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 “Прием онлайн заказов”»

Обучающийся Е.С. Колесник
(Инициалы Фамилия) (личная подпись)

Руководитель к.э. н., доцент Т.А. Раченко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Выпускная квалификационная работа выполнена по теме "Разработка веб-приложения «Прием онлайн-заказов» для организации ООО «Феликс»". Объектом исследования является процесс учета ремонтного обслуживания техники в организации. Актуальность темы обусловлена потребностью в автоматизации процессов обработки заявок на ремонт техники в организации ООО «Феликс». Цель работы - разработка веб-приложения «Прием онлайн-заказов». В работе рассмотрены этапы аналитического обзора, логического и физического проектирования, разработки и тестирования веб-приложения. В результате работы достигнута цель и решены поставленные задачи, практическим результатом является веб-приложение готовое для внедрения в организацию.

Работа включает 5 разделов: введение, аналитический обзор, логическое проектирование, физическое проектирование, заключение, а также список литературы.

В работе проведен аналитический обзор по теме исследования, выявлены проблемы на уровне обработки заявок, предложены пути их решения.

В работе применены современные методы проектирования веб-приложений, такие как UML, IDEF0, ER-модель. Разработана клиент-серверная архитектура приложения, включая фронтенд и бэкенд. Для хранения данных используется база данных.

Таким образом, данная работа представляет собой комплексное исследование процесса автоматизации обработки заявок на ремонт техники в организации ООО «Феликс» и является значимым вкладом в современные технологии разработки веб-приложений.

В работе используется 26 источников, имеется 32 рисунка и 12 таблиц.

Содержание

Введение

В настоящее время онлайн-заказы являются неотъемлемой частью ритма жизни и коммерческой деятельности большинства компаний в различных сферах бизнеса. Онлайн-заказы позволяют клиентам получать услуги в удобное для них время, избегая очередей и стресса, связанного с посещением традиционных магазинов и центров оказания услуг.

Использование концепции онлайн-заказов позволяет сократить затраты на аренду площадей, минимизировать количество персонала и повысить удобство заказа товара или услуги для клиента и компании за счет дистанционного взаимодействия. Кроме того, для компаний, которые предлагают онлайн-заказы – это эффективный инструмент расширения клиентской базы и увеличения прибыли.

Тема настоящей дипломной работы посвящена проблеме разработки веб-приложения для приема онлайн-заказов. Актуальность темы и результатов работы в целом обуславливается потребностью и автоматизацией процессов обработки заказов в целевой организации ООО «Феликс».

Объект исследования: процесс учета ремонтного обслуживания техники в ООО «Феликс».

Предмет исследования: автоматизация процесса хранения и обработки заказов на ремонт техники в ООО «Феликс».

Целью данной дипломной работы является разработка веб-приложения «Прием онлайн-заказов» для организации ООО «Феликс». Для достижения этой цели необходимо решить ряд задач:

- Выполнить аналитический обзор статей и литературы по теме исследования;
- Проанализировать предметную область;
- Выполнить постановку задачи разработки веб-приложения;
- Провести концептуальное моделирование предметной области;
- Произвести постановку задачи на разработку веб-приложения;

- Произвести функциональное моделирование бизнес-процесса «прием онлайн-заявок»;
- Реализовать логическое проектирование веб-приложения;
- Реализовать физическое проектирование веб-приложения;
- Выполнить разработку веб-приложения «Прием-онлайн заказов»;
- Провести тестирование веб-приложения «Прием-онлайн заказов» на основе разработанных сценариев.

В работе использованы как теоретические, так и практические методы исследования, включая анализ литературы, моделирование и проведение экспериментов в практической части выпускной квалификационной работы.

Практическим результатом работы является веб-приложение «Прием онлайн-заказов», возможностями которого могут пользоваться сотрудники профильного подразделения (отдела продаж) организации-объекта исследования ООО «Феликс». Данное решение по автоматизации позволит повысить удобство и качество обслуживания клиентов, а также оптимизирует бизнес-процессы компании.

Апробация решения выполнена на этапе тестирования веб-приложения «Прием онлайн-заказов» посредством методики тест-кейсов. По результатам такого тестирования подтверждена работоспособность функций приложения, внесены дополнения по совершенствованию механизмов обработки и защиты данных на уровне исходных кодов.

Работа включает теоретическую часть, моделирование и практическую часть с разработкой веб-приложения.

1. Аналитический обзор по теме работы

1.1 Описание предметной области

ООО «Феликс» осуществляет деятельность по ремонту компьютеров и периферийного компьютерного оборудования. В том числе, компания предоставляет услуги по диагностике, ремонту и обслуживанию компьютеров, ноутбуков, гаджетов, принтеров, и прочего периферийного оборудования [1].

Сфера деятельности ООО «Феликс» включает в себя оказание услуг по ремонту, апгрейду и обслуживанию компьютеров и периферийного оборудования. Кроме того, компания может предлагать услуги по установке и настройке программного обеспечения, резервного копирования данных, восстановлению данных и др.

На рисунке 1 представлена схема организационной структуры [2] организации-объекта исследования ООО «Феликс».

Рисунок 1 – Организационная структура ООО «Феликс»

Опишем подробнее каждое из подразделений приведенной на рисунке 1 структуры.

Главным управляющим звеном иерархии является генеральный директор. Управляющие звенья следующего уровня – это директор по маркетингу и исполнительный директор. Они курируют работу соответствующих подразделений. Бухгалтерия является отдельным подразделением, подчиняющимся генеральному директору.

Отдел продаж занимается приемом заказов на ремонт компьютеров и периферийного оборудования, продажей запчастей и комплектующих. В отделе работают менеджеры по продажам, операторы колл-центра и консультанты.

Группа логистики в ООО «Феликс» осуществляет планирование, координацию и контроль всех процессов, связанных с доставкой и хранением комплектующих для ремонта техники. Подразделение отвечает за оптимизацию логистических процессов, складской учет, работу транспорта и за взаимодействие с поставщиками.

Отдел маркетинга занимается планированием и осуществлением деятельности по продвижению услуг ремонта техники на рынке. Главная цель отдела маркетинга – увеличение клиентской базы, раскрутка услуг и укрепление конкурентной позиции компании на рынке.

Отдел кадров ООО «Феликс» решает задачи управления персоналом и вопросы, связанные с взаимодействием с сотрудниками. Главная задача отдела кадров – обеспечение эффективного функционирования организации благодаря компетентному управлению кадровыми ресурсами.

IT-отдел в организации занимается управлением информационными технологиями, разработкой и поддержкой программного обеспечения, а также обеспечением безопасности и защитой корпоративной информации.

Отдел ремонтного обслуживания выполняет задачи по ремонту, апгрейду и обслуживанию компьютерной техники и периферийного оборудования. В отделе могут работать мастера по ремонту и техническому обслуживанию компьютерной техники.

Отдел бухгалтерии в организации занимается ведением учета финансовых операций. Перечень задач, решаемых отделом бухгалтерии:

- правильное оформление первичных документов;
- запись операций в учетную систему;
- подготовка отчетности и уплата налогов.
- взаимодействие с контролирующими органами;
- расчетом заработной платы сотрудников;
- выплатой налоговых и социальных взносов.

Отсутствие должного уровня автоматизации при обработке заявок по заказам в ООО «Феликс» негативно сказывается на обслуживании клиентов и на эффективности работы организации в целом.

В настоящее время проблемы обработки заявок наблюдаются на уровне отдела продаж, который работает с клиентскими заказами по обслуживанию и ремонту техники. Клиенты должны связываться с сотрудниками компании по телефону, в мессенджерах или по электронной почте, чтобы сделать заказ на ремонтные работы. Такой подход имеет следующие видимые недостатки:

- возрастает функциональная нагрузка на сотрудников отдела продаж – помимо непосредственного взаимодействия с клиентом они должны искать и упорядочивать заявки на ремонт техники;
- заявки хранятся хаотично;
- возрастает время на поиск определенной заявки;
- сложно управлять заказами и оперативно получать по ним информацию;
- устаревший подход к работе с заказами.

Если у клиента возникают трудности при подаче заявки на ремонт своей техники, это может привести к его неудовлетворенности с дальнейшим негативным влиянием на репутацию компании.

Для решения описанных выше проблем предлагается на уровне подразделения отдела продаж внедрить веб-приложение «Прием онлайн-заказов», которое позволит организовать эффективное управление заказами в ООО «Феликс». Предполагается, что внедренное веб-приложение позволит:

- хранить заявки в едином консолидированном хранилище;
- организовать дополнительный канал связи взаимодействия с клиентами;
- оперативно получать полную информацию по тому или иному заказу;
- обеспечить удобство работы с заявками для персонала отдела продаж и для заказчиков услуг;
- увеличить клиентопоток за счет минимизации ручной работы;
- повысить продуктивность и престиж компании ООО «Феликс» на рынке предоставляемых услуг за счет наличия собственного сайта.

Для разработки веб-приложения необходимо будет использовать соответствующие технологии веб-программирования, такие как HTML, CSS, JavaScript, а также выбрать подходящую базу данных для хранения информации о заказах и клиентах.

При взаимодействии с веб-приложением потребуется обеспечить безопасность корпоративных и конфиденциальных данных, передаваемых через приложение. Для этого необходимо использовать методы шифрования данных и протоколы безопасности.

В целом, внедрение веб-приложения для онлайн заказов на ремонт техники позволит компании ООО «Феликс» улучшить качество обслуживания клиентов и повысить эффективность работы отдела продаж. Также предполагается рост числа заказов за счет нового канала связи и увеличение выручки компании.

1.2 Исследование технологий и подходов к разработке веб-приложений

Исследование технологий разработки веб-приложений включает в себя исследование различных подходов и технологий, которые позволяют создавать высококачественные и эффективные приложения. Рассмотрим некоторые из них.

Фреймворки для разработки веб-приложений являются инструментами, которые позволяют разработчикам быстрее и эффективнее выполнять задачи веб-разработки. Рассмотрим несколько фреймворков, которые могут использоваться для разработки веб-приложения «Прием онлайн-заказов» [3].

Наиболее популярные фреймворки представлены в таблице 1.

Конечный выбор фреймворка для разработки веб-приложения «Прием онлайн-заказов» зависит от многих факторов, таких как язык программирования, требования к производительности и безопасности, уровень опыта разработки и многие другие. Каждый из перечисленных в таблице 1 фреймворков обладает своими преимуществами и недостатками, которые могут быть значимы для конкретного проекта.

База данных представляет собой важную часть веб-приложения «Прием онлайн-заказов». Она служит для хранения и организации данных, связанных с заказами, пользователями, продуктами и т.д. Существует множество различных типов баз данных, каждый из которых может быть подходящим для различных проектов.

Реляционные базы данных (RDBMS) – это наиболее распространенный тип баз данных. Они представляют данные в виде таблиц с рядами и столбцами, связанными между собой по ключам [4]. Это облегчает организацию данных и обеспечивает эффективность доступа к ним. Некоторые из популярных реляционных баз данных, которые могут быть использованы для веб-приложения «Прием онлайн-заказов», включают MySQL, PostgreSQL, Oracle и Microsoft SQL Server.

Таблица 1 – Популярные фреймворки

Фреймворк	Описание	Плюсы	Минусы
Django	Django – это фреймворк на Python, который позволяет быстро создавать крупные и сложные веб-приложения. Он имеет встроенную административную панель, ORM для работы с базой данных, а также множество дополнительных библиотек и расширений.	– Быстрая разработка веб-приложений; – Интуитивно понятный синтаксис; – Мощная административная панель; – Хорошая документация.	– Не подходит для маленьких проектов; – Требует дополнительных знаний в Python для полной реализации потенциала.
Ruby on Rails	Ruby on Rails – это фреймворк на языке Ruby, который также позволяет быстро создавать сложные веб-приложения. Он имеет множество инструментов и библиотек для работы с базой данных, пользовательским интерфейсом и многими другими аспектами веб-разработки.	– Высокая скорость разработки; – Отличная документация; – Множество библиотек и расширений.	– Менее масштабируем, чем некоторые другие фреймворки; – Может иметь проблемы с производительностью на больших проектах.
Flask	Flask – это минималистичный фреймворк на Python, который не имеет многих из встроенных функций Django. Он обладает отличной гибкостью и позволяет разработчику самому выбирать, какие инструменты использовать в проекте.	– Очень гибкий и простой в использовании; – Маленький размер; – Хорошая документация.	– Не имеет встроенной административной панели; – Может быть сложен для использования на больших проектах.

Продолжение таблицы 1

Express	Express – это минималистичный фреймворк на языке JavaScript для создания веб-приложений на Node.js. Он позволяет быстро создавать маршруты, работать с базами данных, создавать API и многое другое.	– Очень быстрый и эффективный; – Хорошая документация; – Огромное количество расширений и модулей.	– Не имеет многих встроенных функций других фреймворков; – Может быть сложен для начинающих разработчиков.
Ruby on Rails	Ruby on Rails – это фреймворк на языке Ruby, который также позволяет быстро создавать сложные веб-приложения. Он имеет множество инструментов и библиотек для работы с базой данных, пользовательским интерфейсом и многими другими аспектами веб-разработки.	– Высокая скорость разработки; – Отличная документация; – Множество библиотек и расширений.	– Менее масштабируем, чем некоторые другие фреймворки; – Может иметь проблемы с производительностью на больших проектах.
Laravel	Laravel – это высокоуровневый фреймворк на языке PHP для создания веб-приложений. Он предоставляет готовые решения для маршрутизации, шаблонов, аутентификации, баз данных и многое другое.	– Простой в использовании и изучении; – Полное руководство по использованию на официальном сайте; – Встроенные инструменты для тестирования.	– Может быть не столь быстрым, как некоторые другие фреймворки; – Требуется базовое знание языка программирования PHP для использования его на полную мощность.

Нереляционные базы данных (NoSQL) – это другой тип баз данных, который используется для хранения и организации данных [5]. Они обычно используются для больших объемов данных и для хранения документов, графов

и ключ-значение пар. Некоторые из популярных NoSQL баз данных, которые потенциально подходят для веб-приложения «Прием онлайн-заказов», включают MongoDB, Cassandra, Couchbase и Redis.

При выборе базы данных для веб-приложения следует учитывать такие факторы, как:

– размер и тип данных;
– производительность и доступность;
– масштабируемость;
– безопасность и простота использования.

В зависимости от конкретных требований проекта, можно выбрать оптимальный вариант СУБД, который будет удовлетворять всем аспектам перечисленных требований.

В направлении веб-разработки выделяют две основные составляющие веб-приложения – фронтенд (frontend) и бэкенд (backend).

Frontend, по сути, представляет собой часть на стороне клиента, которая на практике обычно представлена интерактивным пользовательским интерфейсом [6]. Фронтенд определяет порядок взаимодействия пользователя с приложением в веб-браузере. Наиболее распространенные веб-языки для разработки фронтенда:

HTML (HyperText Markup Language) – язык разметки, который используется для создания структурного каркаса и разметки веб-страницы. Он использует различные теги, которые позволяют определить заголовки, параграфы, изображения, таблицы и другие элементы веб-страницы.

CSS (Cascading Style Sheets) – язык описания внешнего вида веб-страницы. Он позволяет задавать цвета, шрифты, размеры и другие свойства элементов веб-интерфейса. CSS также позволяет создавать анимации и адаптивный дизайн, за счет которого обеспечивается интерактивность и адаптивность на различных устройствах и экранах.

JavaScript – это язык программирования, который используется для добавления динамического поведения веб-страницы. Он позволяет создавать интерактивные элементы, такие как формы, кнопки и меню, а также обрабатывать события и операции, исполняемые в браузере (обновление компонента, выдача предупреждения, сортировка заявок и т.д.).

Бэкенд отвечает за функциональную составляющую веб-приложения, например, события обработки форм, обращение к БД и т.п. В отличие от фронтенда, часть бэкенда находится «под капотом» веб-приложения и скрыта от конечного пользователя [7]. Наиболее популярные веб-языки для разработки бэкенда:

Python – один из наиболее популярных языков для разработки backend веб-приложений. Он прост в изучении и понимании, что делает его очень известным среди начинающих программистов. Python используется во многих областях, включая веб-разработку, анализ данных и машинное обучение.

Ruby – еще один популярный язык для бэкенда. Он используется во многих известных фреймворках, таких как Ruby on Rails, и обычно предпочтительнее для малых и средних проектов.

PHP – является языком программирования для создания веб-страниц и обработки данных на сервере. Он очень популярен в веб-разработке, в частности для создания динамических веб-сайтов. Характеризуется универсальностью – подходит как для небольших, так и сложных нагруженных проектов.

Критически важным аспектом любого веб-приложения является безопасность, особенно если оно обрабатывает конфиденциальную информацию, такую как:

- персональные данные пользователей,
- банковские данные;
- корпоративные данные и сведения о коммерческой тайне;
- информацию о заказах и т.д.

Приложение «Прием онлайн-заказов» должно иметь соответствующие меры безопасности, чтобы защитить пользователей и их данные от злоумышленников.

Некоторые из наиболее распространенных угроз безопасности, с которыми может столкнуться веб-приложение:

Атаки на уровне приложения (инъекции SQL, переполнение буфера, XSS (межсайтовый скриптинг) и CSRF (межсайтовая подделка запроса)), реализуются нарушителями, чтобы получить несанкционированный доступ к приложению и его данным [8].

Уязвимости в сторонних компонентах: веб-приложение может использовать сторонние компоненты, такие как библиотеки, фреймворки и плагины, которые могут иметь свои собственные уязвимости безопасности.

Недостаточная аутентификация и авторизация (слабые пароли, отсутствие двухфакторной аутентификации, неверная реализация механизма авторизации) может привести к возможности несанкционированного доступа к приложению.

Таким образом, безопасность является критически важным аспектом разработки веб-приложения «Прием онлайн-заказов». При проектировании приложения необходимо учитывать возможные уязвимости и применять соответствующие меры защиты. Таковые меры включают использование HTTPS-протокола [9], защиту от XSS и CSRF-атак, аутентификацию и авторизацию пользователей, а также обеспечение безопасности хранения и обработки данных на основе криптографических алгоритмов и протоколов (SSL, TLS и т.д.). Кроме того, необходимо убедиться в безопасности всех компонентов, используемых в приложении, таких как фреймворки, библиотеки и базы данных. В целом, безопасность должна рассматриваться как неотъемлемая часть процесса разработки, чтобы обеспечить защиту для пользователей и сохранить репутацию приложения.

1.3 Постановка задачи разработки веб-приложения «Прием онлайн-заказов»

Постановка задачи разработки веб-приложения «Прием онлайн-заказов» включает в себя определение основных функциональных и нефункциональных требований, которые должны быть реализованы в приложении.

Основная задача веб-приложения «Прием онлайн-заказов» – обеспечить клиентам возможность быстрого и удобного оформления заказа через интернет. Для этого необходимо разработать приложение, которое даст возможность пользователям оставлять заявку на ремонт техники через адаптивную форму.

1.3.1 Основные функциональные требования к веб-приложению «Прием онлайн-заказов»:

- Возможность оставления заявки на услуги ремонта и обслуживания техники со следующими данными: ФИО; Электронная почта; Телефон; Описание неисправности; Дата заявки; Желаемая дата исполнения.
- Проверка корректности введенных данных в форме создания заявки;
- Отслеживание заполнения обязательных полей в форме создания заявки;
- Авторизация от имени администратора;
- Получение информации обо всех заявках в едином интерфейсе администратора;
- Управление статусом заявок и удаление заявок;
- Сортировка заявок в таблице.

1.3.2 Нефункциональные требования веб-приложения «Прием онлайн-заказов»:

- Безопасность: защита от взлома и кражи личных данных пользователей; Обеспечение безопасной обработки платежных данных.

- Производительность: высокая скорость работы приложения; Снижение нагрузки на сервер при высоком трафике.

- Масштабируемость:

- Возможность расширения функционала приложения и добавления новых модулей;

- Возможность добавления информации.

Задача разработки веб-приложения «Прием онлайн-заказов» заключается в создании полноценной системы управления заказами и сбора заявок на ремонт техники отделом продаж, которая обеспечит высокий уровень функциональности, безопасности и удобства использования для клиентов и сотрудников.

В процессе разработки необходимо использовать современные технологии и инструменты, которые позволят создать эффективное и надежное приложение с удобным интерфейсом и безопасными механизмами.

1.4 Концептуальное моделирование

На этапе концептуального моделирования поставлены следующие задачи:

- формализовать используемые классы – для этой цели предлагается использовать диаграмму классов UML, которая позволяет представить статическую структуру веб-приложения с отражением взаимосвязей между сущностями предметной области (объектами, подсистемами и т.д.) [10];

- формализовать пользователей веб-приложения – для этой задачи будет использоваться диаграмма вариантов использования (use case), которая помимо самих пользователей формализует пользовательские сценарии, взаимосвязи между ними и поведение системы в целом [11].

Проектирование соответствующих диаграмм выполнено возможностями пакета для графического моделирования MS Visio.

Диаграмма классов веб-приложения «Прием онлайн-заказов» представлена на рисунке 2.

Для каждого класса диаграммы указываются:

– наименование класса (например, Заказчик);

– список атрибутов с типом значения каждого (например, ФИО заказчика представлено строкой, № заявки – символьным типом и т.д.);

– список доступных операций (например, администратор может изменять, просматривать и удалять заявки).

– Рассмотрим подробнее связи между классами на рисунке 2. Классы Заказчик и Администратор связаны с классом Пользователь связями обобщения. Согласно [10], такой тип связи используется, когда классы-потомки наследуют свойства глобального (родительского) класса. В данном случае от свойства глобального класса Пользователь наследуют два возможных типа пользователей в системе – заказчик и администратор. В зависимости от типа они характеризуются собственными атрибутами (например, заказчик указывает ФИО в заявке, а администратор авторизуется по логину и паролю) и собственным набором операций.

Классы Пользователь и Заявка объединены связью ассоциации. В широком смысле под данной связью в предметной области понимается любое взаимодействие пользователя с заявкой – создавать, просматривать, удалять и т.п.

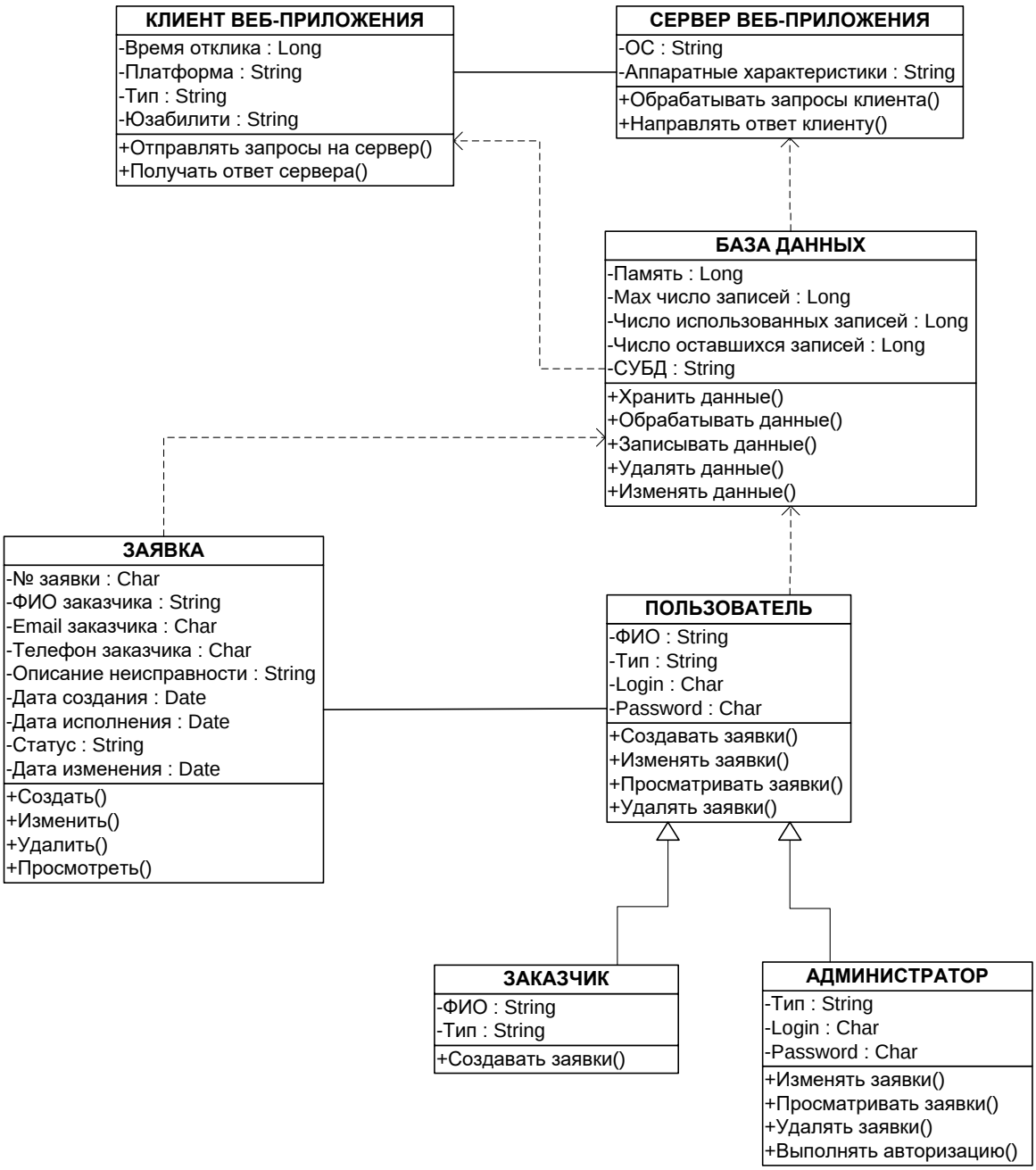


Рисунок 2 – Диаграмма классов веб-приложения «Прием онлайн-заказов»

Поэтому на уровне глобального класса конкретное имя связи опущено, что допускается при моделировании классов [10]. Таким же типом связи объединены классы Клиент Веб-приложения и Сервер Веб-приложения – клиент направляет запросы серверу (например, «Оставить онлайн-заявку на ремонт техники») и по результатам его обработки получает ответ.

