизнес-процессов автосервиса «Кувалда-центр»;
- подобрать систему управления базами данных;
- разработать структуру базы данных по учёту заявок и клиентов;
- разработать интерфейс приложения.

1 Функциональное моделирование предметной области

1.1 Технико–экономическая характеристика компании

Типовое предприятие автосервиса организационно состоит из нескольких служб и подразделений. Как правило это подразделение управлением автосервисом или дирекция, служба организации ремонтов и техобслуживания, логистики, маркетинга, запасных частей. Дирекция автосервиса делится на непосредственного руководителя автосервисом - генерального директора, и его прямых подчинённых, технического и коммерческого директоров.

Для качественного выполнения своих функций в структуре автосервиса чётко определены взаимосвязи между всеми службами и подразделениями.

Организационная структура ООО «Кувалда-центр», представлена на рисунке 1.

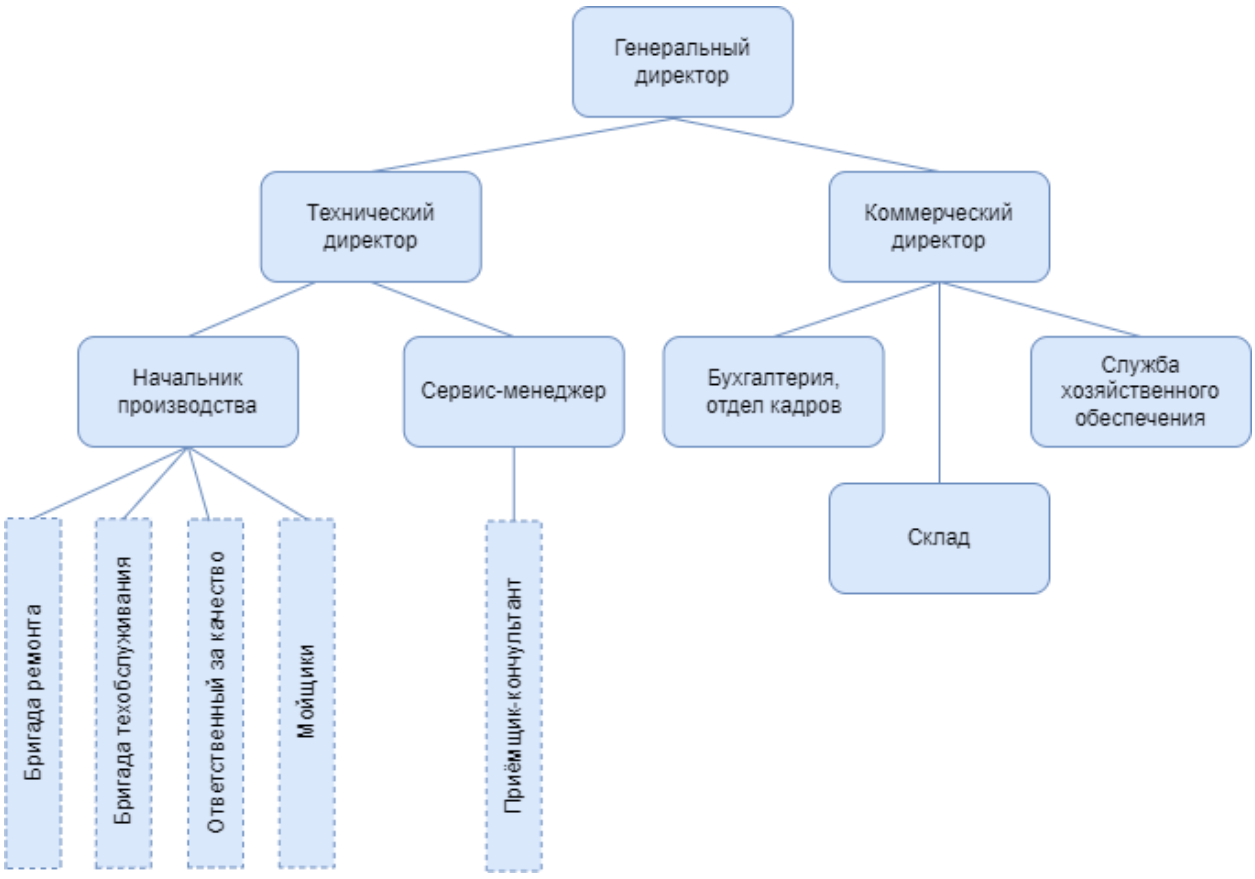


Рисунок 1 - Организационная структура СТО

Генеральный директор осуществляет стратегическое планирование, повышение конкурентоспособности, повышение репутации предприятия, привлекает инвестиции, анализирует ситуацию на рынке автосервисов, анализирует результаты производственной и финансовой деятельности автосервиса, разрабатывает программы развития предприятия, разрабатывает или утверждает долгосрочные планы по удовлетворению будущих потребностей персонала, утверждает организационную структуру и штатное расписание, подписывает в качестве прямого распорядителя договоры с контрагентами и финансовые документы.

Технический директор разрабатывает и внедряет в автосервисе техническую политику, разрабатывает программы качества и работе с рекламациями, разрабатывает инструкции по технике безопасности, пожарной безопасности, охране труда, использованию оборудования, инструментов и средств индивидуальной защиты, и выполняет контроль за их соблюдением, контролирует соблюдение производственной и трудовой дисциплины, контролирует расход материально-технических ресурсов.

Коммерческий директор разрабатывает и организует реализацию планов материально-технического обеспечения, разрабатывает программы развития сервиса, проверяет ежемесячные сводки непродуктивных затрат, и разрабатывает меры по их сокращению; заключает и выполняет договора с поставщиками; руководит работой магазина и склада запчастей, обеспечивает их сохранность.

Приёмка автомобиля на обслуживание может быть решающей для будущих отношений с клиентом. Если СТО серьёзно разочарует заказчика, то вероятнее всего лишится в его лице постоянного клиента. Поэтому необходимо с особым вниманием относится к встрече клиента, его пожеланиям, проблемам и ожиданиям.

Рассмотрим основные функции служб и подразделений автосервиса.

Служба организации технического обслуживания обеспечивает ремонтный персонал всей необходимой документацией по проведению

ремонтов автомобилей, разрабатывает технологические карты по гарантийным ремонтам, контролирует соблюдение технологий ремонта и обслуживания, подготавливает должностные инструкции ремонтного персонала, и программы стажировки для вновь принятых сотрудников.

Ремонтный цех непосредственно выполняет работы по ремонту и обслуживанию автомобилей клиентов, включая механический ремонт, кузовной, ремонт электрооборудования, устанавливает дополнительное оборудование, осуществляет тюнинг и установку дополнительного оборудования, организует работу выездных бригад, осуществляет эвакуацию неисправных автомобилей, обслуживает парк техники самого автосервиса.

Служба хозяйственного обеспечения организует содержание склада запчастей, осуществляет розничную торговлю запчастями, осуществляет учёт и контроль движения товаров, проводит инвентаризацию, анализирует цены конкурентов, контролирует изменение номенклатуры запчастей контрагентами, оформляет счета, накладные и сопроводительные документы, контролирует эффективное использование объёма склада.

Техническое оснащение СТО включает в себя:

- подъёмник, грузоподъёмностью 5т.;
- комбинированная установка для слива/откачки масла;
- пневмоинструмент;
- тестер проверки подвески и амортизаторов;
- сканер ошибок бортового компьютера;
- электронный тестер для аккумуляторных батарей;
- установка для промывки и экспресс-замены жидкости в акпп;
- колонка воздухораздаточная;
- установка для промывки системы охлаждения и экспресс-замены охлаждающей жидкости;
- нагнетатель пластичных смазок;
- установка для обслуживания топливной системы;

– газоанализатор;
– установка для обслуживания кондиционеров.

Кроме этого, центральная диагностическая стойка с компьютером и принтером.

Программная архитектура информационной системы ООО «Кардан-центр» включает следующее основное программное обеспечение:

– операционная система Windows 7;
– Microsoft Office 2007;
– браузеры Opera, Mozilla Firefox, Chrome;
– штатный антивирус Windows Defender;
– платформа 1С:Предприятие (конфигурации Бухгалтерия, Управление персоналом, Управление складом).

Информационная составляющая компании представляет собой совокупность следующих технических средств:

По 2 ПК в СТО, 5 ПК в головном офисе, там же 1 выделенный сервер;

Компьютерная сеть построена по технологии Ethernet 100BASE-T на оборудовании Mikrotik, информационный обмен между СТО и головным офисом осуществляется с помощью технологии vpn, где сервером является маршрутизатор в головном офисе, а клиентами являются маршрутизаторы в СТО. Схема сети представлена на рисунке 2.

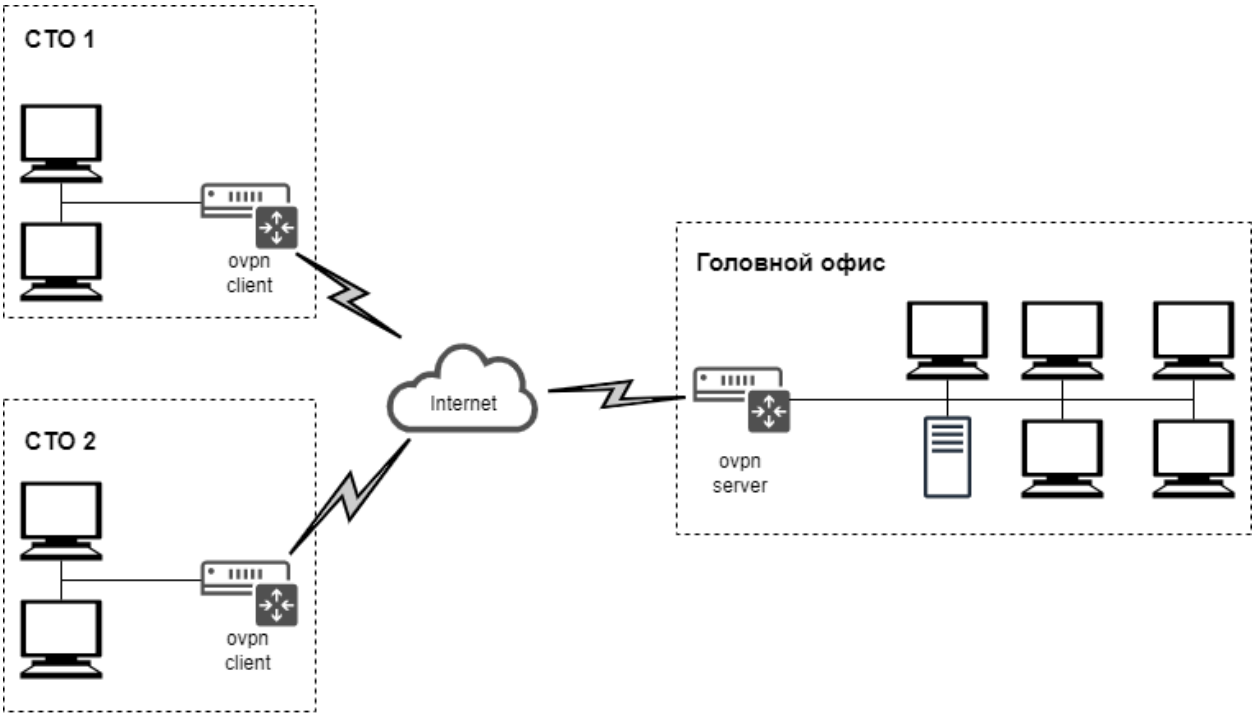


Рисунок 2 – Схема информационной сети СТО

По одному МФУ в СТО и 2 в головном офисе.

Для клиентов СТО доступна гостевая сеть WiFi;

В качестве аппаратной основы сервера используется обычный ПК.

Все ПК закуплены одновременно, и имеют следующую конфигурацию: процессор Intel Core i5 3330, оперативная память 8ГБ, жесткий диск 500ГБ, сетевая карта 100Мбит/сек.

1.2 Концептуальное моделирование предметной области

Целью построения функциональных моделей обычно является выявление наиболее слабых и уязвимых мест деятельности организации, анализ преимуществ новых бизнес-процессов и степени изменения существующей структуры организации деятельности. Анализ недостатков начинают с построения модели «Как есть», т.е. модели существующей организации работы.

Построение модели информационной системы начинается с описания функционирования системы в целом в виде контекстной диаграммы. Контекстную диаграмму процессов, протекающих в СТО можно представить в виде диаграмм IDEF0. Контекстная диаграмма представлена на рисунке 3.

