# Введение

Актуальность современных информационных технологий в повседневной жизни и бизнесе требует эффективных инструментов для планирования задач, организации рабочего процесса и совместной работы команд. В связи с этим, целью данного выпускного квалификационного проекта является разработка веб-приложения для планирования задач, которое обеспечит возможность создания заметок, задач и досок канбан, а также позволит эффективно работать в команде.

Объектом исследования данного проекта является процесс планирования задач, организации рабочего процесса и совместной работы в команде. В рамках исследования рассматриваются организации, где требуется координация деятельности нескольких сотрудников и управление проектами. Основной предмет исследования заключается в реализации веб-приложения, способного удовлетворять потребности пользователей в планировании задач, создании заметок, организации рабочего процесса и совместной работе команды.

Для достижения поставленной цели в рамках дипломного проектирования необходимо решить следующие задачи:

* изучить существующие методы и инструменты, которые предоставляют функционал планирования задач, создания заметок и организации рабочего процесса в командной среде;
* разработать веб-приложение, обеспечивающее создание и управление задачами, заметками и досками канбан, а также функциональностью совместной работы в команде;
* провести тестирование и анализ разработанного веб-приложения с целью оценки его функциональности, производительности и удобства использования;
* определить практическую значимость разработанного веб-приложения для организаций и команд, в которых важна эффективная организация рабочего процесса и совместная работа.

Для достижения поставленных задач будут использованы следующие методы исследования: изучение литературы, анализ существующих инструментов и методов, изучение аналогов, проектирование и разработка веб-приложения, тестирование приложения и сравнительный анализ результатов работы.

Реализация приложения будет осуществляться средствами языка программирования JavaScript, язык разметки документов HTML и язык табличных стилей CSS. В качестве основных инструментов разработки графической составляющей приложения была выбрана библиотека ReactJS, для серверной части в качестве инструмента разработки технологии api используется веб-фреймворк ExpressJS, в качестве системы управления базами данных была выбрана программа HeidiSQL, реализация логической части баз данных был выбран язык MySQL.

Разработка веб-приложения для планирования задач, создания заметок и организации рабочего процесса в командной среде является актуальной и практически значимой задачей, способной повысить эффективность работы организаций и команд. Основываясь на предмете исследования и поставленных целях, и задачах, данная работа имеет важное значение для развития современных информационных технологий и улучшения рабочих процессов в организациях и командах.

# Общая (теоретическая) часть

# Постановка задачи

1.1.1 Обоснование необходимости разработки

Основная задача разрабатываемого веб-приложения заключается в автоматизации процесса планирования задач, создания заметок и управления проектами, с фокусом на совместной работе в команде. Приложение предоставляет пользователям удобный и эффективный инструмент для организации своей работы и повышения производительности.

Одной из ключевых функциональных возможностей приложения является создание и редактирование заметок. Пользователи смогут создавать заметки по необходимости, а также вносить изменения в уже существующие. Это позволит им систематизировать информацию, запоминать важные детали и легко находить нужные записи в дальнейшем.

Кроме того, приложение предлагает возможность создания задач и их последующего редактирования. Пользователи смогут определять цели, устанавливать приоритеты и контролировать ход выполнения задач. Это поможет им планировать свою работу, соблюдать сроки и достигать поставленных целей.

Для удобного организационного управления, в приложении предусмотрено создание проектов и формирование команд. Пользователи смогут объединять связанные задачи и заметки в рамках конкретных проектов, а также приглашать других участников команды для совместной работы. Это способствует улучшению коммуникации и координации усилий между участниками проекта.

Важно отметить, что в разрабатываемом приложении не предусматривается разделение пользователей на различные роли, такие как администраторы и обычные пользователи. Это решение связано с учетом конфиденциальности данных пользователей, где доступ к ним должен быть ограничен. Приложение ориентировано на обеспечение безопасности и приватности пользователей. В связи с этим, выделяются два типа пользователей: незарегистрированный пользователь, который имеет доступ только к общей информации приложения, и зарегистрированный пользователь, который получает полный контроль над своими данными и доступ к личному кабинету.

Разрабатываемое веб-приложение представляет собой эффективный инструмент для планирования задач, создания заметок и организации работы в команде. Оно обладает высокой функциональностью, интуитивно понятным интерфейсом и акцентом на безопасность и приватность пользователей.

1.1.2 Технико-математическое описание задачи

Для разработки веб-приложения был выбран язык программирования JavaScript, который представляет собой высокоуровневый динамический язык программирования, широко применяемый в веб-разработке.