Связью зависимости связаны классы Пользователь, Заявка и База данных. В пределах моделирования предметной области это означает, что любое

изменение на уровне базы данных (например, изменение статуса заявки, изменение данных авторизации и т.д.) будет иметь влияние на значения атрибутов зависимых классов. В свою очередь, класс База данных зависит от родительских классов клиента и сервера веб-приложения, поскольку изменения в базе данных могут произойти по результатам обработки клиентского запроса сервером.

Стоит отметить, что в атрибутах класса могут быть указаны не только количественные, но и качественные атрибуты. Например, «Юзабилити» – это субъективный отзыв пользователей об удобстве работы с интерфейсом (клиентом) приложения «Прием онлайн-заказов».

Статическая структура и классы приложения «Прием онлайн-заказов» формализованы. Перейдем к проектированию use case диаграммы.

Диаграмма вариантов использования приложения «Прием онлайн-заказов» представлена на рисунке 3.

Приведенная на рисунке 3 диаграмма формализует пользователей, сценарии приложения «Прием онлайн-заказов», а также связи этими сценариями. Диаграмма use case поддерживает возможности отображения пакетов выходных данных, формируемых по результатам выполнения сценария. Результирующими данными является актуальная база данных заявок на ремонт техники и пользователей (данные авторизации администраторов).

Варианты использования «Выполнять авторизацию в системе» и «Управлять заявками» связаны отношением включения – «include». На уровне моделирования эта связь означает, что для выполнения глобального сценария (к которому идет стрелка) предварительно должен быть выполнен включаемый сценарий (от которого идет стрелка). Т.е. администратор не получит доступа к функционалу управления заявками пока не пройдет процедуру авторизации.

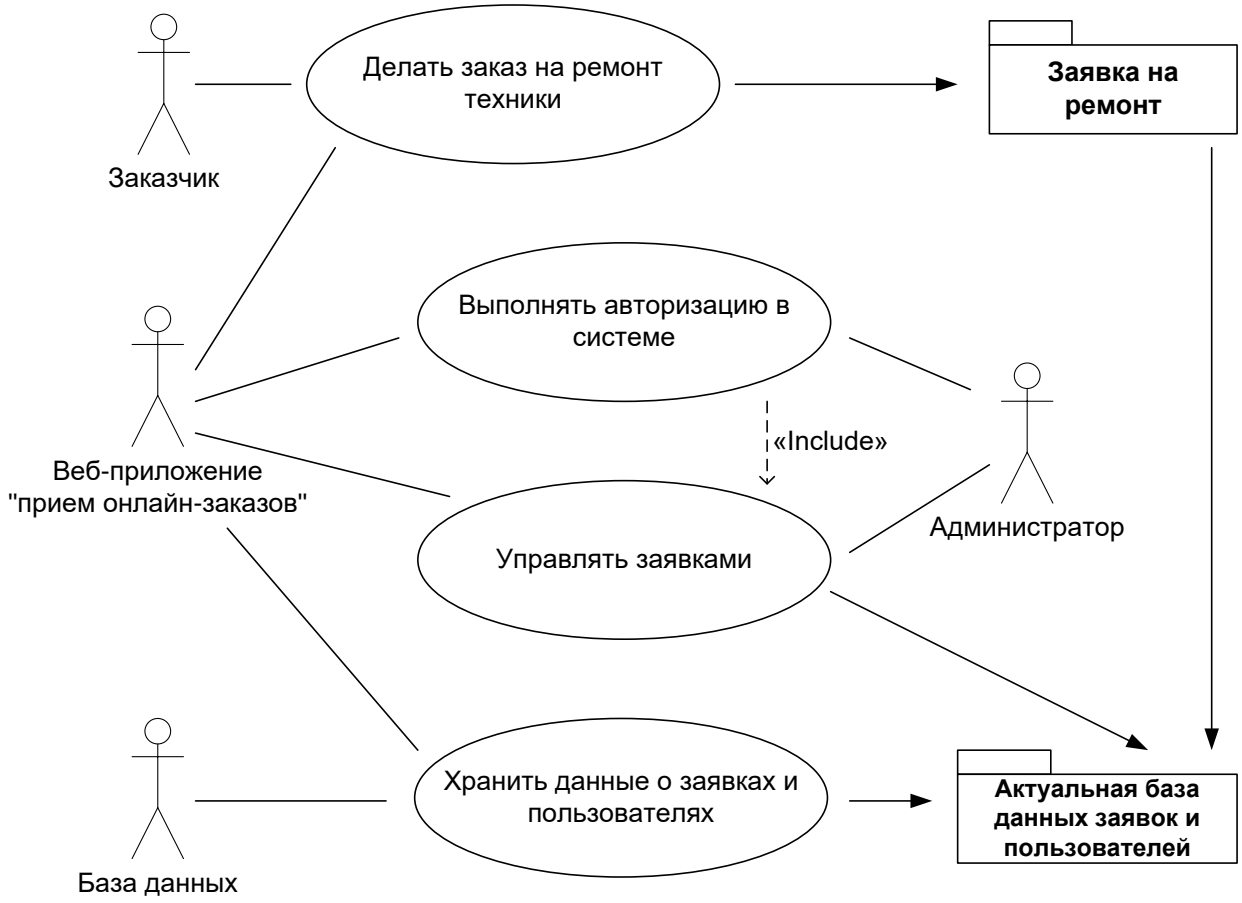


Рисунок 3 – Диаграмма use case

Применительно к диаграмме use case на рисунке 3 стоит отметить, что Веб-приложение «Прием онлайн-заказов» участвует во всех операциях, реализуя и поддерживая логику каждого из сценариев. Это свидетельствует о высокой степени автоматизации на уровне решаемых задач приема онлайн-заказов в ООО «Феликс».

Диаграммы use case предполагают возможность декомпозиции сценариев. Построим use case диаграмму для варианта использования «Выполнять авторизацию в системе», чтобы показать взаимодействие компонентов при выполнении конкретной операции.

Диаграмма сценария «Выполнять авторизацию в системе» представлена на рисунке 4. Данная диаграмма дает представление о том, как клиент и сервер веб-приложения взаимодействуют в процессе выполнения авторизации. Пользователь с ролью «администратор» вводит данные авторизации и

выполняет соответствующую команду. Клиентом формируется запрос к серверу на авторизацию пользователя как администратора.

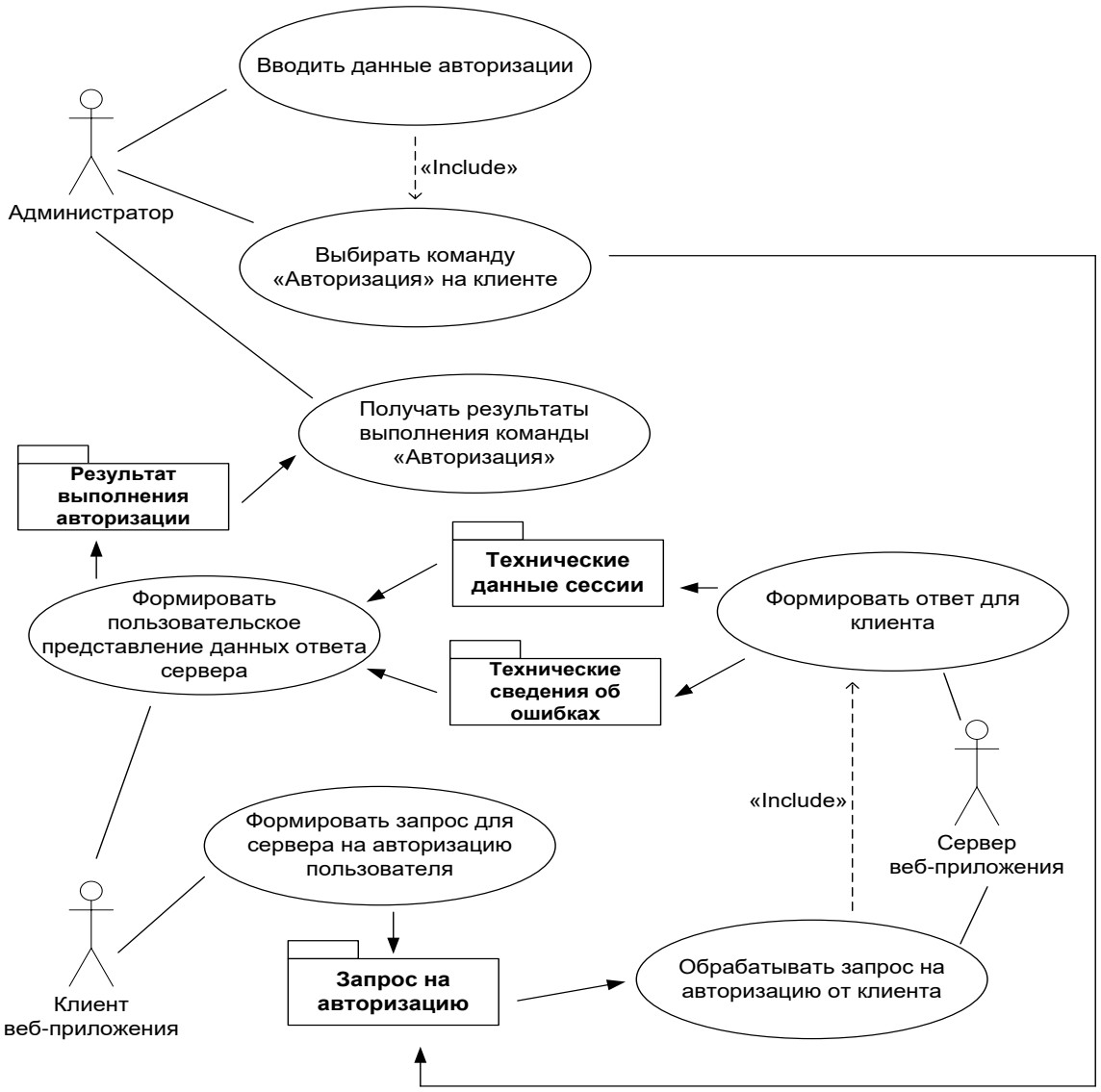


Рисунок 4 – Диаграмма use case варианта использования «Выполнять авторизацию в системе»

Обработав запрос от клиента, сервер в качестве выходных данных может вернуть технические сведения о сессии в случае успеха, либо технические данные об ошибках в случае неуспешной авторизации, например, неверно введенные логин и пароль. Получив эти технические данные, клиент на своей стороне преобразует их в привычное пользовательское представление,

например, в текстовое сообщение: «Имя или Пароль указаны неверно. Попробуйте еще раз» при неверно введенных данных авторизации.

На конечном шаге пользователь получает результат выполнения команды «Авторизация» – вход в систему, либо сообщения об ошибках.

На следующем этапе выпускной квалификационной работы выполним функциональное моделирование процессов предметной области.

1.5 Функциональное моделирование предметной области

Для функционального моделирования процессов предметной области использована нотация IDEF0, которая определяет методологию функционального моделирования [12]. В качестве выходных данных моделирования получены диаграммы в нотации IDEF0 для процессов типа «как есть» (as-is) и «как должно быть» (to-be).

В качестве инструмента для моделирования использован Ramus Educational – программный пакет на основе Java-платформы. При небольшом объеме памяти он располагает всеми необходимыми технологиями для построения диаграмм IDEF0.

На рисунке 5 продемонстрирована контекстная диаграмма «как есть». Она дает общее представление о входных и выходных данных в предметной области приема заказов в ООО «Феликс», исполнителях процессов и управляющих потоках. На рисунке 6 представлена диаграмма декомпозиции глобального процесса «Осуществить прием заявок на ремонт техники в ООО “Феликс”».

Для ситуации «как есть» очевидны следующие проблемы:

– во всех процессах приема заявок участвует менеджер отдела продаж;

– поскольку отсутствует единое хранилище, заявки приходится вручную собирать по всем доступным каналам связи (мессенджеры, социальные сети, телефон, электронная почта и т.д.);

– менеджер вручную структурирует и упорядочивает сведения о заявках;

– менеджер самостоятельно передает данные в отдел ремонтного обслуживания.

Рисунок 5 – Контекстная диаграмма «как есть»

Очевидно, что такой подход к приему заявок нежелателен и имеет ряд уязвимых сторон:

- возрастает нагрузка на менеджера отдела продаж: из-за потребности ручной обработки заявок сокращается время на выполнение прямых обязанностей менеджера – непосредственное взаимодействие с клиентами, поиск и привлечение заказчиков;

- большие временные затраты на поиск и упорядочивание заявок;

- влияние человеческого фактора из-за непосредственного участия во всех процессах;

- риск потери заявок из-за хаотичности и отсутствия единой структуры;

- риск утечки персональных данных клиентов ООО «Феликс» из-за отсутствия алгоритмов защиты.

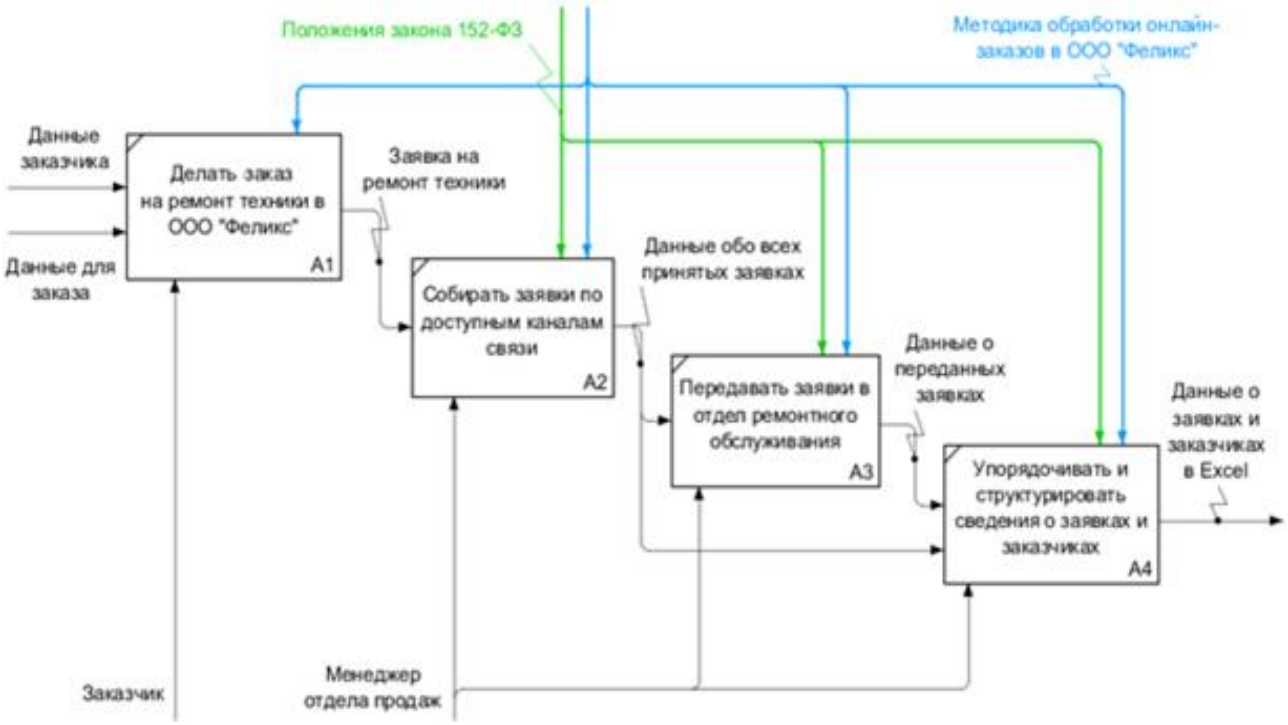


Рисунок 6 – Диаграмма декомпозиции «как есть»

Выходным потоком диаграммы «как есть» являются данные о заявках и заказчиках в Excel. В условиях отсутствия информационной системы в компании ООО «Феликс» используется традиционный подход – учет данных о заказчиках и заявках на ремонт ведутся на разных листах Excel, что в принципе является устаревшим подходом.

На рисунке 7 представлена контекстная диаграмма IDEF0 «как должно быть». В данном случае моделируются процессы приема онлайн-заявок на ремонт с использованием возможностей внедренного веб-приложения «Прием онлайн-заказов».

Рисунок 7 – Контекстная диаграмма «как должно быть»

Рисунок 8 демонстрирует декомпозицию контекстной диаграммы «как должно быть». Следует отметить, что веб-приложение «Прием онлайн-заказов» является исполнителем во всех функциональных процессах предметной области, что свидетельствует о высоком уровне автоматизации решения задачи.

После заполнения формы заказчиком, веб-приложение «Прием онлайн-заказов» автоматически формирует новую заявку на ремонт техники. Внесенные администратором изменения в дальнейшем сохраняются в единой базе данных со сведениями о заказах. Под администратором в данном случае понимается лицо, у которого есть доступ к функционалу управления заявками (имеет логин и пароль учетной записи администратора).

Таким образом, администратором может быть сотрудник отдела ремонтного обслуживания. Это исключает необходимость участия менеджера в передаче заявок в ремонтный отдел – сотрудник сам просматривает и отмечает статусы по заявкам.

Для более подробной демонстрации процесса управления заявок декомпозирован блок А3 «Управлять заявками», рисунок 9. В данном случае администратор просматривает заявки и изменяет их статусы.

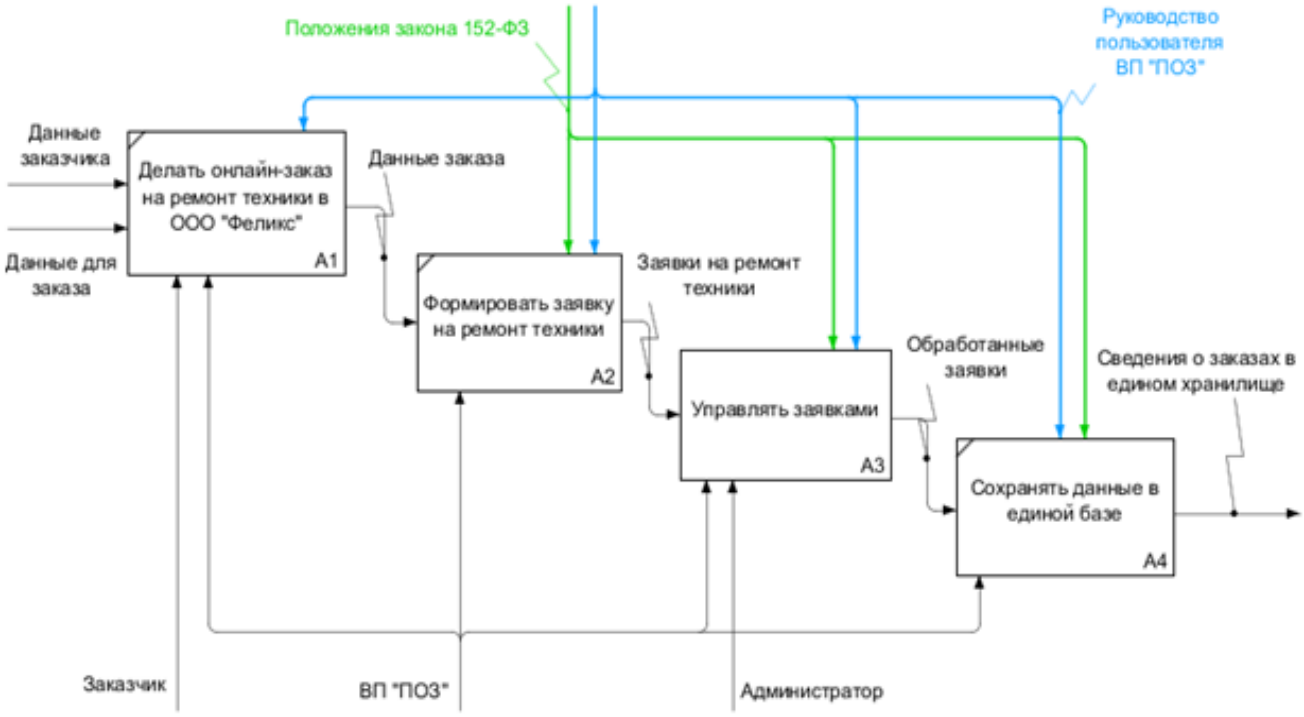


Рисунок 8 – Диаграмма декомпозиции «как должно быть»

(«новая», «в работе», «в архиве»), а веб-приложение «Прием онлайн-заказов» сохраняет эти данные.

Внедрение веб-приложения «прием онлайн-заявок» в ООО «Феликс» обуславливает следующие преимущества:

– минимизация человеческого фактора за счет автоматизации;

– повышение защищенности конфиденциальных данных ввиду использования мер защиты и безопасных протоколов;

– исключение рисков потери заявок;

– устранение хаотичности и хранение всех данных в едином упорядоченном хранилище с возможностью быстрого доступа;

– обработка заявок на ремонт сразу профильным отделом организации ООО «Феликс»;

– организация дополнительного канала связи с клиентами, расширение клиентской базы и рост конкурентоспособности для ООО «Феликс».

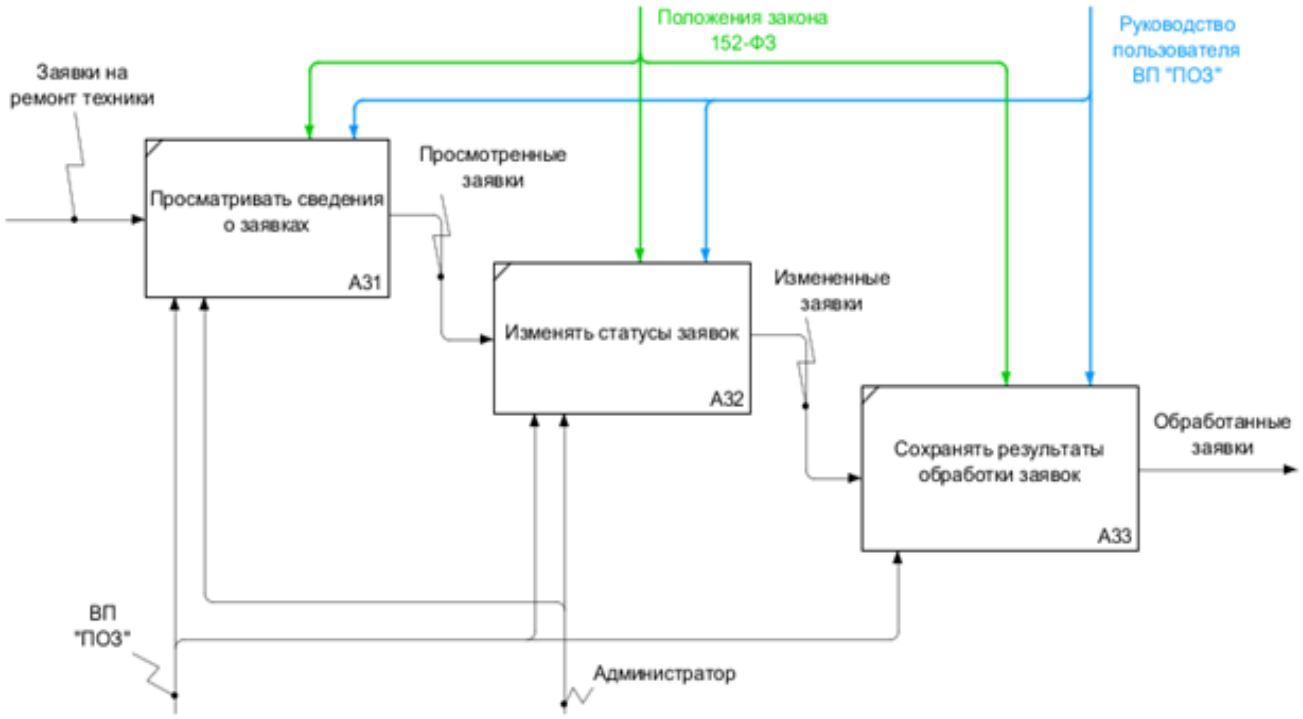


Рисунок 9 – Декомпозиция блока А3 «Управлять заявками»

По результатам моделирования входные потоки «Данные заказчика» и «Данные для заказа» успешно преобразованы в выходной поток «Сведения о заказа в едином хранилище». Следует говорить о достижении поставленных целей функционального моделирования.