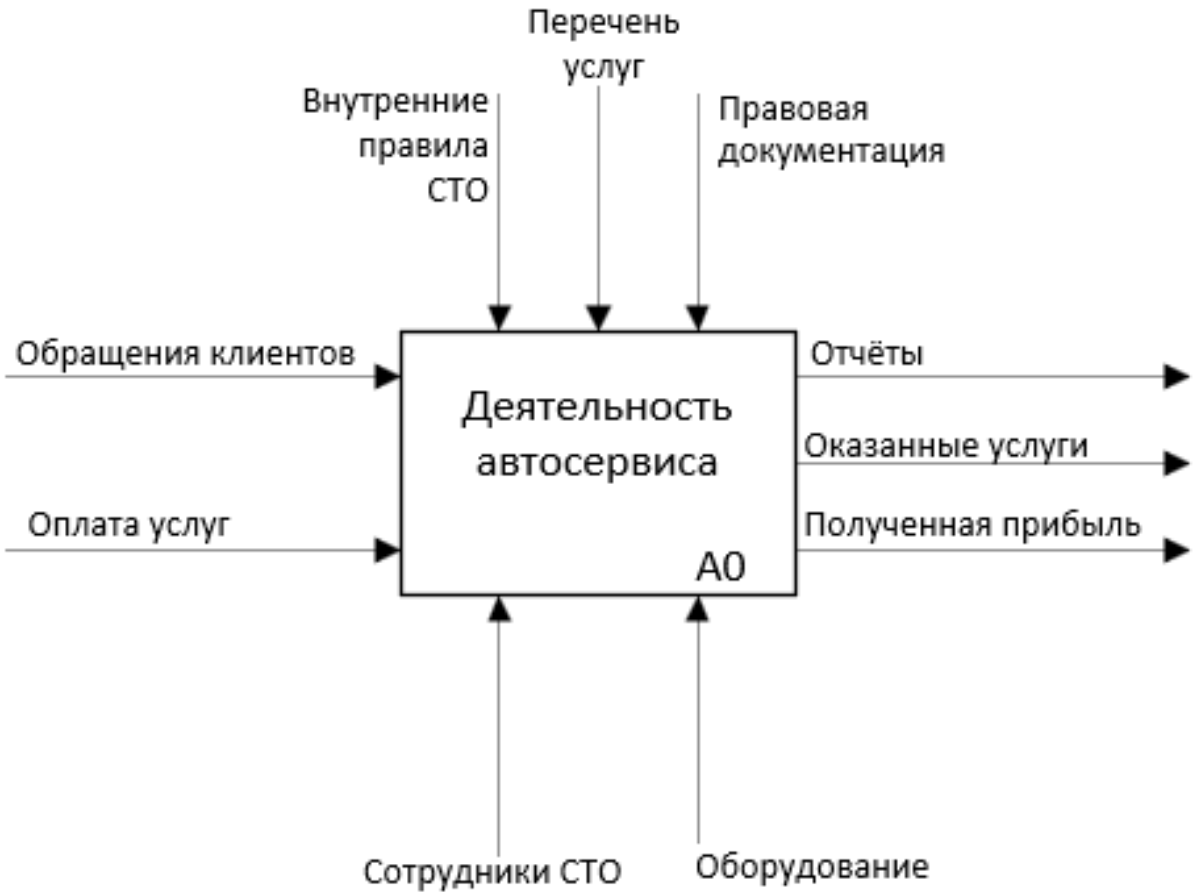


Рисунок 3 – Контекстная диаграмма «Деятельность автосервиса»

Декомпозиция – дробление большой и сложной системы на несколько более примитивных, вложенных систем. Такая форма представления позволяет анализировать процесс, не перегружая представление элементами, излишними для решения текущей задачи. Глубина декомпозиции определяется целями моделирования и, таким образом, задает степень детализации описания процесса. Уровень детализации описания отдельного

бизнес-процесса диктуется необходимостью обеспечить качество понимания бизнес-процесса.

Выполним декомпозицию процесса «Деятельность автосервиса».

Декомпозированная контекстная диаграмма включает следующие подпроцессы: приём заявки от клиента, проведение диагностики авто, выполнение работ, оплата услуг.

Декомпозиция диаграммы «Деятельность автосервиса» представленна на рисунке 4.

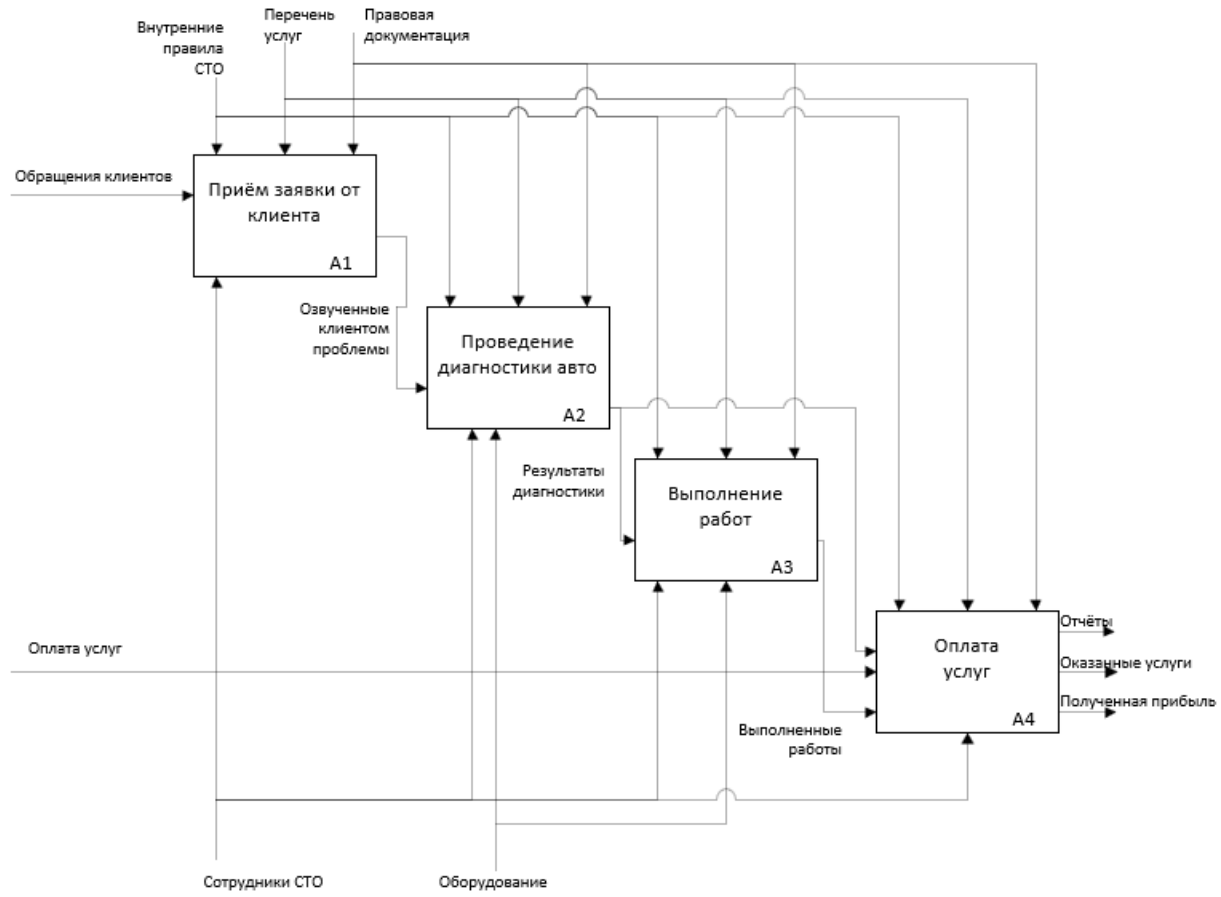


Рисунок 4 – Декомпозиция диаграммы «Деятельность автосервиса»

Если какой-либо шаг процесса при данном уровне детализации остается непонятным, детализацию описания повышают.

Выполним декомпозицию процесса «Приём заявки от клиента».

Декомпозированная контекстная диаграмма включает следующие подпроцессы: регистрация клиента, оформление заказа, расчёт стоимости, подтверждение заказа. Декомпозиция диаграммы «Приём заявки от клиента» представлена на рисунке 5.

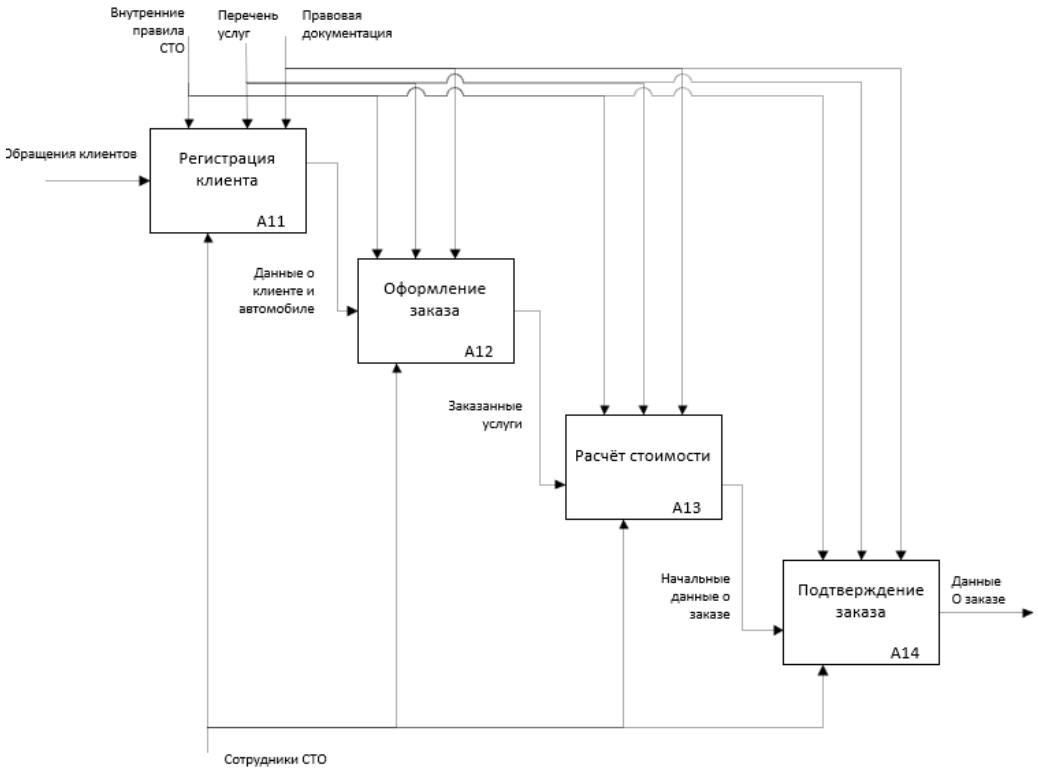


Рисунок 5 – Декомпозиция диаграммы «Приём заявки от клиента»

Перед началом выполнения заказа клиент, или менеджер СТО со слов клиента, должны заполнить формуляр с указанием данных клиента (ФИО, номер телефона), данных автомобиля (гос. номер, марка, модель, цвет), и выбрать услуги. Далее менеджер заполняет электронную таблицу, производит расчёт примерных сроков и стоимости выполнения заказа. В конце клиент подтверждает точность и полноту заказа.

1.4 Постановка задачи на разработку проекта аис

Для оптимизации работы менеджеров автосервиса и сокращения времени на обслуживание клиентов необходимо разработать информационную систему, которая будет учитывать бизнес-процессы автосервиса, и не будет им противоречить. Планируемые функции разрабатываемой системы:

- Оптимизация работы с клиентами, сбор различной информации. Полная информация о автомобилях и клиентах хранятся в одной базе данных, анализ которой позволит принимать эффективные решения;
- Учёт результатов выполнения заказа по обслуживанию (средний чек, наиболее востребованная услуга) поможет принимать дальнейшие управленческие решения и планировать рекламные кампании;
- Учёт времени работы мастеров автосервиса позволит выявить «узкие места» при обслуживании клиентов, оптимизировать ресурсы и расходы.

В процессе разработки приложения необходимо соблюдать следующие правила:

- Единая база данных с информацией о клиентах, автомобилях, заявках, услугах, сотрудниках СТО;
- Учёт и контроль за информацией, которая предоставляется разными категориями пользователей.

1.5 Разработка модели «как должно быть»

Анализ бизнес-процесса показал достаточно высокую загруженность менеджеров СТО при работе с клиентами, без использования информационно-коммуникационных систем эти процессы выполняются медленно и

менеджерами, зачастую, допускаются ошибки, устранить которые можно при помощи использования соответствующих информационных технологий в виде информационной системы.

Предлагается к ресурсам бизнес-процесса добавить автоматизированную информационную систему. После реинжиниринга процесс будет выглядеть следующим образом (рисунок 6).

Рисунок 6 – Контекстная диаграмма «Деятельность автосервиса (Как должно быть)»

Таким образом, в существующую модель обработки бизнес-процессов автосервиса был добавлен механизм «Информационная система», отражающий необходимость использования информационных технологий в

деятельности автосервиса. Внедрение в работу информационной системы позволит автоматизировать деятельность менеджеров автосервиса без изменения существующей технологии работы менеджеров с клиентами, только оптимизировав все бумажные работы.

После автоматизации бизнес-процесса «Деятельность автосервиса» декомпозиция процесса будет выглядеть следующим образом (рисунок 7).

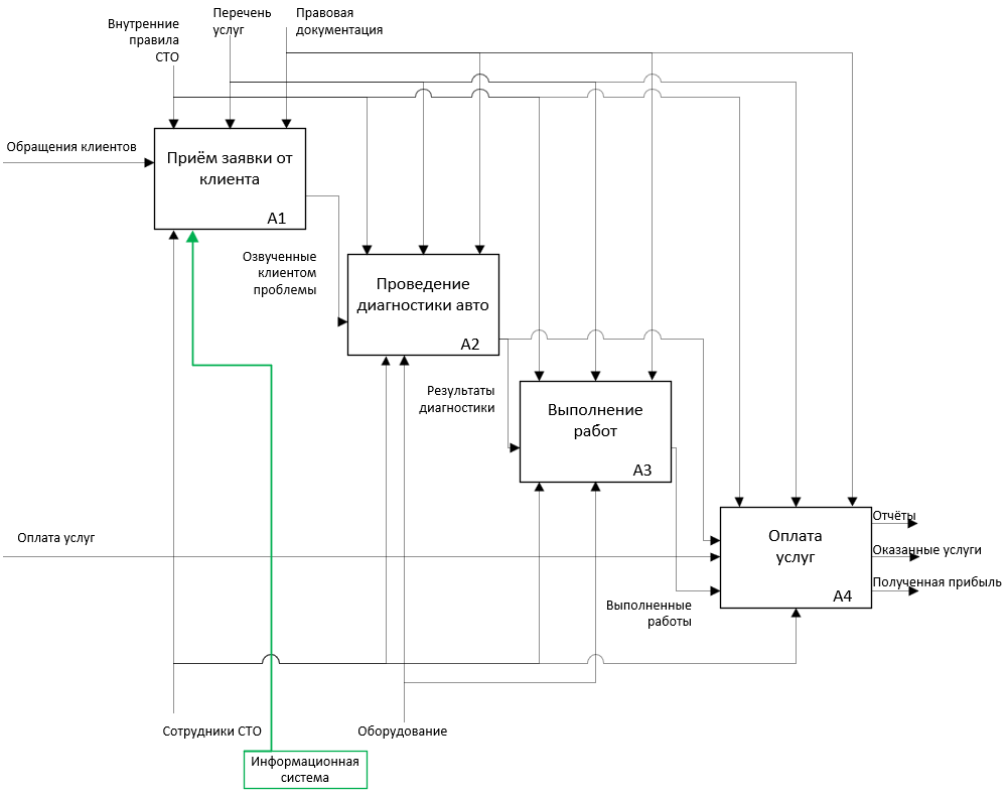


Рисунок 7 – Декомпозиция контекстной диаграммы «Деятельность автосервиса (Как должно быть)»

После автоматизации бизнес-процесса «Приём заявки от клиента» декомпозиция процесса будет выглядеть следующим образом (рисунок 8).

