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

Кафедра «Прикладная математика и информатика»
(наименование)

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

Разработка социальных и экономических информационных систем
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка серверной части сайта (на примере компании АО «ВАЗСИСТЕМ»)»

Обучающийся Р.В. Шубин
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, О.В. Аникина
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, А. В. Егорова
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Бакалаврская работа на тему «Разработка серверной части сайта (на примере компании АО «ВАЗСИСТЕМ»».

Суть задачи заключается в разработке серверной части сайта для приёма заказов на разработку программного обеспечения. В качестве серверного фреймворка используется ASP.NET.

Тема является актуальной по причине сложности подачи технических заданий на разработку в компании АО «ВАЗСИСТЕМ». Разработка, описанная в данной работе, поможет облегчить подачу заявлений на разработку и привлечь больше клиентов. В перспективе это решение можно применять и в других компаниях, желающих увеличить количество заказов.

Работа начинается с описания предприятия и её деятельности. После этого приводиться проблема, существующая на предприятии, и проект её решения. Далее следует моделирование готового решения. Затем описывается разработка технического решения.

Общие сведения по работе: 58 страниц, 45 рисунка, 2 таблицы, 24 литературных источника.

Ключевые слова данной работы: серверная часть, АО «ВАЗСИСТЕМ», Firebase, Elastic Email, ASP.NET, приём заказов на разработку.

Abstract

The title of the graduation work is «Development of the Server-Side of a Website (Using JSC «VAZSYSTEM» as a Case Study)».

The aim of the work is to create the server-side component of a website for receiving software development orders. Employing ASP.NET as the server-side framework, this project addresses the challenge of submitting technical development requests at JSC «VAZSYSTEM». The developed solution aims to streamline the submission process and attract more clients.

The graduation work consists of 58 pages, including 45 figures, 2 tables, and a list of 24 references.

The senior paper may be divided into several logically connected parts which are a presentation of the company, identification of its challenges, detailing modeling of the solution, and detailing the technical development process. It examines the relationship between server-side development and order management effectiveness, aiming to offer a solution that addresses submission complexities at JSC «VAZSYSTEM». The study's conclusion emphasizes the significance of the developed solution in streamlining order submission processes and gives an understanding of next steps.

In conclusion, I'd like to stress that the results of the study underscore the importance of streamlining order submission processes in software development companies. The applicability of this solution does not necessarily have to be within the framework of JSC «VAZSYSTEM» and could potentially benefit other companies seeking to increase the volume of their orders.

Оглавление

Введение

В условиях стремительного развития цифровых технологий, компании сталкиваются с необходимостью внедрения программного обеспечения для оптимизации бизнес-процессов, повышения эффективности и конкурентоспособности. Однако малые бизнесы, не обладающие достаточным опытом и ресурсами, часто испытывают трудности при формировании заказа на разработку программного обеспечения. Это приводит к различным издержкам для компании, связанным с задержками в разработке, упущенной прибыли, потери конкурентоспособности.

Облегчение процесса составления и подачи технического задания является актуальной задачей, поскольку она позволяет малым бизнесам легко и быстро создавать структурированные технические задания, ускоряя процесс разработки программного обеспечения и повышая его качество. Такой инструмент может повысить эффективность взаимодействия между заказчиками и разработчиками, уменьшить количество недопониманий, сократить сроки разработки и снизить общие затраты.

Целью данной работы является разработка серверной части веб-приложения для АО «ВАЗСИСТЕМ», которое предоставит удобный и эффективный инструмент для составления технического задания на разработку программного обеспечения малым бизнесам. Для достижения поставленной цели были выполнены ряд задач:

- описание организации и её деятельности,
- описание бизнес-процесса, выбранного для оптимизации,
- проектирование технического решения,
- разработка технического решения,
- тестирование технического решения.

Объектом исследования будет процесс составления и подачи технических заданий на разработку программного обеспечения.

Предметом исследования выступает разработка серверной части веб-приложения для автоматизации процесса составления технического задания и подачи заказа на разработку.

Для достижения поставленной цели и решения задач мною было решено использовать эмпирический метод исследования, содержащий аналитические и практические подходы. Аналитический аспект включает анализ требований клиентов и особенностей процесса заказа программного обеспечения, а также существующего порядка принятия заказов. Практический аспект предполагает разработку и реализацию серверной части и тестирование полученного решения.

Все ведущие компании разработки программного обеспечения принимают заказы на разработку через электронную почту, что не всегда является удобным решением. Теоретическая значимость проекта заключается в новом варианте подхода к будущим клиентам, желающим заказать разработку программного обеспечения. Практическая значимость проекта заключается в предоставлении малым бизнесам доступного эффективного инструмента для составления технического задания и подачи заказа на разработку, что значительно улучшит их взаимодействие с разработчиками программного обеспечения. Таким образом компании смогут подавать заказы на разработку программного обеспечения без затруднений в понимании процесса подачи технического задания.

Разработанная система онлайн-заказов на разработку программного обеспечения была апробирована на тестовой группе пользователей и показана представителям компании АО «ВАЗСИСТЕМ». Результаты апробации показали, что система является эффективной и понятной для пользователей.

Данная работа содержит введение, три главы основного текста, заключение и список использованной литературы.

Глава 1 Характеристика организации и выбранного бизнес-процесса

1.1 Описание организации

В рамках данной работы, мною было разработано решение для компании АО «ВАЗСИСТЕМ». АО «ВАЗСИСТЕМ» - компания, входящая в состав группы компаний АО «АВТОВАЗ», которая является центром проектирования и разработки программных продуктов. Эти продукты используются для реализации различных ИТ-проектов. В ходе реализации проектов применяются как локальные, так и облачные решения. Количество ежегодно реализуемых проектов измеряется десятками.

Темы проектов АО «ВАЗСИСТЕМ» разнообразны и включают:

- управление HR,
- бухгалтерский и налоговый учет,
- взаимодействие с банками, контрагентами, государственными учреждениями,
- управление материальными ресурсами,
- конструкторская и технологическая подготовка производства,
- управление логистикой,
- управление производством товарной продукции,
- управление вспомогательными производствами,
- управление экономической оценкой деятельности,
- управление клиентами, дилерами, маркетингом,
- управление доставкой готовой продукции,
- управление качеством продукции, отработка претензий потребителей,
- поддержание инфраструктуры предприятия,
- автоматизация производственных процессов,
- управление юридически значимыми документами,
- контроль требований законодательства.

Также компания обеспечивает поддержку и развитие многих других информационных систем, применяемых на заводе и предприятиях группы АО «АВТОВАЗ». Компания имеет более ста шестидесяти программистов штате и считается одной их крупнейших ИТ-компаний Самарской области по численности ИТ-персонала.

В процессе практики у меня есть возможность получить опыт работы в различных проектах и приобрести практические навыки в разработке программного обеспечения. Моя практика проходит под руководством опытных профессионалов, которые помогают мне улучшить свои навыки и расширить знания в области ИТ-разработки. Эта практика позволяет мне применить теоретические знания, полученные в университете, на практике и понять, как функционируют различные информационные системы в реальной среде.

История компании АО «ВАЗСИСТЕМ» началась в 1991 году, когда она была создана в рамках совместного предприятия АО «АВТОВАЗ» и итальянской компании «Logosystem». Изначально АО «ВАЗСИСТЕМ» занималась поставкой компьютерного оборудования и разработкой программного обеспечения для ЭВМ. В 2007 году АО «АВТОВАЗ» стал единственным акционером компании, превратив ее в полностью зависимую дочернюю компанию крупнейшего автозавода в Европе. В 2021 году компания сменила коммерческое название на «INSOFT LADA» после проведения ребрендинга, но внутреннее название осталось «ВАЗСИСТЕМ».

Основной целью компании является разработка и предоставление программных продуктов и услуг, соответствующих потребностям современных предприятий. Они стремятся создавать технические решения высокого качества, которые гарантируют надежность, простоту использования и соответствуют лучшим практикам современных ИТ-технологий. Для успешной реализации проектов и поддержки функционирования систем АО «ВАЗАСИСТЕМ» отслеживает мировые тенденции в построении информационных систем, хранении и защите данных,

передаче информации, обеспечении качества информационного продукта. Главной стратегической целью компании является укрепление ее позиций в области разработки и внедрения сложных программных комплексов в России.

Основные принципы работы компании включают ориентацию на потребности клиента, предоставление услуг высокого качества, постоянное развитие профессиональных навыков сотрудников и соблюдение требований стандартов в сфере информационных технологий.

АО «INSOFT LADA» постоянно стремится улучшать свои предложения и расширять свою деятельность. Компания придает особое значение качественным техническим решениям и уровню удовлетворенности клиентов. Ее история тесно связана с разработкой ИТ-решений в автомобильной промышленности, и она продолжает оставаться лидером в этой области.

Компания АО «ВАЗСИСТЕМ» имеет собственный сайт, включающий публичную часть и портал для сотрудников. Также компания ведет Telegram-канал для сотрудников.

В настоящее время моя выпускная квалификационная работа проходит в центре разработки информационных систем, отвечающем за проектирование, разработку, тестирование и поддержку ИТ-решений для клиентов. Для понимания устройства центра необходимо показать его организационную структуру.

«Организационная структура управления предприятием (организацией) — это сложная, но упорядоченная совокупность взаимосвязанных подразделений, обеспечивающая функционирование предприятия (организации) как единого целого» [8].

Организационную структуру центра можно увидеть на рисунке 1. Центр имеет разветвленную структуру, включающую различные подразделения, которые работают совместно для достижения поставленных целей. Это включает в себя команду разработчиков, дизайнеров, тестировщиков, аналитиков и специалистов по поддержке.

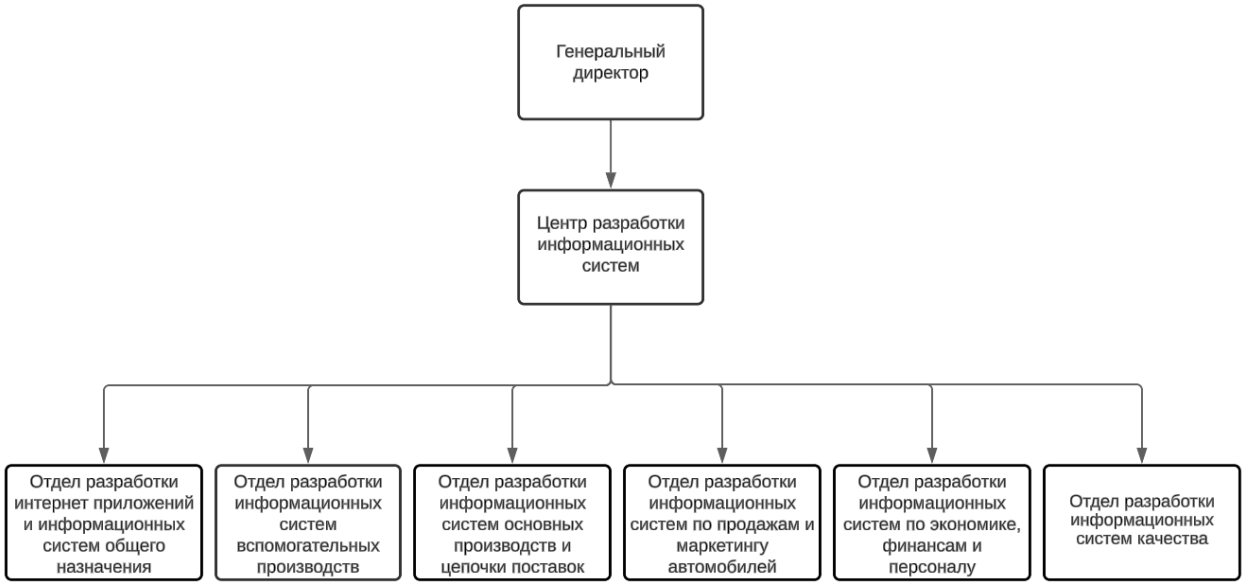


Рисунок 1 – Структура центра разработки информационных систем

«Принципы линейного и функционального управления используются в любой организационной структуре» [4].

В соответствии с уставом акционерного общества, центр разработки информационных систем осуществляет свою работу и выполняет конкретные функции. Для обеспечения эффективной работы подразделения клиенты и другие отделы компании направляют свои запросы через электронный документооборот «АВТОВАЗ», который в интеграции с MS Sharepoint представляет собой CRM-систему. Затем эти запросы обрабатываются и отслеживаются как заявки, которые назначаются конкретным специалистам в зависимости от их знаний и навыков. В ходе работы подразделения создаются закрытые заявки в системе CRM и составляются акты выполненных работ.

Для выполнения своих задач центр разработки информационных систем АО «ВАЗСИСТЕМ» активно использует различные технологии. Список включает следующие технологии:

1С – это платформа для автоматизации бизнес-процессов и разработки информационных систем. Она позволяет создавать и настраивать

программные продукты для управления учетом, торговлей, складскими операциями и другими задачами.

React – это JavaScript-библиотека для создания пользовательских интерфейсов. Она позволяет разрабатывать эффективные и масштабируемые веб-приложения с помощью модульной архитектуры и компонентного подхода. React использует виртуальный DOM для быстрого обновления элементов интерфейса.

Angular – это платформа и фреймворк для разработки веб-приложений. Он использует язык программирования TypeScript и позволяет создавать мощные и масштабируемые одностраничные приложения (SPA). Angular предоставляет инструменты для управления состоянием приложения, создания конечных точек, создания компонентов и связывания данных. Он также обладает множеством расширений и пакетов, которые помогают упростить разработку и повысить эффективность работы. Angular разработан и поддерживается компанией Google.

Java – это универсальный язык программирования, изначально разработанный для создания платформы Java. Он позволяет разрабатывать мобильные приложения, веб-приложения, встроенное программное обеспечение и многое другое. Java имеет богатую библиотеку классов и поддерживает объектно-ориентированное программирование.

Microsoft .NET – это платформа разработки, которая включает несколько языков программирования (например, C# и Visual Basic) и инструменты для создания приложений под Windows, веб-сайтов и веб-служб. .NET предоставляет мощный фреймворк для работы с данными, сетью, графикой и другими компонентами системы.