Выводы к разделу

По результатам выполнения главы представлено комплексное описание предметной области. Подробно исследована организация-объект исследования ООО «Феликс»: представлена организационная структура с описанием подразделений, проанализированы текущие проблемы обработки заявок в организации на уровне отдела продаж. В качестве решения для устранения

выявленных проблем предложено внедрение веб-приложения «прием онлайн-заявок».

Также в ходе выполнения главы были рассмотрены основные технологии, подходы и средства, которые могут быть использованы для разработки веб-приложений. В частности, выделены следующие ключевые технологии и инструменты:

– Фронтенд-языки разработки веб-приложений (HTML, CSS, JS);

– Бэкенд-языки разработки (Python, Ruby, PHP);

– Фреймворки, упрощающие процесс разработки веб-приложений (Django, Ruby on Rails и т.д.);

– СУБД, как средства обработки, хранения и управления данными (MySQL, Oracle, MongoDB, Cassandra и др.);

– Средства обеспечения безопасности корпоративных данных (криптографические алгоритмы и протоколы шифрования, защита от известных типов атак и т.п.).

Сформулированы функциональные и нефункциональные требования в аспектах безопасности, масштабируемости и производительности. Также выполнена постановка задачи на разработку веб-приложения «прием онлайн-заявок».

По итогам выполнения этапа концептуального моделирования спроектированы:

– диаграмма классов UML веб-приложения «Прием онлайн-заказов», отражающая статическую структуру, классы (сущности) с их атрибутами и операциями, а также взаимосвязи между этими классами;

– диаграммы вариантов использования UML, формализующие пользователей веб-приложения «Прием онлайн-заказов», возможные сценарии и поведение системы в целом.

По итогам выполнения этапа функционального моделирования разработаны:

– диаграммы в нотации IDEF0 «как есть», которые отражают текущий подход к приему онлайн-заказов на ремонт техники в организации ООО «Феликс» – для этого подхода проанализированы его недостатки и уязвимые стороны;

– диаграммы в нотации IDEF0 «как должно быть», которые формализуют процессы обработки заявок после внедрения веб-приложения «Прием онлайн-заказов» – для этого подхода отмечены очевидные преимущества и положительный эффект для всей организации ООО «Феликс» в целом.

После внедрения веб-приложения отдел продаж ООО «Феликс» получит удобную и эффективную систему приема заявок на ремонт техники, которая позволит быстро и точно обрабатывать все заявки, упорядочит их и объединит в едином интерфейсе. Полагается, что внедрение концепции онлайн-заявок обусловит положительный эффект – клиенты смогут подать заявку на ремонт в любое время, используя привычное устройство и не выходя из дома. Дополнительный канал связи в виде веб-сайта позволит привлечь новых клиентов и повысить уровень обслуживания текущих клиентов.

2. Логическое проектирование веб-приложения «Прием онлайн-заказов»

2.1 Выбор технологии логического моделирования

На этап логического моделирования веб-приложения «Прием онлайн-заказов» поставлены следующие задачи:

– формализовать архитектуру данного приложения;

– формализовать компоненты веб-приложения и порядок их взаимодействия.

Схема архитектуры веб-приложения должна давать представление об используемых структурных элементах в пределах технической архитектуры, а также о способах взаимодействиях этих элементов.

Выбор технологии для проектирования архитектуры программного обеспечения определяется контекстом предметной области и потребностями. Поскольку веб-приложение «Прием онлайн-заказов» реализовано на основе веб-технологий, для построения архитектуры приложения предлагается использовать графические элементы из теории проектирования сетей. Это позволит наглядно отобразить все основные элементы архитектуры (клиент, сервер, база данных, каналы связи и т.д.). Для построения архитектуры планируется использовать графические обозначения элементов, принятые в программном пакете MS Visio.

Для формализации структуры веб-приложения «Прием онлайн-заказов» на уровне компонентов предлагается использовать принятый для этого инструмент – диаграммы компонентов UML.

Конечное назначение диаграммы компонентов UML может варьироваться в зависимости от контекста [13]. При моделировании бизнес-логики в качестве компонентов могут использоваться бизнес-сущности, например, заказы, ведомости, контрагенты и т.п. Поскольку в данном случае стоит задача

моделирования архитектурного состава веб-приложения «прием онлайн-заказов», в качестве элементов диаграммы будут задействованы технические компоненты приложения.

Помимо представлений о технической архитектуре диаграмма компонентов UML позволит получить представление о структуре исходного кода на глобальном уровне (фронтенд, бэкенд, программные модули т.п.).

Веб-приложение «Прием онлайн-заказов» использует технологию хранения и обработки данных с помощью инструмента баз данных. Проектирование базы данных веб-приложения является отдельным этапом исследования, для которого отдельно необходимо выбрать технологии моделирования.

Для проектирования реляционных баз данных используются диаграммы вида «сущность-связь» (ER или ERD-диаграммы), которые на разных уровнях проектирования позволяют продемонстрировать порядок взаимодействия сущностей внутри системы и типы связей [14].

Таким образом, на разных стадиях проектирования базы данных предлагается использовать инструмент ER-диаграмм с различным уровнем детализации:

– ER-диаграмма логической модели – в общем виде формализует сущности и связи в предметной области без привязки к конкретной технологии;

– ER-диаграмма физической модели – более явно формализует сущности с их атрибутами и связями в системе, привязана к технологиям и используемым инструментам для физической реализации БД.

2.2 Логическая модель веб-приложения

Схема архитектуры веб-приложения «Прием онлайн-заказов» представлена на рисунке 10 Данная схема определяет взаимодействие

архитектурных компонентов в рамках типовой распространенной архитектуры «клиент-сервер».

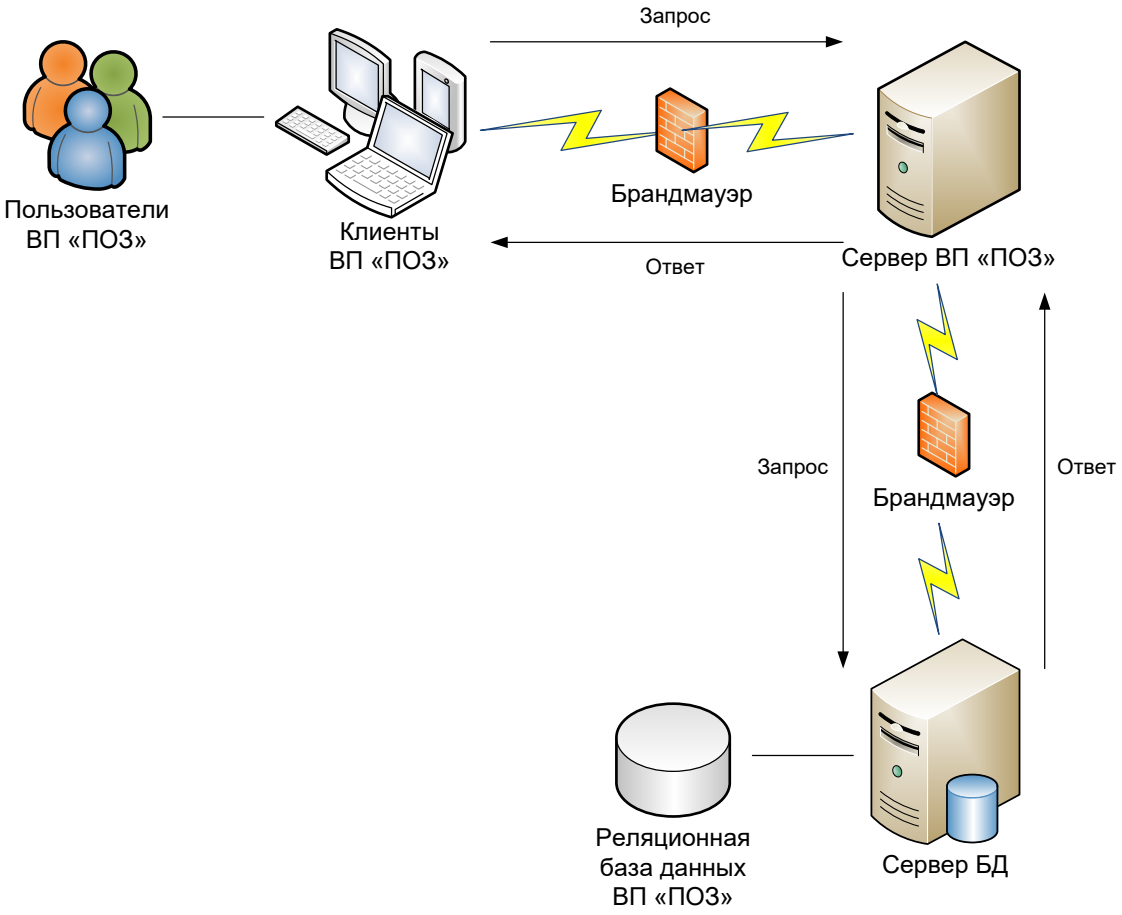
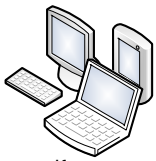


Рисунок 10 – Схема архитектуры веб-приложения «Прием онлайн-заказов»

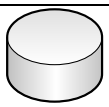
Рассмотрим подробнее структурные элементы архитектуры на рисунке 2.2. Описание элементов с их назначением приводится в таблице 10.

Спроектированная архитектура приложения обеспечивает многопользовательский режим (несколько человек могут работать независимо с разных клиентов), а также безопасную передачу данных за счет использования брандмауэра и безопасных каналов связи.

Таблица 2 – Описание элементов архитектуры веб-приложения «Прием онлайн-заказов»

Обозначение элемента	Описание элемента
 Пользователи ВП «ПОЗ»	Пользователями веб-приложения «Прием онлайн-заказов» являются заказчик услуг ООО «Феликс» на ремонт техники и администраторы. Веб-технологии дают возможность использовать функции приложения множеству пользователей одновременно.
 Клиенты ВП «ПОЗ»	Устройства-клиенты, с помощью которых пользователи взаимодействуют с веб-интерфейсом веб-приложения «Прием онлайн-заказов» Клиентом может быть любое устройство с браузером и выходом в интернет (смартфон, ПК, планшет и т.д.).
	Обозначение выделенных каналов связи, через которые происходит передача данных. Защищенные каналы связи предполагают использование протоколов шифрования для обеспечения безопасности передачи данных.
 Брандмауэр	Стандартное средство обеспечения сетевой безопасности, поставляемое с операционной системой сервера. Передача данных контролируется брандмауэром (файрволлом).
	Графическое обозначение потоков (пакетов) технических данных – запрос от клиента и ответ сервера.
 Сервер ВП «ПОЗ»	Сервер веб-приложения «Прием онлайн-заказов», обрабатывающий клиентские запросы. Функции сервера в данном случае выполняет ПЭВМ под управлением ОС Windows Server.

Продолжение таблицы 2

	Сервер базы данных, предназначенный для обслуживания функций базы данных (хранение, обработка, удаление и т.д.). Сервер БД и сервер обмениваются данными подобно клиенту и серверу, например, сервер может обратиться к серверу БД за извлечением данных.
	База данных, основанная на реляционной модели хранения и обработки данных.
_____	Ассоциативная связь – сервер БД обслуживает базу данных, пользователи используют клиентские устройства.

Формализуем компоненты системы их порядок их взаимодействия. Диаграмма компонентов UML веб-приложения «Прием онлайн-заказов» представлена на рисунке 11.

Диаграмма на рисунке 11 демонстрирует основные компоненты клиент-серверной архитектуры веб-приложения «Прием онлайн-заказов»:

– Клиент веб-приложения «Прием онлайн-заказов» – представляет собой пользовательский веб-интерфейс, с которым пользователь взаимодействует и формирует запросы на сервер (например, «Авторизация»);

– Сервер веб-приложения «Прием онлайн-заказов» – с обращением к компонентам логики и к базе данных обрабатывает запросы и отправляет ответ на сервер;

– Сервер базы данных – обслуживает и поддерживает функции базы данных;

– База данных – реализует основные операции с данным: хранение, удаление, изменение и т.п.

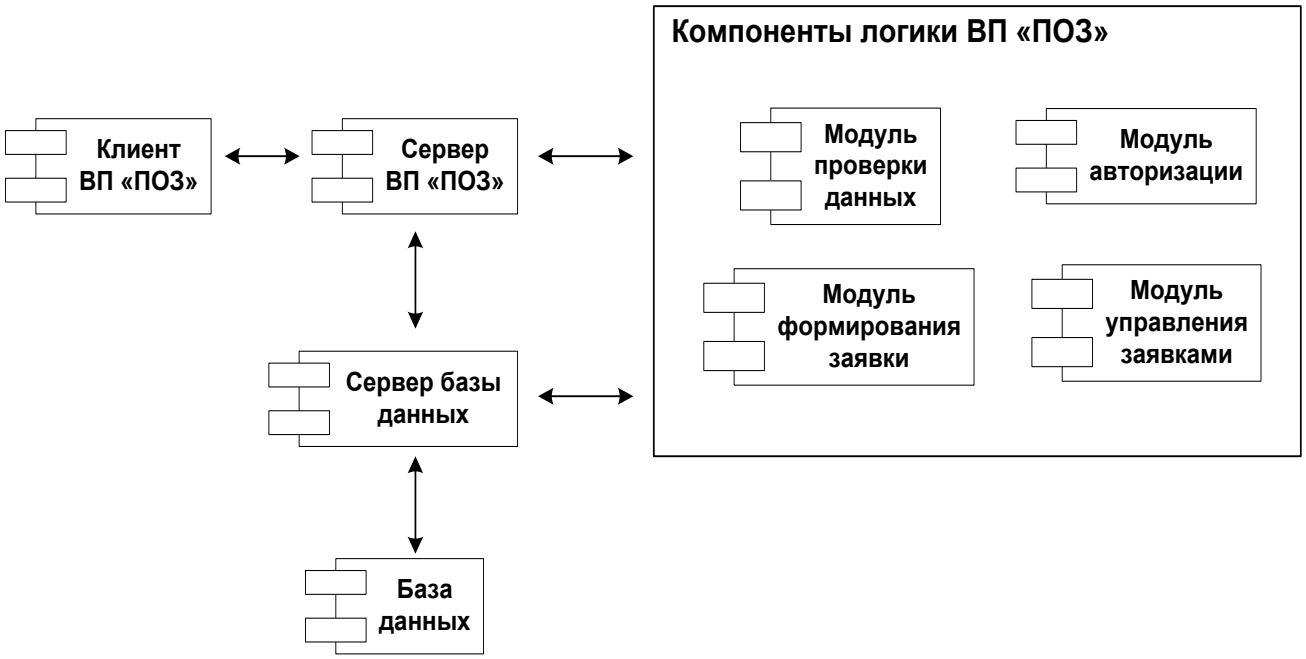


Рисунок 11 – Диаграмма компонентов веб-приложения «Прием онлайн-заказов»

Группа «Компоненты логики веб-приложения «Прием онлайн-заказов» определяет общую логику функционирования веб-приложения. Данная группа дает представление о том, какие модули реализованы на уровне исходного кода:

– Модуль проверки данных – проверяет заполненные данные в формах создания заявки и авторизации на корректность на уровне клиента;

– Модуль авторизации – реализует процедуру авторизации: отправляет запрос с авторизацией на сервер и проверяет их корректность на уровне бэкенда, формирует технические данные сессии;

– Модуль формирования заявки – на основе заполненных данных создает новую запись о заявке в базе данных;

– Модуль управления заявками – на уровне клиента и сервера отвечает за операции с заявками, выполняемые администратором – изменение статуса, удаление и др.

Окончив логическое моделирование веб-приложения, перейдем к этапу проектирования базы данных веб-приложения «прием онлайн-заказов».

2.3 Проектирование базы данных веб-приложения

На этапе логического проектирования базы данных веб-приложения «Прием онлайн-заказов» получена ее логическая модель, которая представлена на рисунке 12 Приведенная модель дает общее представление о взаимодействии сущностей Заказчик, Заявка и Администратор в предметной области.

Сущности Заказчик и Администратор связаны с сущностью Заявка связями типа «один-ко-многим». На уровне моделирования предметной области это означает, что один заказчик может подать более одной заявки на ремонт техники в ООО «Феликс», а также то, что один администратор может управлять множеством заявок.

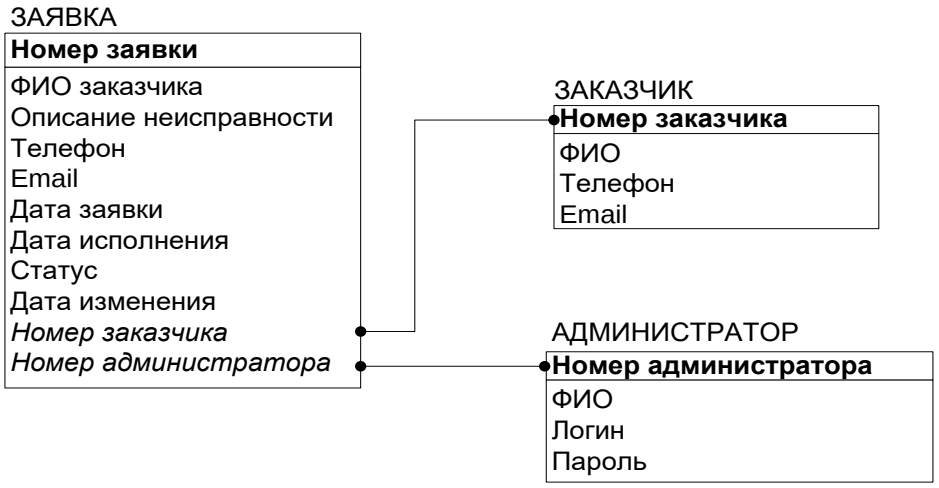


Рисунок 12 – Логическая модель БД веб-приложения

В реляционных моделях сущности связаны по первичному ключу – уникальному признаку, который однозначно идентифицирует экземпляр сущности [15]. Также каждая из сущностей характеризуется собственным

набором атрибутов – полей, значения которых в дальнейшем будут храниться в базе данных.

Опишем атрибуты каждой из сущностей логической модели базы данных в соответствующих таблицах. Атрибуты сущности Заказчик описаны в таблице 3.

Таблица 3 – Атрибуты сущности Заказчик

Название атрибута	Описание атрибута	Тип атрибута
Номер заказчика PK	Присвоенный заказчику корпоративный номер в ООО «Феликс», являющийся натуральным числом, например, 1, 2 и т. д. Первичный ключ, однозначно идентифицирующий заказчиков.	Счетчик
ФИО	Фамилия, имя и отчество заказчика в текстовом формате, например, «Сидоренко Борис Полиграфович».	Строка
Телефон	Номер телефона заказчика для связи в произвольном формате, например, «8(911)0008800».	Символьный
Email	Адрес электронной почты заказчика для связи в формате: «*@.*», например, «Ex2@mail.ru».	Символьный
Номер заявки	Номер заявки, соответствующей заказчику. Внешний ключ для организации связи типа «один-ко-многим».	Символьный

Таблица 4 описывает атрибуты сущности Заявка.

Таблица 4 – Атрибуты сущности Заявка

Название атрибута	Описание атрибута	Тип атрибута
Номер заявки PK	Присвоенный заявке корпоративный номер в ООО «Феликс». Формат номера: «Год_хххх», например, 2023-0001. Первичный ключ, однозначно идентифицирующий заявки.	Символьный
ФИО заказчика	Фамилия, имя и отчество заказчика в текстовом формате.	Строка
Описание неисправности	Описание неисправности техники от заказчика в текстовом формате, например, «На ноутбуке все пропало». Отражает суть проблемы обращения в организацию.	Текст
Телефон	Номер телефона заказчика для связи в произвольном формате.	Символьный
Email	Адрес электронной почты заказчика для связи в формате: «*@.*».	Символьный
Дата заявки	Дата создания заявки в формате: «чч.мм.гггг», например, 05.02.2023.	Дата
Дата исполнения	Дата исполнения заявки в формате: «чч.мм.гггг», например, 08.02.2023.	Дата
Статус	Статус заявки, соответствующий одному из возможных значений: «новая», «в работе», «в архиве», «удалить».	Выпадающий список

Продолжение таблицы 4

Дата изменения	Дата изменения заявки в формате: «чч.мм.гггг чч:мм:сс», например, 08.02.2023 20:12:00.	Дата/Время
Номер заказчика	Внешний ключ для организации связи «один-ко-многим» с родительской таблицей Заказчики.	Счетчик
Номер администратора	Внешний ключ для организации связи «один-ко-многим» с родительской таблицей Администраторы.	Счетчик

Таблица 5 описывает атрибуты сущности Администратор.

Проектирование базы данных на логическом уровне окончено. Перейдем к разработке физической модели базы данных.

Таблица 5 – Атрибуты сущности Администратор

Название атрибута	Описание атрибута	Тип атрибута
Номер администратора PK	Присвоенный администратору корпоративный номер в ООО «Феликс», являющийся натуральным числом, например, 1, 2 и т. д. Первичный ключ, однозначно идентифицирующий администраторов.	Счетчик
ФИО	Фамилия, имя и отчество администратора в текстовом формате.	Строка
Логин	Логин, состоящий из латинских букв, цифр или специальных символов. Ограничения: макс. длина 10 символов, не допускаются русскоязычные буквы.	Символьный

Продолжение таблицы 5

Пароль	Пароль, состоящий из латинских букв, цифр или специальных символов. Ограничения: макс. длина 10 символов, не допускаются русскоязычные буквы.	Символьный

На основе спроектированной на предыдущем этапе логической модели, разработана ER-диаграмма физической модели БД веб-приложение «Прием онлайн-заказов», рисунок 13.

Для каждой сущности обозначен список ее атрибутов. Эта информация в дальнейшем будет храниться и обрабатываться в базе данных веб-приложение «Прием онлайн-заказов».

Помимо первичного ключа для сущности Заявка предусмотрены два внешних ключа – «Номер заказчика» и «Номер администратора». В реляционных моделях внешние ключи используются для реализации отношений (связей) – в данном случае имеют место связи типа «один-ко-многим».

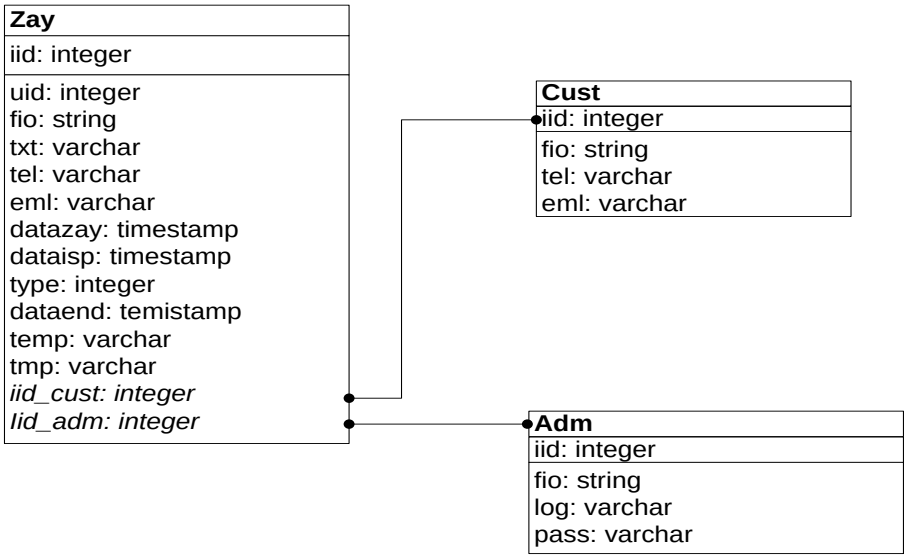


Рисунок 13 – Физическая модель БД веб-приложения

Полученная логическая и физическая модели БД помогут на этапе практической разработки базы данных веб-приложения «Прием онлайн-заказов» с использованием возможностей выбранной СУБД.