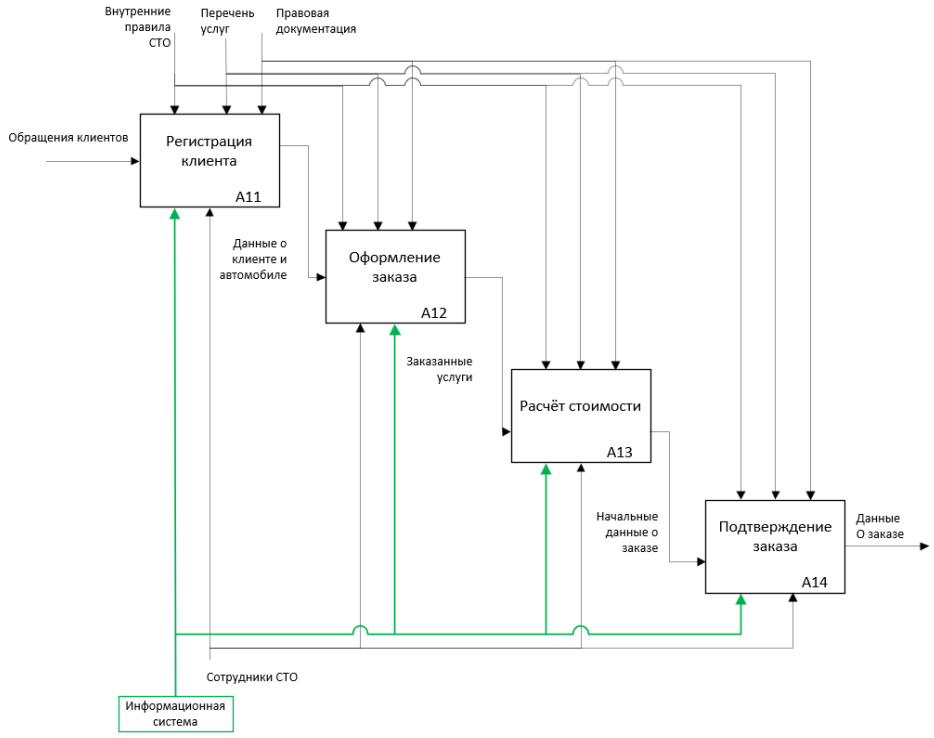


Рисунок 8 – Декомпозиция контекстной диаграммы «Приём заявки от клиента (Как должно быть)»

Важными составляющими после внедрения информационной системы станут:

– упрощение регистрации клиента;
– упрощение учёта заявок на обслуживание;
– частичный отказ от бумажных носителей;
– возможность ведения статистики по ремонтам;
– возможность ведения статистики по обращениям клиентов.

В целом использование информационной системы позволит сократить время на обслуживание клиентов, и оптимизирует работу менеджеров автосервиса, что в свою очередь сокращает издержки автосервиса, повышает конкурентоспособность и как следствие ведёт к увеличению прибыльности предприятия.

Выводы

В данном разделе была исследована деятельность автосервиса, проведен детальный анализ предметной области. На основании исследования установлены бизнес-процессы, препятствующие эффективной работе автосервиса и подлежащие автоматизации с помощью информационной системы, собраны необходимые требования к разрабатываемой информационной системе, осуществлено бизнес-моделирование процессов, разработаны контекстные диаграммы до внедрения информационной системы («Как есть»), и после её внедрения («Как должно быть»), выделены важные составляющие будущей информационной системы, её преимущества после внедрения на предприятии, что позволило приступить к следующей стадии разработки.

2 Логическое проектирование приложения

2.1 Диаграмма прецедентов

Диаграмма прецедентов демонстрирует основные функции информационной системы «Учёта заявок в автосервисе». Из рисунка 9 видно, что на диаграмме прецедентов идёт взаимодействие между тремя актёрами – клиентом, менеджером и администратором.

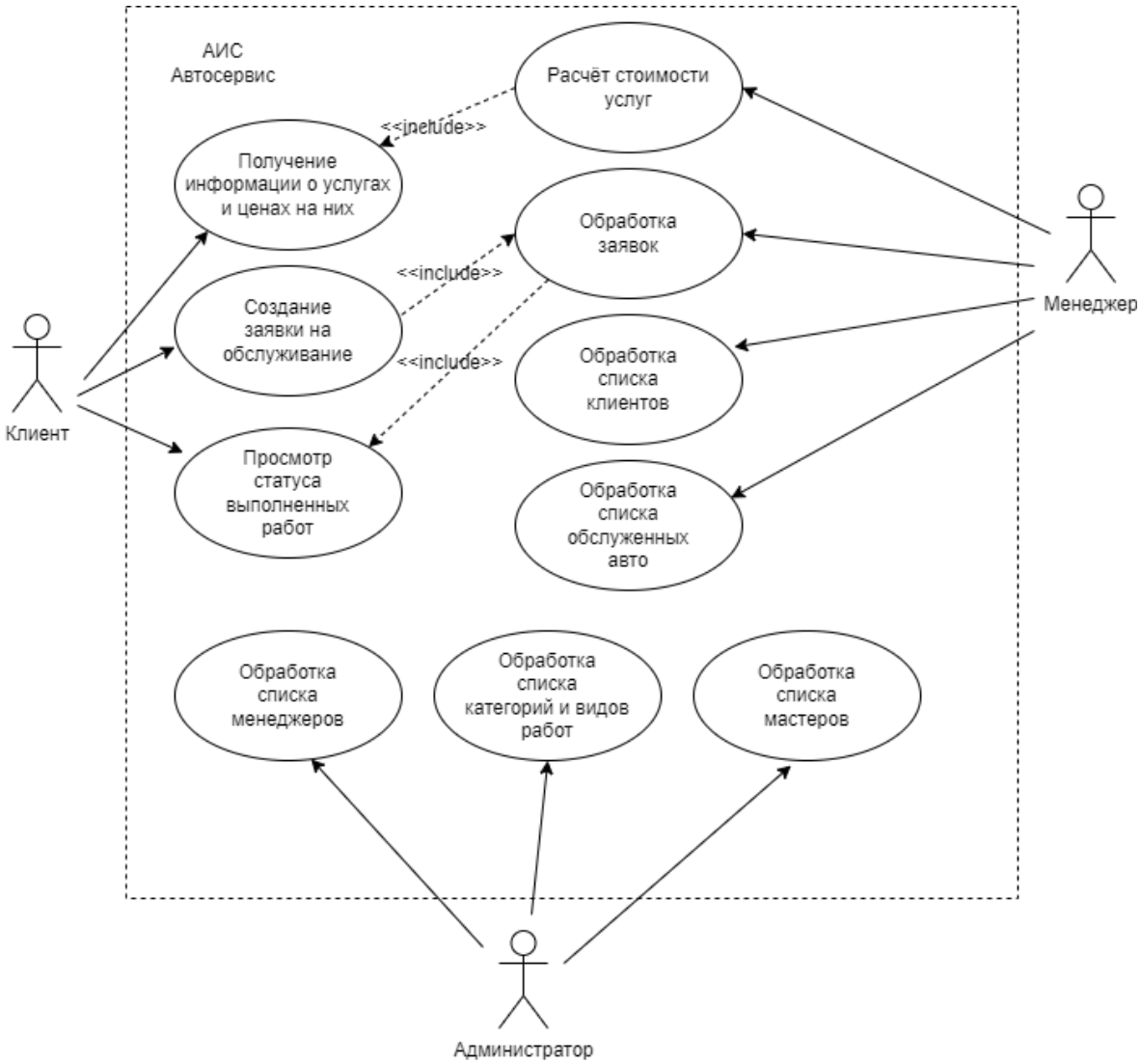


Рисунок 9 – Диаграмма прецедентов

Каждый актёр выполняет свои действия. У клиента есть возможность оформить заявку на обслуживание, посмотреть прайс-лист на услуги в автосервисе, получить отчёт о произведённых работах. Менеджер ведёт список клиентов, автомобилей и обрабатывает заявки клиентов. Администратор системы ведёт список сотрудников автосервиса, список автомобилей по маркам и моделям, список категорий и видов выполняемых автосервисом работ.

2.2 Диаграмма классов

Для спецификации системы, построим диаграмму классов. Проанализировав описание предметной области требований системы, можно увидеть, что следующие сущности встречаются чаще других. Это: автомобили клиентов, оказываемые услуги, заказы на обслуживание, сотрудники автосервиса. Затем, удалив неподходящие и неудачные в создании отдельных классов сущности, выделим определенное количество, подходящих для создания классов.

Выбираем из них те, которых максимально ясное назначение в системе. Они подходят для описания большого количества объектов, имеют характерные наборы свойств, и также имеют отличительные наборы атрибутов.

Клиенты - класс, представляющий собой клиента автосервиса, Менеджеры - класс, представляющий собой менеджера автосервиса, Мастера - класс, представляющий собой мастера автосервиса. Мастера разделяются по специализации обслуживания, Автомобили - класс, представляющий собой автомобиль клиента, Заказ - класс, представляющий собой заказ на ремонт, содержит коллекцию заказанных услуг, Услуга - класс, представляющий собой оказываемую услугу, услуги есть категория время выполнения, предназначенное для планирования времени, и ответственного за выполнения данной услуги.

Диаграмма классов для АИС «Автосервис» представлена на рисунке 10.

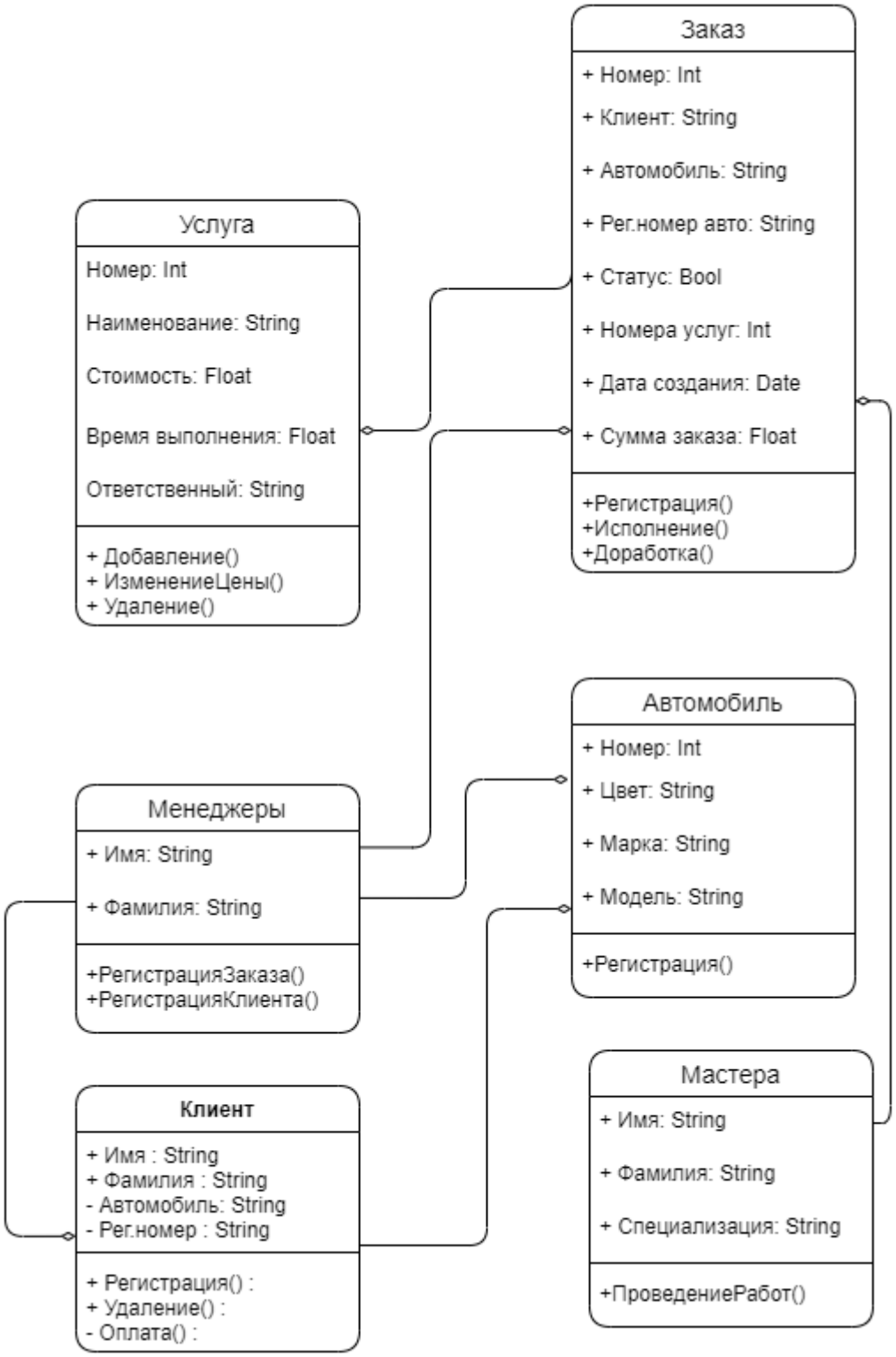


Рисунок 10 – Диаграмма классов АИС «Автосервис»

В конечном итоге в АИС останется 6 классов.