GitLab - веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git, интегрированный с системой тестирования, CI/CD пайплайном и другими функциями.

Python – это простой и удобный язык программирования, который активно используется для разработки веб-приложений, научных вычислений, анализа данных, искусственного интеллекта и других задач. Python обладает чистым синтаксисом и богатой стандартной библиотекой, что делает его популярным выбором для начинающих программистов.

Node.js – это среда выполнения JavaScript, которая позволяет запускать код JavaScript на стороне сервера. Она использует асинхронную модель обработки запросов, что делает Node.js эффективным для разработки высоконагруженных и масштабируемых веб-приложений. Node.js также позволяет использовать один язык программирования как на сервере, так и на клиентской стороне.

Apache Kafka – это распределенная система обмена сообщениями, предназначенная для обработки и передачи потоковых данных в реальном времени. Kafka основан на публикации/подписке и использует протоколы «Producer-Consumer». Он способен обрабатывать большие объемы данных с высокими скоростями записи и чтения. Kafka обладает высокой отказоустойчивостью и масштабируемостью, что делает его популярным выбором для обработки потоков данных в реальном времени.

REST API – это стандартная архитектурная модель для построения веб-сервисов и приложений с использованием HTTP-протокола. REST API позволяет взаимодействовать с сервером и обмениваться данными в формате JSON или XML. Он определяет набор операций (GET, POST, PUT, DELETE) для доступа к ресурсам и поддерживает принципы отсутствия сохранения состояния и кэширования. REST API является широко распространенным и понятным способом взаимодействия между клиентскими и серверными приложениями.

Apache NiFi – это мощная платформа для интеграции данных и автоматизации рабочих процессов. Она предоставляет графический интерфейс для создания, настройки и мониторинга потоков данных. NiFi позволяет обрабатывать, маршрутизировать и трансформировать данные на

основе различных источников и назначения. Платформа также поддерживает масштабирование и распределенную обработку данных.

PostgreSQL – это мощная и открытая реляционная база данных. Она предлагает множество возможностей, включая поддержку многопользовательской работы, транзакций, а также расширяемость через добавление пользовательских типов данных и функций. PostgreSQL имеет широкий набор функций, таких как полнотекстовый поиск, географические запросы, аналитические функции и многое другое. Эта база данных также обладает надежностью, предлагая механизмы резервного копирования, восстановления и репликации данных.

MySQL – это одна из самых популярных свободных реляционных баз данных. Она предлагает надежную и масштабируемую платформу для хранения, управления и извлечения данных. MySQL обладает широким набором функций, включая поддержку транзакций, многопользовательской работы, полнотекстового поиска, репликации данных и многое другое. Она компактна, быстра и проста в использовании, что делает ее популярным выбором для различных приложений, от веб-сайтов до корпоративных систем.

Oracle – это коммерческая реляционная база данных, разработанная корпорацией Oracle. Она предлагает высокую производительность, масштабируемость и надежность в обработке и хранении данных. Oracle обладает мощными возможностями, такими как поддержка транзакций, многопользовательская работа, аналитические функции и многое другое. Она также предоставляет расширяемость с помощью пользовательских типов данных и функций.

Docker – это открытая платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковывать приложения и их зависимости в изолированные и переносимые среды, облегчая развертывание и масштабирование приложений. Docker обеспечивает простой и эффективный способ создания, управления и развертывания контейнеров. Он также обладает инструментарием для оркестрации контейнеров, таких как

Docker Swarm и Kubernetes, что позволяет развертывать и управлять приложениями на нескольких хостах.

UI Path – это платформа для автоматизации процессов, основанная на технологиях искусственного интеллекта и машинного обучения. Она позволяет разработчикам создавать и обучать роботов для автоматического выполнения рутинных задач на рабочих местах. UI Path позволяет снизить рутинную работу сотрудников и повысить производительность бизнес-процессов.

OpenID Connect – это протокол аутентификации и авторизации, построенный на основе протокола OAuth 2.0. Он предоставляет механизмы для безопасной идентификации пользователей и предоставления доступа к ресурсам в распределенных системах. OpenID Connect добавляет слой аутентификации на основе токенов к протоколу OAuth 2.0, позволяя клиентам проверять идентичность пользователей через различные авторизированные серверы. OpenID Connect поддерживает разные сценарии аутентификации, включая открытую регистрацию, аутентификацию на основе пароля и внешнюю аутентификацию через социальные сети.

OAuth 2.0 – это протокол авторизации, который позволяет пользователям давать доступ к своим ресурсам третьим сторонам без необходимости раскрытия своих учетных данных. Он использует механизм выдачи временных токенов, которые затем используются клиентами для получения доступа к защищенным ресурсам. OAuth 2.0 поддерживает различные потоки авторизации, включая авторизацию кода авторизации, неявную авторизацию и поток авторизации с проверкой пароля. Он является широко принятым стандартом в индустрии для реализации безопасной авторизации и делегирования доступа к API и ресурсам.

Figma — это веб-приложение для дизайна интерфейсов, которое позволяет дизайнерам создавать, прототипировать и совместно работать над проектами в реальном времени. Это инструмент, который широко используется в дизайн-индустрии для разработки пользовательских

интерфейсов для веб-сайтов, мобильных приложений, а также для создания других графических проектов.

SonarCloud — это веб-платформа для непрерывного анализа кода, предназначенная для улучшения качества и безопасности программного обеспечения. Она предоставляет различные инструменты и функции статического анализа кода, которые помогают разработчикам обнаруживать потенциальные проблемы, ошибки, недочеты и уязвимости в их коде на ранних стадиях разработки.

Apache Hadoop — это программная экосистема с открытым исходным кодом, предназначенная для распределенной обработки больших наборов данных на кластерах из обычных серверов. Она включает в себя ядро Hadoop, которое обеспечивает распределенное хранение данных и параллельную обработку больших наборов данных через механизм Map Reduce. Hadoop способен масштабироваться от одной машины до тысяч машин, каждая из которых предлагает локальное хранилище и вычислительные ресурсы. Он устойчив к сбоям оборудования и разработан для автоматического выполнения задач параллельно на больших кластерах.

Дав характеристику организации, для которой мною разрабатывается программное решение, необходимо описать выбранный бизнес-процесс.

1.2 Описание выбранного бизнес-процесса

Задание на эту выпускную квалификационную работу подразумевает под собой разработку серверной части сайта для улучшения процесса приёма заказов на разработку, получаемых от заказчиков. Для начала необходимо визуализировать работу бизнес-процесса на данный момент и на момент внедрения программного решения. Это надо для понимания предстоящей работы.

Для отображения работы бизнес-процесса используются различные нотации моделирования. В таблице 1 представлено сравнение нотаций моделирования для бизнес-процессов.

Таблица 1 – Сравнение нотаций моделирования

	Стадия применения	Специфика области применения	Программное обеспечение
IDEF0	Исследование предприятия	Функциональное моделирование бизнес-процессов	Ramus
DFD	Исследование последовательности выполнения процессов	Потоки данных	Ramus
UML	Проектирование информационной системы	Моделирование программного обеспечения	StarUML
BPMN	Исследование бизнес-процессов	Последовательность выполнения бизнес-процессов	BPMN Studio

Так как стоит задача визуализировать работу бизнес-процесса, логично выбрать нотацию IDEF0.

«Нотация IDEFO (Integrated Computer Aided Manufacturing Definition) основана на методологии структурного анализа и проектирования SADT и служит для создания функциональной модели, включающей структурированное описание функций, действий или процессов моделируемой системы» [12].

«Диаграммы IDEF0 используются не только для иллюстрации процессов предприятия, но и в целом для описания схемы его функционирования. Они показывают, с какими ресурсами организация работает, от чего зависит и какой итоговый результат производит» [6].

Диаграмму IDEF0 начинают с контекстной диаграммы. «Создание модели IDEFO системы происходит в стиле декомпозиции, аналогичном используемому для создания иерархии» [20].

На рисунке 2 изображена контекстная диаграмма IDEF0 AI-IS, показывающая процесс получения оформления заказа на разработку программного обеспечения.

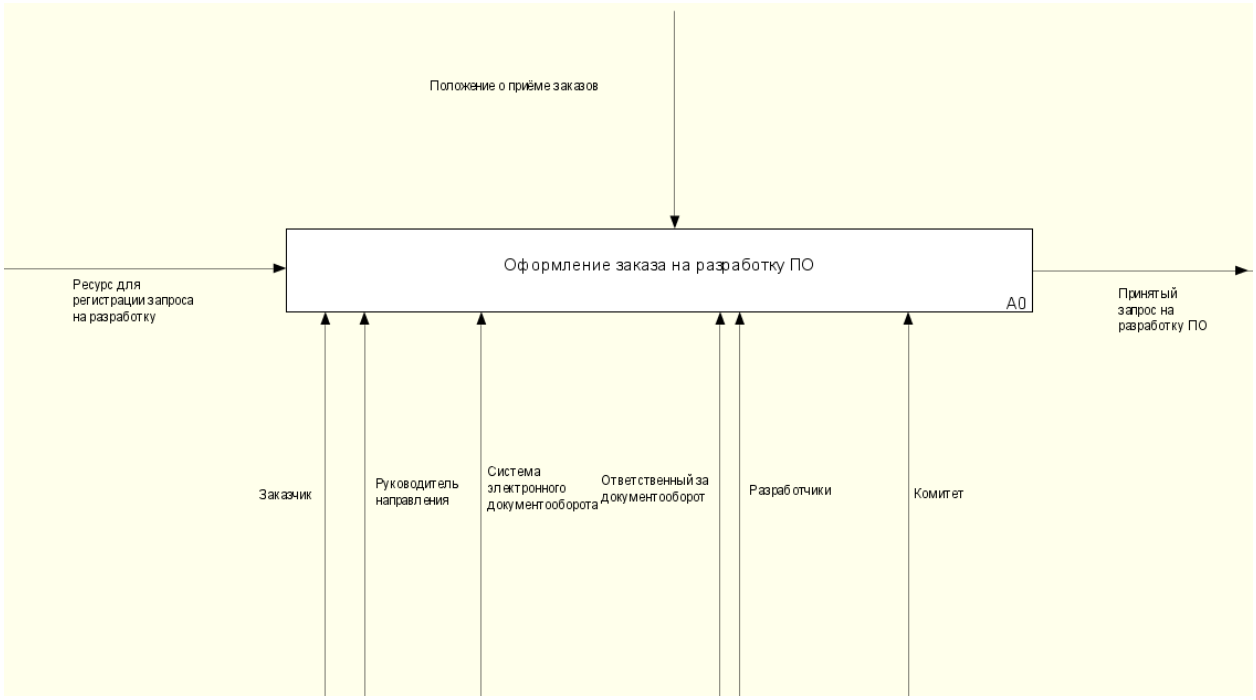


Рисунок 2 – Контекстная диаграмма IDEF0 AI-IS

Рассмотрим процесс принятия заказов на разработку программного обеспечения, представленный на контекстной диаграмме (рисунок 2). На вход поступает ресурс для регистрации запроса на разработку. Процесс регулируется положением о приёме заказов. В процессе учувствуют: заказчик, руководитель направления, система электронного документооборота, ответственный за документооборот, разработчики, комитет. На выходе этого бизнес-процесса находиться принятый запрос на разработку программного обеспечения.

Однако контекстная диаграмма не даёт детализацию. «Какие проблемы связаны с использованием IDEF? На верхнем уровне он требует множество такой информации, которая становится доступной лишь при достаточной детализации» [3]. Для полноты информации, данную контекстную диаграмму следует дополнить. рисунком 3 отображающем диаграмму IDEF0 AI-IS.

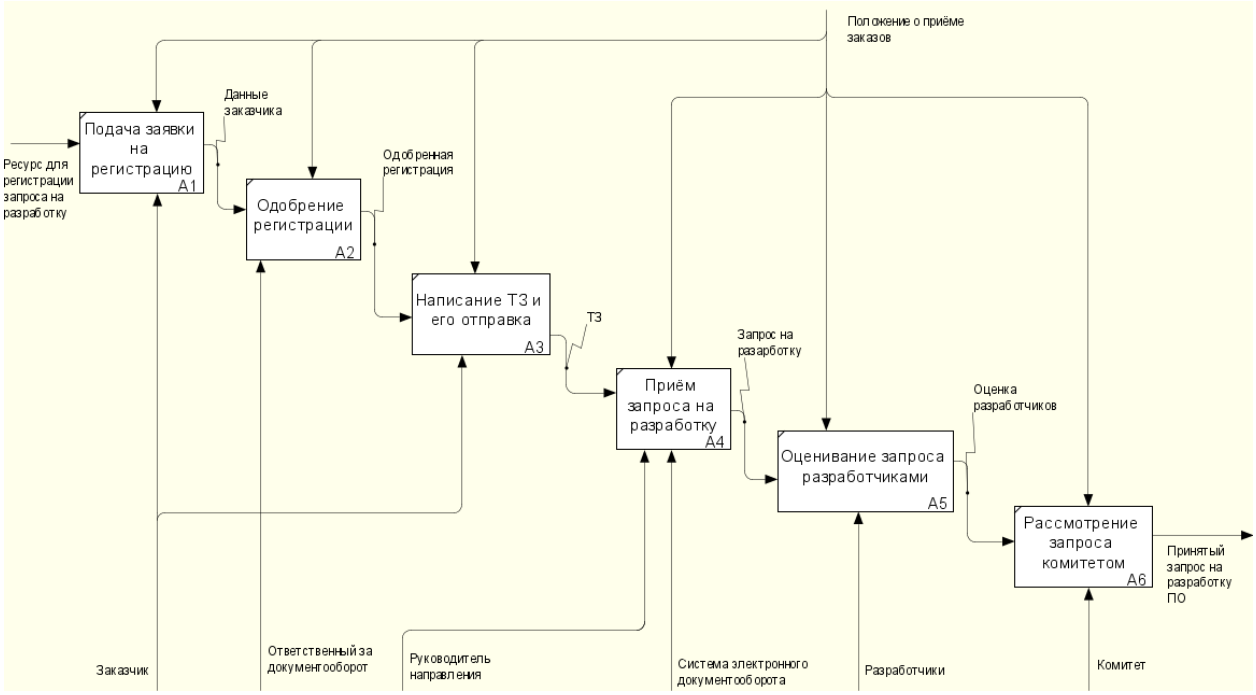


Рисунок 3 - Диаграмма IDEF0 AI-IS

Данная диаграмма показывает действия, происходящие во время процесса принятия заказов на разработку программного обеспечения. Сначала заказчик подаёт заявку на регистрацию. Затем ответственный за документооборот одобряет регистрацию. После одобрения заказчик может подать техническое задание. Техническое задание проходит через руководителя направления. Это делается для создания запроса на разработку. Сам запрос поступает на оценку разработчикам. Оценив запрос, разработчики высылают его на рассмотрение комитету. После рассмотрения комитетом, запрос становиться принятым. Действия A1 и A2 отображают один из недостатков текущего решения по приёму заказов. Заказы принимаются по

почте и ещё требуют одобрения регистрации заказчика в системе. Поэтому исключение одобрения регистрации поможет облегчить процесс подачи заказов на разработку программного обеспечения.