2.4 Требования к аппаратно-программному обеспечению веб-приложения «Прием онлайн-заказов»

На данном этапе исследования поставим задачу оценки существующей ИТ-инфраструктуры в ООО «Феликс» как организации-объекта, куда предполагается внедрить разработанное веб-приложение «Прием онлайн-заказов».

На этапе анализа предметной области выяснено, что текущая ИТ-инфраструктура для приема заказов предполагает хаотический и не системный подход. Графически такая инфраструктура может быть представлена следующим образом, рисунок 14.

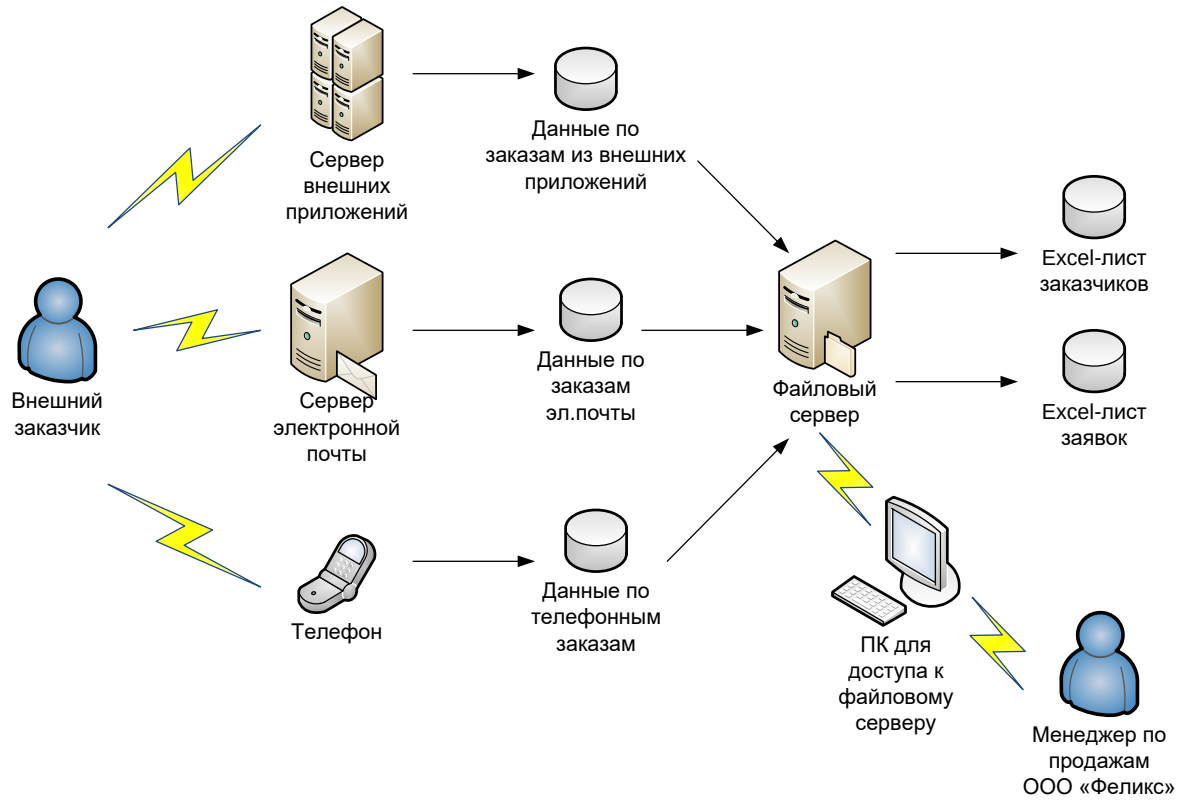


Рисунок 14 – Схема ИТ-инфраструктуры приема заказов в ООО «Феликс»

Для схемы на рисунке 14 можно отметить отсутствие четко выраженной архитектуры как таковой. Рисунок 15 схематически описывает текущий подход к обработке заказов в ООО «Феликс».

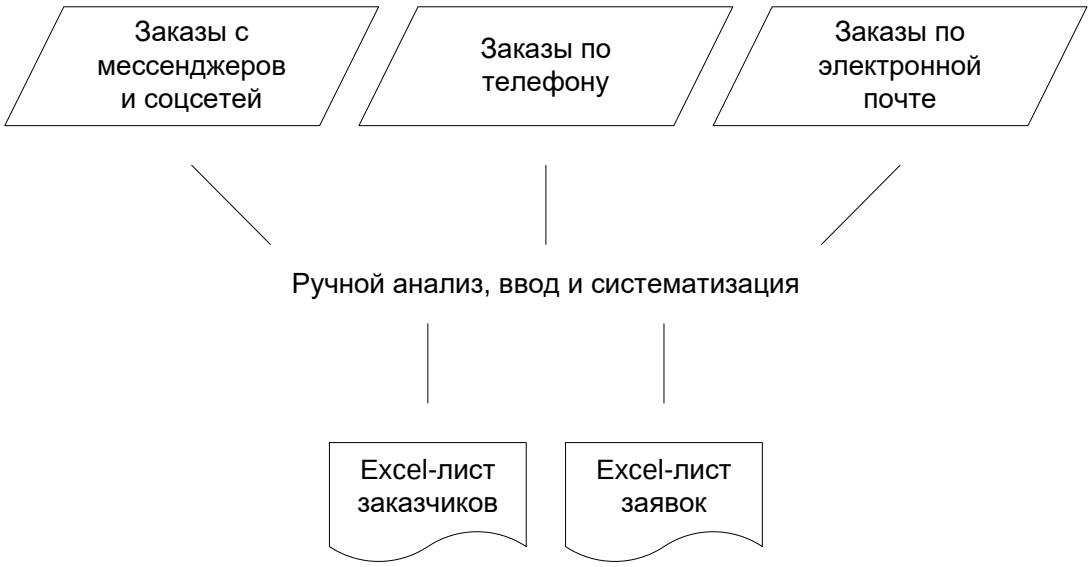


Рисунок 15 – Текущий подход к обработке заявок в ООО «Феликс»

Данные по заказам, поступающие из внешних каналов связи после ручной обработки сохраняются на корпоративном файловом сервере. Используя для доступа к серверу ПК, менеджер формирует Excel-листы с соответствующими итоговыми данными.

Существенные недостатки такого подхода проанализированы на этапах концептуального и функционального моделирования (хаотичность хранения данных, риск потери заявок, влияние человеческого фактора и т.д.). Еще одним существенным недостатком является отсутствие явной и четкой технической архитектуры.

В качестве решения по улучшению предложена клиент-серверная архитектура с концепцией единого консолидированного хранилища. Данная архитектура разработана на этапе логического моделирования и представлена на рисунке 15 Очевидные преимущества предложенной архитектуры, следующие:

– понятный и широко распространенный тип архитектуры, используемый при реализации веб-приложений;

– устранение хаотичности и исключение риска потери данных за счет использования единого консолидированного хранилища;

– повышение безопасности корпоративных данных;

– повышение надежности и защищенности общей архитектуры;

– возможность масштабирования архитектуры;

– возможность перехода на облачный способ хранения данных при потребности (например, в случае отказа оборудования).

Используемый в рамках предложенной архитектуры подход к обработке заявок схематически представлен на рисунке 16.

При постановке требований к веб-приложению «Прием онлайн-заказов» следует учитывать особенности его технической архитектуры и используемые в основе технологии.

Рисунок 16 – Предлагаемый подход к обработке заявок

Программное обеспечение будет выполнено на основе веб-технологий и предполагает кроссплатформенное использование за счет возможности запуска на различных клиентских устройствах с наличием веб-браузера и выходом в интернет.

Для реализации предложенной архитектуры клиент-сервер необходимо использовать следующее программное обеспечение:

– редактор исходного кода, поддерживающий синтаксис фронтенд и бэкенд-языков веб-технологий;

– веб-браузер (внешний обозреватель) для запуска и тестирования приложения, для отладки исходных кодов веб-приложение «Прием онлайн-заказов»

А также следующее аппаратное обеспечение:

– ПК стандартной конфигурации с возможностью выхода в интернет.

Технические характеристики ПК, который планируется использовать для разработки веб-приложения, приводятся в таблице 6.

Описанные в таблице технические характеристики ПК позволят полноценно использовать все функции программного обеспечения для разработки, объективно тестировать веб-приложение «Прием онлайн-заказов» с требуемым быстродействием.

Таблица 6 – Технические характеристики ПК для разработки веб-приложения «Прием онлайн-заказов»

Наименование	Требование
Операционная система	Ubuntu
Процессор	AMD A10-9620P RADEON R5, 10 COMPUTE CORES 4C+6G
Тактовая частота	2,5 ГГц
ОЗУ	8 Гб
Видеопамять	512 МБ
Разрешение экрана	1366x768
Доступное пространство на жестком диске	589 ГБ

Выводы к разделу

Данный раздел посвящен логическому проектированию веб-приложения «Прием онлайн-заказов».

На начальном этапе логического проектирования веб-приложения выполнено обоснование выбора технологии для разработки логических моделей. Выбраны следующие методы разработки логической модели:

- для формализации архитектуры веб-приложения – обозначения из теории проектирования сетей;
- для формализации компонентов веб-приложения их взаимодействия – диаграмма компонентов UML;
- для разработки логической и физической моделей базы данных – инструмент ER-диаграмм.

По результатам проведения работ по логическому моделированию получены:

- схема архитектуры веб-приложения «Прием онлайн-заказов»;
- диаграмма компонентов UML веб-приложения;

На отдельном этапе исследования выполнено комплексное проектирование базы данных приложения. По результатам проектирования разработаны:

- ER-диаграмма логической модели БД веб-приложения «Прием онлайн-заказов»;
- ER-диаграмма физической модели БД веб-приложения;
- таблицы с описанием атрибутов сущностей предметной области.

На заключительном этапе раздела проведен анализ требований к программно-аппаратному обеспечению веб-приложения «Прием онлайн-заказов». Графически представлена текущая ИТ-инфраструктура обработки заказов, используемая в ООО «Феликс». Проанализированы недостатки данной инфраструктуры и описаны преимущества предложенного архитектурного решения на основе внедрения веб-приложения «прием онлайн-заказов».

3. Физическое проектирование веб-приложения «Прием онлайн-заказов»

3.1 Выбор архитектуры веб-приложения

Для реализации веб-приложения «Прием онлайн-заказов» выбрана классическая архитектура «клиент-сервер», широко используемая для построения кроссплатформенных веб-приложений [16]. Выбранная архитектура предполагает два основных звена – клиент (1 звено) и сервер (2 звено), поэтому такая архитектура еще называется двухзвенной, рисунок 17.

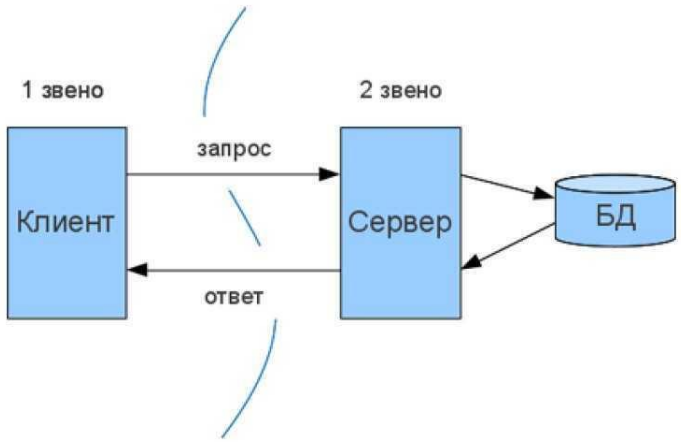


Рисунок 17 – Порядок взаимодействия звеньев в двухзвенной архитектуре клиент-сервер

Сервер – как правило, представляет собой высокопроизводительный компьютер, предоставляющий некоторый сервис/группу сервисов в пределах глобальной сети Интернет или частной локальной сети.

Клиент – устройство, отправляющее запрос серверу посредством веб-браузера и получающее от сервера данные, содержащие запрашиваемую пользователя приложения информацию. Как-правило, это компьютер (ноутбук или десктоп) либо мобильное устройство, для которого доступен функционал

браузера. Этот фактор определяет основную особенность и достоинство веб-приложений – кроссплатформенность.

Схема клиент-серверной архитектуры веб-приложения разработана на этапе логического моделирования (см. рисунок 17).

Клиент-серверная архитектура веб-приложения предполагает разработку в двух классических направлениях:

– Фронтенд – клиентская часть веб-приложения (интерфейс), с которой пользователь взаимодействует, отправляет запросы на сервер, получает от него ответ и формирует данные для отображения в браузере пользователя.

– Бэкенд – серверная часть веб-приложения, непосредственно реализующая его функциональные возможности.

Приложение дополнительно использует возможности базы данных, которая поддерживается отдельным сервером. С учетом этого следует говорить, что используемая архитектура клиент-сервер является двухуровневой [17].

На следующем этапе написания работы будет сделан выбор технологий и средств разработки веб-приложения «Прием онлайн-заказов».

3.2 Выбор технологий и средств разработки веб-приложения

При выборе инструментальных средств для практической разработки веб-приложения «Прием онлайн-заказов», согласно Приложения А следует учитывать описанные ранее особенности архитектуры.

Для разработки фронтенда предлагается использовать традиционные языки веб-технологий: HTML, CSS, JavaScript. Подробное описание выбранных технологий приводится в таблице 7.

Таблица 7 – Языки и технологии, выбранные для разработки фронтенда веб-приложения «Прием онлайн-заказов»

Язык/технология	Описание
HTML	Язык разметки гипертекста, представляющий основанную на тегах систему разметки документов, определяющую структуру и отдельные компоненты того, что в итоге компилируется в объектную модель документа, или DOM.
JavaScript	Высокоуровневый интерпретируемый скриптовый язык, с помощью которого прописывается и выполняется все поведение приложения и его реакция на действия пользователя.
CSS	Каскадная таблица стилей, являющаяся фреймворком для стилизации приложения, с помощью которого идентифицируют и оформляют различные части DOM видимой области страницы. CSS предоставляет возможности вроде выбора элементов по их ID, классу и отношению к другим элементам DOM.

Преимущества выбранного языка JavaScript, выгодно отличающие его от аналогов [18]:

– универсальность – подходит для разработки приложений любого типа;

– способен работать на стороне серверной части;

– интерпретируется браузером без дополнительных пакетов;

– повышает интерактивность веб–приложения;

– большое количество обучающих материалов в сети.

Для разработки бэкенда предлагается использовать популярный бэкенд-язык программирования PHP. Преимущества данного инструмента [19]:

– мощность и гибкость – оптимален как для разработки масштабных сервисов, так и небольших веб–приложений;

– надежный и давно зарекомендовавший себя бэкенд–язык;

– свободно распространяемый и полностью бесплатный;

– относительно прост в освоении и использовании;

– популярность и большое интернет–комьюнити.

Выбор языков и технологий для разработки фронтенд и бэкенд-части выполнен. Перейдем к выбору инструментальных средств для реализации веб-приложения. Для разработки серверной части достаточно наличие веб-сервера, совмещающего в себе также функционал сервера базы данных (СУБД). Разрабатываемое приложение не имеет особых требований к производительности сервера, размеру дискового пространства. Суммарный объем файлов (основной html-страницы фронтэнда, исполняемых на бэкэнде приложения скриптов на языке программирования PHP, файлов используемых на фронтэнде библиотек JavaScript, файлов стилей CSS и файлов изображений) не превышает одного мегабайта.

Как правило, для написания исходного кода используется редактор кода, поставляемый с IDE. Интегрированные среды разработки (IDE) и редакторы кода (CE) – это программные приложения, используемые для написания и редактирования кода. IDE обычно имеют больше функций, чем редакторы кода, но некоторые редакторы кода могут быть настроены так, чтобы иметь функции, аналогичные функциям IDE.

Для написания и редактирования исходных кодов планируется использовать редактор кода со встроенными функциями подсветки синтаксиса распространенных языков программирования – «Kwrite» для Linux-систем. Функциональных возможностей данного редактора вполне достаточно для написания кодов на языках программирования JavaScript и PHP, файла, либо блока описания стилей CSS, а также HTML-разметки основного документа.

Отказ от IDE и использование редактора для реализации проекта приложения обусловлен тем, что для исполнения кода на JavaScript и PHP не требуется его компиляция (JS – высокоуровневый интерпретируемый скриптовый язык), а проверить функциональность JS-скрипта и наличие ошибок в коде можно простой перезагрузкой страницы. При этом, веб-страница может отображаться без потери своего функционала даже локально, без размещения ее на веб-сервере.

Таким образом, в качестве дополнительного инструмента разработки, заменяющим возможности IDE предлагается использовать веб-браузер, например, Mozilla Firefox. С использованием веб-браузера будут решаться следующие задачи разработки веб-приложения:

– тестирование и отладка кода;

– проверка наличия синтаксических и логических ошибок (для браузера Mozilla этот функционал вызывается нажатием клавиши F12);

– проверка работоспособности скриптов.

Использование описанных возможностей браузера значительно облегчит разработку приложения и не потребует установки дополнительного программного обеспечения.

Выбранные инструменты для разработки с перечнем решаемых задач описаны в таблице 8.

Таблица 8 – Инструменты для разработки веб-приложения

Инструмент	Решаемые задачи
Редактор кода Kwrite	– написание исходных кодов фронтенда; – написание исходных кодов бэкенда.
Веб-браузер Mozilla Firefox	– тестирование и отладка кода; – проверка наличия синтаксических и логических ошибок (F12); – проверка работоспособности скриптов.

Дополнительным компонентом архитектуры веб-приложения «Прием онлайн-заказов» является база данных. На следующем этапе разработки осуществим выбор СУБД для веб-приложения.

3.3 Выбор СУБД для веб-приложения

В ходе выбора СУБД для веб-приложения проведен сравнительный анализ существующих популярных решений систем управления базами данных. Результаты сравнительного анализа представлены в таблице 9.

Таблица 9 – Сравнительный анализ существующих решений СУБД

Название СУБД	Достоинства	Недостатки
Oracle	– инновационные технологии и функции; – высокая надежность; – мощная корпоративная поддержка; – оптимальная для масштабных проектов.	– полноценное использование функций будет платным; – требований значительных вычислительных ресурсов; – нецелесообразно использовать для крупных проектов.
MySQL	– бесплатное распространение; – оптимальная для небольших проектов; – простая установка на Unix-системах за счет включения данной СУБД в стандартные репозитории дистрибутивов; – хорошо документирована.	– некоторые функции, типовые для других СУБД приходится реализовывать дополнительно (например, создание инкрементных резервных копий); – отсутствуют встроенные XML и OLAP.

Продолжение таблицы 9

MongoDB	– быстродействие; – простота использования; – собственный подход к хранению данных на основе коллекций.	– предполагает использование собственной NoSQL модели, которую нужно дополнительно осваивать; – привычный SQL не используется в качестве языка запросов; – долгая установка и настройка.
PostgreSQL	– поддержка формата JSON; – легко масштабируется; – поддерживает нагруженную обработку (до терабайтов данных); – множество предопределенных функций.	– слабо проработанная документация; – низкая производительность для запросов чтения и пакетных операций; – относительно сложна в освоении неподготовленным пользователем.

Рассмотрим ряд особенностей веб-приложения с точки зрения обработки данных, которые помогут выбрать оптимальный тип СУБД:

– веб-приложение «Прием онлайн-заказов» не является крупным масштабным проектом – это проект среднего уровня;

– в приложении не предполагается обработка огромных объемов данных;

– база данных в веб-приложения «Прием онлайн-заказов» основана на классической реляционной модели данных;

– разработка приложения выполняется на ПК под управлением ОС Linux;

– для разработки целесообразно использовать бесплатно распространяемое решение СУБД.

С учетом описанных выше особенностей проекта можно говорить о том, что оптимальным решением СУБД является MySQL.

Таким образом, в качестве базы данных для хранения информации предлагается использовать СУБД MySQL. Выбор данной СУБД также обусловлен её широким распространением, легкостью интеграции с различными серверными языками программирования, в том числе с выбранным в качестве основного языка для создания бэкенда – языка программирования PHP.

3.4 Разработка веб-приложения

3.4.1 Разработка базы данных веб-приложения

Согласно инфологической и физической модели, разработанных на этапе моделирования базы данных (см рисунки 2.4, 2.5), БД веб-приложения должна содержать сведения о сущностях в виде таблиц:

Заказчики: Номер заказчика (PK), ФИО, Телефон, Email.

Заявки: Номер заявки (PK), ФИО заказчика, Описание неисправности, Телефон, Email, Дата заявки, Дата исполнения, Статус, Дата изменения, Номер заказчика (ВК), Номер администратора (ВК).

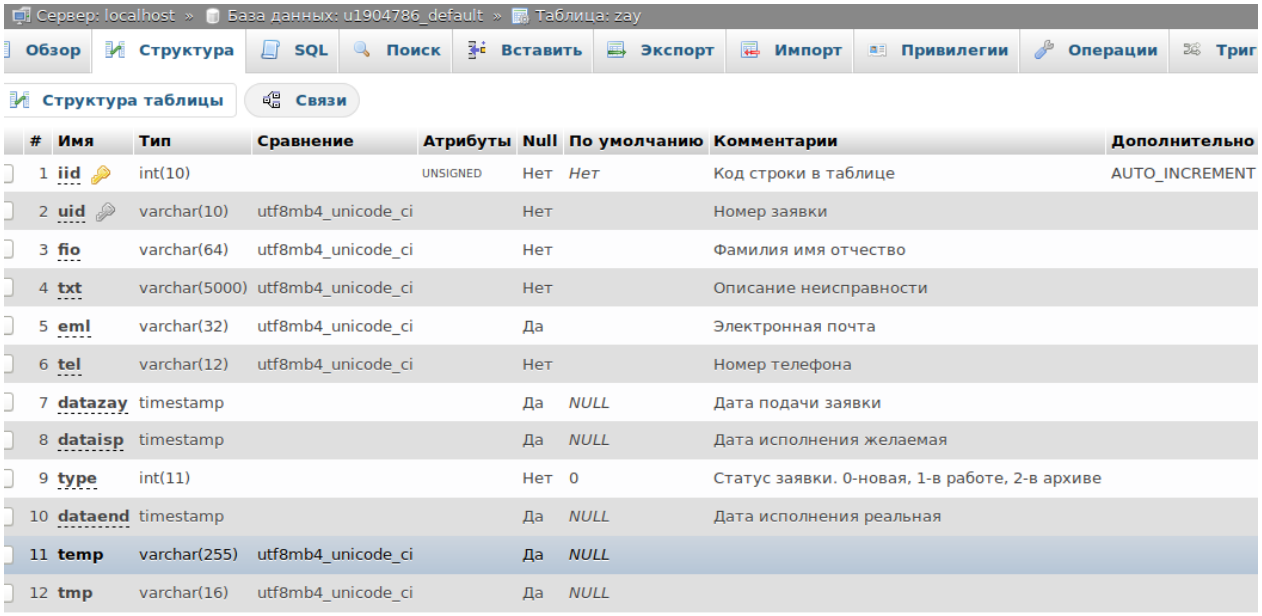
Администраторы: Номер администратора (PK), ФИО, Логин, Пароль.

Создание таблиц и заполнение их данными в базе производилось с использованием приложения PhpMyAdmin [20]. Это веб–приложение с

открытым кодом, написанное на языке PHP и представляющее собой веб–интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL.

Работа со всеми таблицами веб-приложения (создание, наполнение данными, задание типов полей для атрибутов) ведется по единому аналогичному сценарию. Рассмотрим алгоритм создания таблиц в БД на примере таблицы с заявками, как самой емкой и информативной составляющей БД.

Структура и поля таблицы с заявками ООО «Феликс» в интерфейсе phpMyAdmin продемонстрирована на рисунке 18.