На диаграмме верхняя часть класса содержит имя класса. В средней части содержится список атрибутов, а в нижней - список функций. Атрибуты хранят данные класса, а функции описывают поведение объектов класса.

2.3 Логическая модель базы данных.

При разработке АИС «Автоматизация работы автосервиса» было выделено 10 таблиц.

Структура таблицы с наименованием полей, и их типом данных сведены в таблицу 1.

Таблица 1- Таблицы базы данных

Таблица	Поле	Тип данных	Ключевое поле
Марки автомобилей	Id	Int	PK
	Марка	String	
Модели автомобилей	Id	Int	PK
	Марка	Int	FK
	Модель	String	
	Год выпуска	Int	
Автомобили	Id	Int	PK
	Марка	Int	FK
	Номер	String	
	Цвет	String	
Клиенты	Id	Int	PK
	Имя	String	
	Фамилия	String	
	Телефон	String	

Продолжение таблицы 1

Таблица	Поле	Тип данных	Ключевое поле
Заказы	Id	Int	PK
	Дата создания	DateTime	
	Клиент	Int	FK
Виды работ	Менеджер	Int	FK
	Автомобиль	Int	FK
	Стоимость	Float	
	Время выполнения	Float	
Специальности	Id	Int	PK
	Специальность	String	
Мастера	Id	Int	PK
	Имя	String	
	Фамилия	String	
	Специальность	Int	FK
Сведения о заказе	Id	Int	PK
	Заказ	Int	FK
	Мастер	Int	FK
	Работы	Int	FK

Таблица «Марки автомобилей» содержит информацию о марках автомобилей.

Таблица «Модели автомобилей» содержит информацию о моделях автомобилей и годе выпуска. Связь осуществляется с таблицей «Марки автомобилей» в отношении один-ко-многим.

Таблица «Автомобили» содержит информацию о автомобилях, обслуживаемых на СТО, их регистрационном номере и цвете кузова. Связь

осуществляется с таблицей «Модели автомобилей» в отношении один-ко-многим.

Таблица «Клиенты» содержит информацию о клиентах СТО.

Таблица «Заказы» содержит информацию о заявках на обслуживание в СТО. Связь осуществляется с таблицами «Клиенты» в отношении один-ко-многим, «Менеджеры» в отношении один-ко-многим и «Автомобили» в отношении один-ко-многим.

Таблица «Менеджеры» содержит информацию о менеджерах СТО.

Таблица «Виды работ» содержит информацию о всех видах оказываемых услуг на СТО.

Таблица «Специальности» содержит информацию специализации мастеров СТО.

Таблица «Мастера» содержит информацию о мастерах СТО. Связь осуществляется с таблицей «Специальности» в отношении один-ко-многим.

Таблица «Сведения о заказе» содержит информацию о работах, включённых в заявку на обслуживание и кем эта работа выполняется. Связь осуществляется с таблицами «Заказы» в отношении один-ко-многим, «Мастера» в отношении один-ко-многим и «Виды работ» в отношении один-ко-многим.

Таким образом, данная структура позволит хранить всю необходимую информацию в упорядоченном виде.

2.4 Проектирование концептуальной модели.

Основным этапом проектирования базы данных является создание концептуальной модели данных. Концептуальная модель является основой любой системы баз данных. На этом этапе создаются подробные модели пользовательских представлений данных, затем они объединяются в концептуальную модель базы данных.

Модель должна содержать формальное описание предметной области, которое легко воспринимается не только специалистами, и не должно быть привязано к конкретной БД. Концептуальная модель представлена на рисунке 11.

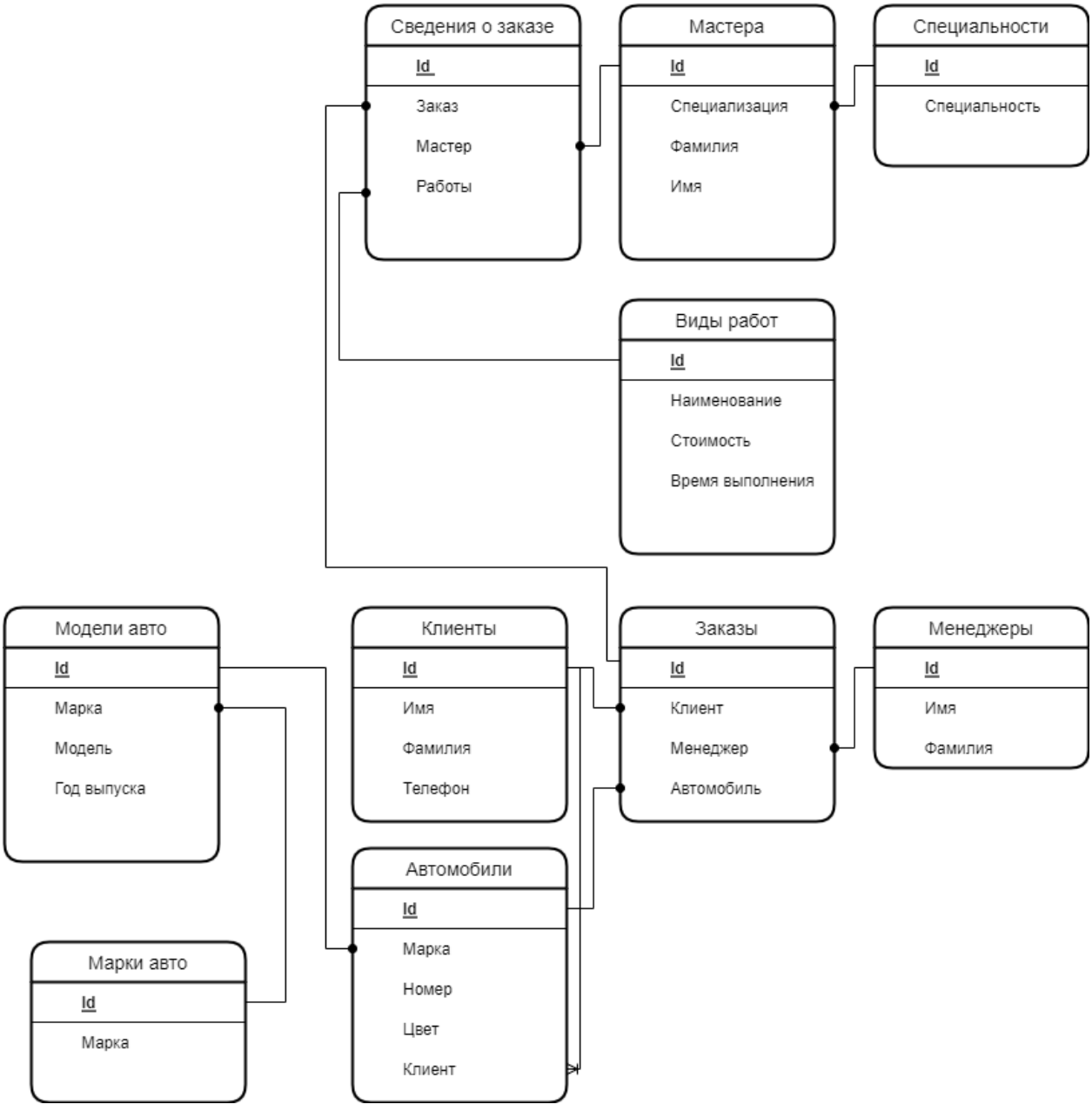


Рисунок 11 - Концептуальная модель

Следующим шагом будет выделение основных сущностей в логической модели данных. Её диаграмма, для последующего преобразования в таблицы

БД, выполняется с помощью функционала ERD Tool, приложения PgAdmin4. На рисунке 12 представлена логическая модель данных приложения.

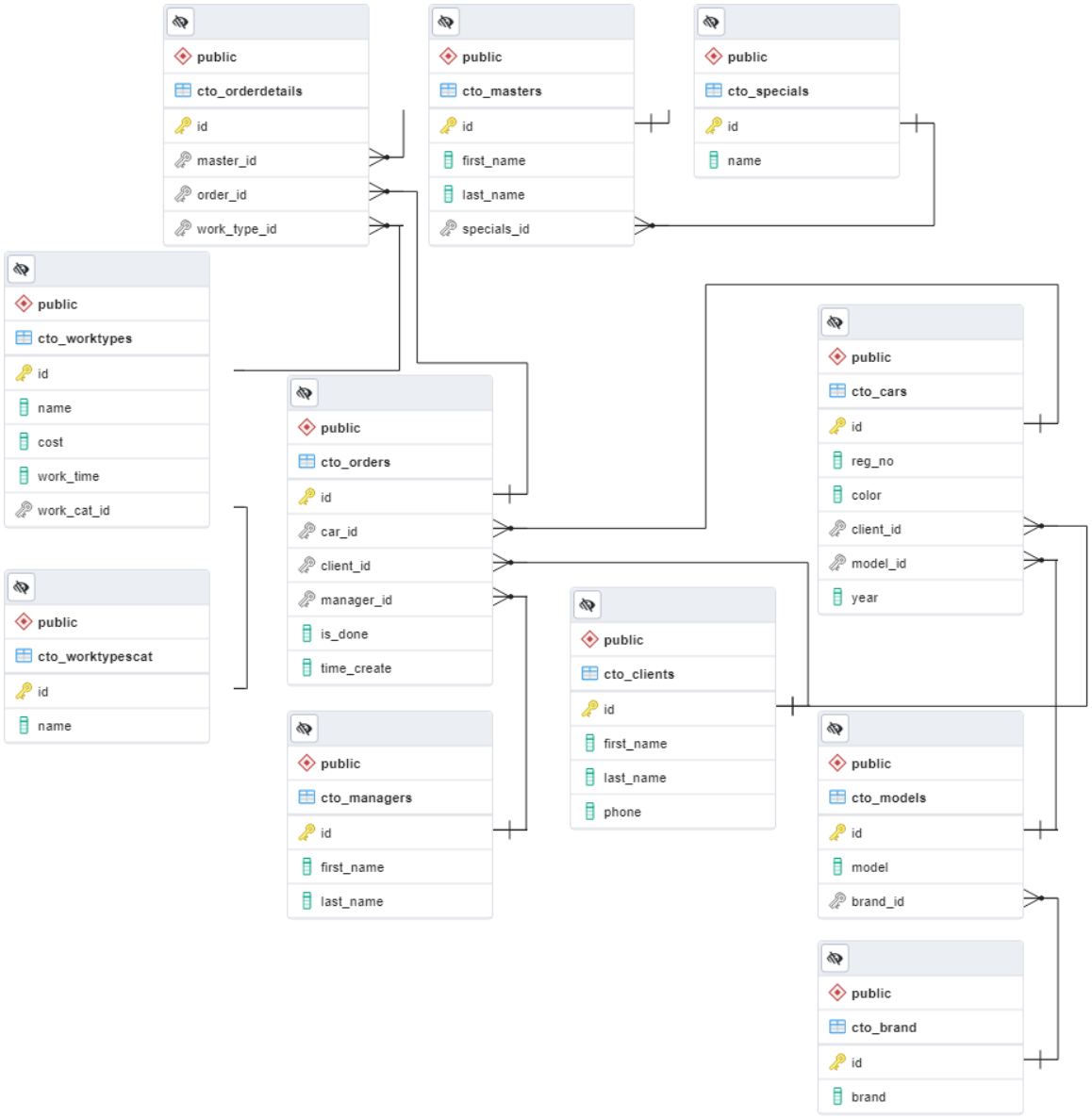


Рисунок 12 - Логическая модель данных

На логическом уровне модели данных содержится информация обо всех объектах базы данных. Отношения, разработанные на стадии формирования концептуальной модели, преобразуются в таблицы, атрибуты в поля таблиц, для ключевых атрибутов создаются первичные ключи.

2.5 Требования к аппаратному и программному обеспечению информационной системы

В настоящее время информационная система предприятия, реализована в виде таблиц Excel, функционирующей под операционной системой Windows, с помощью которых ведётся обработка данных.

Основными недостатками данной системы являются:

- отсутствие разграничения прав пользователей системы;
- отсутствие возможности совместной работы с данными;
- процесс работы с таблицами является трудоёмким, что в свою очередь обеспечивает превышение трудозатрат.

Для устранения недостатков в информационной системе предприятия было предложено разработать новую информационную систему. Новая система будет основана на использовании базы данных PostgeSQL, обращение к базе данных будет осуществляться с использованием ORM (Object Relational Mapping), фреймворка Django. Интерфейс системы будет построен с использованием фреймворка Bootstrap. Использование новой информационной системы облегчит процесс работы, создания отчётов. Также, новая информационная система должна быть кроссплатформенной, обеспечить разграничение прав пользователей, т.к. существующая не обеспечивает на должном уровне защищённость данных.

Немаловажным значением в современных реалиях будет снижение зависимости от крупных вендоров программного обеспечения, и использованием в разработке свободного ПО, или ПО, разрабатываемого в России. Так как многие крупные зарубежные вендоры программного обеспечения отказались от продажи и поддержки своих программных продуктов в России.

Так как разрабатываемое приложение будет функционировать в браузере, основным аппаратным требованием к пользовательским терминалам будет возможность запуска любого интернет-браузера. Требованием к серверу будут. Физический сервер или виртуальная машина с:

- 2х ядерным, x86 совместимым процессором с частотой от 2ГГц;
- оперативная память не менее 4ГБ;
- жёсткий диск от 32ГБ.

2.6 Обоснование выбора программных средств для реализации информационной системы

2.6.1 Система управления базами данных

В настоящее время на рынке программного обеспечения присутствует большое количество реляционных СУБД. Наиболее популярные из них:

- MS SQL Server;
- PostgeSQL;
- Oracle.

В таблице 2 произведен сравнительный анализ этих СУБД.