На рисунке 4 показана контекстная диаграмма IDEF0 TO-BE.

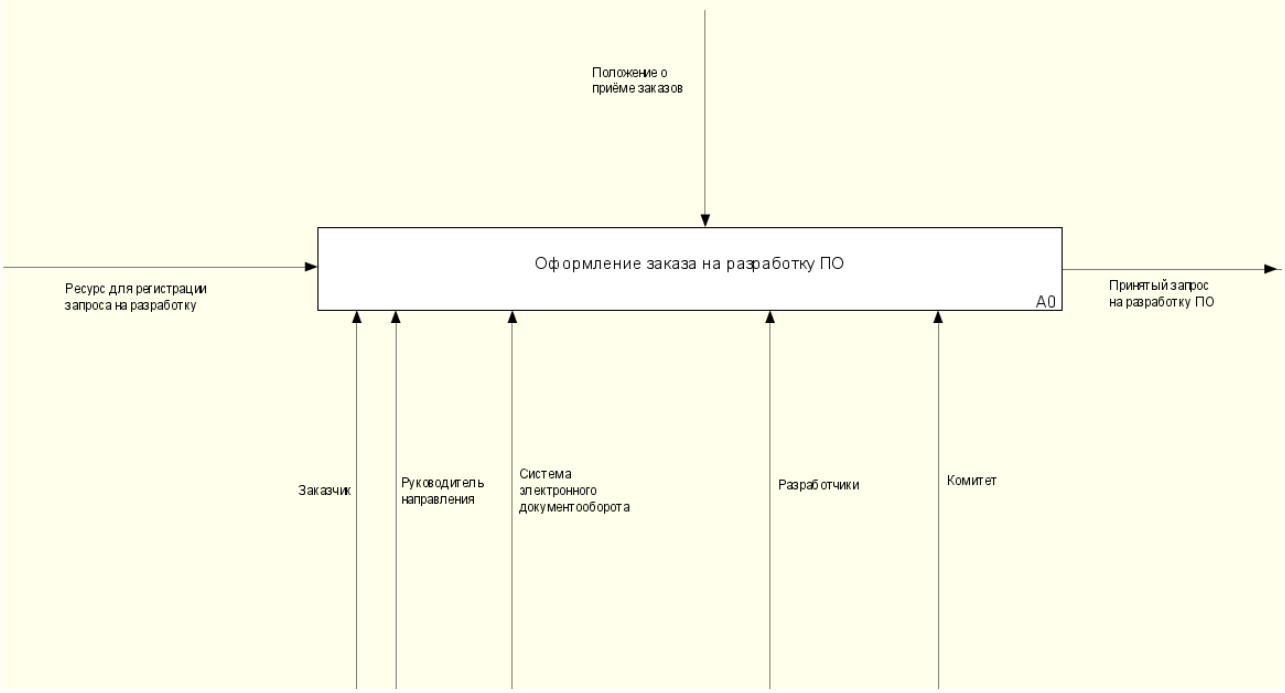


Рисунок 4 - Контекстная диаграмма IDEF0 TO-BE

Как видно из неё, бизнес-процесс имеет почти такие же параметры, как и раньше, но теперь нет необходимости в ответственном за документооборот. Это означает, что бизнес-процесс был модернизирован. Более подробно рассмотреть новую структуру процесса оформления заказа на разработку программного обеспечения можно на рисунке 5.

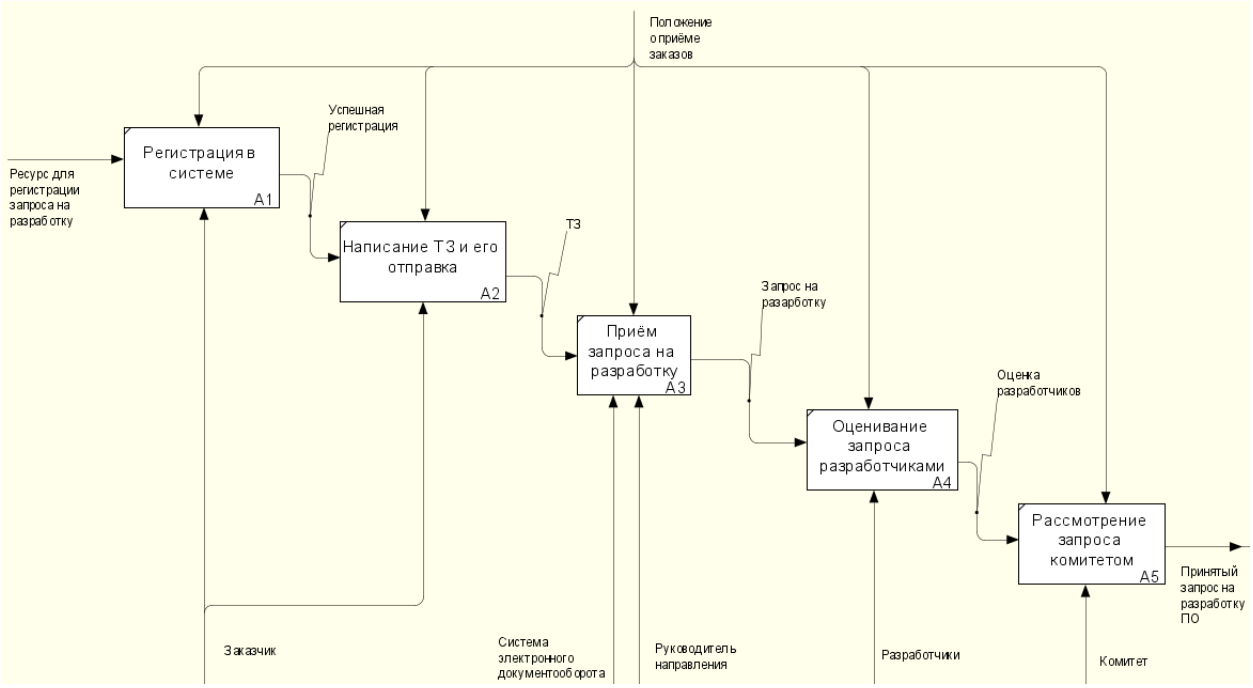


Рисунок 5 - Диаграмма IDEF0 TO-BE

Из этой диаграммы видно, что сократилось количество этапов бизнес-процесса, а также видно, что теперь регистрация проходит без одобрения со стороны ответственного лица.

Выводы по главе 1

В первой главе была описана организация, для которой разрабатывается программное решение. Также был описан бизнес-процесс, нуждающийся в оптимизации. Тем самым было выяснено, что компания АО «ВАЗСИСТЕМ», являющаяся ведущей в области разработки программного обеспечения, нуждается в оптимизации процесса оформления заказа на разработку.

Глава 2 Проектирование технического решения

2.1 План работ

В самом начале работы, необходимо знать какие действия необходимо выполнить для её успешного выполнения. «Планирование может быть сложным и трудоемким делом, и это, несомненно, объясняет, почему люди изобретают способы и средства для упрощения задачи» [24]. Для планирования работ можно составить диаграмму Ганта.

«Диаграмма Ганта (англ. Gantt chart, ленточная диаграмма, график Ганта, календарный график) — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту» [1].

«Диаграмма состоит из полос, ориентированных вдоль оси времени. Каждой полосе соответствует задача из списка слева. При этом задачи располагаются последовательно: или по приоритетности, или по логике выполнения» [5]. «Длина полосы показывает время, затрачиваемое на завершение работы или деятельности» [21].

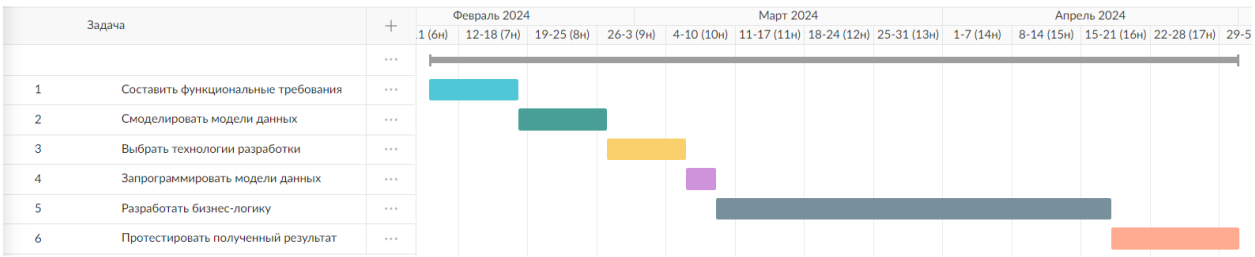


Рисунок 6 – План работ

На рисунке 6 мы можем увидеть план работ. Этот план отображает последовательность действий, необходимых для разработки серверной части сайта. Также этот план отображает временные промежутки, отведённые для каждого этапа.

План работ подразумевает, что необходимо последовательно выполнить следующие задачи:

- составить функциональные требования,
- смоделировать модели данных,
- выбрать технологии разработки,
- запрограммировать модели данных,
- разработать бизнес-логику,
- протестировать полученный результат.

После выполнения всех задач, будет получен результат в виде готового программного продукта.

2.2 Моделирование программного решения

Модернизация бизнес-процессов всегда должна иметь цель. Целью данной модернизации является разработка готового решения для АО «ВАЗСИСТЕМ», которое предоставит простой и эффективный инструмент для составления технического задания на разработку программного обеспечения малым бизнесам. Имея цель, можно узнать, что из себя будет представлять готовый продукт. Для начала следует указать функциональные требования будущего программного решения. Это следует сделать с использованием нотации UML.

«Унифицированный язык моделирования UML является методологией объектно-ориентированного подхода, представляющей набор диаграмм для проектирования информационных систем» [7]. Он позволяет визуализировать функциональные и нефункциональные требования к программному обеспечению, а также его архитектуру и дизайн. UML является мощным инструментом, который помогает разработчикам создавать более качественное, надежное и конкурентоспособное программное обеспечение.

«Всего существует 14 различных типов диаграмм UML, разделённых на две общие категории: структурные и поведенческие диаграммы» [19].

Функционал будущего веб-приложения лучше всего будет показать с помощью UML диаграммы вариантов использования, которая относится к категории поведенческих диаграмм UML. «Диаграмма вариантов использования (Use Case Diagram) описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования» [2]. «На диаграмме вариантов использования прямоугольник представляет границу системы, овалы представляют варианты использования, а человекоподобные фигурки представляют действующих лиц» [17].

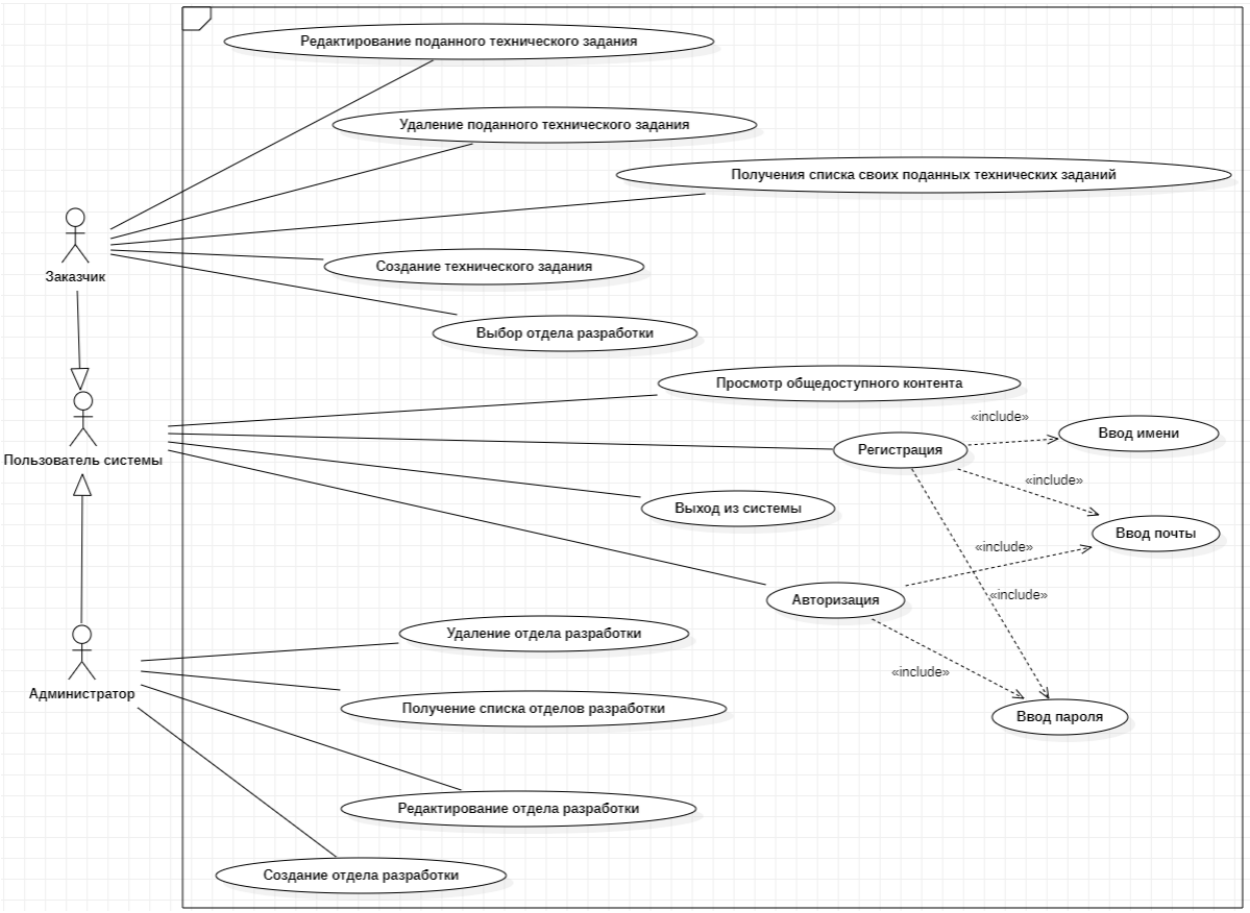


Рисунок 7 – Диаграмма вариантов использования

На рисунке 7 мы можем увидеть UML диаграмму вариантов использования. На ней мы видим три основные категории пользователей

системы: заказчики, обычные пользователи системы, администраторы. Каждая категория обладает определённым количеством доступных ей функций. Заказчики и администраторы имеют некоторые общие функции, потому как могут быть обобщены до пользователей системы. Эти общие функции включают: просмотр общедоступного контента, регистрацию, авторизацию, выход из системы. Заказчики могут выбирать отделы разработки, создавать технические задания, редактировать технические задания, удалять технические задания и просматривать список своих поданных технических заданий. Администраторы же могут создавать, редактировать, удалять и просматривать имеющиеся отделы разработки.