Для дополнительной разработки веб-приложения были использованы язык разметки HTML (HyperText Markup Language) и язык табличных стилей CSS (Cascading Style Sheets). HTML предоставляет структуру и содержимое веб-страницы, в то время как CSS определяет внешний вид и стилизацию элементов на странице. Для удобства работы с CSS были также использованы препроцессоры SASS (Syntactically Awesome Style Sheets) и SCSS (Sassy CSS), которые предоставляют дополнительные возможности и инструменты для более эффективного написания стилей.

Для разработки графической составляющей приложения была выбрана библиотека ReactJS. ReactJS - это мощная и популярная графическая JavaScript-библиотека для создания пользовательских интерфейсов. Она позволяет разрабатывать компоненты, которые могут быть повторно использованы, и обеспечивает эффективное обновление только изменяющихся частей интерфейса приложения. Это улучшает производительность и повышает отзывчивость веб-приложения. Вместе с библиотекой ReactJS была использована библиотека MobX. MobX - это библиотека для управления состоянием приложения. Она позволяет удобно и эффективно организовывать управление данными и реактивно отслеживать изменения состояния, обновляя соответствующие компоненты при необходимости. MobX упрощает процесс разработки, делая код более читаемым и поддерживаемым.

Для разработки информационной части веб-приложения был выбран язык MySQL, данный язык имеет огромную пользовательскую поддержку, а также является одним из самых производительных и отказоустойчивых языков. Для разработки баз данных в качестве системы управления базами данных была выбрана HeidiSQL, из-за открытого исходного кода и огромного функционала.

В качестве основной среды выполнения служит NodeJS версии 18.16. Для реализации серверной части и технологии api использовался веб-фреймворк ExpressJS, данный фреймворк позволяет быстро реализовать и настроить разрабатываемый сервер. Для получения данных с базы данных использовалась библиотека mysql2. Конфиденциальные данные пользователей хранятся в захешированном виде, используемый алгоритм хеширования SHA256, а библиотека для шифрования crypto.

1.1.3 Характеристика бизнес-процессов

Основная задача приложения заключается в автоматизации планирования задач, создании заметок и совместной работы над проектами в командной среде. Приложение стремится эффективно выполнять следующие функции: создание и редактирование заметок, создание и управление задачами, а также создание и управление проектами и соответствующими командами.

Важно отметить, что при разработке данного веб-приложения не было предусмотрено разделение ролей, включая отсутствие административной роли. Это решение было принято в связи с конфиденциальной природой пользовательских данных, и, следовательно, отсутствием необходимости предоставления доступа администратору к этим данным. Вместо этого, в приложении выделены два основных типа пользователей.

Незарегистрированные пользователи имеют доступ только к общей информации приложения, что позволяет им ознакомиться с его функциональностью. Зарегистрированные пользователи, с другой стороны, получают расширенные возможности, включая доступ к личному кабинету. Личный кабинет предоставляет зарегистрированным пользователям полный контроль над их данными, позволяя им эффективно управлять своими задачами, заметками и проектами.

Таким образом, данное приложение стремится обеспечить пользователей инструментами для удобного планирования и управления задачами, создания и редактирования заметок, а также совместной работы в команде. Оно предлагает простой и безопасный доступ к функционалу приложения, гарантируя конфиденциальность пользовательских данных и обеспечивая удобство использования для каждого типа пользователя.

1.1.4 Требования программы

Минимальные требования к программе для пользователей:

* версия браузера 7.0 или выше;
* оперативная память: минимум 2 гигабайта;
* постоянная память: минимум 256 мегабайт;
* процессор: Intel Atom 3-серии или аналогичный.

Дополнительные требования:

* для обеспечения плавной работы приложения рекомендуется более высокий объем оперативной памяти и постоянной памяти;
* рекомендуется использовать актуальную версию браузера для обеспечения полной поддержки функциональности приложения.

Минимальные системные требования для сервера:

* процессор: Intel Xeon 4300 с частотой 3.2 Гц или аналогичный;
* оперативная память: минимум 2 гигабайта;
* постоянная память: объем не менее 20 гигабайт.

Дополнительные требования:

* рекомендуется использовать более производительное оборудование, если предполагается большая нагрузка или увеличение числа пользователей;
* операционная система должна поддерживать запуск NodeJS, необходимого для работы приложения;
* рекомендуется использовать надежное сетевое соединение для обеспечения стабильной работы сервера и обмена данных с клиентами.

# Обоснование проектных решений

1.2.1 Обоснование выбора языков программирования

При разработке веб-приложения мы выбрали ReactJS - графическую библиотеку, основанную на JavaScript и TypeScript, для упрощения разработки и обеспечения масштабируемости. ReactJS предоставляет компонентный подход, позволяющий разделить приложение на отдельные компоненты, которые могут быть многократно использованы. Библиотека имеет широкую поддержку как со стороны разработчиков, так и от сообщества.