Таблица 2 – Сравнение СУБД

	MS SQL Server	Oracle	PostgeSQL
Поддерживаемые ОС	Linux Windows	AIX, HP-UX, Linux(red hat based), OS X, Solaris, Windows, z/OS	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows

Продолжение таблицы 2

	MS SQL Server	Oracle	PostgeSQL
API-интерфейсы и другие методы доступа	ADO.NET, JDBC, ODBC, OLE DB, Tabular Data Stream	JDBC, ODBC, ODP.NET, Oracle Call Interface (OCI)	ADO.NET, JDBC, native C library, ODBC, streaming API for large objects
Поддерживаемые языки программирования	C#, C++, Delphi, Go, Java, JavaScript (Node.js), PHP, Python, R, Ruby, Visual Basic	C, C#, C++, Clojure, Cobol, Delphi, Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp, Objective C, Ocaml, Perl, PHP, Python, R, Ruby, Scala, Tcl	.Net, C, C++, Delphi, Java, JavaScript (Node.js), Perl, PHP, Python, Tcl
Поддержка транзакций	Есть	Есть	Есть
Отказоустойчивость	Есть	Есть	Есть
Поддержка сообществом	Слабая	Есть, по платной подписке	Есть, как платно, так и бесплатно
Контроль доступа к данным	Детализированные права доступа в соответствии со стандартом SQL	Детализированные права доступа в соответствии со стандартом SQL	Детализированные права доступа в соответствии со стандартом SQL
Лицензия	Коммерческая	Коммерческая	Свободная

PostgreSQL – объектно-реляционная система управления базами данных с открытым исходным кодом. К ее сильным сторонам относится расширяемость, она успешно конкурирует с основными реляционными СУБД: Oracle, SQL Server и MySQL. Благодаря разнообразию расширений и открытой лицензии PostgreSQL часто применяется в исследовательских проектах, но её код также лежит в основе многих открытых и коммерческих СУБД, например, Greenplum и Vertica. К тому же стартапы часто отдают предпочтение PostgreSQL в силу условий лицензии и изобилия компаний, оказывающих коммерческую поддержку. [6, с. 16]

На основе приведенного анализа, и учитывая стоимость, а также учитывая невозможность легального владения некоторых из представленных СУБД, основной СУБД для создания ИС для учёта заявок в СТО выбрана PostgerSQL. Даная СУБД создана для работы с реляционными базами данных, включающая все необходимые инструментальные средства для создания многопользовательской системы на основе клиент-серверной архитектуре.

2.6.2 Фреймворк веб-приложения (бекэнд)

Фреймворк (от англ, framework - каркас) - это программная библиотека, реализующая большую часть типовой функциональности разрабатываемого продукта. Своего рода каркас, на который разработчик конкретного продукта “навешивает” свои узлы, механизмы и детали декора. [7, с. 17]

Возможность создавать веб-приложения не встроена в основные функции Python. Для реализации задач подобного класса на Python нужен дополнительный инструментарий. Таким инструментарием является веб-фреймворк Django. Django считается лучшим веб-фреймворком, написанным на Python. Этот инструмент удобно использовать для создания сайтов, работающих с базами данных, он делает веб разработку на Python очень качественной и удобной. [12, с. 15]

Django - это высокоуровневый веб-фреймворк на Python, который поощряет быструю разработку и чистый, прагматичный дизайн, написанный на языке программирования Python и следует архитектурному шаблону MVC-

MVT. Django упрощает создание лучших веб-приложений быстро и с меньшим количеством кода.

Шаблон Model-View-Template (MVT) немного отличается от MVC. На самом деле основное различие между двумя шаблонами заключается в том, что Django сам заботится о части контроллера (программном коде, который управляет взаимодействиями между моделью и представлением), оставляя нас с шаблоном. Шаблон представляет собой HTML-файл, смешанный с Django Template Language (DTL). Рисунок 13 иллюстрирует, как каждый из компонентов шаблона MVT взаимодействует друг с другом для обслуживания пользовательского запроса.

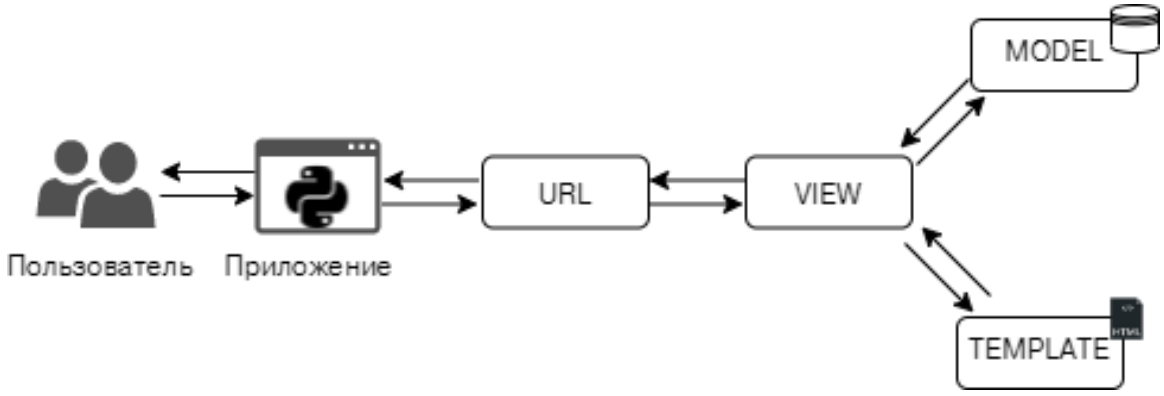


Рисунок 13 - Схема шаблона MVT

Разработчики Django придерживаются следующей философии:

- Loosely Coupled (Слабо связанный): Django стремится сделать каждый элемент своего стека независимым от других.
- Less Coding (Меньше кода): Меньше кода, что, в свою очередь, обеспечивает быструю разработку.
- Don't Repeat Yourself (DRY): Все должно разрабатываться в одном месте, а не повторяться снова и снова.
- Fast Development (Быстрая разработка): Философия Django заключается в том, чтобы делать всё возможное для сверхбыстрой

разработки.

– Clean Design (Чистый дизайн): Django строго поддерживает чистый дизайн во всем своем собственном коде и позволяет легко следовать лучшим практикам веб-разработки.

Преимущества Django:

Object Relational Mapping (ORM) Объектно-реляционное отображение – универсальный механизм взаимодействия с таблицами базы данных, через объекты классов языка Python, не привязанной к конкретной СУБД. Схема взаимодействия Django с базами данных представлена на рисунке 14.

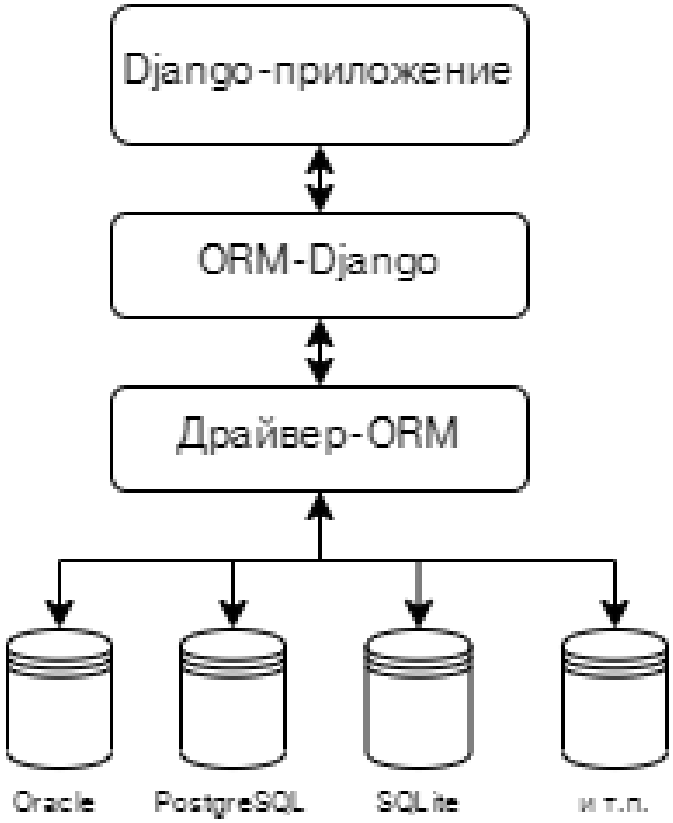


Рисунок 14 - Схема взаимодействия Django с базами данных

Практически любое веб-приложение, так или иначе, работает с базой данных. При этом с системой управления базами данных (СУБД) приложение общается каким-нибудь универсальным способом - например, посредством

языка SQL. Однако программисту чаще всего хочется иметь некую абстракцию, позволяющую большую часть времени работать с привычными сущностями языка. Такой абстракцией является Object Relational Mapping (ORM) - отображение сущностей предметной области и их взаимосвязей в объекты, удобные для использования программистом. [12, с. 232]

В Django задача ORM - моделировать базу данных, однако ещё существует вторая система, которая отвечает за фактическое создание базы данных, - миграция. Её задача – давать возможность добавлять и удалять таблицы и столбцы, основываясь на изменениях, вносимых в файлы models.py. [11 с.102]

Манипуляции с данными в Django проводятся с помощью операций CRUD.

Аббревиатура CRUD обозначает четыре основные операции, которые используются при работе с базами данных. Этот термин представляет собой сочетание первых букв английских слов: создание (Create), чтение (Read), модификация (Update) и удаление (Delete). Это, по своей сути, стандартная классификация функций для манипуляций с данными. В SQL этим функциям соответствуют операторы Insert (создание записей), Select (чтение записей), Update (редактирование записей) и Delete (удаление записей). В Django при создании моделей данных они наследуют свое поведение от класса django.db.models.Model, который предоставляет базовые операции с данными (добавление, чтение, обновление и удаление). [12, с. 240]

Django обеспечивает мост между моделью данных и ядром базы данных и поддерживает большой набор систем баз данных, включая MySQL, Oracle, Postgres и т.д. Django также поддерживает базу данных NoSQL через Django-nonrel fork. На данный момент единственными поддерживаемыми базами данных NoSQL являются MongoDB и google app engine.

Django поддерживает многоязычные веб-сайты с помощью своей встроенной системы интернационализации. Таким образом, можно разработать своё веб-приложение, с поддержкой нескольких языков.

Django имеет встроенную поддержку асинхронных запросов к серверу, генерацию ленты новостей, кэширования и различных других фреймворков и библиотек.

Django предоставляет встроенный мощный и готовый к использованию пользовательский интерфейс для административных задач, позволяющий осуществлять как манипулирование пользовательскими учётными данными, так и любыми таблицами базы данных, созданными в процессе разработки приложения.

Django поставляется с легким веб-сервером для облегчения сквозной разработки и тестирования приложений.

2.6.3 Фреймворк веб-приложения (фронтэнд)

Bootstrap - это самый популярный, бесплатный фреймворк с открытым исходным кодом для создания адаптивного макета. Он содержит HTML, CSS и JS компоненты для создания форм, кнопок, навигации, выпадающего списка, модальных окон, макета и многого другого.

До Bootstrap веб-дизайнерам, приходилось работать с медиа-запросами CSS, чтобы создать адаптивный веб-дизайн. Bootstrap упростил это, самостоятельно позаботившись о медиа-запросах. Это устранило сложность и сделало процесс быстрым и легким.

Веб-дизайнеры и разработчики любят использовать Bootstrap в своих проектах. Они используют его для создания адаптивного веб-дизайна, который выглядит идеально точно на экранах всех размеров (смартфоны, планшеты, ноутбуки и ПК).

Преимущества, которые предоставляет Bootstrap:

- быстрое создание объектов, используя предопределенные классы и шаблоны проектирования;
- адаптивный дизайн – С Bootstrap не нужно применять медиа-запросы в CSS-файле. Он выполняет динамическую настройку веб-страницы на всех размерах экрана.

- не нужно беспокоиться ни о каком браузере, так как он совместим с последними версиями всех браузеров – Google Chrome, Firefox, Opera, Safari и Edge;
- несложно использовать в веб-дизайне. Нужны только базовые знания HTML и CSS;
- согласованность обеспечит неслаженность между вашими проектами и другими разработчиками;
- отличная документация;
- бесплатный и с открытым исходным кодом.

2.6.4 Выбор интегрированной среды IDE

Для ускорения процесса написания программного кода удобно использовать специализированную инструментальную среду - так называемую интегрированную среду разработки (IDE, Integrated Development Environment). Эта среда включает полный комплект средств, необходимых для эффективного программирования на Python. Обычно в состав IDE входят текстовый редактор, компилятор или интерпретатор, отладчик и другое программное обеспечение. Использование IDE позволяет увеличить скорость разработки программ (при условии предварительного обучения работе с такой-инструментальной средой). [12, с. 14]

Доступный в виде кроссплатформенного приложения, PyCharm совместим с платформами Linux, macOS и Windows. Изящно вписываясь в число лучших IDE Python, PyCharm обеспечивает поддержку как версий Python 2 (2.7), так и Python 3 (3.5 и выше).

PyCharm поставляется с множеством модулей, пакетов и инструментов для ускорения разработки на Python. Кроме того, PyCharm может быть настроен в соответствии с требованиями разработки и личными предпочтениями. Впервые он был выпущен для широкой публики еще в феврале 2010 года. IDE включает в себя инструменты анализа кода, отладчик, инструменты тестирования, а также опции управления версиями. Он также

помогает разработчикам в создании плагинов Python с помощью различных доступных API. IDE позволяет работать с несколькими базами данных напрямую, не интегрируя её с другими инструментами. Хотя он специально разработан для Python, файлы HTML, CSS и Javascript также могут быть созданы с помощью этой IDE. Он также поставляется с красивым пользовательским интерфейсом, который можно настроить в соответствии с потребностями с помощью плагинов.