Также для описания работы будущего программного решения необходимо разработать его архитектуру (рисунок 8). На этапе выбора технологий данная архитектура поможет составить список необходимых инструментов.

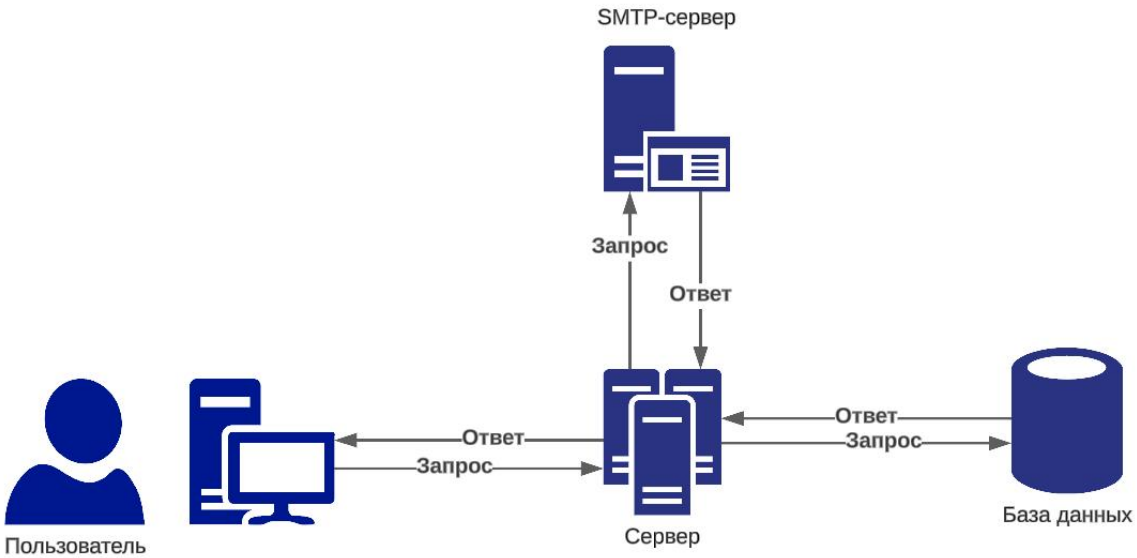


Рисунок 8 – Архитектура приложения

На рисунке 8 изображена архитектура будущего программного решения. Из этой архитектуры видно, что компьютер пользователя взаимодействует с сервером, который в свою очередь взаимодействует с SMTP-сервером и базой данных.

Теперь, поняв функционал будущего решения для удобного оформления заказа на разработку программного обеспечения, необходимо разработать модели данных, которые будут соответствовать данным, передаваемым между сервером и клиентом. Для этого подойдёт UML диаграмма классов. «Диаграмма классов (class diagram) - диаграмма, которая служит для представления совокупности декларативных или статических элементов модели, таких как классы, типы, а также разнообразных отношений между ними» [10]. UML диаграмма классов относиться к логическому уровню описания данных. «В современных технологиях различают логический уровень описания данных в модели и физический уровень. На логическом уровне данные носят абстрактный характер, они представляются и называются, как и в реальном мире и непосредственно связаны с исследуемыми процессами» [9].

«Общее представление класса содержит его имя, вместе с его значимыми атрибутами и операциями» [22]. В разрабатываемой системе, у классов есть атрибуты и отсутствует необходимость иметь операции. «Атрибут — это фрагмент информации, связанный с классом» [11]. Важно, чтобы атрибутов в моделях было достаточно для полноценной работы будущего программного решения. Также в составлении моделей данных, важным является указание типов данных. Знание типа данных атрибута модели становиться важным на этапе программирования.

Стоит начать с построения диаграммы классов, представляющей собой схему отношений между классами. Это поможет понять связь между классами и понадобиться при программировании моделей данных.

На рисунке 9 мы можем наблюдать диаграмму классов.

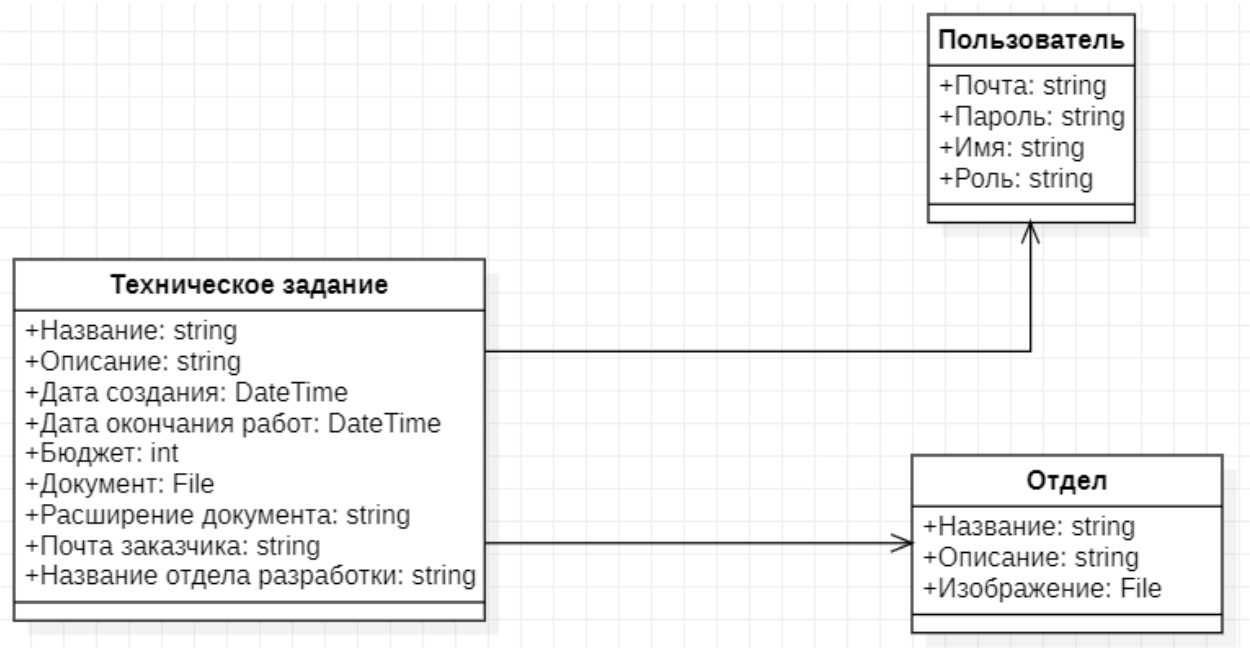


Рисунок 9 – Диаграмма классов

Теперь зная отношения между классами, можно проанализировать каждый класс по отдельности.

На рисунке 10 мы можем увидеть модель «Техническое задание».

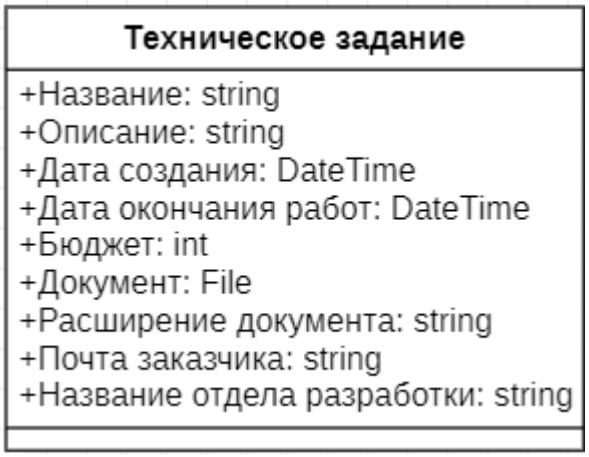


Рисунок 10 – Модель «Техническое задание»

В этой модели есть следующие атрибуты:

- название: string,

- описание: string,
- дата создания: DateTime,
- дата окончания работ: DateTime,
- бюджет: int,
- документ: File,
- расширение документа: string,
- почта заказчика: string,
- название отдела разработки: string.

Этот класс будет использоваться для приёма и передачи технических заданий. Поле «Расширение документа» необходимо поскольку тип данных «File» не хранит расширение документов, а также не позволяет записывать новые данные в свои поля. А поле «Название отдела разработки» нужно для того, чтобы знать какой стек разработки выбрал заказчик.

На рисунке 11 мы можем увидеть модель «Отдел».

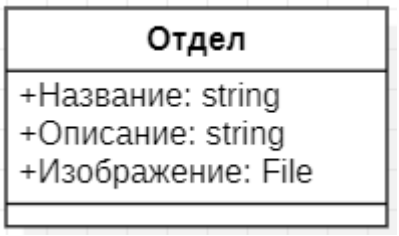


Рисунок 11 – Модель «Отдел»

Модель содержит следующие атрибуты:

- название: string,
- описание: string,
- изображение: File.

Данный класс будет использоваться для отделов разработки, между которыми заказчики будут выбирать при выборе типа программного обеспечения.

На рисунке 12 мы можем увидеть модель «Пользователь».

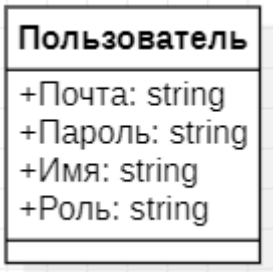


Рисунок 12 – Модель «Пользователь»

Видно, что модель содержит следующие атрибуты:

- почта: string,
- пароль: string,
- имя: string,
- роль: string.

Модель пользователя понадобиться для создания функционала регистрации и авторизации клиентов.

2.3 Выбор технологий

Важным аспектом является выбор технологий с помощью, которых будет разработано программное решение. В моём случае необходимо выбрать фреймворк, базу данных и сервис электронной почты. Следует начать с выбора фреймворка. В данном случае был выбран ASP.NET, поскольку он используется на предприятии АО «ВАЗСИСТЕМ», а также на нём проходит моё обучение в университете.

«ASP.NET – это серверный фреймворк, разработанный и созданный компанией Microsoft для создания динамических веб-страниц» [18]. Фреймворк предоставляет удобные средства для работы с базами данных, аутентификации и авторизации, управления состоянием и многими другими аспектами веб-разработки. Важно упомянуть, что в ASP.NET используется

MVC паттерн. «MVC расшифровывается как Model-View-Controller, что представляет собой шаблон проектирования, описывающий структуру приложения» [15].

Определив фреймворк для серверной части, можно выбрать систему управления базами данных. В данном случае было решено выбрать облачную базу данных, поскольку облачные базы данных позволяют в онлайн режиме получить доступ к данным. На рынке существует множество облачных баз данных, поэтому была составлена таблица 2, сравнивающая необходимые параметры наиболее популярных решений.

Таблица 2 – Сравнение облачных баз дынных

База данных	Тип	Масштабируемость	Наличие пакета для ASP.NET
Firebase	NoSQL	Высокая	+
MongoDB Atlas	NoSQL	Высокая	-
Neo4j Aura	NoSQL	Высокая	-
DataStax Astra DB	NoSQL	Высокая	-
CockroachDB	SQL	Высокая	-

Проведя сравнение облачных баз данных, мною была выбрана база дынных Firebase. Это было сделано по причине наличия пакета Firebase для ASP.NET. Также стоит отметить, что будут использованы следующие модули данной базы данных: Firebase Authentication, Firebase Realtime Database, Firebase Storage.

Firebase — это облачная система управления базами данных, разработанная Google. Она предоставляет разработчикам удобные API и средства аутентификации, аналитики, управления пользователями и другие инструменты для облегчения разработки и масштабирования приложений. Благодаря своей облачной архитектуре, Firebase также обеспечивает отказоустойчивость, высокую производительность и масштабируемость, что

делает его привлекательным выбором для разработчиков, которым нужна надежная и масштабируемая база данных для их приложений.

Firebase является нереляционной базой данных или по другому NoSQL. «Нереляционные базы данных хранят данные в неструктурированном формате. В отличие от SQL-баз данных с фиксированной схемой, NoSQL-базы данных хранят данные в виде документов (JSON), пар ключ-значение, графов или таблиц переменного размера» [16].

Также для отправки почтовых сообщений при регистрации пользователей будет задействован сервис Elastic Email. Сервис был выбран поскольку он легко интегрируется с ASP.NET.