#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
1	iid	int(10)		UNSIGNED	Нет	*Нет*	Код строки в таблице	AUTO_INCREMENT
2	uid	varchar(10)	utf8mb4_unicode_ci		Нет		Номер заявки	
3	fio	varchar(64)	utf8mb4_unicode_ci		Нет		Фамилия имя отчество	
4	txt	varchar(5000)	utf8mb4_unicode_ci		Нет		Описание неисправности	
5	eml	varchar(32)	utf8mb4_unicode_ci		Да		Электронная почта	
6	tel	varchar(12)	utf8mb4_unicode_ci		Нет		Номер телефона	
7	datazay	timestamp			Да	*NULL*	Дата подачи заявки	
8	dataisp	timestamp			Да	*NULL*	Дата исполнения желаемая	
9	type	int(11)			Нет	0	Статус заявки. 0-новая, 1-в работе, 2-в архиве	
10	dataend	timestamp			Да	*NULL*	Дата исполнения реальная	
11	temp	varchar(255)	utf8mb4_unicode_ci		Да	*NULL*		
12	tmp	varchar(16)	utf8mb4_unicode_ci		Да	*NULL*		

Рисунок 18 – Таблица со сведениями о заявках ООО «Феликс»

Первичный индекс таблицы заявок (поле iid) служит для обеспечения уникальности записей и содержит параметр AUTO_INCREMENT, который позволяет при добавлении новых записей (командой INSERT) автоматически увеличивать значение поля iid на единицу. Второй индекс в таблице создан по полю uid – в данном поле содержится уникальный номер заявки, формируемый на серверной части PHP скрипта бэкэнда приложения. По uid осуществляется

поиск необходимой заявки в базе данных для изменения ее статуса (например, изменение статуса с значения «новая» на статус «в работе» или «в архиве»).

Список SQL-запроса для создания таблицы в СУБД MySQL и необходимых индексов для ускорения работы поисковых запросов представлен в таблице 10.

Таблица 10 -- SQL-запрос для создания таблицы заявок

```
DROP TABLE IF EXISTS u1904786_default.zay;
CREATE TABLE  u1904786_default.zay (
  `iid` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT 'Код строки в таблице',
  `uid` varchar(10)  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '' COMMENT 'Номер заявки',
  `fio` varchar(64)  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '' COMMENT 'Фамилия имя отчество',
  `txt` varchar(5000) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '' COMMENT 'Описание неисправности',
  `eml` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT '' COMMENT 'Электронная почта',
  `tel` varchar(12) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '' COMMENT 'Номер телефона',
  `datazay` timestamp NULL DEFAULT NULL COMMENT 'Дата подачи заявки',
  `dataisp` timestamp NULL DEFAULT NULL COMMENT 'Дата исполнения желаемая',
  `type` int NOT NULL DEFAULT '0' COMMENT 'Статус заявки. 0-новая, 1-в работе, 2-в архиве',
  `dataend` timestamp NULL DEFAULT NULL COMMENT 'Дата исполнения реальная',
  `temp` varchar(255)  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `tmp` varchar(16)  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (iid) USING BTREE,
  KEY uid (uid) USING BTREE
)  ENGINE=InnoDB  DEFAULT  CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
```

Для понимания назначения полей в таблице, большинство из них имеют текстовый комментарий, который отображается в соответствующем столбце при работе с таблицами базы данных посредством приложения PhpMyAdmin.

При запуске скрипта SQL-запроса, представленного на рисунке 3.2, перед созданием необходимой для работы приложения таблицы, скрипт проверяет наличие такой таблицы в СУБД и при необходимости удаляет её командой «drop table». Поля, которые используются для хранения данных, содержащих текстовые значения, такие как «fio» (фамилия, имя, отчество), «txt» (описание неисправности) имеют тип «varchar» и для их корректного отображения во фронтэнде приложения (браузере), таким полям задана кодировка «utf8». Также задана необходимая размерность (длина для строк).

Поля, предназначенные для хранения календарных дат (даты обращения, даты подачи заявки), желаемой и реальной дат исполнения (закрытия) заявки имеют тип «timestamp», который помимо даты (день, месяц, год) может содержать и сведения о времени (часы, минуты, секунды). При формировании заявки через форму в приложении, в поле с датой подачи заявки посредством скрипта PHP на бэкэнде приложения, в базу данных заносится и информация о времени ее заведения. При закрытии заявки и изменении ее статуса на значение «в архиве», аналогично автоматически заполняется поле «dataend» с указанием времени операции закрытия заявки.

Поле в таблице с именем «type» (статус заявки) является цельночисленным словарным и может принимать следующие значения:

– 0 – «новая»;
– 1 – «в работе»;
– 2 – «в архиве»;
– 3 – «удалить».

Кроме того, помимо полей, непосредственно участвующих в работе приложения, в таблице созданы два резервных текстовых поля («temp» и «tmp») для технических целей на стороне бэкенда.

3.4.2 Разработка фронтенда веб-приложения

При разработке веб-приложения «Прием онлайн-заказов», формирование дизайн-макета UX/UI-интерфейса производилось, в первую очередь, с учетом требований логики работы и функциональности веб-приложения, а также обеспечения интуитивной понятности компонентов интерфейса. Ввиду отсутствия явных требований к дизайну приложения от заказчика (ООО «Феликс»), интерфейс приложения выполнен без учета требований к дизайнерскому наполнению графического представления.

В случае необходимости интеграции приложения в существующий веб-сайт, либо иное приложение заказчика, имеющее свое уникальное дизайнерское наполнение, каждому элементу интерфейса в приложении присвоен уникальный код тега «id» (продублирован тегом «name» для совместимости с некоторыми устаревшими браузерами), а стиль отображения элементов интерфейса с каждым кодом «id» описан в блоке описания стилей «style» основного файла приложения (отдельного файла описания стилей CSS) и при корректировке в блоке описания стилей формата отображения соответствующих элементов, может быть легко адаптирован под требования заказчика к графическому отображению элементов интерфейса приложения.

Из особенностей дизайнерского функционала, в соответствии с техническим заданием от заказчика, отдельным графическим стилем выполнены буквы логотипа, стилизованные под объемный текст. Логотип заказчика (текст «ООО Феликс») расположен в левой верхней части экрана, и отображается во всех режимах работы приложения (режим подачи заявки и в режиме администратора). Вследствие того, что данный графический объект отображается во всех режимах работы приложения (пользователь и администратор), он помимо функции логотипа, выполняет функции ссылки для перехода на главную (стартовую) страницу приложения (переключение в режим пользователя для подачи заявки).

Стартовая страница веб-приложения «Прием онлайн-заказов» с формой для подачи заявки представлена на рисунке 19.

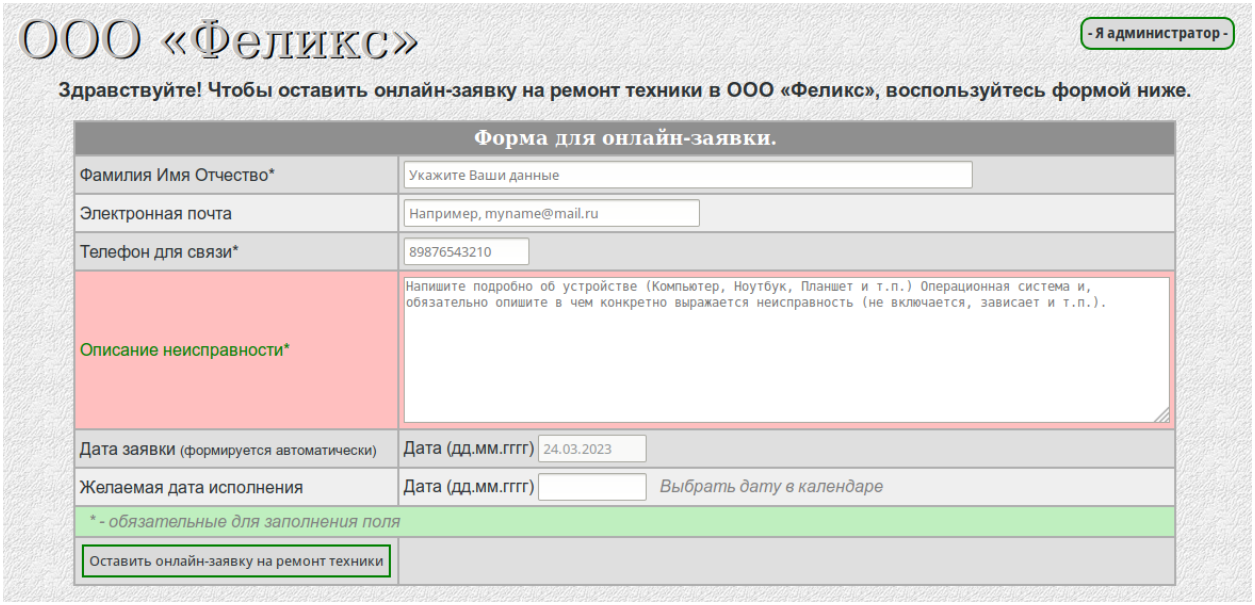


Рисунок 19 – Дизайн стартовой страницы веб-приложения

Также, в соответствии с требованиями заказчика, создан графический объект с эффектом «бегущая строка», содержащая текст «Здравствуйте! Чтобы оставить онлайн-заявку на ремонт техники в ООО «Феликс», воспользуйтесь формой ниже», который появляется при первом запуске приложения, а также, при переходе на главную страницу посредством нажатия на форму с текстом логотипа («ООО Феликс»). Объект в большей мере служит для привлечения внимания и информирует пользователя приложения о возможности подать онлайн-заявку на ремонт техники, заполнив форму заявки, которая расположена ниже объекта «бегущая строка».

В форме подачи заявки добавлена возможность отмечать желаемую дату выполнения заказа посредством выбора ее в графическом изображении календаря, которое появляется/исчезает ниже формы для заполнения заявки при нажатии на соответствующую ссылку в форме. Календарь выполнен с использованием стандартного модуля datepicker библиотеки jquery-ui.js, подключенной в программном коде JavaScript приложения. Там же, в

соответствующем разделе кода, подключается файл для отображения стилей календаря (jquery-ui.css).

Для корректного отображения календаря с учетом локализации (названий месяцев, дней недели и т. п.), при первом запуске приложения в блоке скрипта «локализация datepicker» производится его первичная инициализация: при этом заполняются соответствующие переменные и массивы модуля.

Окно выбора даты из графической формы календаря представлено на рисунке 20.

После заполнения всех обязательных полей (в форме помечены знаком звездочка – *), на месте кнопки «Оставить заявку на ремонт техники» отображается кнопка «Подтверждаю, что заявка заполнена правильно» и запрос согласия на обработку персональных данных.

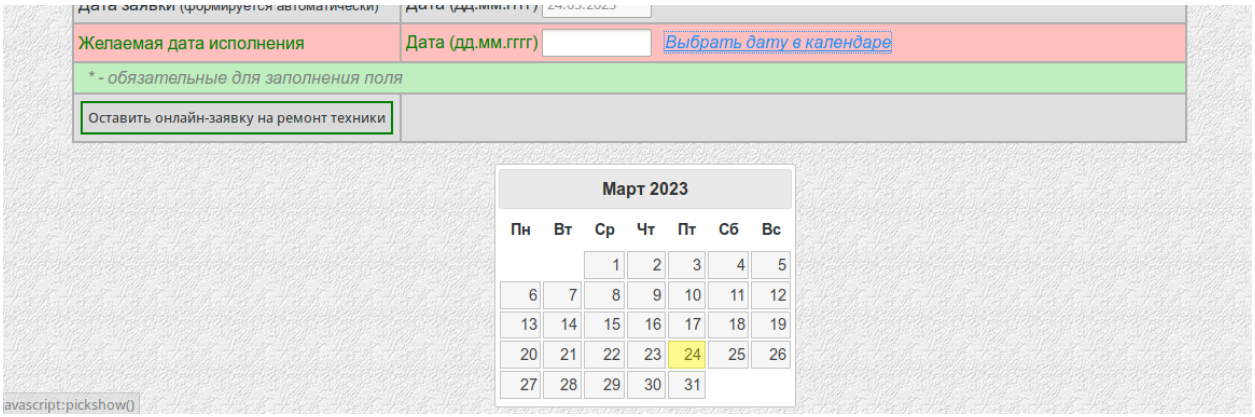


Рисунок 20 – Выбор даты из календаря

Данные с формы подачи заявки отправляются в базу приложения только после проставления «галочки» и нажатия на кнопку «Подтверждаю...». Форма для подтверждения отправки данных представлена на рисунке 21.

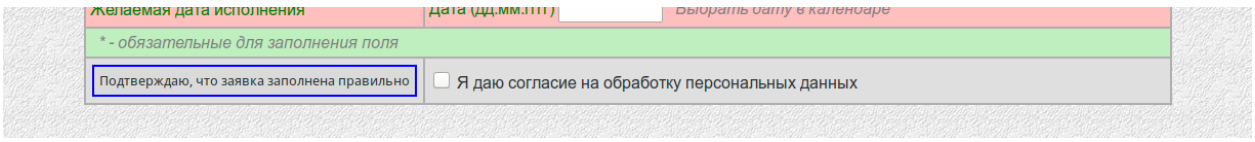


Рисунок 21 – Форма подтверждения отправки заявки

Веб-приложение «Прием онлайн-заказов» имеет два режима работы (режим пользователя и администратора). Переход из страницы режима пользователя в режим администратора осуществляется по кнопке «я администратор», расположенной в правом верхнем углу. При нажатии на кнопку, отображается страница авторизации, а на ее месте появляется кнопка «Выйти», которая отображается при дальнейшей работе в роли администратора и служит для перехода в режим пользователя.

Функционал администратора для работы с поступившими заявками доступен только после авторизации в системе – для этого необходимо в полях формы ввести логин и пароль. Окно авторизации представлено на рисунке 22.

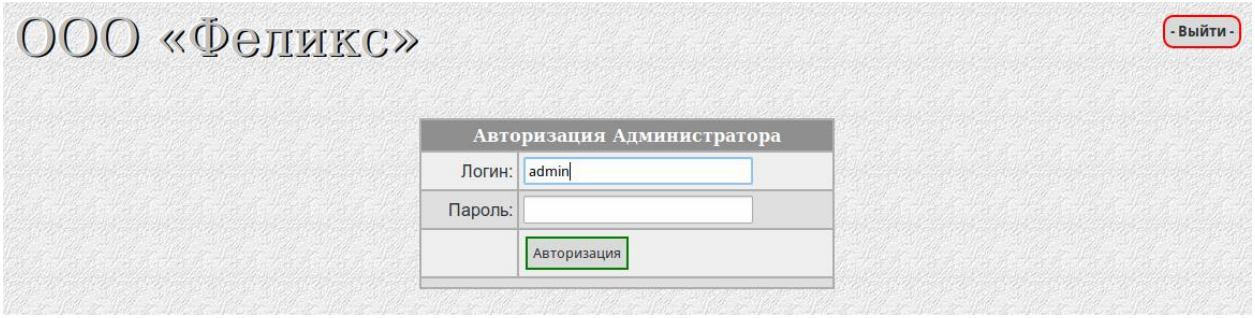


Рисунок 22 – Окно авторизации администратора

В режиме администратора в интерфейсе веб-приложения отображаются поступившие заявки в виде таблицы с полями, аналогичными полям на странице подачи заявки. В столбце «статус заявки» отображается на какой стадии обработки находится поступившая заявка («новая», «в работе» и «в архиве»). Здесь же реализована возможность удаления заявки.

В столбце «дата изменения» отображается дата и время изменения статуса заявки, например, при переходе из состояния «новая» в состояние «в работе». Изменение статуса заявки, а также ее удаление, производится нажатием на кнопку в столбце «статус заявки» на соответствующей заявке строке. При изменении статуса заявки, изменяется и цвет подсветки соответствующей строки.

Страница, содержащая таблицу со списком заявок в кабинете администратора, представлена на рисунке 23.

Кроме подсветки строк, содержащих информацию о заявках в соответствии с их статусом (этапом исполнения), даты исполнения для просроченных заявок подсвечиваются красным цветом для акцента внимания на них, что можно видеть на рисунке 23.

По умолчанию данные о заявках в таблице отображаются в порядке их поступления, что, как правило, совпадает с порядком номеров заявок. Для удобства работы с заявками, предусмотрена возможность сортировки всего списка по любому полю (столбцу) таблицы.

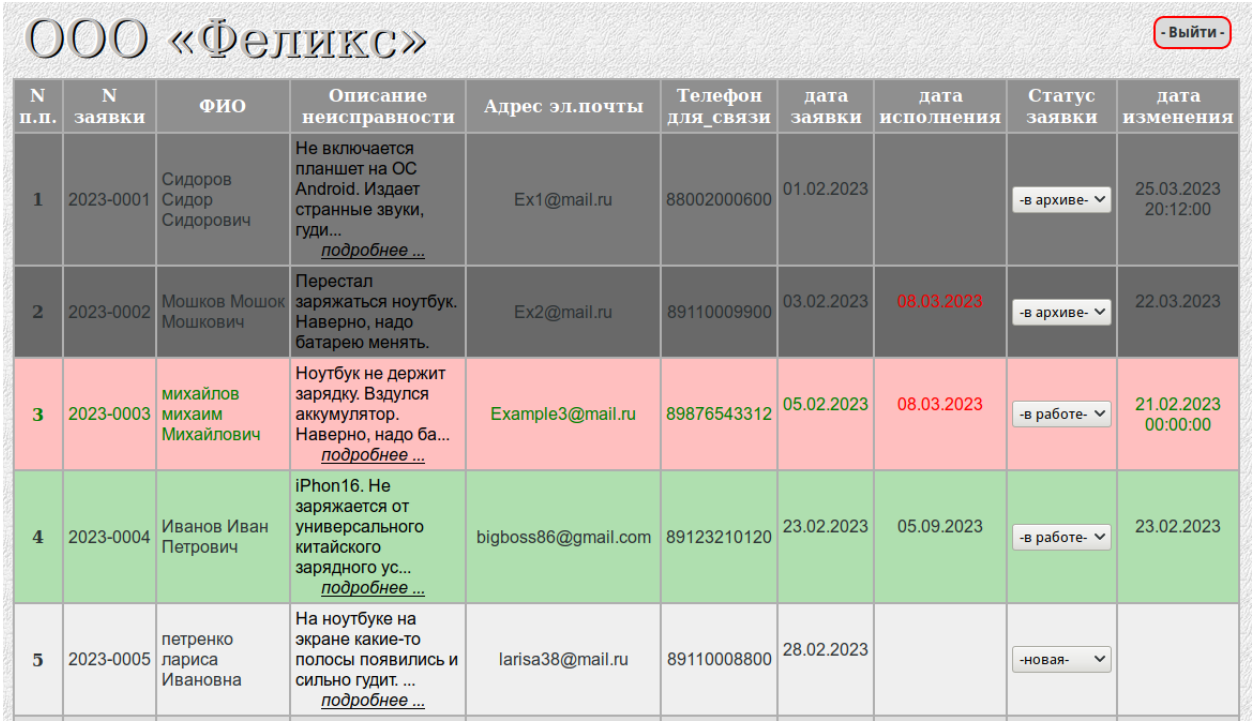


Рисунок 23 – Таблица заявок в кабинете администратора

При наведении курсора на название столбца в шапке таблицы, текст подсвечивается, а при нажатии на название поля, производится сортировка списка. Список можно отсортировать по любому полю как по возрастанию, так и по убыванию значений. Рядом с названием столбца отображается знак

стрелки, указывающий на порядок сортировки. Пример сортировки по полю «ФИО» в порядке возрастания (по алфавиту) представлен на рисунке 24.

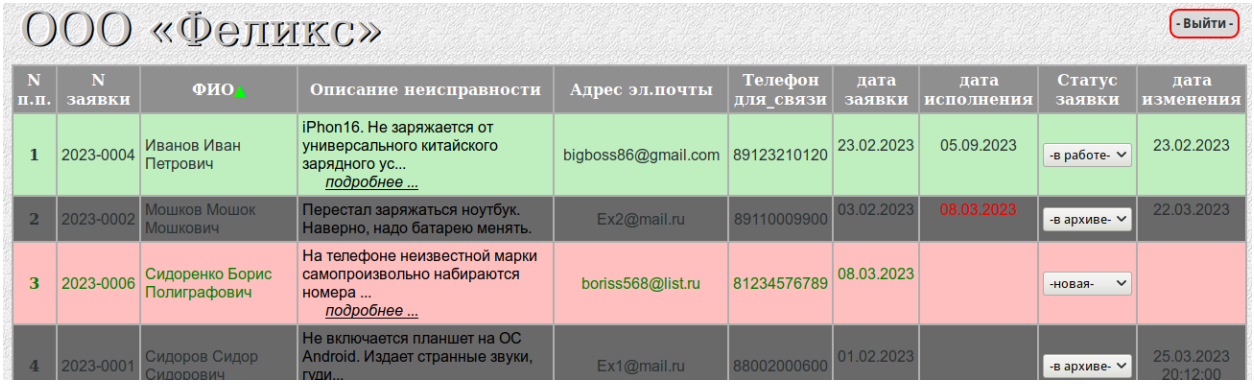


Рисунок 24 – Пример сортировки формы отображения заявок

При необходимости удаления заявки, например, ошибочно направленной или дублированной, при выборе в столбце «статус заявки» значения «удалить», значения в строке соответствующей заявки подсвечиваются другим цветом (красного оттенка) а текст в них отображается перечеркнутым. При этом, сверху формы (рядом с кнопкой «Выйти») отображаются кнопки для подтверждения удаления заявки, либо отмены действия. Пример удаления заявки (номер 2 в списке) представлен на рисунке 25.

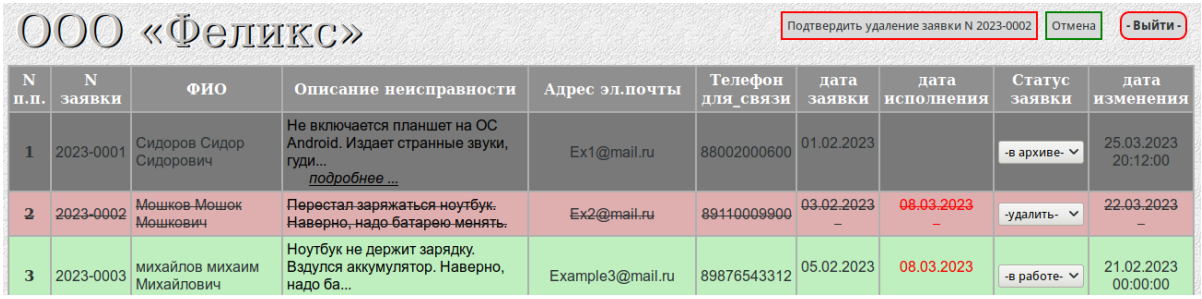


Рисунок 25 – Пример удаления заявки

Функционал клиентской части приложения практически полностью выполнен на скриптовом языке программирования JavaScript без использования сторонних дополнений и фреймворков, за исключением модуля jQuery, который служит для передачи данных между клиентской и серверной частями приложения (между фронтэндом и бэкэндом). Данные с сервера (с бэкэнд на фронтэнд) передаются по технологии AJAX в формате массивов данных JSON.

AJAX – метод построения интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером [21]. В результате, при обновлении данных, поступивших на фронтэнд приложения с его бэкэнда, открытая пользователем в браузере веб-страница не перезагружается полностью. За счет использования такого подхода веб-приложения становятся быстрее и удобнее в использовании.

JSON – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми [22]. Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования.

3.4.3 Разработка бэкенда веб-приложения

Серверная часть (бэкенд) веб-приложения выполнена на языке программирования PHP. При использовании приложения в составе частной локальной сети, установка языка PHP на веб-сервер не представляет трудностей. Кроме того, существуют готовые сборки, имеющие в своем составе веб-сервер с модулями PHP и MySQL. Установка таких сборок производится буквально «одним нажатием клавиши».

Для взаимодействия приложения с СУБД MySQL использовано стандартное расширение mysqli для языка программирования PHP.

MySQL Improved – это расширение для PHP, которое добавляет в язык программирования полную поддержку баз данных MySQL [23]. Данное

расширение поддерживает множество возможностей современных версий MySQL. Расширение, как правило, сразу подключено и не требует дополнительных настроек. Посмотреть установлено ли это расширение для PHP и текущую версию можно командой phpinfo().