Выводы

Было выполнено проектирование базы данных информационной системы, а именно построены концептуальная, логическая и физическая модели, был произведен подробный анализ части доступных на рынке программного обеспечения систем управления базами данных. Также обоснован выбор веб-фреймворков, как для разработки ядра системы, так и для разработки пользовательского интерфейса проектируемой информационной системы. Рассмотрены современные паттерны, используемые при разработке веб-приложений, технологии взаимодействия приложения и базы данных, функции и методы манипулирования данными, приёмы разработки современных, адаптивных интерфейсов, подстраивающихся под клиентское разрешение устройства пользователя.

Стоит отметить, что процесс проектирования системы является одним из наиболее важных этапов разработки программного обеспечения. Когда система чётко разбита и спроектирована, упрощается процесс её эксплуатации и поддержки.

3 Описание функциональности приложения

В этом разделе рассмотрена функциональность приложения учёта заявок автосервиса с точек зрения разных ролей пользователя.

Так как клиент не авторизован в системе, и соответственно не обладает необходимыми правами на весь функционал системы, ему доступен только базовый функционал приложения. На экране своего устройства, или киоска информирования автосервиса он видит только меню с запросом номера заявки и ссылку на прайс-лист автосервиса. Интерфейс приложения, доступный клиентам изображён на рисунке 15.

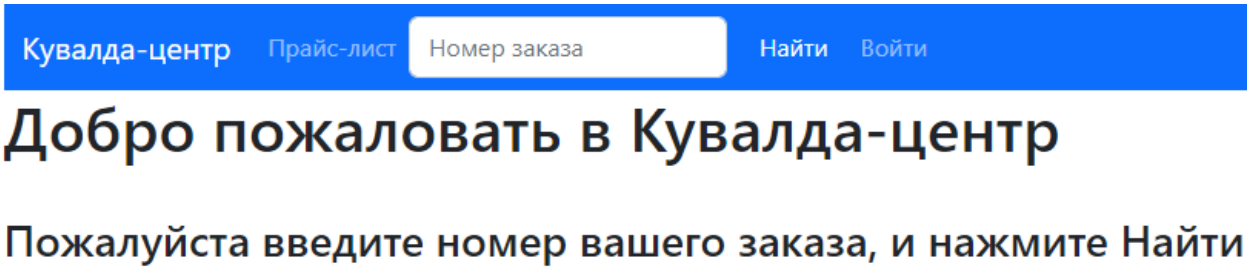


Рисунок 15 - Интерфейс приложения, доступный клиентам СТО

Когда клиент вводит номер заказа ему предоставляется полная информация по заказу, включающая перечень заказанных услуг с их стоимостью, итоговую сумму заказа и статус выполнения.

На рисунке 16 изображена детализация заказа клиента.

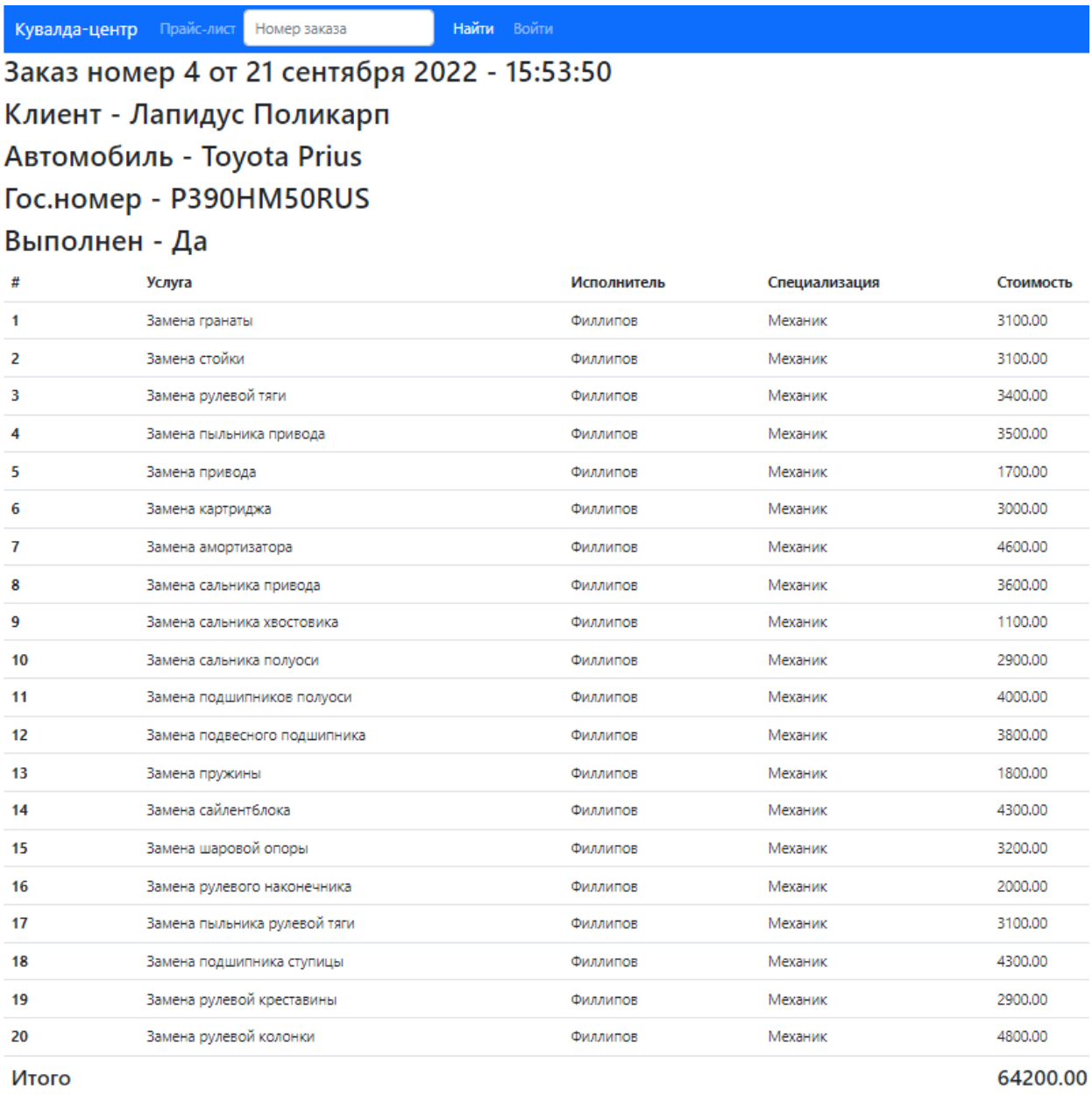
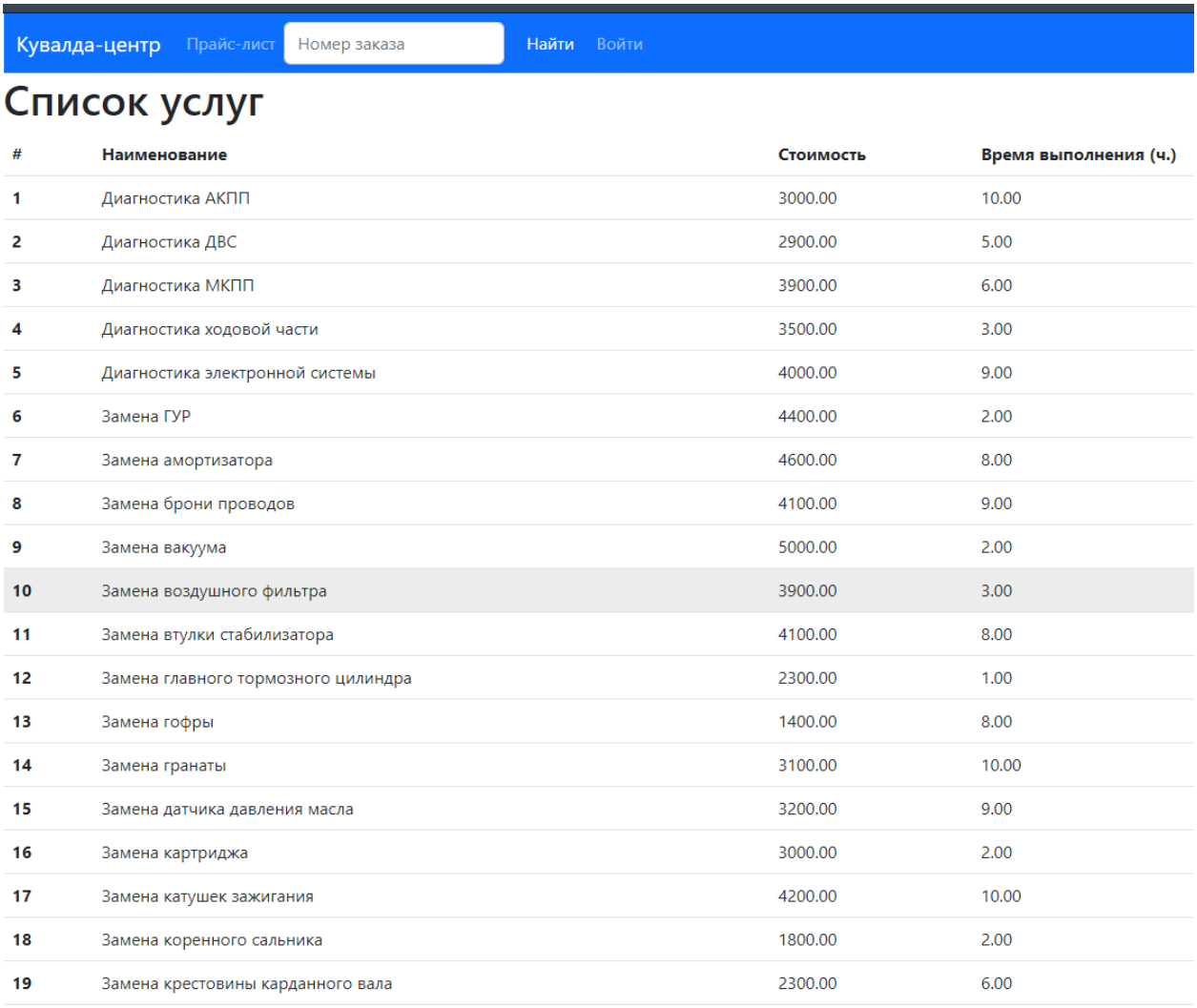


Рисунок 16 - Детализация заказа клиента

Также клиент может самостоятельно ознакомится с прайс-листом на услуги, выполняемые автосервисом. На рисунке 17 представлен экран, при выборе клиентом пункта меню «Прайс-лист».



#	Наименование	Стоимость	Время выполнения (ч.)
1	Диагностика АКПП	3000.00	10.00
2	Диагностика ДВС	2900.00	5.00
3	Диагностика МКПП	3900.00	6.00
4	Диагностика ходовой части	3500.00	3.00
5	Диагностика электронной системы	4000.00	9.00
6	Замена ГУР	4400.00	2.00
7	Замена амортизатора	4600.00	8.00
8	Замена брони проводов	4100.00	9.00
9	Замена вакуума	5000.00	2.00
10	Замена воздушного фильтра	3900.00	3.00
11	Замена втулки стабилизатора	4100.00	8.00
12	Замена главного тормозного цилиндра	2300.00	1.00
13	Замена гофры	1400.00	8.00
14	Замена гранаты	3100.00	10.00
15	Замена датчика давления масла	3200.00	9.00
16	Замена картриджа	3000.00	2.00
17	Замена катушек зажигания	4200.00	10.00
18	Замена коренного сальника	1800.00	2.00
19	Замена крестовины карданного вала	2300.00	6.00

Рисунок 17 – «Прайс-лист» автосервиса

При авторизации в системе, персоналу автосервиса становится доступен более широкий функционал приложения, исходя из его прав.

Менеджерам автосервиса доступен весь остальной функционал приложения. Создание и просмотр заказов, данные клиентов и их автомобилей и т.п. На рисунке 18 представлен интерфейс пользователя из группы «Менеджеры», который доступен после авторизации в системе.

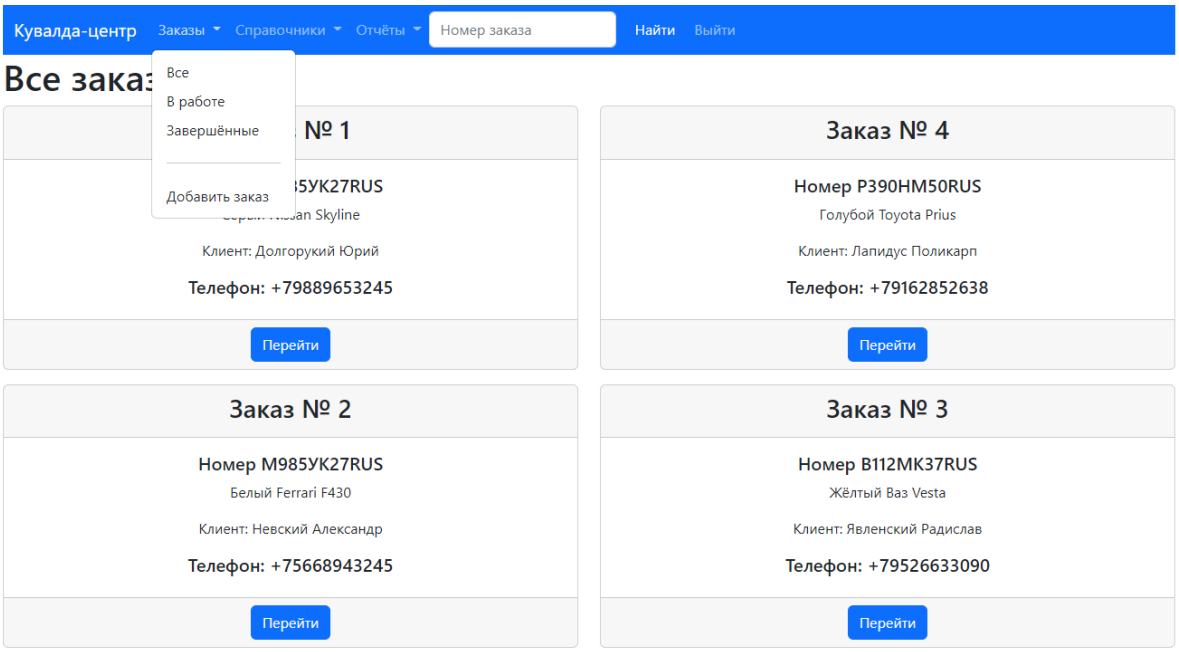


Рисунок 18 - Интерфейс пользователя «Менеджер»

Наивысшими правами обладает администратор системы, ему доступны для редактирования все таблицы базы данных. Также администратор управляет пользователями и группами пользователей системы, назначает им права.