Elastic Email — это сервис электронной почты и маркетинга, который предоставляет разнообразные инструменты для отправки и управления электронными письмами. Он позволяет пользователям создавать и настраивать кампании электронной почты, автоматизировать отправку сообщений, отслеживать результаты и анализировать эффективность своих кампаний.

Выводы по главе 2

Во второй главе было выполнено проектирование технического решения. А именно, было выполнено следующее: составлен план работ, смоделированы модели данных, функционал и архитектура программного решения, в качестве технологий разработки выбраны: ASP.NET, Firebase, Elastic Email.

Глава 3 Разработка технического решения

3.1 Разработка моделей данных

Программирование — это процесс создания программного обеспечения, включая написание и отладку кода, тестирование программы и обеспечение ее работоспособности. Программирование помогает человеку создавать множество вещей. В этом деле всегда важно принимать поступательные действия.

Программирование следует начать с создания моделей данных. Модели данных будут использоваться для передачи информации между клиентской и серверной частями. Стоит отметить, что Firebase является не реляционной базой данных, по этой причине идентификаторы записей в базе данных будут генерироваться самостоятельно.

Для управления данными технических заданий необходимо создать несколько версий модели данных. Это необходимая мера потому как для разных действий с этими данными необходим разный набор полей. Если бы небыли созданы несколько версий данной модели данных, то постоянно бы возникали поля с нулевыми значениями в них. Разные версии модели будут использоваться для работы с базой данных и представлениями, в которых содержаться формы и места для вывода данных.

Стоит начать с моделей для данных о технических заданиях, чтобы знать какие поля понадобятся для разных ситуаций. Стоит отметить, что данные о технических заданиях будут храниться с помощью следующих модулей Firebase: Firebase Realtime Database, Firebase Storage. Firebase Storage будет применяться для хранения файлов.

На рисунке 13 мы можем увидеть запрограммированную модель «Техническое задание» для действия создания технического задания.

```
public class TSViewModelForCreate
{
    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 3
    public string Name { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 3
    public string Description { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 1
    public DateTime CreationDate { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 3
    public DateTime Deadline { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    [Range(1, float.MaxValue, ErrorMessage = "Цена должна быть выше 0")]
    Ссылок: 3
    public int Budget { get; set; }
    Ссылок: 6
    public IFormFile? Document { get; set; }
    Ссылок: 1
    public string? DocumentExt { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 1
    public string ClientEmail { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 1
    public string DepartmentName { get; set; }
}
```

Рисунок 13 – Модель «Техническое задание» для действия создания

Данная модель будет применяться для подачи технических заданий на странице с выбором отдела разработки. Данная модель имеет аннотации для полей потому, как будет применяться для создания представления, где необходима валидация. Эти аннотации ещё иногда называют атрибутами. «Валидация на основе атрибутов связана с свойством, к которому применен атрибут валидации (некоторые атрибуты валидации также могут быть применены к классам)» [23].

На рисунке 14 показана запрограммированная модель «Техническое задание» для действия просмотра списка поданных технических заданий.

```
public class TSModelForIndex
{
    Ссылок: 3
    public string Id { get; set; }
    Ссылок: 2
    public string Name { get; set; }
    Ссылок: 2
    public string Description { get; set; }
    Ссылок: 2
    public DateTime CreationDate { get; set; }
    Ссылок: 2
    public DateTime Deadline { get; set; }
    Ссылок: 2
    public int Budget { get; set; }
    Ссылок: 2
    public string? DocumentExt { get; set; }
    Ссылок: 2
    public string DepartmentName { get; set; }
    Ссылок: 2
    public string? DownloadUrl { get; set; }
}
```

Рисунок 14 - Модель «Техническое задание» для действия просмотра списка

Данная модель применяется в представлении, которое относиться к личному кабинету пользователя.

На рисунке 15 продемонстрирована запрограммированная модель «Техническое задание» для действия редактирования технического задания.

```
public class TSViewModelForUpdate
{
    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public string Name { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public string Description { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public DateTime Deadline { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    [Range(1, float.MaxValue, ErrorMessage = "Цена должна быть выше 0")]
    Ссылок: 4
    public int Budget { get; set; }
    Ссылок: 6
    public IFormFile? Document { get; set; }
}
```

Рисунок 15 - Модель «Техническое задание» для действия редактирования

Эта модель также как и модель для создания имеет аннотации полей, потому что тоже используется в представлении, где необходима валидация.

На рисунке 16 мы видим запрограммированную модель «Техническое задание» для действия удаления технического задания.

```
public class TSModelForDelete
{
    Ссылок: 2
    public string Name { get; set; }
    Ссылок: 2
    public string Description { get; set; }
    Ссылок: 2
    public DateTime CreationDate { get; set; }
    Ссылок: 2
    public DateTime Deadline { get; set; }
    Ссылок: 2
    public int Budget { get; set; }
    Ссылок: 4
    public string? DocumentExt { get; set; }
    Ссылок: 2
    public string DepartmentName { get; set; }
    Ссылок: 4
    public string? DownloadUrl { get; set; }
}
```

Рисунок 16 - Модель «Техническое задание» для действия удаления

На рисунке 17 мы можем наблюдать запрограммированную модель «Техническое задание» для базы данных.

```
public class TSModelForDatabase
{

    public string Name { get; set; }

    public string Description { get; set; }
    Ссылок: 0
    public DateTime CreationDate { get; set; }
    Ссылок: 1
    public DateTime Deadline { get; set; }
    Ссылок: 1
    public int Budget { get; set; }
    Ссылок: 3
    public string? DocumentExt { get; set; }
    Ссылок: 0
    public string ClientEmail { get; set; }
    Ссылок: 0
    public string DepartmentName { get; set; }
}
```

Рисунок 17 - Модель «Техническое задание» для базы данных

Данная модель отражает данные, которые хранятся в Firebase Realtime Database.

Запрограммировав все модели для работы с данными о технических заданиях, можно приступить к программированию моделей для данных об отделах разработки. Отдельной модели для действия удаления не потребуется поскольку данные о отделах разработки хранятся только в Firebase Realtime Database.

На рисунке 18 видно запрограммированную модель «Отдел» для действия создания отдела разработки.

```
public class DepartmentModelForCreate
{
    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public string Name { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public string Description { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 4
    public IFormFile Image { get; set; }
}
```

Рисунок 18 - Модель «Отдел» для действия создания

Данная модель имеет аннотации полей для валидации формы создания.

На рисунке 19 показана запрограммированная модель «Отдел» для действия просмотра списка отделов разработки.

```
public class DepartmentModelForIndex
{
    Ссылок: 5
    public string Id { get; set; }
    Ссылок: 2
    public string Name { get; set; }
    Ссылок: 2
    public string Description { get; set; }
    Ссылок: 1
    public byte[] Image { get; set; }
}
```

Рисунок 19 - Модель «Отдел» для действия просмотра списка

На рисунке 20 мы видим запрограммированную модель «Отдел» для действия редактирования отдела разработки.

```
public class DepartmentModelForUpdate
{
    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 5
    public string Name { get; set; }

    [Required(ErrorMessage = "Это поле обязательно")]
    Ссылок: 5
    public string Description { get; set; }
    Ссылок: 4
    public IFormFile? Image { get; set; }
}
```

Рисунок 20 - Модель «Отдел» для действия редактирования

Модель имеет аннотации полей для валидации формы редактирования.

На рисунке 21 продемонстрирована запрограммированная модель «Отдел» для базы данных.

```
public class DepartmentModelForDatabase
{
    Ссылок: 9
    public string Name { get; set; }
    Ссылок: 8
    public string Description { get; set; }
    Ссылок: 7
    public byte[] Image { get; set; }
}
```

Рисунок 21 - Модель «Отдел» для базы данных

Теперь необходимо запрограммировать модели для авторизации и регистрации пользователей системы. Данный функционал необходим, поскольку составленные ранее функциональные требования говорят, что необходимо разграничивать доступный функционал в зависимости от роли

пользователя в системе. Хранение данных о пользователях системы будет производиться с помощью модуля Firebase Authentication.

На рисунке 22 мы можем наблюдать запрограммированную модель «Авторизация».

```
public class LoginModel
{
    Ссылок: 1
    public string Email { get; set; }

    Ссылок: 1
    public string Password { get; set; }
}
```

Рисунок 22 - Модель «Авторизация»

Для входа в систему будет необходимо ввести почту и пароль, введённые на этапе регистрации.

На рисунке 23 мы можем просмотреть запрограммированную модель «Регистрация».

```
public class RegistModel
{
    Ссылок: 3
    public string Email { get; set; }

    Ссылок: 1
    public string Password { get; set; }

    Ссылок: 2
    public string Username { get; set; }
}
```

Рисунок 23 - Модель «Регистрация»

Данный класс представляет то, как в базе данных будут храниться данные о пользователях, а также используется на этапе регистрации пользователя.

3.2 Разработка бизнес-логики

В интернете существует множество веб-приложений, но их всех объединяет наличие бизнес-логики. Бизнес-логика веб-приложений – это набор правил и процессов, которые определяют, как приложение должно функционировать согласно бизнес-требованиям и целям.

Программирование бизнес-логики стоит начать с программирования функционала для администраторов системы потому как без отделов разработки нельзя будет создавать технические задания.

На рисунке 24 можно наблюдать метод контроллера «Admin» для отображения списка отделов разработки.

```
[HttpGet]
Ссылок: 0
public async Task<ActionResult> Index()
{
    try
    {
        var departments = await _firebaseClient
        .Child("departments")
        .OnceAsync<DepartmentModelForDatabase>();

        var departmentListToView = departments.Select(d => new DepartmentModelForIndex
        {
            Id = d.Key,
            Name = d.Object.Name,
            Description = d.Object.Description,
            Image = d.Object.Image
        }).ToList();

        return View(departmentListToView);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 24 – Метод контроллера «Admin» для отображения списка отделов разработки

На рисунке 25 показан метод контроллера «Admin» для создания отдела разработки.

```
[HttpGet]
Ссылок: 0
public ActionResult CreateDepartment()
{
    return View();
}

[HttpPost]
Ссылок: 0
public async Task<ActionResult> CreateDepartment(DepartmentModelForCreate department)
{
    try
    {
        DepartmentModelForDatabase depart = new DepartmentModelForDatabase();
        depart.Name = department.Name;
        depart.Description = department.Description;
        using (var memoryStream = new MemoryStream())
        {
            department.Image.CopyTo(memoryStream);
            depart.Image = memoryStream.ToArray();
        }

        await _firebaseClient
            .Child("departments")
            .PostAsync(depart);

        return RedirectToAction("Index", "Admin");
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 25 – Метод контроллера «Admin» для создания отдела разработки

Потому как данные об отделах разработки хранятся только в Firebase Realtime Database, изображения хранятся в виде массива байт.

На рисунке 26 продемонстрирован метод контроллера «Admin» для редактирования отдела разработки.

```
[HttpGet("[controller]/UpdateDepartment/{uid}")]
Ссылок: 0
public async Task<ActionResult> UpdateDepartment(string uid)
{
    try
    {
        var department = await _firebaseClient
            .Child("departments")
            .Child(uid)
            .OnceSingleAsync<DepartmentModelForDatabase>();

        if (department != null)
        {
            var departmentModel = new DepartmentModelForUpdate
            {
                Name = department.Name,
                Description = department.Description
            };

            return View(departmentModel);
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

```
[HttpPost("[controller]/UpdateDepartment/{uid}")]
Ссылок: 0
public async Task<ActionResult> UpdateDepartment(string uid, DepartmentModelForUpdate department)
{
    try
    {
        var departmentToUpdate = await _firebaseClient
            .Child("departments")
            .Child(uid)
            .OnceSingleAsync<DepartmentModelForDatabase>();

        if (departmentToUpdate != null)
        {
            departmentToUpdate.Name = department.Name;
            departmentToUpdate.Description = department.Description;
            if (department.Image != null)
            {
                using (var memoryStream = new MemoryStream())
                {
                    department.Image.CopyTo(memoryStream);
                    departmentToUpdate.Image = memoryStream.ToArray();
                }
            }

            await _firebaseClient
                .Child("departments")
                .Child(uid)
                .PutAsync(departmentToUpdate);

            return RedirectToAction("Index", "Admin");
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 26 – Метод контроллера «Admin» для редактирования отделов разработки

В методе Get не передаётся изображение из-за правил безопасности браузеров по отношению к полям ввода.

На рисунке 27 мы видим метод контроллера «Admin» для удаления отдела разработки.

```
[HttpGet("[controller]/DeleteDepartment/{uid}")]
Ссылок: 0
public async Task<ActionResult> DeleteDepartment(string uid)
{
    try
    {
        var department = await _firebaseClient
            .Child("departments")
            .Child(uid)
            .OnceSingleAsync<DepartmentModelForDatabase>();

        if (department != null)
        {
            return View(department);
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

```
[HttpPost("[controller]/DeleteDepartment/{uid}")]
Ссылок: 0
public async Task<ActionResult> DeleteDepartmentPost(string uid)
{
    try
    {
        var departmentToDelete = await _firebaseClient
            .Child("departments")
            .Child(uid)
            .OnceSingleAsync<DepartmentModelForDatabase>();

        if (departmentToDelete != null)
        {
            await _firebaseClient
                .Child("departments")
                .Child(uid)
                .DeleteAsync();

            return RedirectToAction("Index", "Admin");
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 27 – Метод контроллера «Admin» для удаления отдела разработки

Создав функционал для отделов разработки, можно создать метод для главной страницы, на которой клиенты будут выбирать отдел разработки.

На рисунке 28 мы можем увидеть метод контроллера «Home» для вывода списка отделов разработки.

```
[HttpGet]
Ссылок: 0
public async Task<ActionResult> Index()
{
    try
    {
        var departments = await _firebaseClient
        .Child("departments")
        .OnceAsync<DepartmentModelForDatabase>();

        var departmentListToView = departments.Select(d => new DepartmentModelForDatabase
        {
            Name = d.Object.Name,
            Description = d.Object.Description,
            Image = d.Object.Image
        }).ToList();

        return View(departmentListToView);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 28 – Метод контроллера «Home» для вывода списка отделов разработки

Он отличен от метода в контроллере «Admin» тем, что в нём не передаются идентификаторы отделов разработки. Они не передаются потому как нету необходимости создавать динамические ссылки в представлении

Далее следует создать функционал регистрации и входа в систему.

Рисунок 29 показывает метод контроллера «Account» для регистрации клиентов в системе.