Фрагмент кода PHP-сценария, в котором реализован процесс взаимодействия с СУБД MySQL с формированием выходных данных в формате JSON для последующей передачи информации фронтэнду приложения, представлен на рисунке 26.

```
18  $table = $kod;
19  $count = 0;
20  $error = FALSE;
21  $lines = array();
22
23  header('Content-Type: application/json');  // заголовок http ответа, указывает на формат json
24
25  // Create connection
26  $conn = new mysqli($serv, $user, $pass, $dbnm);
27
28  // Check connection
29  if ($conn->connect_error) { $error = TRUE; }
30  else
31  {
32  $conn->set_charset("utf8");
33  $sql = "SELECT * FROM $kod";
34  $result = $conn->query($sql);
35  $count  = $result->num_rows;
36
37  if ($count > 0) { while ($row = $result->fetch_array(MYSQLI_NUM)) { $lines[] = $row; } }
38  $conn->close();
39  }
40
41  $array = array();
42  $array['Table'] = $table;
43  $array['Count'] = $count;
44  $array['Error'] = $error;
45  $array['Lines'] = $lines;
46
47  $json = json_encode($array, JSON_PRETTY_PRINT);
48  echo $json;
49  exit;
50
```

Рисунок 26 – Листинг PHP-сценария работы с СУБД MySQL

Процесс работы скрипта PHP сценария с СУБД MySQL и формирование выходных данных в формате массива JSON (результат выборки в формате массива JSON представлен на рисунке 26) показан на примере команды «SELECT *» для выборки из таблицы «zay».

Рассмотрим подробнее сам сценарий (алгоритм работы) скрипта.

Строка 26 - установка подключения к серверу СУБД, передача необходимых для соединения с базой данных параметров (адрес хоста, где функционирует СУБД (как правило, это localhost), порт подключения (обычно имеет значение 3306) логин и пароль для авторизации, имя базы данных).

Строка 29 - контроль успешности подключения (корректность обработки сервером СУБД переданных в строке 26 параметров соединения). При неудачном соединении, параметр «Error», который в дальнейшем передается фронтэнду приложения, принимает значение «TRUE». По умолчанию данный параметр имеет значение «FALSE».

Строка 32 - установка кодовой страницы (utf-8) для корректного отображения текстовых данных, поступивших с сервера.

Строка 33 - формирование и передача в СУБД на исполнение SQL запроса (на примере запроса на чтение данных из таблицы — команда «SELECT *»).

Строки с 35 по 37 - получение от СУБД массива данных и занесение его в массив PHP сценария для дальнейшего формирования результирующих данных для передачи их на сторону фронтэнда приложения при успешном исполнении команды на стороне сервера СУБД. При этом параметр «Count», который так же передается на сторону фронтэнда, принимает значение, соответствующее количеству выбранных записей. По умолчанию данный параметр имеет нулевое значение.

Строка 38 — завершение работы с СУБД, корректное закрытие соединения.

Строки с 41 по 47 — формирование из полученного массива и других служебных параметров (имя базы данных, количество полученных записей, наличие ошибок) объекта в формате JSON для передачи его фронтэнду приложения.

Фрагмент сформированного в формате JSON ответа, поступившего с PHP сценария бэкэнда приложения на его фронтэнд, в окне отладки консоли браузера, представлен на рисунке 27.

Полученный со стороны бэкэнда в формате JSON ответ заносится в JS-массив для дальнейшего использования в работе фронтэнда веб-приложения «Прием онлайн-заказов». Ответ содержит шесть записей, соответствующих количеству поступивших заявок и будет из таблицы «zay» СУБД MySQL занесен в соответствующий JavaScript-массив фронтэнда.

Обеспечение защищенности корпоративных данных ООО «Феликс» сводится к реализации защиты от SQL-инъекций, как самой вероятной угрозы.

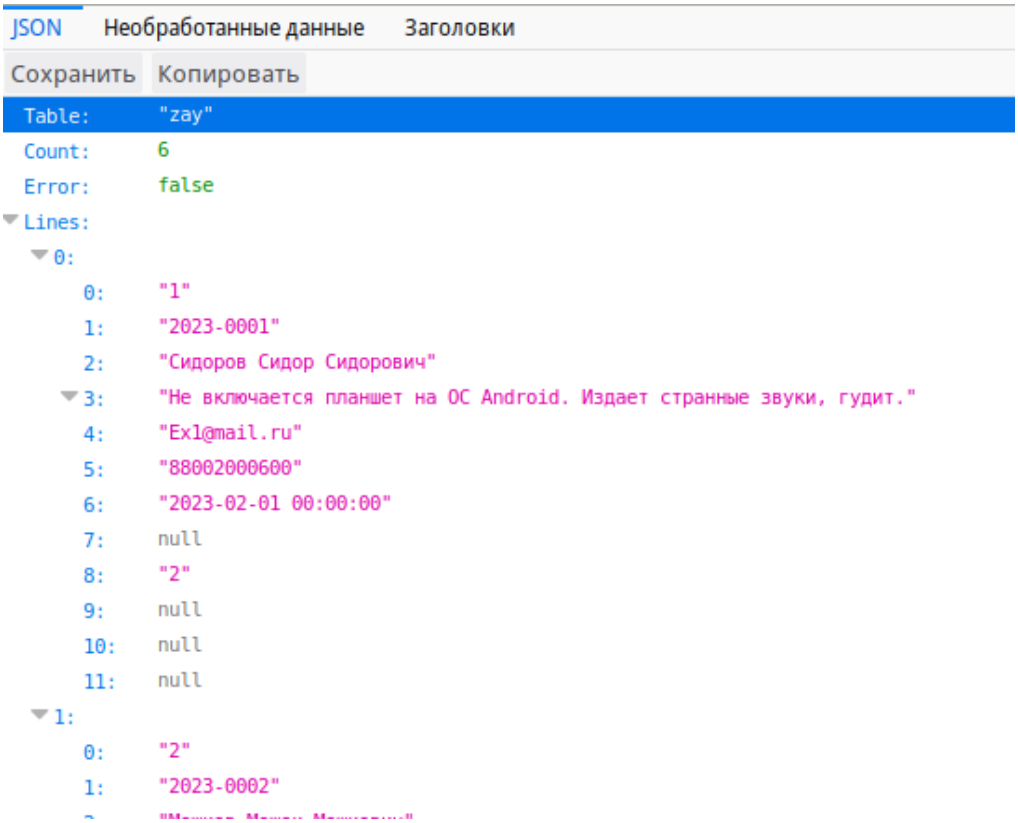


Рисунок 27 – Фрагмент JSON ответа на фронтэнде приложения

SQL-инъекция или SQLi – уязвимость, которая позволяет атакующему использовать фрагмент вредоносного кода на языке структурированных запросов (SQL) для манипулирования базой данных и получения доступа к потенциально ценной информации [24]. Атаки на основе таких уязвимостей одни из самых распространенных и опасных: они могут быть нацелены на любое веб-приложение или веб-сайт, которые взаимодействуют с базой данных SQL (а подавляющее большинство баз данных реализованы именно на SQL).

Веб-приложение также использует в своей архитектуре базу данных под управлением СУБД MySQL.

Для защиты от данной уязвимости в коде приложения предусмотрено ограничение на размер и формат данных, которые пользователь вносит в поля ввода, и которые впоследствии передаются для записи в таблицы СУБД MySQL. На фронтэнде приложения для ряда полей ограничен максимальный размер вносимых данных параметром «maxlength» для полей типа «input» и производится контроль корректности данных, например, исключаются символы латинского алфавита и другие символы кроме разрешенных. На бэкэнде полученные с фронтэнда для записи в СУБД данные перед формированием SQL-запроса на ввод или обновление информации также проверяются на наличие служебных символов и команд.

3.5 Тестирование веб-приложения

При тестировании веб-приложений ручным способом принято использовать ряд следующих подходов [25]:

- функциональное тестирование;
- тестирование пользовательского интерфейса;
- тестирование удобства использования;
- нагрузочное и стрессовое тестирование;
- проверка ссылок и HTML-кода;
- тестирование безопасности.

Функциональное тестирование и тестирование пользовательского интерфейса приложения производилось разработчиком непосредственно в процессе написания программного кода и его отладке, а также, с использованием тест-кейсов.

Тестирование удобства использования, нагрузочное и стрессовое тестирование приложения не производилось по причине малого количества

специалистов, принимающих участие в ручном тестировании, и как следствие, невозможности объективной оценки и анализа полученных результатов. Данный раздел тестирования может быть выполнен по факту полного внедрения веб-приложения «Прием онлайн-заказов» в организацию ООО «Феликс» и после некоторого опыта его практической эксплуатации.

Для проверки ссылок, корректности HTML-кода, кода JavaScript и состояния переменных в различных ситуациях и местах исполняемого кода использовался стандартный функционал языка программирования JavaScript:

– отладочная печать в консоль браузера (команда console.log («текстовое сообщение и значение переменной/переменных») языка программирования JavaScript);

– стандартные механизмы отладки, встроенные в браузер, показывающие ошибки в синтаксисе кода JavaScript с указанием номера строки в коде скрипта, в которой обнаружена ошибка.

Пример сообщения об ошибке в строковом литерале и об обращении к необъявленной ранее функции, отображаемый в окне отладки консоли браузера представлен на рисунке 28.

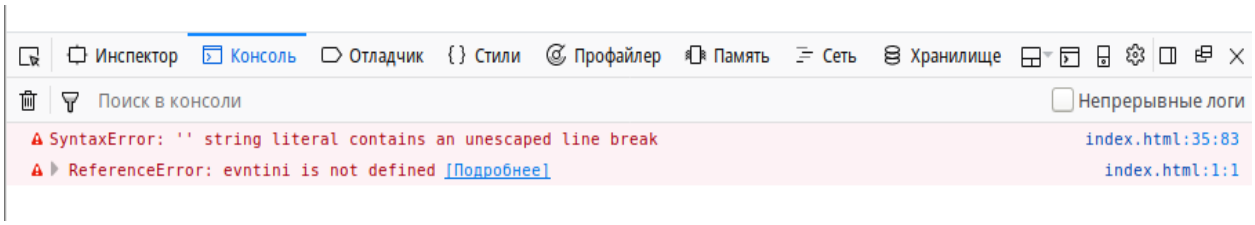


Рисунок 28 – Окно с сообщениями об ошибке в консоли браузера

Анализ сообщений в отладочной консоли браузера, а также, анализ результатов отладочной печати помогают оперативно вносить в проект необходимые изменения и дополнения, а также исправлять синтаксические ошибки языка программирования на стадии написания кода.

3.5.1 Разработка сценариев тестирования

Тестирование работоспособности приложения предлагается выполнить путем распространенного метода тест-кейса [26]. Тесты-кейс позволит выявить соответствие заявленного функционала приложения реальному результату.

Тест-кейс, помимо последовательного описания шагов тестирования с необходимыми действиями, содержат предусловие и послеусловие. Предусловие необходимо для проверки штатного состояния приложения непосредственно перед началом выполнения шагов тестирования. Послеусловие необходимо для приведения приложения к исходному виду.

Тест-кейс для тестирования веб-приложения «Прием онлайн-заказов» при работе с ролью пользователя при подаче заявки представлен в таблице 10. Тестированию подлежит механизм контроля вводимых пользователем данных в полях формы подачи заявки. Успешный результат выполнения шагов теста условимся отмечать знаком «+», неуспешный знаком «-», неожиданный/непонятный результат знаком «?».

Таблица 11 – Тест-кейс тестирования роли пользователя приложения

Действие	Ожидаемый результат	Результат
Предусловие:		
Откройте главную страницу приложения «Прием онлайн-заявок»	Открылась страница с надписью в левом верхнем углу «ООО Феликс», бегущей строкой «Здравствуйте...». Ниже размещено окно «Форма для подачи онлайн-заявки». Форма имеет шесть текстовых полей и кнопку в левом нижнем углу.	+
Шаги теста:		
1. Нажмите кнопку «Оставить онлайн заявку», не заполняя никаких текстовых полей	Справа от кнопки отобразится текст «Ошибка. Поле фамилия имя отчество должно быть заполненным»	+
2. Введите в поле «фамилия имя отчество» текстовые данные, содержащие буквы на латинской раскладке и цифровые значения, затем нажмите кнопку «Оставить...»	Данные, набранные в поле ввода исчезли. Вывелось сообщение «Ошибка...», как в п.1.	+
3. Введите в поле «фамилия имя отчество» текстовые данные, содержащие буквы на латинской и русской раскладке клавиатуры и цифровые значения, затем нажмите кнопку «Оставить...»	В поле ввода остались только буквы, набранные в русской раскладке, остальные символы исчезли. Вывелось сообщение «Ошибка. Поле телефон для связи должен быть заполнен».	+
4. Введите в поле «телефон для связи» текстовые данные, содержащие буквы на латинской и русской раскладке клавиатуры и цифровые значения, затем нажмите кнопку «Оставить...»	Данные, кроме цифровых значений, введенные в поле ввода исчезли. Вывелось сообщение «Ошибка...», как в п.4	+

Продолжение таблицы 11

5. Введите в поле «телефон для связи» только цифровые значения, не менее 11 символов затем нажмите кнопку «Оставить...»	Вывелось сообщение «Ошибка. Описание неисправностей должно быть более подробным».	+
6. Введите в поле «Описание неисправности» текстовые данные, содержащие буквы на латинской и русской раскладке клавиатуры и цифровые значения, затем нажмите кнопку «Оставить...»	Вывелось сообщение «Ошибка...», как в п.6	+
7. Введите в поле «Описание неисправности» текстовые данные, содержащие буквы на латинской и русской раскладке клавиатуры и цифровые значения, с общим количеством не менее 40 символов, затем нажмите кнопку «Оставить...»	Вместо кнопки «Оставить...» появилась кнопка «Подтверждаю, что заявка заполнена правильно». В поле, где писалось «Ошибка...» появился не отмеченный чекбокс и текст «Я даю согласие на обработку персональных данных»	+
8. Нажмите кнопку «Подтверждаю...», не отмечая «галочкой» чекбокс	Ничего не изменилось.	+
9. Отметьте «галочкой» чекбокс и нажмите кнопку «Подтверждаю...»,	Кнопка исчезла. На ее месте кнопки появился текст «N заявки...», а на месте сообщения об ошибке текст «Поздравляем! Заявка на ремонт успешно добавлена». Форма содержит введенные ранее значения.	?

Продолжение таблицы 11

Послеусловие:		
«Кликните» по логотипу «ООО Феликс» в верхнем левом углу страницы приложения	Открылась страница с надписью в левом верхнем углу «ООО Феликс», бегущей строкой «Здравствуйте...» и формой для подачи онлайн-заявки. При этом в форме отсутствуют введенные в процессе тестирование данные. Форма имеет вид, как при первом входе в приложение.	+

Тест-кейс для тестирования реакции веб-приложения на действия в режиме администратора при работе с поданными пользователями заявками представлен в таблице 11.

Таблица 12 – Тест-кейс тестирования роли администратора приложения

Действие	Ожидаемый результат	Результат
Предусловие:		
Откройте главную страницу приложения «Прием онлайн-заявок»	Открылась страница с надписью в левом верхнем углу «ООО Феликс», бегущей строкой «Здравствуйте...». Ниже размещено окно «Форма для подачи онлайн-заявки». Форма имеет шесть текстовых полей и кнопку в левом нижнем углу. Поля в форме имеют демонстрационные значения.	+
Шаги теста:		
1. Нажать кнопку «Я администратор», расположенную в правом верхнем углу формы.	Вместо формы для подачи заявки отобразилась форма «Авторизация Администратора» с полями «логин», «пароль» и кнопкой «авторизация»	+

Продолжение таблицы 12

2. Произвести попытки авторизоваться, используя заведомо некорректную пару «логин-пароль», либо авторизоваться без ввода пароля.	Авторизация не происходит. Ниже кнопки «авторизация» отображается сообщение с текстом «Имя или Пароль указаны неверно. Попробуйте еще раз»	+
3. Авторизоваться, используя пару «логин-пароль» (1-1 или admin-admin)	Вместо формы «Авторизация Администратора» отобразился список поданных заявок со столбцами, аналогичными полям в форме подачи заявки.	+
4. Убедиться в наличии тестовой записи, сформированной в режиме пользователя при тестировании приложения по предыдущему тест-кейсу.	В списке поданных заявок существует тестовая запись, сформированная при тестировании приложения предыдущим тест-кейсом. Номер заявки и данные в таблице соответствуют введенным при тестировании.	+
5. Оценить корректность подсветки строк таблицы, в зависимости от статуса заявки.	Строки, отображающие заявки с разным статусом, подсвечены разными цветами	+
	«новая» - светло серый	+
	«в работе» - зеленый	+
	«в архиве» - темно серый	+
6. Оценить корректность подсветки значений в ячейке «дата исполнения» зависимости от ожидаемой и текущей даты.	В ячейках «дата исполнения» с датой меньше текущего значения, текст отображается красным цветом.	+

Продолжение таблицы 12

7. Изменить статус заявки с статуса «новая» на статус «в работе», «в архиве», изменяя значение в ячейке «статус...»	Строки отображающие заявки при изменении статуса подсвечиваются в цвета как в п. 4 теста.	+
	Изменить статус заявки на значение «новая» из любого другого статуса невозможно.	+
	При изменении статуса заявки в ячейку «дата изменения» заносится текущее значение даты-времени.	+
8. Изменить статус произвольной заявки на значение «удалить».	При изменении статуса на значение «удалить», строка соответствующей заявки подсвечивается красным, а шрифт становится перечеркнутым.	+
	В верхней правой части формы рабом с кнопкой «выйти» появляется запрос на подтверждение удаления заявки с ее номером и кнопкой отмены.	+
	При нажатии на кнопку «подтвердить удаление», заявка удаляется и в списке не отображается.	+
	При нажатии на кнопку «отмена», заявка не удаляется и ее статус не изменяется.	+
9. Проверить сортировку по всем столбцам нажатием на наименование столбца.	При нажатии на наименование столбца, рядом с текстом отображается треугольник-стрелка, указывающий порядок сортировки.	+
	При повторном нажатии, порядок сортировки меняется на обратный.	+
	Сортировка производится в алфавитном порядке (прямом или обратном).	-

Продолжение таблицы 12

10. Выйти из режима администратора, нажав на кнопку «выйти» в правом верхнем углу приложения.	Открылась страница с надписью в левом верхнем углу «ООО Феликс», бегущей строкой «Здравствуйте...». Ниже размещено окно «Форма для подачи онлайн-заявки».	+
Послеусловие:		
«Кликните» по логотипу «ООО Феликс» в верхнем левом углу страницы приложения	Открылась страница с надписью в левом верхнем углу «ООО Феликс», бегущей строкой «Здравствуйте...» и формой для подачи онлайн-заявки. При этом в форме отсутствуют введенные в процессе тестирование данные. Форма имеет вид, как при первом входе в приложение.	+

Тестированию подлежит механизм изменения статуса поступивших заявок, их удаление, сортировка общего списка подсветка строк и значений. Аналогично тест-кейсу для тестирования веб-приложения в режиме пользователя, успешный результат выполнения условимся отмечать знаком «+», неуспешный знаком «-», непонятный результат знаком «?».

3.5.2 Проведение тестирования и анализ полученных результатов

Результаты проведения шагов тестирования по тест-кейсам, описанным в таблице 9, проставлены в столбце «результат» условными символами в соответствии с результатом выполнения соответствующего шага тестирования.

Рассмотрим результаты, отмеченные знаками «?» (неожиданный/ непонятный результат) и «-» (результат отрицательный).

Результат тестирования в п.10 первого тест-кейса и п.4 второго тест-кейса получил неожиданный результат (сформированная заявка в режиме пользователя и в режиме администратора успешно сформирована, но данные содержат произвольный и нелогичный набор символов). В п.9 (сортировка по произвольному столбцу в алфавитном порядке) второго тест-кейса результат не совпал с заявленным ожидаемым результатом, остальные шаги тестирования по обоим тест-кейсам пройдены успешно.

Тестирование приложения в п.10 первого тест-кейса и п.4 второго тест-кейса показало неожиданный результат несмотря на то, что формально все шаги тестирования по тест-кейсу пройдены успешно и заявленная реакция приложения на действия пользователя в режиме подачи заявки совпала с заявленным ожидаемым результатом, данные в форме содержат произвольный и не осмысленный набор символов.

Такой результат при заполнении формы подачи заявки не может считаться ошибкой функционирования приложения, так как все введенные в поля формы данные прошли условия логического контроля, а появление в форме заявки произвольного набора символов, некорректного содержания с логической точки зрения обусловлено человеческим фактором, при этом данные, введенные в форму, приняли вид, представленный на рисунке 29.

Для обеспечения корректности вносимых в поля заявки данных, некоторые логические условия контроля корректности введенных в форму подачи заявки данных были сформированы непосредственно в процессе

написания JavaScript кода фронтэнда приложения и ограничены максимально допустимой длиной данных для полей ввода, другие условия контроля были скорректированы и добавлены при проведении тестировании приложения и анализа результатов тестов.

Например, для поля «фамилия, имя, отчество» допустимы только символы кириллицы в верхнем и нижнем регистре, для поля «адрес электронной почты» допустимы только латинские символы, которые переведены в нижний регистр и символ «@» и «.» (точка), в поле «телефон для связи» допустимы только цифры и знак «+», также это поле должно иметь длину не менее 11 символов.

Во фрагменте листинга JavaScript-кода фронтэнда веб-приложения, представленном на рисунке 30, реализован контроль заполнения полей формы как по минимальной длине введенных данных, так и их соответствию типу данных для поля формы.

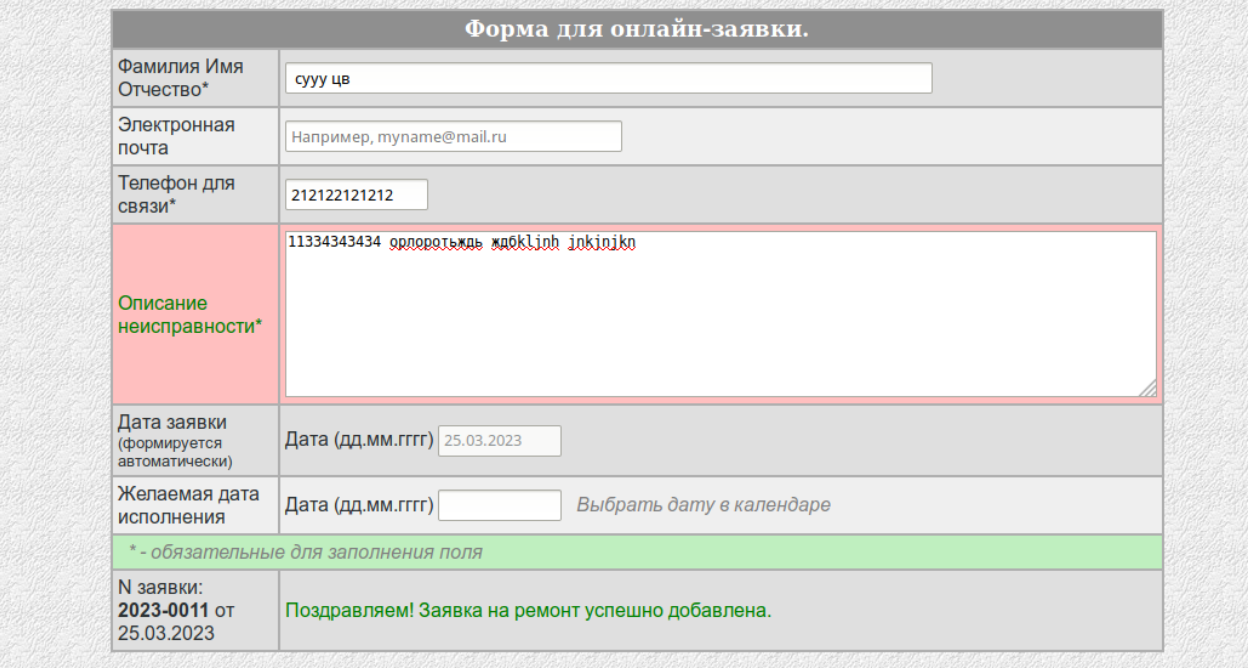


Рисунок 29 – Заполненная форма подачи заявки для тестирования

При проведении тестирования работы приложения в режиме администратора, по составленному тест-кейсу не совпал с ожидаемым

результатом такой шаг тестирования, как сортировка записей в алфавитном порядке (прямая и обратная) по произвольному полю (столбцу таблицы).

```
document.getElementById("eml").value.toLowerCase();
var newna = document.getElementById("nam").value.replace(/[^a-яёA-ЯЁ ]/g, '');
var newml = document.getElementById("eml").value.replace(/[^a-z\s\@\.]+/ig, '');
var newtl = document.getElementById("tel").value.replace(/[^0-9\+]/g, '');
var newtx = document.getElementById("nwa").value;
var newd2 = document.getElementById("datepicker_value").value.replace(/[^0-9\.]/g, '');;

document.getElementById("nam").value = newna;
document.getElementById("tel").value = newtl;
document.getElementById("eml").value = newml;
document.getElementById("datepicker_value").value = newd2;

if ( newd2.length < 10 && newd2.length >0 )
                        { msg = msf + 'Ошибка. Неправильная дата / желаемое окончание работ.</font>';  }
if ( newtx.length < 40 )     { msg = msf + 'Ошибка. Описание неисправностей должно быть более подробным.</font>';  }
if ( newtl.length < 11 )     { msg = msf + 'Ошибка. Поле "телефон для связи" должно быть заполнено.</font>';  }
if ( newna.length < 10 )     { msg = msf + 'Ошибка. Поле "фамилия имя отчество" должно быть заполнено.</font>';  }

if ( msg != '' )
```

Рисунок 30 – Фрагмент JavaScript кода с условиями контроля

При проведении тестирования работы приложения в режиме администратора, по составленному тест-кейсу не совпал с ожидаемым результатом такой шаг тестирования, как сортировка записей в алфавитном порядке (прямая и обратная) по произвольному полю (столбцу таблицы).