На рисунке 19 представлен административный интерфейс приложения.

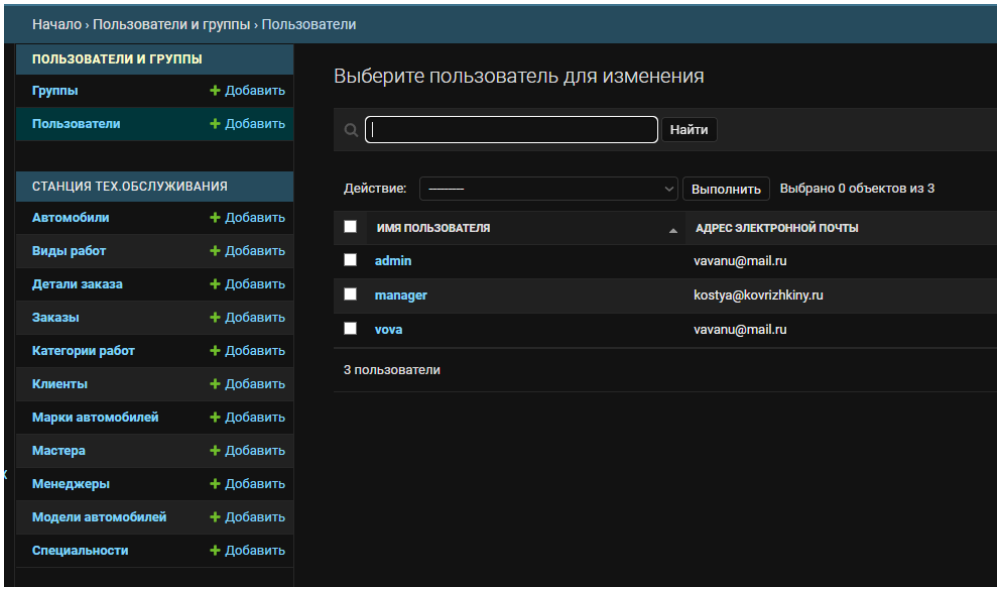


Рисунок 19 – Административный интерфейс приложения

Также менеджеру доступны справочники и отчёты. На рисунке 20 представлен справочник «Клиенты».

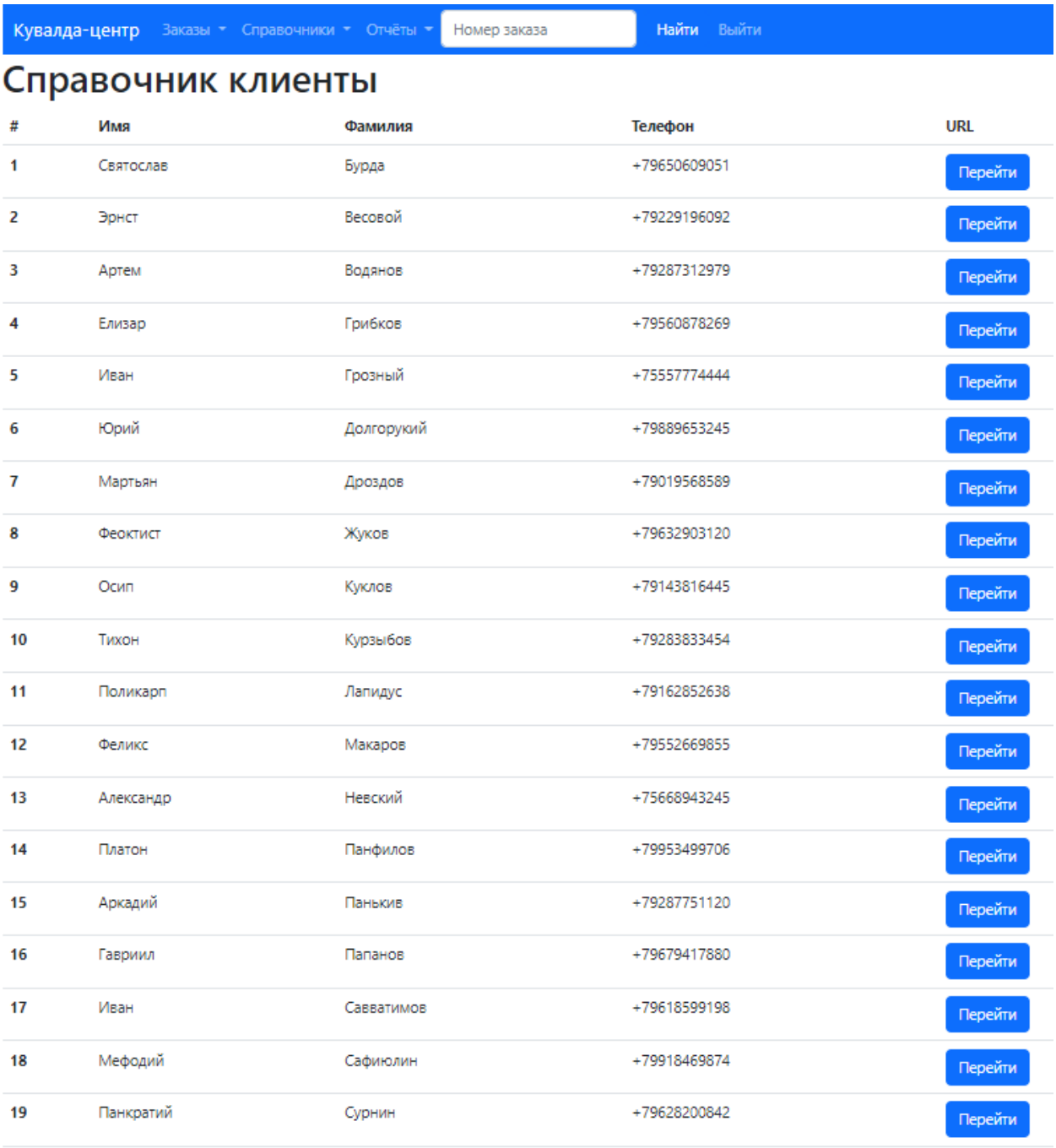


Рисунок 20 – Справочник «Клиенты»

Так как администратор системы обладает всеми правами на таблицы базы данных он должен внимательно ознакомится с инструкцией по

эксплуатации системы, знать все функциональные возможности системы и его устройстве, структуру входных и выходных данных.

Выводы

Разработанный в данном проекте интерфейс приложения, интуитивно понятен и отвечает требованиям эргономики для любых категорий пользователей, позволяет быстро ориентироваться в расположении интересующей его информации. В то же время, каждый разработанный вид имеет не только своё собственное меню, но также несёт на себе определённый набор представлений информации. Данные средства реализации интерфейса тесно взаимодействуют между собой, позволяя выдавать определённым пользователям доступную только им информацию.

Использование фреймворка Bootstrap позволяет использовать данное приложение на мобильных телефонах, коммуникаторах, планшетных и стационарных компьютерах. Разработанный интерфейс не зависит от интернет-браузера, и выглядит одинаково на любом из них, также может работать даже в браузере с очень низкой производительностью.

Таким образом, для любой роли пользователя, на любом используемом устройстве, гарантирован успех в решении задачи.

4 Тестирование приложения

Тестирование приложений - это часть процесса разработки программного приложения, которая выполняется для проверки правильности и валидации функциональных возможностей разрабатываемого приложения. Как часть требований клиента, функциональные спецификации поддерживают качество программного обеспечения, чтобы избежать потери времени на ошибки, допущенные во время разработки, следить за тем, чтобы не превышать запланированные усилия и время, информировать проектную команду о достигнутом прогрессе.

Тестирование помогает командам выпускать надежные приложения без ошибок. Это также позволяет командам выявлять ошибки на ранних стадиях разработки и экономить время разработки.

Тестирование приложений может проводиться в нескольких областях, таких как графический интерфейс, функциональность, база данных (серверная часть), нагрузочное тестирование и так далее.

Тестирование приложений проводится в два этапа – тестирование внешнего интерфейса или пользовательского интерфейса и внутреннее тестирование, которое проверяет поведение базы данных.

Существует два основных способа тестирования приложения:

Ручное тестирование - подход к ручному тестированию чаще используется разработчиками на начальных этапах разработки для тестирования конкретных функций и разовых сценариев. Тестировщики вручную просматривают различные разделы веб-сайта или функции приложения, чтобы выявить ошибки, недочеты, аномалии и тому подобное.

Автоматизированное тестирование - после того, как программное приложение полностью разработано, тестировщики автоматизируют сценарии тестирования с точки зрения конечного пользователя для оценки удобства использования, функциональности и производительности приложения. Тестировщики настраивают фреймворки и создают тестовые сценарии,

которые автоматизируют действия пользователя, необходимые для тестирования веб-сайта или приложения.

Инструмент тестирования приложений - это любая программа, которая помогает тестировщикам управлять процессом тестирования и регулировать его. Решение о том, какое программное обеспечение для тестирования приложений или фреймворк использовать, зависит от характера приложения.

Рассмотрим самые популярные фреймворки автоматизации тестирования.

Selenium - самый популярный набор инструментов для автоматизации тестирования веб-приложений. Это позволяет тестировщику проверять кроссбраузерную совместимость веб-приложения с помощью Selenium WebDriver.

Apache JMeter - это чисто Java-программное обеспечение с открытым исходным кодом, предназначенное для нагрузочного тестирования, функционального поведения и измерения производительности. JMeter используется для анализа и измерения производительности веб-приложений или различных служб. Первоначально JMeter использовался для тестирования веб-приложений. В настоящее время он используется для функционального тестирования, тестирования сервера баз данных и т.д.

LoadRunner - это инструмент для тестирования производительности, впервые разработанный компанией Mercury в 1999 году. LoadRunner поддерживает различные инструменты разработки, технологии и протоколы связи. Фактически, это единственный инструмент на рынке, который поддерживает такое большое количество протоколов для проведения тестирования производительности. Результаты тестирования производительности, полученные с помощью программного обеспечения LoadRunner, используются в качестве эталона по сравнению с другими инструментами.

IBM Rational Functional tester, широко известный как (IBM RFT), является одним из самых передовых объектно-ориентированных

инструментов тестирования для функционального и регрессионного тестирования JAVA, HTML, Silverlight, Eclipse, Siebel, Flex, и Ajax приложений Microsoft для Windows. В качестве платформы он использует Microsoft и Red Hat Linux. Кроме того, RFT также можно использовать для документов Adobe PDF, приложений zSeries и pSeries.

Cypress framework - это платформа сквозного тестирования на основе JavaScript, построенная поверх Mocha - многофункциональной платформы тестирования JavaScript, работающей в браузере и внутри него, что делает асинхронное тестирование простым и удобным. Он также использует библиотеку утверждений BDD / TDD и браузер для сопряжения с любой платформой тестирования JavaScript.

TestNG – это платформа автоматизации тестирования для Java. Философия TestNG – охватить все категории автоматизации тестирования - модульное тестирование, функциональное тестирование, сквозное тестирование, интеграционное тестирование. Устраняя большинство ограничений старого фреймворка – Junit, этот фреймворк автоматизации тестирования Java дает разработчику возможность писать более гибкие и мощные тесты. Хотя, для того, чтобы иметь возможность использовать эту платформу автоматизации, необходимо иметь хотя бы базовые знания языка программирования Java.

Так как приложение в данном проекте разрабатывалось на языке программирования Python целесообразно использовать для тестирования фреймворк Selenium.

Чтобы выполнить тестирование воспользуемся инструментом фреймворка Selenium WebDriver.

Webdriver - это библиотека для управления браузером из кода. Данную библиотеку можно использовать для автоматизации тестирования приложений со сложной логикой, сложной проверкой постусловий, а также для управления браузером (например, мы хотим пройтись по всем строкам

таблицы на стороннем сайте и импортировать их в свою базу данных). [14, с. 140]

WebDriver - это интерфейс, который взаимодействует с браузером. Согласно документации Selenium, методы в этом интерфейсе делятся на 3 категории:

- управление самим браузером;
- выбор веб-элементов;
- средства отладки.

Поскольку Selenium – это средство функционального тестирования, то критерием успешности теста является наличие на странице всех элементов, которые использовались в тесте (гиперссылки, поля ввода и т. д.). Если какого-то элемента нет (например, программисты удалили сам элемент, поменяли его идентификатор или загрузилась другая страница, где его физически нет), то мы имеем дело с нарушением функциональности. [14, с. 136]

Проведём тестирование аутентификации пользователя на странице входа.

Входные данные: пользователь вводит корректные имя пользователя и пароль, нажимает кнопку ввода.

Ожидаемый результат: пользователь авторизован. Проверка, что кнопка поменяла название с «Вход» на «Выход».

Чтобы написать функциональный тест для приложения необходимо найти путь xpath до элемента страницы. Для этого воспользуемся средством, встроенным в каждый современный браузер «Инструменты разработчика», позволяющем детально просматривать свойства каждого тэга html документа, и применённых к нему стилей оформления.

Также в «Инструменты разработчика» встроен язык запросов xpath, используемый для навигации и поиска html тегов документов. Вид утилиты «Инструменты разработчика» представлен на рисунке 21.