```
[HttpGet]
Ссылок: 0
public IActionResult Regist()
{
    return View();
}

[HttpPost]
Ссылок: 0
public async Task<IActionResult> Regist(RegistModel user)
{
    try
    {
        var userCredential = await _client.CreateUserWithEmailAndPasswordAsync(user.Email, user.Password, user.Username);

        await FirebaseAuth.DefaultInstance.SetCustomUserClaimsAsync(userCredential.User.Uid, new Dictionary<string, object>
        {
            { "role", "user" }
        });

        var emailActionLink = await FirebaseAuth.DefaultInstance.GenerateEmailVerificationLinkAsync(user.Email);

        _emailService.SendConfirmationEmail(user.Email, emailActionLink, user.Username);

        ViewBag.ConfirmationMessage = "На вашу почту было отправлено письмо с подтверждением";
        return View();
    }
    catch (Exception ex)
    {
        if (ex.Message.Contains("EMAIL_EXISTS"))
        {
            ViewBag.Error = "Пользователь с такой почтой уже существует!";
            return View();
        }
        else
        {
            return BadRequest(ex.Message);
        }
    }
}
```

Рисунок 29 – Метод контроллера «Account» для регистрации

Помимо данных из формы регистрации, в базу данных производиться запись роли пользователя, а также высылается письмо с подтверждением.

Рисунок 30 демонстрирует метод для отправки почты.

```
public void SendConfirmationEmail(string recipientEmail, string confirmationLink, string username)
{
    string smtpHost = "smtp.gmail.com";
    int smtpPort = 587;
    string smtpUsername = _configuration.GetValue<string>("SMTP_Username");
    string smtpPassword = _configuration.GetValue<string>("SMTP_Password");

    MailMessage message = new MailMessage();
    message.From = new MailAddress(_configuration.GetValue<string>("SMTP_Username"));
    message.To.Add(new MailAddress(recipientEmail));
    message.Subject = "Подтверждение регистрации";
    message.BodyEncoding = System.Text.Encoding.UTF8;
    message.IsBodyHtml = true;

    string htmlTemplate = System.IO.File.ReadAllText(Path.Combine("wwwroot", "html", "confirmation_email_template.html"));
    htmlTemplate = htmlTemplate.Replace("{{username}}", username);
    htmlTemplate = htmlTemplate.Replace("{{confirmationLink}}", confirmationLink);
    message.Body = htmlTemplate;

    SmtpClient smtpClient = new SmtpClient(smtpHost, smtpPort);
    smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
    smtpClient.EnableSsl = true;
    smtpClient.UseDefaultCredentials = false;
    smtpClient.Credentials = new NetworkCredential(smtpUsername, smtpPassword);

    smtpClient.SendMailAsync(message);
}
```

Рисунок 30 – Метод для отправки почты

Данный метод используется при регистрации новых пользователей.

На рисунке 31 мы видим метод контроллера «Account» для авторизации клиентов в системе.

```
[HttpGet]
public IActionResult Login(string? returnUrl)
{
    if (returnUrl != null)
    {
        var options = new CookieOptions
        {
            MaxAge = TimeSpan.FromMinutes(5),
            HttpOnly = true,
            Secure = true,
            SameSite = SameSiteMode.Strict
        };

        Response.Cookies.Append("returnUrl", returnUrl, options);
    }
    return View();
}
```

```
[HttpPost]
public async Task<IActionResult> Login(LoginModel user)
{
    try
    {
        var userCredential = await _client.SignInWithEmailAndPasswordAsync(user.Email, user.Password);
        string uid = userCredential.User.Uid;

        if (!string.IsNullOrEmpty(uid))
        {
            var userData = await FirebaseAuth.DefaultInstance.GetUserAsync(uid);

            var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType, userData.DisplayName),
                new Claim(ClaimTypes.Email, userData.Email)
            };

            if (userData.CustomClaims != null && userData.CustomClaims.TryGetValue("role", out var roleValue))
            {
                string role = roleValue?.ToString();
                claims.Add(new Claim(ClaimsIdentity.DefaultRoleClaimType, role));
            }

            var claimsIdentity = new ClaimsIdentity(claims, "Cookies");
            var claimsPrincipal = new ClaimsPrincipal(claimsIdentity);
            await HttpContext.SignInAsync(claimsPrincipal);

            var returnUrl = Request.Cookies["returnUrl"];
            Response.Cookies.Delete("returnUrl");
            return Redirect(returnUrl ?? "/");
        }
        return BadRequest();
    }
    catch (Exception ex)
    {
        if (ex.Message.Contains("INVALID_LOGIN_CREDENTIALS"))
        {
            ViewBag.Error = "Пользователь не найден или не правильно введены данные !";
            return View();
        }
        else
        {
            return BadRequest(ex.Message);
        }
    }
}
```

Рисунок 31 – Метод контроллера «Account» для авторизации

При авторизации устанавливаются куки необходимые для работы веб-приложения.

На рисунке 32 можно наблюдать метод контроллера «Account» для выхода из системы. В нём происходит удаление всех куки файлов.

```
[HttpGet]
Ссылок: 0
public async Task<IActionResult> Logout()
{
    await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    return RedirectToAction("Index", "Home");
}
```

Рисунок 32 – Метод контроллера «Account» для выхода из системы

Теперь, когда готов функционал работы с отделами разработки, регистрации и авторизации, можно реализовать функционал технических заданий.

На рисунке 33 мы можем видеть метод контроллера «TS» для вывода списка поданных технических заданий.

```
[HttpGet]
Ссылок: 0
public async Task<ActionResult> Index()
{
    try
    {
        var tsList = await _firebaseClient
            .Child("ts")
            .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
            .OnceAsync<TSModelForDelete>();

        foreach (var ts in tsList)
        {
            if (ts.Object.DocumentExt != null)
            {
                var storageClient = _firebaseStorage.Child("ts").Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_")).Child(ts.Key + ts.Object.DocumentExt);
                var downloadUrl = await storageClient.GetDownloadUrlAsync();
                ts.Object.DownloadUrl = downloadUrl.ToString();
            }
        }

        var tsListToView = tsList.Select(d => new TSModelForIndex
        {
            Id = d.Key,
            Name = d.Object.Name,
            Description = d.Object.Description,
            CreationDate = d.Object.CreationDate,
            Deadline = d.Object.Deadline,
            Budget = d.Object.Budget,
            DocumentExt = d.Object.DocumentExt,
            DepartmentName = d.Object.DepartmentName,
            DownloadUrl = d.Object.DownloadUrl
        }).ToList();

        return View(tsListToView);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 33 – Метод контроллера «TS» для вывода списка технических заданий

Можно заметить, что данные берутся из Firebase Realtime Database, а файлы из Firebase Storage. Для того, чтобы клиенты могли посмотреть файлы, которые они прикладывали с техническими заданиями, создаются ссылки на их скачивание.

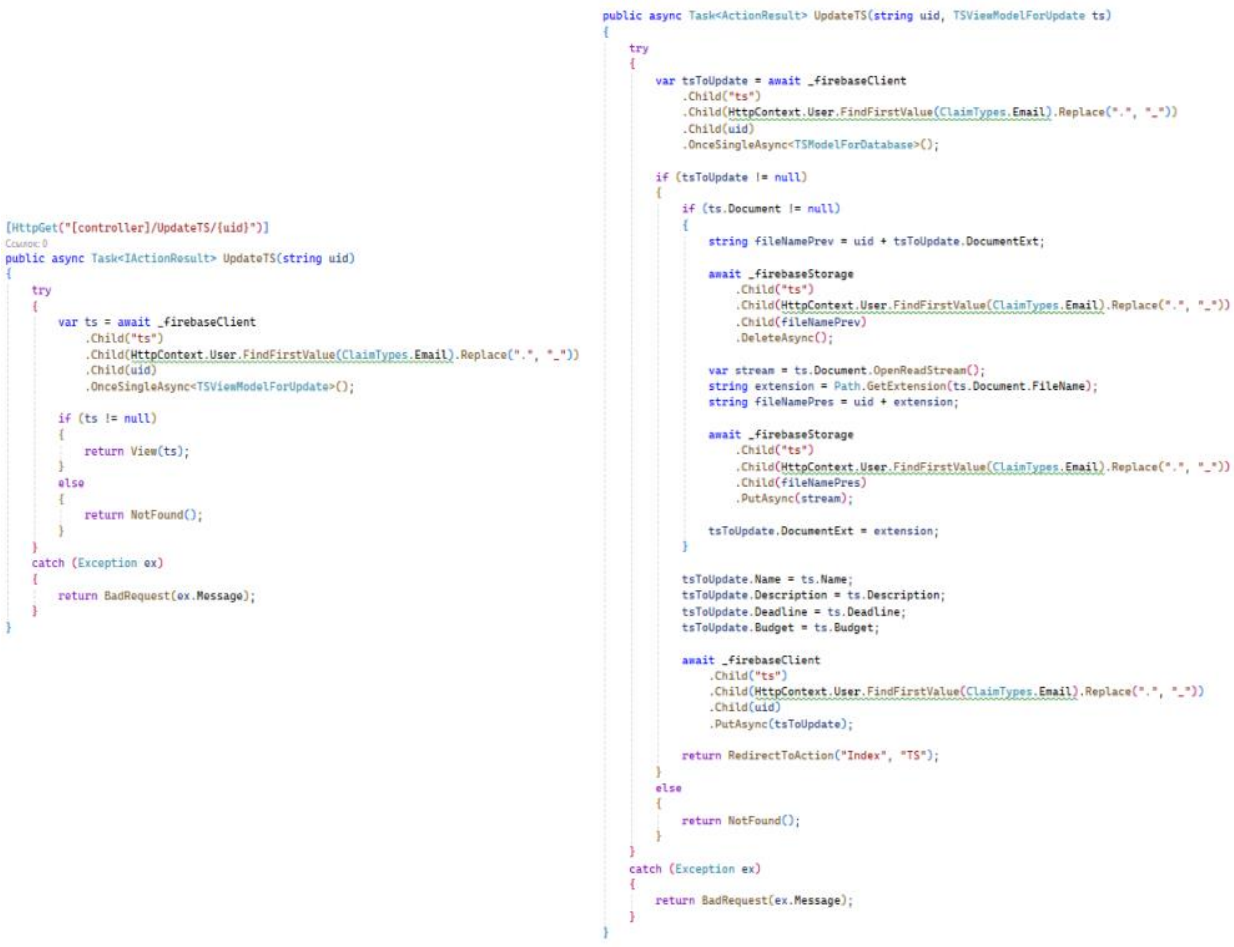
На рисунке 34 показан метод контроллера «TS» создания технического задания.

```
[HttpGet]
Ссылок: 0
public IActionResult CreateTS(string DepartmentName)
{
    return View();
}

[HttpPost]
Ссылок: 0
public async Task<ActionResult> CreateTS(TSViewModelForCreate ts)
{
    try
    {
        if (ts.Document != null)
        {
            var stream = ts.Document.OpenReadStream();
            string extension = Path.GetExtension(ts.Document.FileName);

            ts.Document = null;
            ts.DocumentExt = extension;

            var postResponse = await _firebaseClient.Child("ts").Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_")).PostAsync(ts);

            string fileName = postResponse.Key;

            await _firebaseStorage
                .Child("ts")
                .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                .Child(fileName + extension)
                .PutAsync(stream);
        }
        else
        {
            var postResponse = await _firebaseClient.Child("ts").Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_")).PostAsync(ts);
        }
        return RedirectToAction("Index", "TS");
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 34 - Метод контроллера «TS» для создания технического задания

Для каждого клиента существует своя папка с его техническими заданиями. Даже зайдя в интерфейс базы данных можно найти поданные технические задания определённого клиента.

Рисунок 35 иллюстрирует метод контроллера «TS» для редактирования технического задания.



```
[HttpGet("[controller]/UpdateTS/{uid}")]
Ссылок: 0
public async Task<IActionResult> UpdateTS(string uid)
{
    try
    {
        var ts = await _firebaseClient
            .Child("ts")
            .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
            .Child(uid)
            .OnceSingleAsync<TSViewModelForUpdate>();

        if (ts != null)
        {
            return View(ts);
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

```
public async Task<ActionResult> UpdateTS(string uid, TSViewModelForUpdate ts)
{
    try
    {
        var tsToUpdate = await _firebaseClient
            .Child("ts")
            .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
            .Child(uid)
            .OnceSingleAsync<TSModelForDatabase>();

        if (tsToUpdate != null)
        {
            if (ts.Document != null)
            {
                string fileNamePrev = uid + tsToUpdate.DocumentExt;

                await _firebaseStorage
                    .Child("ts")
                    .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                    .Child(fileNamePrev)
                    .DeleteAsync();

                var stream = ts.Document.OpenReadStream();
                string extension = Path.GetExtension(ts.Document.FileName);
                string fileNamePres = uid + extension;

                await _firebaseStorage
                    .Child("ts")
                    .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                    .Child(fileNamePres)
                    .PutAsync(stream);

                tsToUpdate.DocumentExt = extension;
            }

            tsToUpdate.Name = ts.Name;
            tsToUpdate.Description = ts.Description;
            tsToUpdate.Deadline = ts.Deadline;
            tsToUpdate.Budget = ts.Budget;

            await _firebaseClient
                .Child("ts")
                .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                .Child(uid)
                .PutAsync(tsToUpdate);

            return RedirectToAction("Index", "TS");
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 35 - Метод контроллера «TS» для редактирования технического задания

Из-за реализации пакета Firebase для ASP.NET, для перезаписи файла в Firebase Storage требуется сначала его удалить и только потом записать новый файл.

Рисунок 36 демонстрирует метод контроллера «TS» для удаления технического задания.