Анализ результата шага тестирования (п.9) с неудачным результатом показал, что сортировка по произвольному полю производится, но сначала в отсортированном списке идут символы в верхнем регистре, а затем в нижнем. Для устранения ошибки сортировки потребовалось внести изменения в одну строку кода, для перевода всех символов, содержащихся в поле, по которому производится сортировка, в верхний регистр. Причем, при отображении значений в соответствующем столбце формы, они остаются в том виде и том регистре, в котором были изначально введены в форму и записаны в базу. Фрагмент листинга фрагмента JavaScript кода с функцией сортировки представлен на рисунке 31. Строка, в которую внесены изменения, подсвечена и выделена курсором.

Табличная форма списка обращений с сортировкой по столбцу «фамилия, имя отчество» при выполнении п.9 тест-кейса представлена на рисунке 32.

```
function evntsrt(k,a)  // сортировка строк массива (текстовая) k - номер поля по которому сортировать
   {
    for (let i = 0; i < Zay.length; i++)
      { let s = Zay[i];
        if (s.indexOf(';')>0)          // разделитель полей в строке - массиве точка с запятой ;
          { var tmp = s.split(';');
            tmp[0] = '0' + tmp[k].toUpperCase();          // В массиве заявок в первый столбец дублируем данные
            Zay[i] = tmp.join(';');
            }
      }
    Sort = k;                          // k - номер поля по которому сортировали присвоем глобальной переменной
    Zay.sort();                        // Сортировка штатной функцией JS сортируем массив как текстовые строки с
    Strl = Strl + 1;
    if (Strl>1) { Zay.reverse(); Strl = 0; }  // если нужна обратная сортировка, то реверс массива и присвоение
    evntfn(a);                                // перерисовываем таблицу с мероприятиями параметр 'a' просто пер
   }  // evntsrt
```

Рисунок 31 – Функция сортировки массива заявок

В левом столбце представлен результат сортировки до внесения изменений в JavaScript-кода, в правом после внесения корректировок в алгоритм JavaScript кода.

Использованные методы контроля корректности, введенной пользователем в поля формы заполнения заявки информации, рассчитаны на обычного пользователя и служат больше для подсказки порядка заполнения формы и не в полной мере могут предотвратить неправомерный доступ со стороны злоумышленника высокой квалификации и опыта. Современные инструменты для отладки приложений помимо помощи программисту при написании проекта, предоставляют возможность не только просмотреть, но и изменить значения переменных на разных этапах выполнения программного кода, внедрить в код JavaScript свои модули и функции, модифицировать код на стороне клиентской части приложения, подменить значения, обойти логический контроль и прочее.

Для защиты данных приложения от квалифицированного злоумышленника можно использовать методы обфускации кода (сделать код трудночитаемым и логически непонятным), запретить обработку сторонних JSON запросов на бэкэнде приложения, и даже применить методы криптографии и шифрования, при этом необходимо тщательно протестировать уязвимость приложения возможным угрозам не только ручным способом, но и с использованием специального программного обеспечения.

N п.п.	N заявки	ФИО
1	2023-0004	Иванов Иван Петрович
2	2023-0002	Мошков Мошок Мошкович
3	2023-0006	Сидоренко Борис Полиграфович
4	2023-0001	Сидоров Сидор Сидорович
5	2023-0009	Федорчук лариса Ивановна
6	2023-0003	михайлов михаим Михайлович
7	2023-0005	петренко лариса Ивановна
8	2023-0010	семенов семен петрович
9	2023-0008	степанович петр борисовия
10	2023-0007	федоркович федр степанович

N п.п.	N заявки	ФИО
1	2023-0004	Иванов Иван Петрович
2	2023-0003	михайлов михаим Михайлович
3	2023-0002	Мошков Мошок Мошкович
4	2023-0005	петренко лариса Ивановна
5	2023-0010	семенов семен петрович
6	2023-0006	Сидоренко Борис Полиграфович
7	2023-0001	Сидоров Сидор Сидорович
8	2023-0008	степанович петр борисовия
9	2023-0007	федоркович федр степанович
10	2023-0009	Федорчук лариса Ивановна

Рисунок 32 – Пример сортировки по алфавиту для столбца «ФИО»

Эти и другие вопросы защиты приложения и обрабатываемых в нём данных от неправомерного доступа со стороны злоумышленника высокой квалификации и опыта требуют более подробного рассмотрения и выходят за рамки целей и задач данного проекта.

Выводы к разделу

Данная глава посвящена физическому проектированию веб-приложения «Прием онлайн-заказов».

Рассмотрены вопросы выбора архитектуры веб-приложения. Выбрана двухуровневая клиент-серверная архитектура, сформулированы особенности разработки таких архитектур.

Решена задача выбора языков программирования для создания фронтэнда и бэкэнда приложения, а также прикладных инструментов для разработки веб-приложения «Прием онлайн-заказов». В результате, для написания фронтэнда веб-приложения использованы стандартные возможности разметки HTML с использованием таблицы стилей CSS. Для обеспечения функциональности

приложения был использован скриптовый язык программирования JavaScript. Для взаимодействия фронтэнда с бэкэндом выбрана и использована технология AJAX с форматом обмена данными в формате JSON, для удобства реализации обмена данными к фронтэнду приложения подключена библиотека jquery. Для облегчения работы пользователя приложения с выбором дат, использована библиотека для формирования графического представления календаря jquery-ui.

В качестве бэкенда приложения выбран язык программирования PHP с подключенным модулем для работы с СУБД MySQL, которая выбрана в качестве базы для хранения данных, формирующихся в процессе функционирования приложения.

Рассмотрены вопросы создания интерфейса клиентской части приложения (фронтэнда). Приняты решения о режимах работы — режим пользователя и режим администратора. Для переключения для работы в веб-приложении с ролью администратора реализован механизм авторизации по логину и паролю.

Для работы в веб-приложении в качестве пользователя разработан интерфейс, содержащий необходимые поля для формирования заявки. Для удобства заполнения желаемой даты выполнения заявки, подключен модуль с графическим календарем. Реализован контроль корректности вводимой пользователем информации.

Для работы в качестве администратора реализован механизм изменения статуса поступивших заявок, перевода их в различные технологические стадии – состояние «в работе», «в архиве». Предусмотрено удаление заявок, сортировка общего списка поступивших заявок. Для улучшения информативности интерфейса использованы методы визуализации, такие как подсветка строк заявок, находящихся на различных технологических стадиях разной цветовой гаммой, изменение цвета отображения даты исполнения для «просроченных» заявок.

На заключительном этапе написания главы выполнено тестирование работоспособности веб-приложение «Прием онлайн-заказов» на основе технологии тест-кейса. Результатами работ является полностью функциональное веб-приложение, лишенное благодаря исправлениям большинства недостатков, выявленных при проведении мероприятий по тестированию.

Использование данной методики в части анализа данных ручного тестирования и анализ результатов тест-кейсов непосредственно на стадии написания программного кода, с немедленным исправлением выявленных при тестировании недостатков показывает, что процесс тестирования разрабатываемого приложения, особенно на стадии его написания, является неотъемлемым и обязательным этапом при создании качественного программного продукта.

Заключение

По результатам выполнения выпускной квалификационной работы достигнута ее цель, решены поставленные задачи.

Первая глава работы посвящена аналитическому обзору предметной области. На начальном этапе исследования выполнено комплексное описание предметной области с построением организационной структуры ООО «Феликс», которое является компанией-объектом для внедрения решения по автоматизации. Выявлены проблемы на уровне подразделения обработки заявок, предложены возможны пути решения в виде внедрения веб-приложения. Выполнен обзор популярных технологий и существующих подходов к разработке веб-приложений. В качестве основных инструментов выделены веб-фреймворки, СУБД, фронтенд и бэкенд-языки программирования. Сформулирована постановка задачи на разработку веб-приложения «Прием онлайн-заказов».

По результатам выполнения этапа концептуального моделирования спроектированы:

– диаграмма классов UML веб-приложения, отражающая статическую структуру, классы (сущности) с их атрибутами и операциями, а также взаимосвязи между этими классами;

– диаграммы вариантов использования UML, формализующие пользователей приложения, возможные сценарии и поведение системы в целом.

По итогам выполнения этапа функционального моделирования разработаны:

– диаграммы в нотации IDEF0 «как есть», которые отражают текущий подход к приему онлайн-заказов на ремонт техники в организации ООО «Феликс» – для этого подхода проанализированы его недостатки и уязвимые стороны;

– диаграммы в нотации IDEF0 «как должно быть», которые формализуют процессы обработки заявок после внедрения веб-приложения «Прием онлайн-заказов» – для этого подхода отмечены очевидные преимущества и положительный эффект для всей организации ООО «Феликс» в целом.

Вторая глава выпускной квалификационной работы посвящена задачам логического моделирования веб-приложения «Прием онлайн-заказов». На начальном этапе логического проектирования веб-приложения выполнено обоснование выбора технологии для разработки логических моделей. Выбраны следующие методы разработки логической модели:

– для формализации архитектуры веб-приложения – обозначения из теории проектирования сетей;

– для формализации компонентов веб-приложения их взаимодействия диаграмма компонентов UML;

– для разработки логической и физической моделей базы данных – инструмент ER-диаграмм.

По результатам проведения работ по логическому моделированию получены:

– схема архитектуры веб-приложения;

– диаграмма компонентов UML веб-приложения.

На отдельном этапе исследования выполнено комплексное проектирование базы данных приложения.

По результатам проектирования разработаны:

– ER-диаграмма логической модели БД веб-приложения «Прием онлайн-заказов»;

– ER-диаграмма физической модели БД веб-приложения;

– таблицы с описанием атрибутов сущностей предметной области.

На заключительном этапе раздела проведен анализ требований к программно-аппаратному обеспечению приложения. Графически представлена

текущая ИТ-инфраструктура обработки заказов, используемая в ООО «Феликс». Проанализированы недостатки данной инфраструктуры и описаны преимущества предложенного архитектурного решения на основе внедрения веб-приложения «Прием онлайн-заказов».

В третьей главе представлены результаты работ по физическому проектированию веб-приложения «Прием онлайн-заказов». В качестве технической архитектуры веб-приложения выбрана двухуровневая клиент-серверная архитектура. Описаны особенности подходов к разработке архитектуры такого типа.

С анализом аналогов, преимуществ и недостатков существующих решений выбраны оптимальные технологии, инструменты и языки программирования для реализации веб-приложения.

Подробно, с экранными демонстрациями и листингами описаны этапы практической разработки веб-приложения «Прием онлайн-заказов»:

– разработка базы данных;
– разработка фронтенда;
– разработка бэкенда.

На заключительном этапе написания главы разработаны тест-кейсы для тестирования работоспособности веб-приложения «Прием онлайн-заказов». В дальнейшем на основе данных тест-кейсов проведено тестирование функций веб-приложения с выявлением уязвимостей приложения и их последующим устранением.

Практическим результатом выпускной квалификационной работы является разработанное веб-приложение «Прием онлайн-заказов», готовое к внедрению и дальнейшей эксплуатации в профильном подразделении организации-объекта ООО «Феликс». Внедрение данного приложения в ООО «Феликс» позволит:

– хранить заявки в едином консолидированном хранилище;

– организовать дополнительный канал связи взаимодействия с клиентами;

– оперативно получать полную информацию по тому или иному заказу;

– обеспечить удобство работы с заявками для персонала отдела продаж и для заказчиков услуг;

– повысить продуктивность и престиж компании ООО «Феликс» на рынке предоставляемых услуг за счет наличия собственного сайта.

Веб-приложение «Прием онлайн-заказов» характеризуется высоким уровнем защищенности данных, прозрачностью и надежностью архитектуры, возможностью масштабирования.

Список используемых источников

1. А.С. Мишушина. Разработка автоматизированных тестов для веб-приложения с использованием библиотеки Selenium WebDriver. Бакалаврская работа - ТГУ, 2019.
2. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. Современные фреймворки для разработки web-приложений [Электронный ресурс] // Интеллектуальные технологии на транспорте. 2020. №4 (24). URL: https://cyberleninka.ru/article/n/sovremennye-freymvorki-dlya-razrabotki-web-prilozheniy (дата обращения: 14.03.2023).
3. Блог статей Oracle, Определение JSON, [электронный ресурс], URL: https://www.oracle.com/cis/database/what-is-json/ (дата обращения: 12.05.2023).
4. Блог Kaspersky, Что такое SQL-инъекция? Определение и описание, [электронный ресурс], URL: https://www.kaspersky.ru/resource-center/definitions/sql-injection (дата обращения: 12.05.2023).
5. Блог Я.Практикума, С чего начинается тестирование: что такое тест-кейс, зачем он нужен и как его писать, [электронный ресурс], URL: https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/ (дата обращения: 12.05.2023).
6. Веб-ресурс продукта PhpMyAdmin, [электронный ресурс], URL: https://www.phpmyadmin.net/ (дата обращения: 12.05.2023).
7. Документация Microsoft. Нереляционные данные и базы данных NoSQL [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data (дата обращения: 15.03.2023).
8. Журнал Тинькофф. Путь в ИТ: бэкенд-разработчик [Электронный ресурс]. URL: https://journal.tinkoff.ru/becoming-backend-engineer/ (дата обращения: 16.03.2023).

9. Лиманова Н.И, Селезнев И.А. Анализ эффективности клиент-серверной архитектуры // Бюллетень науки и практики. 2022. №7. URL: https://cyberleninka.ru/article/n/analiz-effektivnosti-klient-servernoy-arhitektury (дата обращения: 11.05.2023).
10. Леоненков А.В. Самоучитель UMLб 2-е издание. ГЛАВА 5. Диаграмма классов (class diagram) [Электронный ресурс]. URL: http://www.telenir.net/uchebniki/samouchitel_uml/p5.php (дата обращения: 10.05.2023).
11. Леоненков А.В. Самоучитель UMLб 2-е издание. ГЛАВА 4. Диаграмма вариантов использования (use case diagram) [Электронный ресурс], URL:http://www.telenir.net/uchebniki/samouchitel_uml/p4.php (дата обращения: 10.05.2023).
12. НОУ ИНТУИТ. Курс: Общие вопросы технической защиты информации. Лекция 9: Угрозы несанкционированного доступа к информации. Основные классы атак в сетях на базе TCP/IP [Электронный ресурс]. URL: https://intuit.ru/studies/professional_skill_improvements/4495/courses/591/lecture/12691?page=2&ysclid=lfk7xbwaf3358123100 (дата обращения: 16.03.2023).
13. Официальная документация PHP, mysqli.overview, [электронный ресурс], URL: https://www.php.net/manual/en/mysqli.overview.php (дата обращения: 12.05.2023).
14. РД IDEF 0 - 2000. МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ IDEF0. Руководящий документ.
15. СОВКОМБЛОГ. Организационная структура предприятия как главный борец с хаосом в бизнес-процессах [Электронный ресурс]. URL: https://sovcombank.ru/blog/biznesu/organizatsionnaya-struktura-predpriyatiya-kak-glavnii-borets-s-haosom-v-biznes-protsessah?utm_referrer=https%3A%2F%2Fyandex.ru%2F (дата обращения: 13.03.2023).

16. Статьи Amazon. Что такое реляционная база данных? [Электронный ресурс]. URL: https://aws.amazon.com/ru/relational-database/ (дата обращения: 15.03.2023).
17. Справочник по базам данных на GitHub. Что такое первичный ключ? [электронный ресурс], URL: https://github.com/dpopkov/notes/blob/master/sql_jdbc.md#q01 (дата обращения: 10.05.2023).
18. Codernet. Преимущества и недостатки языка программирования PHP. / [электронный ресурс], URL: https://codernet.ru/articles/web/preimushhestva_i_nedostatki_yazyika_programmirovaniya_php/?ysclid=le4q8sa6yn8457391 (дата обращения: 11.05.2023).
19. jQuery API Documentation, jQuery.ajax() [Электронный ресурс]. URL: https://api.jquery.com/jquery.ajax/ (дата обращения: 12.05.2023).
20. Jang D., Choe K.M. Points-to analysis for JavaScript. – Proceedings of the ACM Symposium on Applied Computing, 2009.
21. Lucidchart. Что такое ER-диаграмма и как ее создать? [Электронный ресурс]. URL: https://www.lucidchart.com/pages/ru/erd-диаграмма (дата обращения: 10.05.2023).
22. Ler C., Rosenblum D. UML Component Diagrams and Software Architecture. – Experiences from the Wren Project, 2001.
23. Rusprofile. Сведения об организации ООО «Феликс» [Электронный ресурс]. URL: https://www.rusprofile.ru/id/7259604?ysclid=lfk7gcm920134327539 (дата обращения: 12.03.2023).
24. Skillbox. Чем различается фронтенд и бэкенд-разработка [Электронный ресурс]. URL: https://skillbox.ru/media/code/frontend_i_backend_razrabotka/ (дата обращения: 16.03.2023).

25. Sulyman S. Client-Server Model. – IOSR Journal of Computer Engineering, 16, 2014.
26. Thankappan M. Network Forensic Investigation of HTTPS Protocol. – International Journal of Engineering Research, 2013.