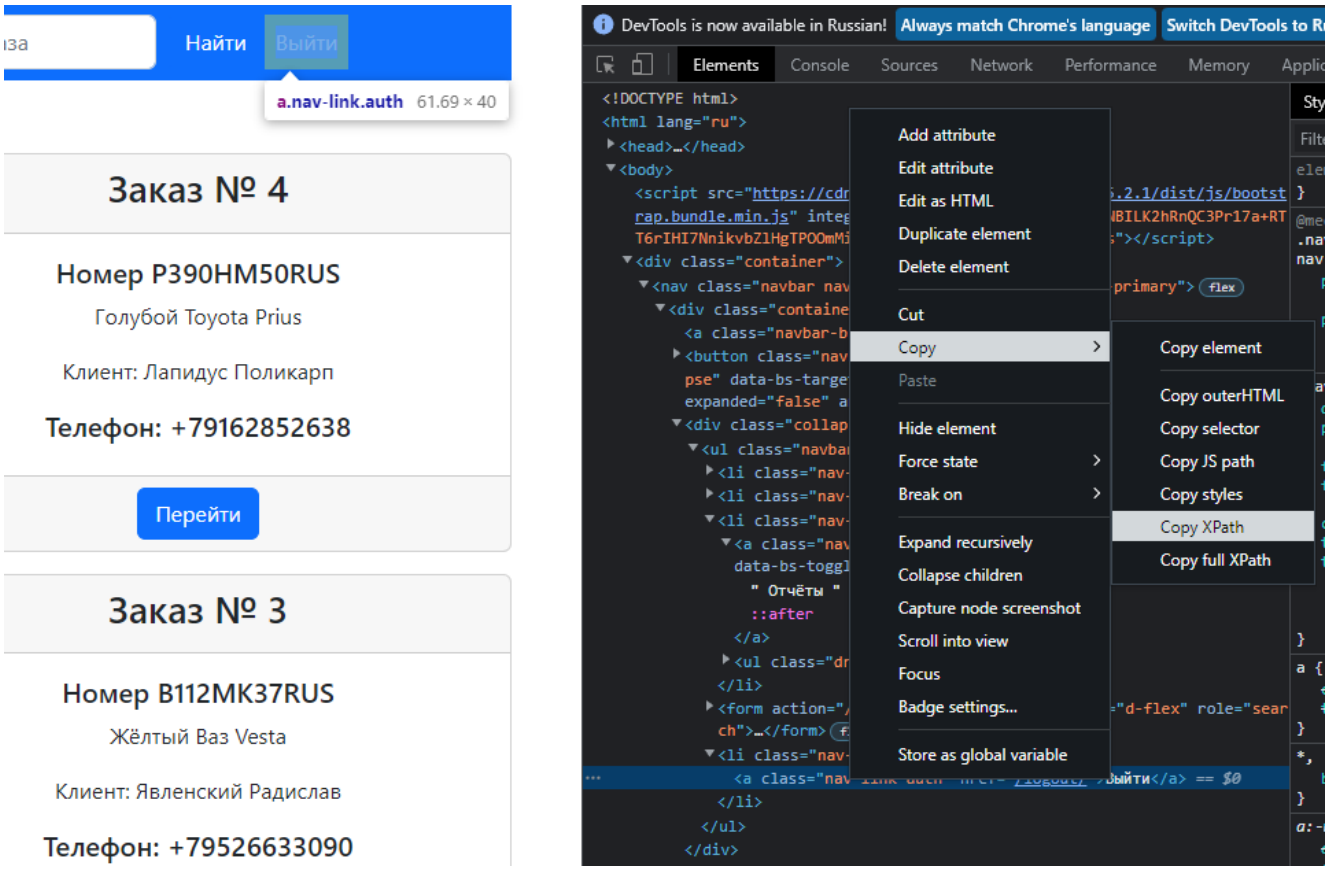


Рисунок 21 – Утилита «Инструменты разработчика»

Так как путь до контрольного элемента известен, создадим тест, который ищет html элемент кнопки авторизации, и проверяет, изменяется, или нет заголовок кнопки при успешной авторизации. Тест считается успешным, если заголовок кнопки поменял своё значение с Вход на Выход. Фрагмент исходного кода теста представлен на рисунке 22.

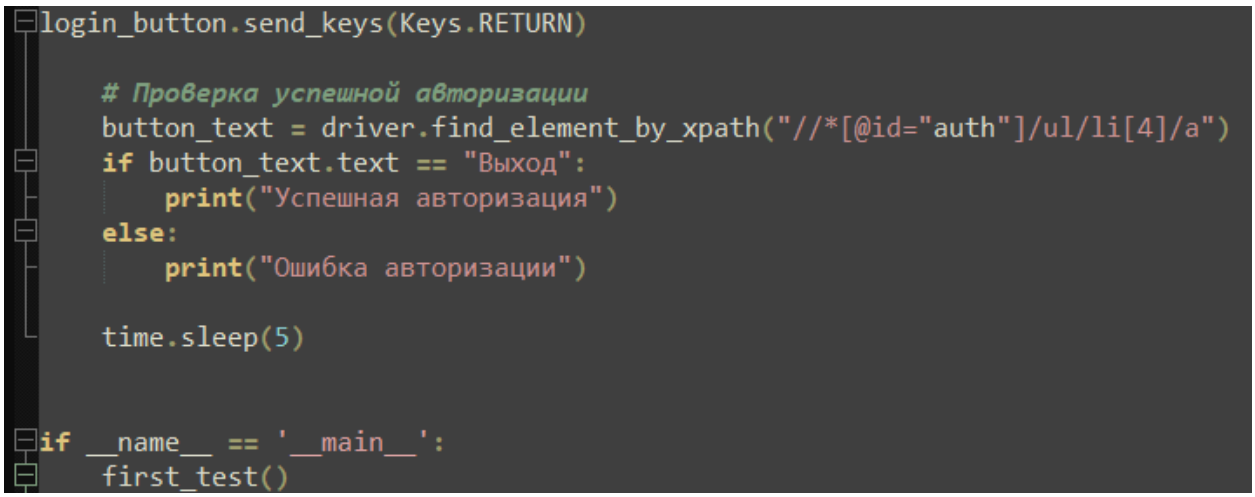


Рисунок 22 – Исходный текст теста авторизации

Этот тест сравнивает текст найденного элемента с ожидаемым значением и выводит сообщение в консоль. Таким же методом осуществляется тестирование остальных составляющих приложения.

Выводы

При разработке приложения важной составляющей является его тестирование. Было выполнено функциональное тестирование приложения, с использования средства Selenium, посредством сравнения полученных, и ожидаемых результатов. Также обоснован выбор фреймворка для тестирования проектируемой информационной системы.

Полученные при тестировании данные показали корректную работу отдельных модулей, и приложения в целом. Пользователи высоко оценили функционал информационной системы, и согласились, что она действительно отвечает предъявляемым к ней требованиям

В результате исследований и тестирования была разработана эффективная информационная система, которая может служить стандартным приложением в любом автосервисе.

5 Развертывание приложения

Технологии меняются и влияют на развитие промышленности, которая, в свою очередь, предъявляет новые сложные требования к технологиям. Менее чем за 2 десятилетия мы перешли от эры коммутируемых модемов со скоростью 56 Кбит до 100-гигабитных сетей Ethernet. С увеличением скорости передачи данных выросли требования к скорости работы программного обеспечения, для разработки которого были созданы более совершенные и высокоуровневые языки программирования. Аналогично в области систем мы перешли от мейнфреймов к высокоскоростным серверам, а затем воплотили на этих серверах облачные технологии и технологии виртуализации. Теперь все чаще в разговорах специалистов стал встречаться термин «контейнеризация», обозначающий новую технологию, помогающую более эффективно использовать ресурсы. [9, с. 16]

Как виртуальные машины, так и контейнеры можно использовать для изоляции приложений друг от друга при работе на одном хосте. Дополнительная степень изоляции в виртуальных машинах обеспечивается гипервизором, и виртуальные машины являются многократно проверенной в реальных условиях технологией. Контейнеры представляют собой сравнительно новую технологию, и многие организации не вполне доверяют методике изоляции функциональных компонентов в контейнерах, пока не увидят воочию весомых доказательств преимуществ контейнеров. По этой причине часто встречаются гибридные системы, в которых контейнеры работают внутри виртуальных машин для совмещения преимуществ обеих технологий. [10, с. 19]

Для развёртывания разработанного приложения подойдёт любая операционная система так как приложение кроссплатформенное. Для работы СУБД будет использоваться существующий сервер баз данных PostgreSQL, который работает под управлением операционной системы Debian Linux 10. В СУБД будет создана дополнительная схема для приложения. Само

приложение, после прохождения опытной эксплуатации в режиме отладки, будет собрано в Docker контейнер, и запущено на сервере баз данных.

До появления Docker в конвейере разработки обычно использовались комбинации различных технологий для управления движением программного обеспечения, такие как виртуальные машины, инструменты управления конфигурацией, системы управления пакетами и комплексные сети библиотечных зависимостей. Все эти инструменты должны были управляться и поддерживаться специализированными инженерами, и у большинства из них были свои собственные уникальные способы настройки. [8, с. 20]

Выбор Docker обусловлен тем, что для правильной работы приложения должны быть установлены все правильные версии библиотек, приложений и служб, установленные на компьютере разработчика. Docker позволяет создавать контейнерные приложения, которые содержат необходимые версии всех зависимостей. Таким образом приложения будут одинаково выполняться на разных машинах.

Выводы

При разработке любого проекта, необходимо учитывать множество критериев. Приступая непосредственно к выполнению работы по созданию информационной системы, разработчик должен обладать всей полнотой информации о будущей системе, которая позволит без задержек и переделок закончить работу в срок и передать её в промышленную эксплуатацию, с гарантированным качеством.

Заключение

Целью выпускной квалификационной работы являлась разработка приложения по учёту клиентов и заявок на обслуживание клиентов СТО.

В процессе выполнения работы были решены следующие задачи:

- изучена функциональная структура автосервиса «Кувалда-центр»;
- произведён анализ бизнес-процессов автосервиса «Кувалда-центр»;
- подобрана система управления базами данных;
- разработана структура базы данных по учёту заявок и клиентов;
- разработан интерфейс приложения.

Разработанное приложение позволяет вести учёт клиентов автосервиса, заявок на обслуживание, обслуженных автомобилей. Получать клиентам информацию о услугах автосервиса, и ценах на них.

Работа позволила приобрести навыки работы с моделями данных, а также навыки разработки веб-приложения, основанного на клиент-серверной технологии, с использованием современных паттернов и средств разработки. Также в результате выполнения работы приобретены навыки работы с учебной, справочной и методической литературой.

Разработанная автоматизированная система обладает всем необходимым функционалом, требуемым в работе автосервиса.

Дальнейшее развитие приложения предусматривает его публикацию в интернет, с добавлением функционала предварительной записи.

В настоящий момент приложение проходит опытную эксплуатацию в СТО.

В результате применения разработанного приложения достигнуто существенное сокращение времени на регистрацию заявки на обслуживание, и её последующий учёт, обеспечена оптимизация и эффективность работы менеджеров автосервиса.

Таким образом, все задачи были решены, цели достигнуты.

Список используемой литературы и используемых источников

1. Афонин В.В. Моделирование систем: учебно-практическое пособие / В.В. Афонин, С.А. Федосин. - М.: Интуит, 2016. – 231 с.
2. Васильков А.В. Безопасность и управление доступом в информационных системах [Текст]: учебное пособие / А.В. Васильков, И.А. Васильков. - Москва: ФОРУМ: ИНФРА-М, 2017. - 368 с.
3. Валитов Ш.М. Современные системные технологии в отраслях экономики: Учебное пособие / Ш.М. Валитов, Ю.И. Азимов, В.А. Павлова. - М.: Проспект, 2016. – 504 с.
4. Венделева М.А. Информационные технологии в управлении.: Учебное пособие для бакалавров / М.А. Венделева, Ю.В. Вертакова. - Люберцы: Юрайт, 2016. – 462 с.
5. Гвоздева В.А. Основы построения автоматизированных информационных систем: учебник. - Москва: ИД «ФОРУМ»: ИНФРА-М, 2017. – 320 с.
6. Джуба С., Волков А. Изучаем PostgreSQL 10 / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2019. - 400 с.
7. Дронов В. А. Django 3.0. Практика создания веб-сайтов на Python. — СПб.: БХВ-Петербург, 2021. - 704 с.
8. Иан Милл, Эйдан Хобсон Сейерс. Docker на практике / пер. с англ. Д.А. Беликов. – М.: ДМК Пресс, 2020. – 516 с.
9. Кочер П. С. Микросервисы и контейнеры Docker / пер. с анг. А. Н. Киселева. – М.: ДМК Пресс, 2019. – 240 с.
10. Моуэт Э. Использование Docker / пер. с англ. А. В. Снастина; науч. ред. А. А. Маркелов. - М.: ДМК Пресс, 2017. – 354 с.
11. Персиваль Г. Python. Разработка на основе тестирования. / пер. с англ. Логунов А. В. – М.: ДМК Пресс, 2018. – 622 с/

12. Постолит А. В. Python, Django и PyCharm для начинающих. -СПб.: БХВ-Петербург, 2021. -464 с.
13. Рогов Е. В. PostgreSQL изнутри. - М.: ДМК Пресс, 2022. — 660 с.
14. Старолетов С. М. Основы тестирования и верификации программного обеспечения: учебное пособие - 22е изд., стер. - СПб: Лань, 2020. - 344 с.
15. Титков А. В. Создание веб-приложений: учебное пособие / А. В. Титков, С. А. Черепанов. — Томск: Эль Контент, 2014. — 72 с.
16. Шениг Г.-Ю PostgreSQL 11. Мастерство разработки / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2019. – 352 с.
17. Django documentation [Электронный ресурс] URL: https://docs.djangoproject.com/en/4.1/ (дата обращения: 1.10.2022)
18. PostgreSQL documentation [Электронный ресурс] URL: https://www.postgresql.org/docs/current/ (дата обращения: 1.10.2022)
19. Bootstrap documentation [Электронный ресурс] URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/ (дата обращения: 1.10.2022)
20. Docker documentation [Электронный ресурс] URL: https://docs.docker.com/ (дата обращения: 1.10.2022)