```
[HttpGet("[controller]/DeleteTS/{uid}")]
Ссылок: 0
public async Task<ActionResult> DeleteTS(string uid)
{
    var ts = await _firebaseClient
            .Child("ts")
            .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
            .Child(uid)
            .OnceSingleAsync<TSModelForDelete>();

    if (ts.DocumentExt != null)
    {
        var storageClient = _firebaseStorage.Child("ts").Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_")).Child(uid + ts.DocumentExt);
        var downloadUrl = await storageClient.GetDownloadUrlAsync();
        ts.DownloadUrl = downloadUrl.ToString();
        return View(ts);
    }
    else if (ts != null)
    {
        return View(ts);
    }
    else
    {
        return NotFound();
    }
}
```

```
[HttpPost("[controller]/DeleteTS/{uid}")]
Ссылок: 0
public async Task<ActionResult> DeleteTSPost(string uid)
{
    try
    {
        var tsToDelete = await _firebaseClient
            .Child("ts")
            .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
            .Child(uid)
            .OnceSingleAsync<TSModelForDatabase>();

        if (tsToDelete != null)
        {
            await _firebaseClient
                .Child("ts")
                .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                .Child(uid)
                .DeleteAsync();

            if (tsToDelete.DocumentExt != null) {
                string fileName = uid + tsToDelete.DocumentExt;

                await _firebaseStorage
                    .Child("ts")
                    .Child(HttpContext.User.FindFirstValue(ClaimTypes.Email).Replace(".", "_"))
                    .Child(fileName)
                    .DeleteAsync();
                }

            return RedirectToAction("Index", "TS");
        }
        else
        {
            return NotFound();
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Рисунок 36 - Метод контроллера «TS» для удаления технического задания

Таким образом, были представлены основные этапы разработки приложения.

3.3 Тестирование

Тестирование является заключительным этапом в разработке программного обеспечения до его выпуска в релиз. Этот процесс играет ключевую роль в обеспечении качества разрабатываемого программного продукта. Оно направлено на обнаружение ошибок, недочетов и проблем, которые могут возникнуть при работе программы.

Для проверки работоспособности методов контроллеров были созданы представления с помощью компонента ASP.NET, Razor Pages. «Razor Pages связывает одно представление с классом, который предоставляет ему функции, и использует файловую систему маршрутизации для сопоставления URL-адресов» [14].

Однако нет необходимости показывать все представления. Важно продемонстрировать функционал разграничения ролей пользователей и

функционал подачи заявок на разработку. Показывать работоспособность, стоит начать с главной страницы (рисунок 37).

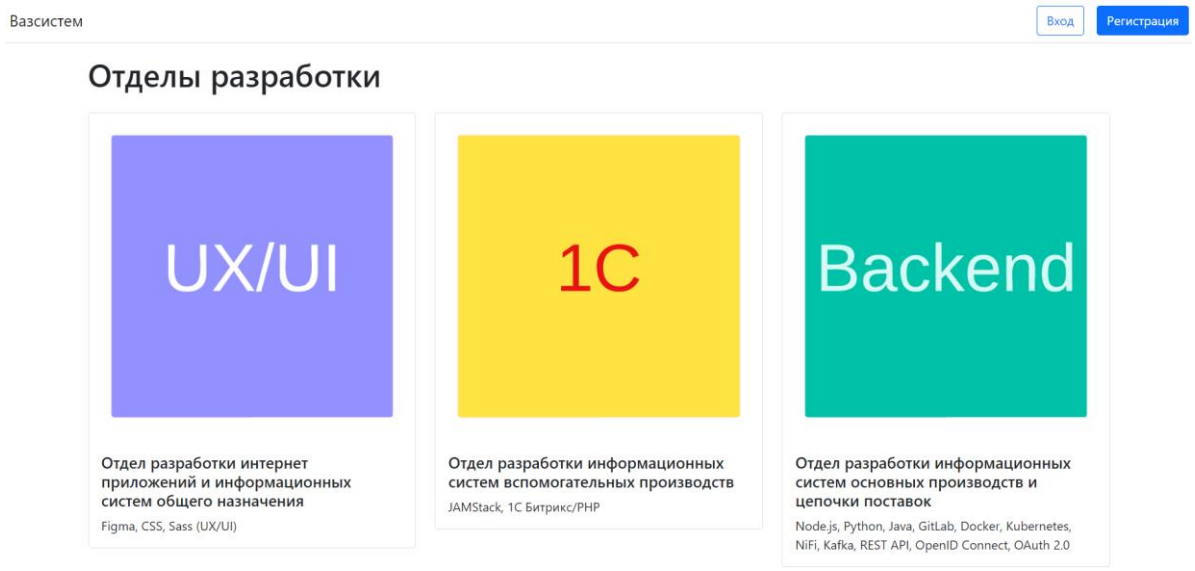


Рисунок 37 – Главная страница без входа в систему

На рисунке 37 мы можем видеть то, как выглядит главная страница без входа в систему. Именно на ней клиенты смогут выбирать нужный отдел разработки (рисунок 38).

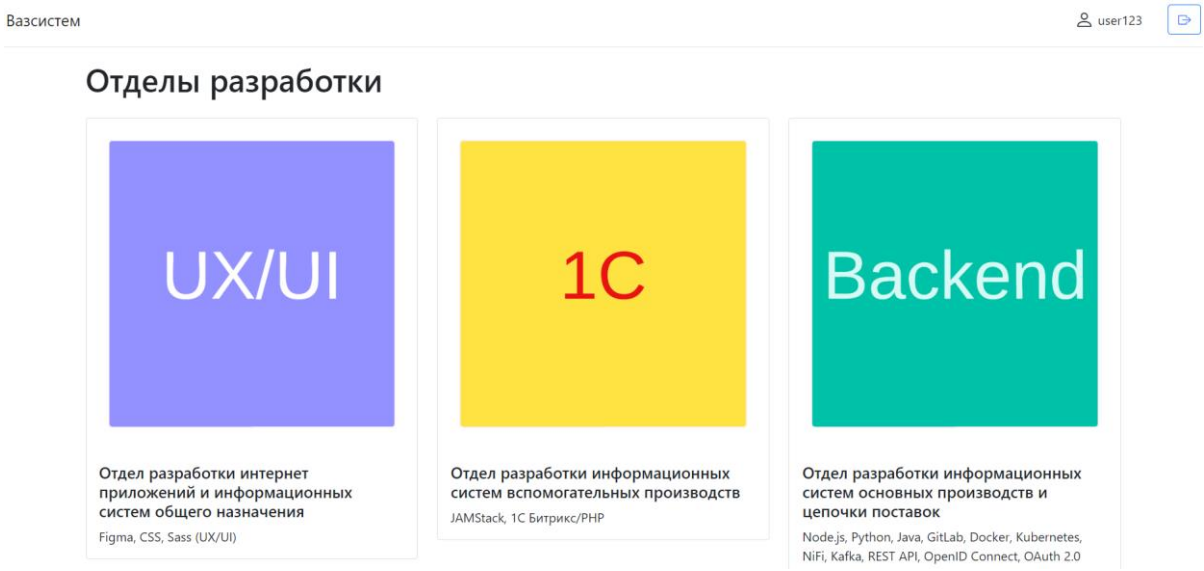


Рисунок 38 – Главная страница от лица заказчика

На рисунке 38 продемонстрирована главная страница после входа в систему под обычным пользователем. Здесь можно заметить замещение блока регистрации и авторизации на кнопку выхода и блок с иконкой пользователя и именем, которое заказчик указал при регистрации.

На рисунке 39 мы видим главную страницу после входа в систему под администратором.

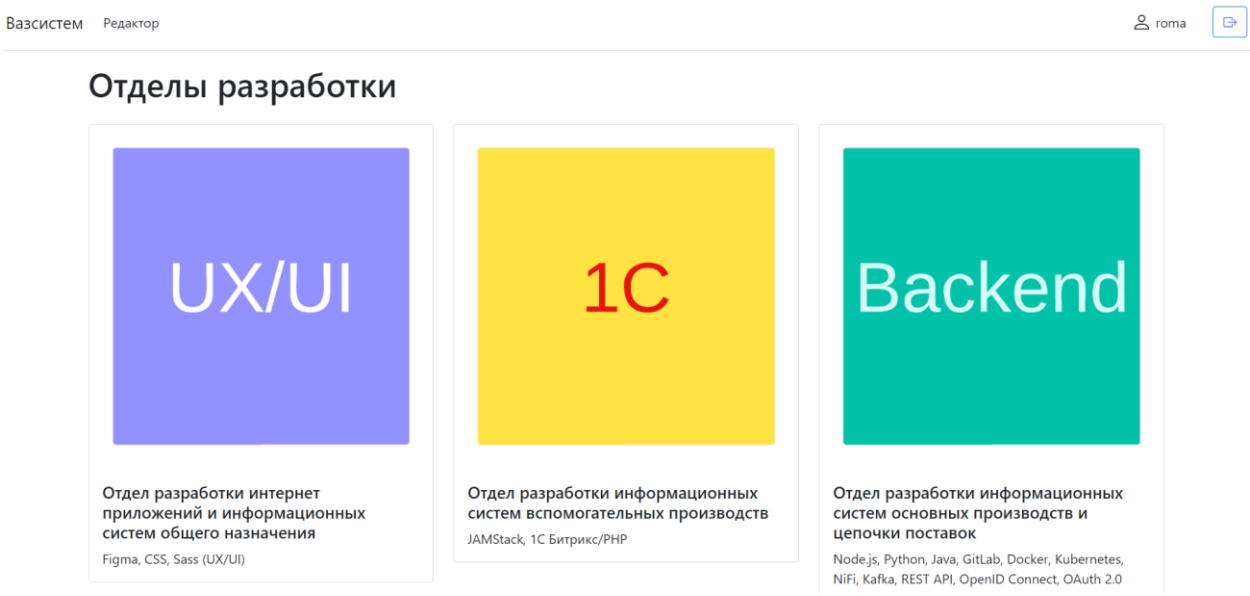


Рисунок 39 – Главная страница от лица администратора

Можно заметить, что администраторам доступна вкладка «Редактор» слева сверху.

Проверив работу разделения ролей пользователей, стоит показать страницы, связанные с регистрацией и авторизацией.

На рисунке 40 продемонстрирована страница для ввода данных при регистрации.

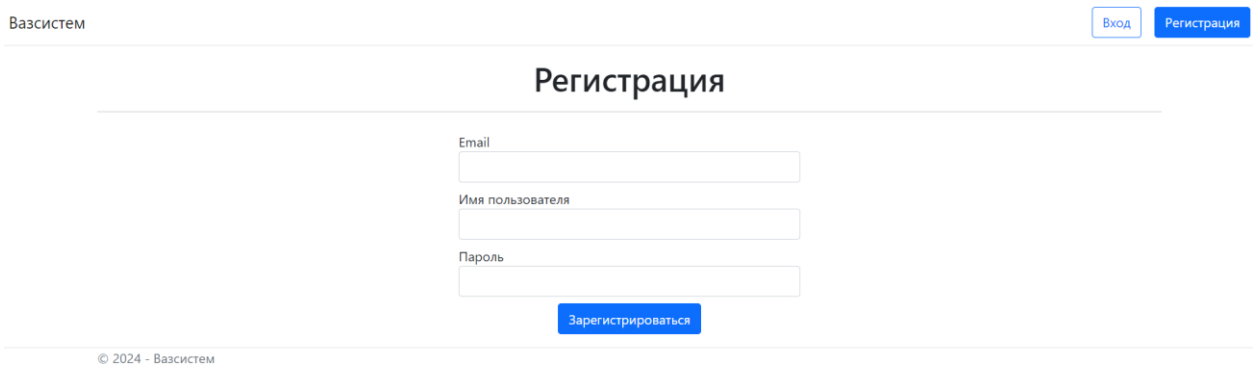


Рисунок 40 – Страница регистрации

На рисунке 41 показана страница для авторизации пользователей системы.

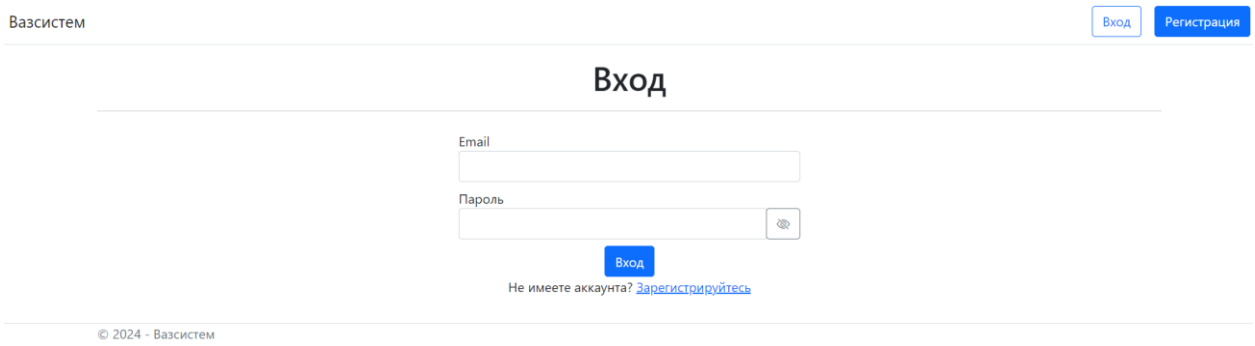


Рисунок 41 – Страница авторизации

Дальше стоит показать панель «Редактор», доступную администраторам, для добавления отделов разработки.

Рисунок 42 демонстрирует страницу «Редактор», которая доступна только пользователям с ролью «admin».