Приложение А

Исходный код веб-приложения «Прием онлайн-заказов»

```
<html>
<head>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
<link rel="stylesheet" href="./jquery-ui.css">
<script src="./jquery.min.js"></script>
<script src="./jquery-ui.js"></script>

<style type="text/css">#datepicker {width: 250px;margin: 0 auto;}</style>

<script>
/* Локализация datepicker */
$.datepicker.regional['ru'] = {
        closeText: 'Закрыть',
        prevText: 'Предыдущий',
        nextText: 'Следующий',
        currentText: 'Сегодня',
        monthNames: ['Январь','Февраль','Март','Апрель','Май','Июнь','Июль','Август','Сентябрь','Октябрь','Ноябрь','Декабрь'],
        monthNamesShort: ['Янв','Фев','Мар','Апр','Май','Июн','Июл','Авг','Сен','Окт','Ноя','Дек'],
        dayNames: ['воскресенье','понедельник','вторник','среда','четверг','пятница','суббота'],
        dayNamesShort: ['вск','пнд','втр','срд','чтв','птн','сбт'],
        dayNamesMin: ['Вс','Пн','Вт','Ср','Чт','Пт','Сб'],
        weekHeader: 'Не',
        dateFormat: 'dd.mm.yy',
        firstDay: 1,
        isRTL: false,
        showMonthAfterYear: false,
        yearSuffix: ''
};
$.datepicker.setDefaults($.datepicker.regional['ru']);

// Глобальные переменные - массивы
var Zay = new Array ();     // Список заявок общий
var Adm = new Array ();        // Список администраторов

// Глобальные переменные
var DEBUG = 1;              // Отладка и для Демонстрации (0 - работа с СУБД MySQL)  1 - вкл хранение данных в массивах 2 - LocalStorage

var Sort = 0;             // Поле по которому сортируем номер поля в масиве Zay
var Strl = 0;                    // Стрелка вверх/вниз. порядок сортировки (убывание, возрастание)
var logid = '';               // Администратор
var delid = '';               // для удаления
var pickf = 0;         // календарь выкл.
// Глобальные переменные end.

function sysdate() // формирует текущую дату и время в формате yyyy-mm-dd HH:MM:00
   {
    var today = new Date();
    var                 dd                  =
String(today.getDate()).padStart(2, '0');
    var mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!
    var yyyy = today.getFullYear();
    var hh = '0'+today.getHours(); // 10
    var ii = '0'+today.getMinutes(); // 0
    var sdat = yyyy + '-' + mm + '-' + dd + ' ' + hh.substr(-2) + ':' + ii.substr(-2) + ':00';
    return(sdat);
   } // sysdate

function dateymd(d)  // преобразует дату из dd.mm.yyyy в формат yyyy-mm-dd 00:00:00
   { let out = '';
     if ( d.length>3) { out = d.substr(6,4) + '-' + d.substr(3,2) + '-' + d.substr(0,2) + ' 00:00:00'; }
     return(out);
   } // dateymd

function datedmy(d)  // преобразует дату из yyyy-mm-dd в формат dd.mm.yyyy<br>HH:MM:II
   { let out = '';
     if ( d.length>3) { out = d.substr(8,2) + '.' + d.substr(5,2) + '.' + d.substr(0,4) + '<br> ' + d.substr(11) + ' '; }
     return(out);
   } // datedmy

function selphp(kod,dat,idx,mas)
   {
    $.ajax({
        type: "POST",
        dataType: "json",
       url:"./sel.php",
        cache: false,
//        async: false,
       data: {kod:kod,dat:dat,idx:idx},
        error: function() { alert('Сервер не отвечает'); },
        success: function(otvet)
        {
        var rs = "Table:"+otvet.Table+"<br>Count:"+otvet.Count+"<br>Error:"+otvet.Error+"<br>Lines"+otvet.Lines[0][2];
        var ms = otvet.Lines[0].join(';');
        for (let i = 0; i < otvet.Count; i++) {mas[i] = otvet.Lines[i]; }
//          $("#res").html(rs + '<br><br>' + ms);
//        console.log(raz.data[0]);
        }
      });
   } // selphp

function evntfn(a)    // Основная функция. Отображает таблицу со списком заявок
   {
   var strelka = ['', '', '', '', '', '', '', '', '', '', '', ''];    // Обнуляем массив стрелок для отображения порядка сортировки
   if    (Strl>0)    {    strelka[Sort]='<font color="lime">&#9650;</font>'    }    else    { strelka[Sort]='<font color="lime">&#9660;</font>'; } // стрелки - треугольники вверх-вниз
   var j = 0;    // для зебры при подсветке строк
```

```
   var k = 0;   // номер строки
   var s = 's';

   document.getElementById('evnt-
del').style.display ='none';
   document.getElementById('evnt-
can').style.display ='none';

   var txt = '<br><table border="0"
cellpadding="4" cellspacing="2"
bgcolor="#afafaf" width="99%">';
   // Строка - заголовок таблицы. В описании
колонок ссылки на вызов функции сортировки с
изображением стрелки порядка сортировки
   txt = txt + '<tr
class="v"><th>N<br>п.п.</th>' +
        '<th><a
href="javascript:evntsrt(1,\'' + a + '\')">N
заявки' + strelka[1] + '</a></th>' +
        '<th width="20%"><a
href="javascript:evntsrt(2,\'' + a +
'\')">ФИО' + strelka[2] + '</a></th>' +
        '<th width="40%"><a
href="javascript:evntsrt(3,\'' + a +
'\')">Описание неисправности' + strelka[3] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(4,\'' + a +
'\')">Адрес эл.почты' + strelka[4] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(5,\'' + a +
'\')">Телефон<br>для_связи' + strelka[5] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(6,\'' + a +
'\')">дата<br>заявки' + strelka[6] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(7,\'' + a +
'\')">дата<br>исполнения' + strelka[7] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(8,\'' + a +
'\')">Статус заявки' + strelka[8] +
'</a></th>' +
        '<th><a
href="javascript:evntsrt(9,\'' + a +
'\')">дата<br>изменения' + strelka[9] +
'</a></th></tr>';

    for (let i = 0; i < Zay.length; i++)
      { let str = Zay[i];
       if (str.indexOf(';')>0)
         { let tmp = str.split(';');
           let dis = datedmy(tmp[7]);
           let pp = '';
           if (tmp[3].length > 63) { pp =
tmp[3].substr(0,64) + '... <br>    
  <i><u>подробнее ...</u></i>'; } else {
pp= tmp[3]; }
           if (a==tmp[1]) { pp= tmp[3]; }
           if (sysdate() > tmp[7]) { dis =
'<font color="red">' + datedmy(tmp[7]) +
'</font>'; }
           let s = 's';
           if (tmp[8]==9) { continue; }  //
не отображать уделенные
           if (tmp[8]==1) { s = 'm'; }
           if (tmp[8]==2) { s = 'z'; }
           if (tmp[1]==delid) { s = 'd';

document.getElementById('evnt-del').value  =
'Подтвердить удаление заявки N ' + tmp[1];

document.getElementById('evnt-
del').style.display ='';

document.getElementById('evnt-
can').style.display ='';
                        }
           k = k + 1; j = j + 1; if (j>1) {
j = 0; }  // для зебры при подсветке строк
           txt = txt + '<tr class="' + s + j
+ '">' +
           '<th>' + k + '</th>' +
           '<td class="r">' + tmp[1] +
'</td>' +
           '<td>' + tmp[2] + '</td>' +
           '<td><a class="p"
href="javascript:evntpod(\'' + tmp[1] +
'\')">' + pp + '</a></td>' +
           '<td class="r">' + tmp[4] +
'</td>' +
           '<td class="r">' + tmp[5] +
'</td>' +
           '<td class="r">' +
datedmy(tmp[6]) + '</td>' +
           '<td class="r">' + dis + '</td>'
+
           '<td class="r"><select
name="stat-za" id="stat-za"
onchange="statza(this,\'' + tmp[1] +
'\')"><option ';
           if (tmp[8]==0) { txt = txt +
'selected '; }
           txt = txt + 'style="color:
white;" value="0"> -новая- </option><option
';
           if (tmp[8]==1) { txt = txt +
'selected '; }
           txt = txt + 'value="1"
style="color: lime;"> -в работе-
</option><option ';
           if (tmp[8]==2) { txt = txt +
'selected '; }
           txt = txt + 'value="2"
style="color: gray;"> -в архиве-
</option><option ';
           if (tmp[1]==delid) { txt = txt +
'selected '; }
           txt = txt + 'value="9"
style="color: red;"> -удалить-
</option></select></td>';
           txt = txt + '<td class="r">' +
datedmy(tmp[9]) + '</td>';
           txt = txt + '</tr>';
         }
      }
   txt = txt + '</table>';
   document.getElementById('evnt-
table').innerHTML = txt;
   }  // evntfn

function evntpod(a)
   { evntfn(a);
   }  // evntpod
```

```
function statza(a,b)  //  отображение формы операций с заявками
   { let aa = a.value;
     let bb = b;
     delid = '';
     if (aa==0) { evntfn(); return; }  // нельзя изменить статус на новый
     for (let i = 0; i < Zay.length; i++)
      { let str = Zay[i];
        if (str.indexOf(';')>0)
          { let tmp = str.split(';');
            if ( bb == tmp[1] )
              {
              if (aa==2) { let upd = aa + ';'+sysdate()+';;0;';  evntupd(Zay,upd,8,bb); }  //  в архив с датой
              if (aa==1) { let upd = aa + ';'+sysdate()+';;0;';  evntupd(Zay,upd,8,bb); }  //  в работу
              if (aa==9) { //let upd = aa + ';'+sysdate()+';;0;';  evntupd(Zay,upd,8,bb); //  на удаление
                           delid = bb;
                          }
              }
          }
      }
    evntfn(b);
   }  // statza

function evntdl()
   {  let  upd  =  '9;'+sysdate()+';;0;'; evntupd(Zay,upd,8,delid);
     document.getElementById('evnt-del').style.display ='none';
     document.getElementById('evnt-can').style.display ='none';
     delid = '';
     evntfn();
   }  // evntdel

function evntcn()
   { delid = '';
     document.getElementById('evnt-del').style.display ='none';
     document.getElementById('evnt-can').style.display ='none';
     evntfn();
   }  // evntdel

function animatein()
    {  if (parseInt(crossheader.left)<0)
crossheader.left=parseInt(crossheader.left)+20  // СКОРОСТЬ ПОЯВЛЕНИЯ
       else
       { crossheader.left=0
         crossheader.fontStyle="normal"
         clearInterval(start)
       }
   }  // animatein

function evntnw()  //  отображение формы ввода новой заявки
   {
document.getElementById('datepicker').style.display ='none'; pickf = 0;

document.getElementById('flyin').style.display ='';
     document.getElementById('evnt-logn').style.display ='';
     document.getElementById('evnt-exit').style.display ='none';
     document.getElementById('evnt-del').style.display ='none';
     document.getElementById('evnt-can').style.display ='none';

     if (document.getElementById||document.all)
        {                                  var crossheader=document.getElementById? document.getElementById("flyin").style : document.all.flyin.style
         crossheader.left     =     "-1200"; start=setInterval("animatein()",100)
        }

     var txt = '';
txt    =    txt    +    '<table    border="0" cellpadding="4"                cellspacing="2" bgcolor="#afafaf" width="66%">' +
           '<tr                class="v"><th colspan="2"><font  size=+1>Форма  для  онлайн-заявки.</font></td></tr>' +
           '<tr                class="s0"><td width="30%">Фамилия  Имя  Отчество*<td><input class="nw"  name="nam"  id="nam"  size="64" maxlength="64"  placeholder="Укажите  Ваши данные"></td>' +
           '<tr  class="s1"><td>Электронная почта<td><input     class="nw"     name="eml" id="eml"       size="32"       maxlength="32" placeholder="Например, myname@mail.ru"></td>' +
           '<tr  class="s0"><td>Телефон  для связи*<td><input     class="nw"     name="tel" id="tel"       size="12"       maxlength="12" placeholder="89876543210"></td>' +
           '<tr      class="s1"><td>Описание неисправности*<td><textarea        cols="100" rows="8"          name="nwa"          id="nwa" placeholder="Напишите подробно об устройстве (Компьютер,  Ноутбук,  Планшет  и  т.п.) Операционная система и, обязательно опишите в чем  конкретно  выражается  неисправность  (не включается,            зависает            и т.п.)."></textarea></td>' +
           '<tr  class="s0"><td>Дата  заявки <font                  size=-1>(формируется автоматически)</font><td>  Дата  (дд.мм.гггг) <input  disabled  class="nw"  name="nd1" id="nd1" size="10" maxlength="10" value="' + datedmy(sysdate().substr(0,10)).substr(0,10) + '"></td>' +
           '<tr class="s1"><td>Желаемая дата исполнения<td>  Дата  (дд.мм.гггг)  <input type="text"  id="datepicker_value"  size="10" maxlength="10">         <a    class="k" href="javascript:pickshow()"> <i>Выбрать дату в календаре</i></a></td>' +
           '<tr                class="m1"><td colspan="2"><font color="gray"><i>  * - обязательные          для          заполнения поля</i></font></td>' +
           '<tr          class="s0"><td><div name="but-new"            id="but-new"><input type="button"        id="zp"        name="zp" onclick="evntnew(0)" style="border: 2px solid green;    padding:    5px;    margin:    2px; cursor:pointer;"   value="Оставить   онлайн-заявку на ремонт техники"></div></td><td><div name="evnt-new" id="evnt-new"></div></td>';
```

Продолжение приложения А

```
    txt = txt + '</table>';
    document.getElementById('evnt-table').innerHTML = txt;
    }  // evntnw

function pickshow()   //   отображение формы ввода даты календаря
   { if (pickf == 0)
          {
document.getElementById('datepicker').style.display ='';     pickf = 1; }
     else                                  {
document.getElementById('datepicker').style.display ='none'; pickf = 0; }
   }

function evntnew(a)                        // Ввод нового
   {
   let msg = '';
   let ins = '';
   let msf = '<font color="red">';     // Подсветка сообщения об ошибке
   let end = a;


document.getElementById("eml").value.toLowerCase();
   var                 newna                 = document.getElementById("nam").value.replace( /[^a-яёA-ЯЁ ]/g, '');
   var                 newml                 = document.getElementById("eml").value.replace( /[^a-z0-9\@\.]+/ig, '');
   var                 newtl                 = document.getElementById("tel").value.replace( /[^0-9\+]/gi, '');
   var                 newtx                 = document.getElementById("nwa").value.replace( /[^a-яёA-ЯЁa-zA-Z0-9\s\-]/gi, '');
   var                 newd2                 = document.getElementById("datepicker_value").value.replace(/[^0-9\.]/g, '');;

   document.getElementById("nam").value      = newna;
   document.getElementById("tel").value      = newtl;
   document.getElementById("eml").value      = newml;
   document.getElementById("nwa").value      = newtx;

document.getElementById("datepicker_value").value = newd2;

   if ( newd2.length < 10 && newd2.length >0 )
                                      { msg = msf + 'Ошибка.  Неправильная  дата  /  желаемое окончание работ.</font>';  }
   if ( newtx.length < 40 )     { msg = msf + 'Ошибка. Описание неисправностей должно быть более подробным.</font>';  }
   if ( newtl.length < 11 )     { msg = msf + 'Ошибка. Поле "телефон для связи" должно быть заполнено.</font>';  }
   if ( newna.length < 10 )     { msg = msf + 'Ошибка. Поле "фамилия имя отчество" должно быть заполнено.</font>';  }

   if ( msg == '' )
      {    if    (    end    ==    1    && document.getElementById('form-chk').checked )

      { let len = 1 + Zay.length;
        let uim = 0 + len + 100000;
        let uis = '' + uim;
        uim = sysdate().substr(0,4) + '-' + uis.substr(-4);
        ins = '' + len + ';' + uim + ';' + newna + ';' + newtx + ';' + newml + ';' + newtl + ';' + sysdate() + ';' + dateymd(newd2) + ';0;;;;;0;';
        evntzap(ins);
      } else
      {         document.getElementById('but-new').innerHTML  =  '<input  type="button" id="zp"    name="zp"    onclick="evntnew(1)" style="border: 2px solid blue; padding: 5px; margin:         2px;         cursor:pointer;" value="Подтверждаю,  что  заявка  заполнена правильно">';
       msg    =    '<input    type="checkbox" id="form-chk" name="form-chk"> Я даю согласие на обработку персональных данных';
      }
      }
   if    (    msg    !=    ''    )    { document.getElementById('evnt-new').innerHTML = msg; }
   }  // evntnew

function evntzap(a)                        // Ввод нового
   { let ins = a;
     let len = 1 + Zay.length;
     let uim = 0 + len + 100000;
     let uis = '' + uim;
     uim  =  sysdate().substr(0,4)  +  '-'  +  uis.substr(-4);
     if  (  evntins(Zay,ins,len)  ==  0  ) // вызов функции добавления в массив или в СУБД
       {      document.getElementById('evnt-new').innerHTML           =           '<font color="green">Поздравляем! Заявка на ремонт успешно добавлена.</font>';

document.getElementById('datepicker').style.display ='none'; pickf = 0;
//        document.getElementById('but-new').innerHTML  =  '<input  type="button" id="zp" name="zp" onclick="evntpod(' + (len-1)  +  ')"  style="border:  2px  solid  blue; padding: 5px; margin: 2px; cursor:pointer;" value="Перейти в Новое мероприятие">';
          document.getElementById('but-new').innerHTML = 'N заявки: <b>' + uim + '</b> от ' + datedmy(sysdate()).substr(0,10);
       }
   }  // evntzap

function  evntlog(logrl)  //  Авторизация Администраторя
   {
    Sort = 0; // Поле по которому сортируем номер поля
```

```
    Strl = 0; // Стрелка вверх/вниз. порядок сортировки (убывание, возрастание)

document.getElementById('flyin').style.display ='none';

document.getElementById('datepicker').style.display ='none'; pickf = 0;
    document.getElementById('evnt-table').innerHTML = '';
    document.getElementById('evnt-logn').style.display ='none';
    document.getElementById('evnt-exit').style.display ='';

    var txt = '<br><br><br><table border="0" cellpadding="4" cellspacing="2" bgcolor="#afafaf" width="33%">';

    txt = txt + '<tr class="v"><th colspan="2">Авторизация Администратора';
          txt = txt + '<tr class="s1" name="lg-ps1" id="lg-ps1"><td align="right">Логин:<td><input type="text" size="24" name="user-id" id="user-id" value="' + logid + '"></td></tr>' +
          '<tr class="s0 name="lg-ps0" id="lg-ps0""><td align="right">Пароль:<td><input type="password" size="24" name="user-pw" id="user-pw" value=""></td></tr>';
          txt = txt + '<tr class="s1"><td><td><input type="button" id="lg" name="lg" onclick="loginlog()" style="border: 2px solid green; padding: 5px; margin: 2px; cursor:pointer;" value="Авторизация">' +
          '<tr class="s0"><td colspan="2"><div name="login-log" id="login-log"></div></td>';
   txt = txt + '</table>';

   document.getElementById('evnt-table').innerHTML = txt;
   } // evntlog

function loginlog() // Проверка на валидность пар Логин-Пароль
   {
    let msg = '<font color="red">Имя или Пароль указаны неверно. Попробуйте еще раз.</font>';
    let logok = '';

    logid = document.getElementById("user-id").value;
    logpw = document.getElementById("user-pw").value;

    for (let i = 0; i < Adm.length; i++)
    { let str = Adm[i];
      if (str.indexOf(';')>0)
      { var ttp = str.split(';');
       if ( (ttp[3] == logid) && (ttp[4] == logpw) )
          { logok = ' ok-log'; }
      }
    } // Перебор зарегистрированных пользователей
   if (logok == '') // ошибка авторизации. пишем сообщение об ошибке
        { document.getElementById('login-log').innerHTML = msg;
        }
   else { evntfn(0); // перерисовываем таблицу со списком заявок, скрываем ненужные и показываем нужные кнопки в шапке
         document.getElementById('evnt-logn').style.display ='none';
         document.getElementById('evnt-exit').style.display ='';
        }
   } // loginlog

function evntcl() // очистка списка и условий сортировки. Выход. разлогинивание
   {
    Sort = 0; // Поле по которому сортируем номер поля
    Strl = 0; // Стрелка вверх/вниз. порядок сортировки (убывание, возрастание)
    document.getElementById('evnt-table').innerHTML = '';
    document.getElementById('evnt-logn').style.display ='';
    document.getElementById('evnt-exit').style.display ='none';

document.getElementById('datepicker').style.display ='none'; pickf = 0;
    document.getElementById('evnt-del').style.display ='none';
    document.getElementById('evnt-can').style.display ='none';
    evntnw();
   } // evntcl

function evntsrt(k,a) // сортировка строк массива (текстовая) k - номер поля по которому сортировать
   {
    for (let i = 0; i < Zay.length; i++)
      { let s = Zay[i];
       if (s.indexOf(';')>0) // разделитель полей в строке - массиве точка с запятой ;
         { var tmp = s.split(';');
           tmp[0] = '0' + tmp[k].toUpperCase(); // В массиве заявок в первый столбец дублируем данные из поля для сортировки
           Zay[i] = tmp.join(';');
         }
      }
    Sort = k; // k - номер поля по которому сортировали присвоем глобальной переменной Sort
    Zay.sort(); // Сортировка штатной функцией JS сортируем массив как текстовые строки с разделителями ';'
    Strl = Strl + 1;
    if (Strl>1) { Zay.reverse(); Strl = 0; } // если нужна обратная сортировка, то реверс массива и присвоение переменной Strl направления стрелки
    evntfn(a);
// перерисовываем таблицу с мероприятиями параметр 'a' просто передаем в другую функцию
   } // evntsrt
```

Продолжение приложения А

```
function evntupd(t,s,l,d)  // UPDATE (таблица, строка новые данные, начальное поле, индекс-uid строки)
   {  let res = 0;
      let upd = '';
      let idx = 0;
      for (let i = 0; i < t.length; i++)
      { let str = t[i];
       if (str.indexOf(';')>0)
         { let tt = str.split(';');
           if ( d == tt[1])
              { for (let j = 0; j < l; j++) { upd = upd + tt[j] + ';'; }
                idx = i;
              }
         }
      }  // for

    switch (DEBUG)
        {
        case 0:
console.log('debug:' + DEBUG + " Upd mysql;\n", "\n");
        break;
        case 1:
        t[idx] = upd + s;
console.log('debug:' + DEBUG + ' idx:' + idx + " Upd array;\n" + upd + s + "\n");
       break;
        case 2:
        t[d-1] = upd + s;
console.log('debug:' + DEBUG + ' idx:' + d + " Upd Local Storage;\n" + s + "\n");
            break;
         default:
            break;
        }
   return(res);
   }  // evntupd

function evntins(t,s,l)  // INSERT (таблица, длина таблицы строка)
   {  let res=0;
   switch (DEBUG)
        {
        case 0:
console.log('debug:' + DEBUG + " Ins mysql;\n", "\n");
        break;
        case 1:
        t.push(s);
console.log('debug:' + DEBUG + ' length:' + l + " Ins array;\n" + s + "\n");
       break;
        case 2:
        t.push(s);
console.log('debug:' + DEBUG + ' length:' + l + " Ins Local Storage;\n" + s + "\n");
            break;
         default:
            break;
        }
   return(res);
   }  // evntins

function evntini()  // получение сортированного массива заявок
   {
    Sort = 0; // Поле по которому сортируем номер поля
    Strl = 0; // Стрелка вверх/вниз. порядок сортировки (убывание, возрастание)

       switch (DEBUG)
        {
        case 0:
console.log('debug:' + DEBUG + " Init mysql;\n", "\n");
        break;
        case 1:
console.log('debug:' + DEBUG + " Init array;\n", "\n");

Zay[0] = '1;2023-0001;Сидоров Сидор Сидорович;Не включается планшет на ОС Android. Издает странные звуки, гудит.;Ex1@mail.ru;88002000600;2023-02-01;;2;2023-03-25 20:12:00;;0;';
Zay[1] = '2;2023-0002;Мошков Мошок Мошкович;Перестал заряжаться ноутбук. Наверно, надо батарею менять.;Ex2@mail.ru;89110009900;2023-02-03;2023-03-08;2;2023-03-22;;;0;';
Zay[2] = '3;2023-0003;михайлов михаим Михайлович;Ноутбук не держит зарядку. Вздулся аккумулятор. Наверно, надо батарею менять.;Example3@mail.ru;89876543312;2023-02-05;2023-03-08;1;2023-02-21 00:00:00;;;0;';
Zay[3] = '4;2023-0004;Иванов Иван Петрович;iPhon16. Не заряжается от универсального китайского зарядного устройства. Что-то пишет не по-русски;bigboss86@gmail.com;89123210120;2023-02-23;2023-09-05;1;2023-02-23;;;0;';
Zay[4] = '5;2023-0005;петренко лариса Ивановна;На ноутбуке на экране какие-то полосы появились и сильно гудит. Кошка уронила.;larisa38@mail.ru;89110008800;2023-02-28;;0;;;;0;';
Zay[5] = '6;2023-0006;Сидоренко Борис Полиграфович;На телефоне неизвестной марки самопроизвольно набираются номера из контактов даже когда заблокирован экран.;boriss568@list.ru;81234576789;2023-03-08;;0;;;;0;';
Zay[6] = '7;2023-0007;федоркович федр степанович;длинное описание для демонстрации показа всей ячейки или части описания неисправности. вот такой аппарат, а работать не хочет скидка на Великой Китайской распродаже на AliExpress! - Мини-ПК Beelink U59 Pro, Windows 11, Intel 11, N5105, DDR4, 8 ГБ, 16 ГБ, 512 Гб SSD, Двойной Wi-Fi, 1000M LAN, игровой мини-компьютер GK Mini S всего за 13 877,63 руб. проверка для демонстрации длинного описание;fedirik@mailt.ru;89876543201;2023-03-18;2023-04-01;0;;;;0;';
Zay[7] = '8;2023-0008;степанович петр борисовия;скидка на Великой Китайской распродаже на AliExpress! - Мини-ПК Beelink U59 Pro, Windows 11, Intel 11, N5105, DDR4, 8 ГБ, 16 ГБ, 512 Гб SSD, Двойной Wi-Fi, 1000M LAN, игровой мини-компьютер GK Mini S всего за 13 877,63 руб.;stepan1820@glist.ru;81233856789;2023-03-18;;0;;;;0;';
Zay[8] = '9;2023-0009;Федорчук лариса Ивановна;На ноутбуке все пропало.;larisa3338@mail.ru;89110008856;2023-03-24;;0;;;;0;';
```

Продолжение приложения А

```
Zay[9] = '10;2023-0010;семенов семен петрович;СРОЧНО ПОЧИНИТЕ ноутбук. залил сладким чаем и крышка не открывается;semensemen@simail.ru;89145008856;2023-03-24;2023-03-24;0;;;;0;';

Adm[0] = '1;1;Администратор;admin;admin;;;0;';
Adm[1] = '2;2;Администратор;1;1;;;0;';

        break;
         case 2:
console.log('debug:' + DEBUG + " Init Local Storage;\n", "\n");
        break;
        }
evntnw();
//   evntfn(0);  // Формируем таблицу заявок
    }  // evntini

</script>
<style>
a{text-decoration:none;color:white;}
a:hover{color:dodgerblue; text-decoration:underline;}

a.p{text-decoration:none;color:black;}
a.p:hover{color:dodgerblue; text-decoration:underline;}

a.k{text-decoration:none;color:gray;}
a.k:hover{color:dodgerblue; text-decoration:underline;}
.v{font-family:calibri;font-weight:bold;text-align:center;background:#8f8f8f;color:white;}
td{font-family:arial;font-weight:normal;}
td.r{font-family:arial;text-align:center;font-weight:normal;}
.s0{ background:#dfdfdf;}
.s1{ background:#efefef;}
.m0{ background:#afdfaf;}
.m1{ background:#bfefbf;}
.z0{ background:#696969;}
.z1{ background:#797979;}
.d0{ background:#dfafaf;text-decoration: line-through;}
.d1{ background:#efbfbf;text-decoration: line-through;}
.l0{ background:#ddabdd; }
.s0:hover{ background:#ffbfbf;color:green;}
.s1:hover{ background:#ffbfbf;color:green;}
.m0:hover{ background:#ffbfbf;color:green;}
.m1:hover{ background:#ffbfbf;color:green;}
.z0:hover{ background:#ffbfbf;color:green;}
.z1:hover{ background:#ffbfbf;color:green;}
.d0:hover{ background:#ffbfbf;color:green;text-decoration: line-through;}
.d1:hover{ background:#ffbfbf;color:green;text-decoration: line-through;}

.light2 { top:-1; left:-2; color:white; cursor:pointer; }
.shade2 { top:+1; left:+2; color:black; cursor:pointer; }
.fille2 { top:0; left:0; color:silver; cursor:pointer; }

</style>
</head>

<body bgcolor="#ffffff" background="./felix.png" onload='evntini()'>

<font style="position:absolute" size="+4" onclick="evntnw()">
<div style="position:absolute; cursor:pointer;" class="light2"> ООО «Феликс»</div>
<div style="position:absolute; cursor:pointer;" class="shade2"> ООО «Феликс»</div>
<div style="position:absolute; cursor:pointer;" class="fille2"> ООО «Феликс»</div>
</font>

<center>
<div name='evnt-login' id='evnt-login'>
<table border="0" width='99%'><tr>
<td align='left'>
<td align='right'>
<input type='button' name='evnt-del' id='evnt-del' onclick='evntdl()' style="border: 2px solid red; padding: 5px; margin: 2px; cursor:pointer; display:none;" value=''>
<input type='button' name='evnt-can' id='evnt-can' onclick='evntcn()' style="border: 2px solid green; padding: 5px; margin: 2px; cursor:pointer; display:none;" value='Отмена'>  
<input type='button' id='evnt-logn' name='evnt-logn' onclick='evntlog()' style="font-weight:bold; border: 2px solid green; padding: 5px; margin: 5px; border-radius: 10px; cursor:pointer;" value='- Я администратор -'>
<input type='button' id='evnt-exit' name='evnt-exit' onclick='evntcl()' style="font-weight:bold; border: 2px solid red; padding: 5px; margin: 5px; border-radius: 10px; cursor:pointer; display:none;" value='- Выйти -'>
</table>
</div>

<h3 id="flyin" style="position:relative;left:-1300;font-style:italic;"><font face="Arial">Здравствуйте! Чтобы оставить онлайн-заявку на ремонт техники в ООО «Феликс», воспользуйтесь формой ниже.</font></h3>

<div name='evnt-table' id='evnt-table'>
</div>
<br>
<div id="datepicker" style="display:none;" >

<script>

$(function(){
        $("#datepicker").datepicker({
                showOn: "both",
                onSelect: function(date){

        $('#datepicker value').val(date)
                }
```

Продолжение приложения А

```
        });
        $("#datepicker").datepicker("setDate"
, $('#datepicker_value').val());
});

if (document.getElementById||document.all)
   { var crossheader=document.getElementById?
document.getElementById("flyin").style      :
document.all.flyin.style; }

</script>

</center>
</body>
</html>
```