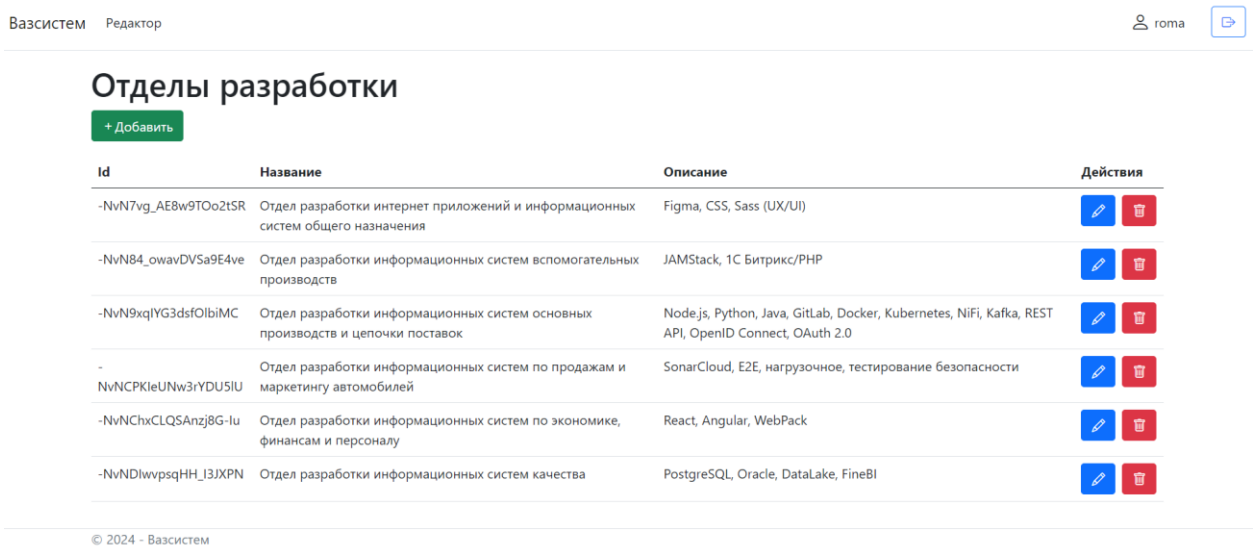


Рисунок 42 – Страница «Редактор»

Эта страница даёт возможность создавать, редактировать и удалять отделы разработки.

Следующим шагом будет показана страница для подачи технического задания (рисунок 43).

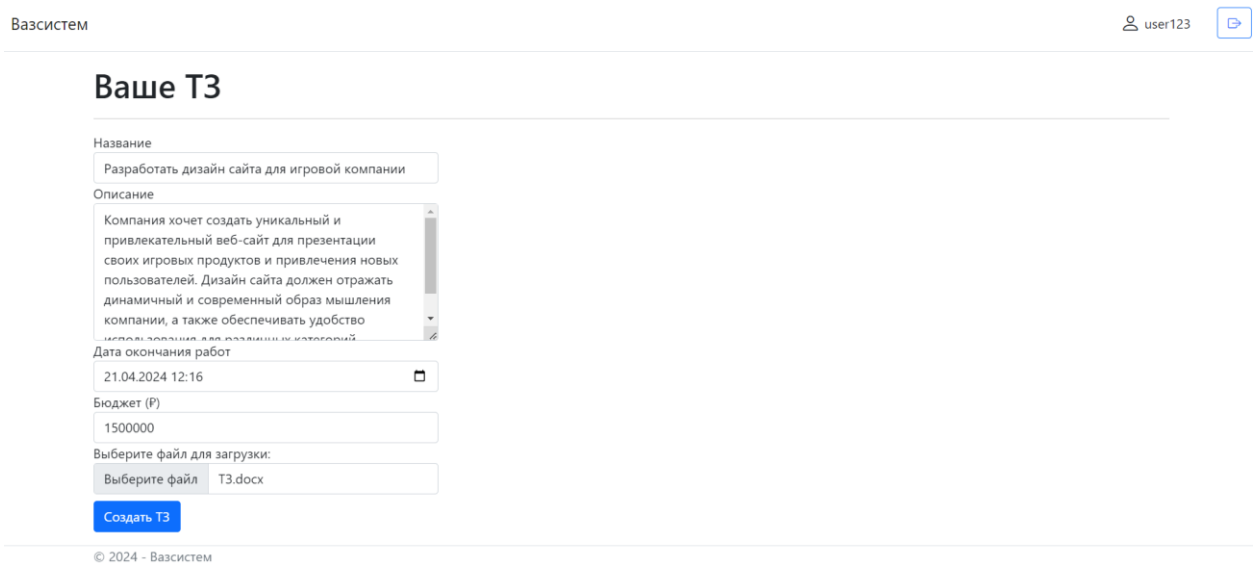


Рисунок 43 – Страница для подачи технических заданий

Рисунок 43 показывает страницу для создания технических заданий. Данная страница доступна только заказчикам при клике по одному из отделов разработки на главной странице.

Рисунок 44 иллюстрирует страницу с поданными техническими заданиями.

Рисунок 44 – Страница с поданными техническими заданиями

Данная страница доступна только заказчику и позволяет редактировать и удалять поданные технические задания. Для перехода на эту страницу заказчику необходимо нажать на блок с иконкой пользователя и его именем, указанным при регистрации.

Во всех представленных страницах есть стили. Для того чтобы страницы имели готовые стили в Razor Pages используется Bootstrap. «Bootstrap – это открытый и бесплатный HTML, CSS и JS фреймворк, с которым работает веб-разработчик для быстрой верстки адаптивных дизайнов сайтов и веб-приложений» [13].

Теперь можно провести функциональное тестирование (рисунок 45). Для этой цели будет использован Selenium.

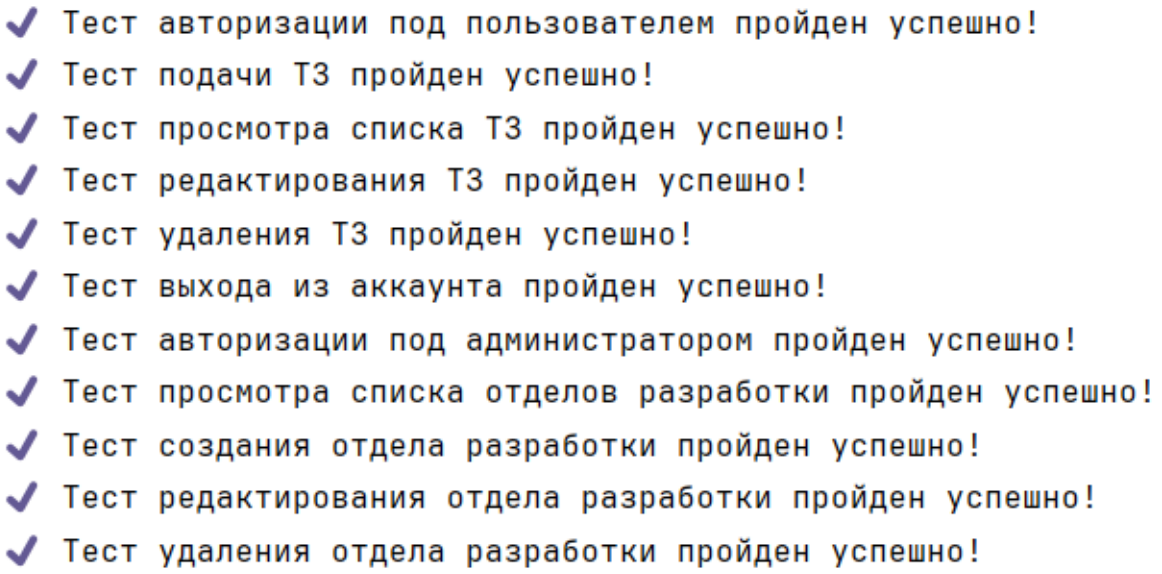


Рисунок 45 - Результат функционального тестирования

На рисунке 45 можно увидеть результат функционального тестирования. Результат показывает, что весь функционал веб-приложения работает исправно.

Выводы по главе 3

Третья глава посвящена тестированию. В этой главе были разработаны модели данных и бизнес логика приложения. Из проверки работоспособности веб-приложения можно выявить, что подача технического задания стала проще и удобней.

Заключение

Все компании постоянно совершенствуют свою продукцию. Это позволяет добиться расположения большего числа клиентов и поддерживать конкурентоспособность компании.

Данная работа была начата по причине сложного и неудобного процесса подачи технических заданий на разработку в АО «ВАЗСИСТЕМ».

В результате проделанной работы было разработано готовое решение для АО "ВАЗСИСТЕМ" – веб-приложение, обеспечивающее удобный и эффективный процесс составления и подачи технического задания на разработку программного обеспечения малым бизнесам. Это означает, что цель данной работы была выполнена.

В результате выполнения поставленных задач были достигнуты следующие результаты:

- описаны характеристики организации и её деятельности, что способствовало дальнейшему выбору подхода решения существующей проблемы;
- описан интересуемый бизнес-процесс. Данный результат позволил составить план рефакторинга;
- спроектировано техническое решение, что позволило определить необходимые функциональные возможности программного решения;
- разработана серверная часть веб-приложения, обеспечивающая автоматизацию процесса составления технического задания и подачи заказа на разработку. Данное решение позволит привлечь новых клиентов;
- проведено тестирование разработанного решения, результаты которого подтвердили его работоспособность и соответствие поставленным требованиям.

Результаты данного исследования имеют научную и практическую значимость. Научная значимость была реализована благодаря новому подходу

компании к будущим клиентам, желающим заказать разработку ПО. Практическая значимость проекта заключается в том, что был предоставлен доступный и эффективный инструмент для малых бизнесов, который существенно улучшает их взаимодействие с ИТ-сферой и позволяет подавать заказы на разработку ПО без затруднений.

В заключении следует отметить, что разработанное веб-приложение имеет потенциал для успешного внедрения и применения в коммерческой сфере компании АО «ВАЗСИСТЕМ». Оно позволяет сократить время и издержки на создание технического задания, а также улучшить взаимодействие между заказчиками и разработчиками. Результаты работы могут быть использованы как основа для дальнейших исследований в области улучшения взаимодействия заказчика и исполнителя. Опыт, полученный при выполнении данной работы, поможет мне в моей профессиональной карьере.

Список используемой литературы

1. Бобровников А. Э. Введение в управление проектами внедрения ERP-систем / А. Э. Бобровников. - 1С-Паблишинг, 2021 – 289 с.
2. Брежнев Р. В. Методы и средства проектирования информационных систем и технологий : учебное пособие / Р. В. Брежнев, 2021. - 216 с.
3. Вишняков О. Преимущество повторяемости. Практическое руководство по бизнес-процессам. Процессы и их описание. - СПб.: Питер, 2022. - 304 с.
4. Волкова В. Н. Теория систем и системный анализ : учебник для вузов / В. Н. Волкова, А. А. Денисов. - 3-с изд. - Москва : Издательство Юрайт, 2021. - 562 с.
5. Завертайлов В. В. Настольная книга project-менеджера. Что нужно знать, чтобы управлять IT, digital и другими проектами с учетом российских реалий / В. В. Завертайлов. – Бомбора, 2022. – 752 с.
6. Зараменских Е. П. Информационные системы: управление жизненным циклом : учебник и практикум для среднего профессионального образования / Е. П. Зараменских. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 431 с.
7. Зараменских Е. П. Управление жизненным циклом информационных систем : учебник и практикум для вузов / Е. П. Зараменских. - 2-е изд. - Москва : Издательство Юрайт, 2021. - 497 с.
8. Кирильчук С. П. Экономика предприятия : учебник для среднего профессионального образования / С. П. Кирильчук [и др.] ; под общей редакцией С. П. Кирильчук. - 2-е изд., перераб, и доп. - Москва : Издательство Юрайт, 2023. - 458 с.
9. Лисяк В. В. Разработка информационных систем : учебное пособие : [16+] / В. В. Лисяк ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2019. – 97 с.

10. Лосев К.Ю. Объектно-ориентированное инфографическое моделирование : учебно-методическое пособие / К.Ю. Лосев. — Москва : Издательство МИСИ – МГСУ, 2022. – 45 с.

11. Мельников П. П. Проектирование информационных систем : учебник и практикум для вузов / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук ; под общей редакцией Д. В. Чистова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. – 293 с.

12. Романова Ю. Д. Информационные технологии в менеджменте (управлении) : учебник и практикум для среднего профессионального образования / Ю. Д. Романова [и др.] ; под редакцией Ю. Д. Романовой. - 2-е изд., перераб. и доп. - Москва : Издательство Юрайт, 2020. - 411 с.

13. Эберт Елена. Шпаргалки для начинающего верстальщика HTML/CSS. - Издательские решения, 2021. - 148 с.

14. Adam Freeman. Pro ASP.NET Core 3 Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. – Apress, 2020. – 1080 p.

15. Adam Freeman. Pro ASP.NET Core 7, Tenth Edition. – Manning, 2023. - 1256 p.

16. Agnieszka Koziorowska, Wojciech Marusiak. Google Cloud Associate Cloud Engineer Certification and Implementation Guide. Master the Deployment, Management, and Monitoring of Google Cloud Solutions. - Packt Publishing, 2023. – 606 p.

17. Gopinath Karmakar, Amol Wakankar, Ashutosh Kabra, Paritosh Pandya. Development of Safety-Critical Systems. Architecture and Software, 2023. – 360 p.

18. Kemal Birer. ASP.NET Core for Jobseekers: Build Career in Designing Cross-Platform Web Applications Using Razor and Entity Framework Core. - BPB Publications, 2021. – 348 p.

19. Manuel A. Serrano, Mario Piattini, Ricardo Perez-Castillo. Quantum Software Engineering. - Springer International Publishing, 2022. – 302 p.

20. Patrick J. Driscoll, Gregory S. Parnell, Dale L. Henderson. Decision making in systems engineering and management. - John Wiley & Sons, 2023. — 576 p.

21. Rajesh Kumar R. Operations Management. - Jyothis Publishers, 2022. — 208 p.

22. Ravi Sethi. Software Engineering. Basic Principles and Best Practices. - Cambridge University Press, 2023. – 342 p.

23. Ricardo Peres. Modern Web Development with ASP.NET Core 3. - Packt Publishing, 2020. – 802 p.

24. Roel Grit. Project Management. A Practical Approach. - Taylor & Francis, 2022. – 226 p.