JavaScript был выбран в качестве основного языка программирования для разработки. Мы выбрали его из-за его популярности и уникальности. JavaScript является основным языком для создания интерфейсов во всех браузерах и обладает высокой скоростью работы по сравнению с конкурирующими языками, такими как Python, Ruby и PHP. JavaScript также обладает быстрой скоростью разработки благодаря своей динамичности, что позволяет избежать указания типов переменных и сэкономить время на разработке механизмов взаимодействия. Хотя JavaScript поддерживает ООП, следует отметить, что его ООП-возможности не являются полноценными по сравнению с языками, такими как C и Java.

JavaScript соответствует всем требованиям, которые мы установили при архитектурном проектировании. Он обеспечивает высокую скорость работы, быструю разработку и низкое потребление ресурсов компьютера. Кроме того, JavaScript широко распространен и имеет активную поддержку от сообщества разработчиков.

ExpressJS был выбран в качестве веб-фреймворка, из-за производительности и быстроты разработки API, а также огромное количество библиотек, написанных для данного фреймворка. ExpressJS также имеет методы защиты.

Для создания базы данных мы используем MySQL - основной язык баз данных. MySQL отличается высокой скоростью работы и подходит для обработки больших объемов данных. Он также обладает надежностью, отказоустойчивостью и имеет обширную поддержку в сообществе разработчиков.

1.2.2 Инструментальные средства

Для разработки логической части клиентского приложения в рамках исследования был выбран язык программирования JavaScript. Этот язык является широко применяемым и позволяет разработчикам создавать динамические и интерактивные веб-приложения. Он обладает мощными инструментами и богатым экосистемой, что обеспечивает гибкость и эффективность процесса разработки.

Для создания графической составляющей была разработана графическая библиотека ReactJS, которая также использует JavaScript в своей основе. ReactJS позволяет разработчикам создавать модульные компоненты пользовательского интерфейса, что способствует повышению переиспользуемости кода и облегчает масштабирование приложения. Библиотека имеет широкую поддержку со стороны сообщества разработчиков и предоставляет множество инструментов и расширений для ускорения процесса разработки. Удобство контроля состояниями осуществляется с помощью библиотеки MobX.

В рамках разработки серверной части приложения также был выбран язык JavaScript, используемый на движке NodeJS. Для удобства и эффективности разработки, был выбран веб-фреймворк ExpressJS. Он предоставляет удобные инструменты и возможности для создания REST-API, что обеспечивает взаимодействие между серверной частью и клиентским приложением.

Для обеспечения взаимодействия с базой данных была использована библиотека mysql2. Это надежное и эффективное решение, которое позволяет осуществлять операции с базой данных MySQL с высокой производительностью и надежностью. Сама же база данных разработана с помощью MySQL.

Выбор данных инструментальных средств обусловлен их популярностью, широкой поддержкой в сообществе разработчиков, а также их способностью ускорить процесс разработки и обеспечить высокую гибкость и эффективность в работе с клиентской и серверной частями приложения.

1.2.3 Обоснование выбора среды программирования

В качестве среды программирования для разработки был выбран Visual Studio Code. Она предлагает множество преимуществ, которые подтверждают выбор данной среды.

Во-первых, Visual Studio Code обладает широким набором функций, которые значительно облегчают процесс разработки. Встроенное автодополнение кода, интеграция с отладчиками и системами контроля версий позволяют повысить эффективность работы и ускорить процесс разработки.

Во-вторых, данная среда разработки обладает простым и интуитивно понятным интерфейсом. Это позволяет разработчикам быстро освоиться и начать работу без длительного обучения.

В-третьих, Visual Studio Code является легкой и быстрой средой, что позволяет сократить время отклика и повысить общую производительность работы. Она также предлагает возможность расширения функциональности с помощью различных плагинов, что позволяет настроить среду под индивидуальные потребности разработчика.

Кроме того, Visual Studio Code активно поддерживается сообществом разработчиков, что гарантирует обновления и исправление ошибок. Большое количество доступных ресурсов и документации делает процесс разработки более удобным и эффективным.

В качестве дополнительных расширений использовались ConsolasNinja позволяющей выводить информацию о переменных прямо в редактор. AutorenameTag расширения, которые добавляет автоматическое переименования парных html-тегов. Import Cost данное расширение позволяет видеть вес импортируемых компонентов, и DrawIO Integration, которая дополняет функционал среды разработки, позволяя создавать диаграммы и открывать их прямо в редакторе. Данные дополнения позволяют приблизить Visual Studio Code, к полноценной IDE.

В итоге, выбор Visual Studio Code в качестве среды программирования полностью удовлетворяет требованиям проекта, обеспечивая надежную и эффективную среду для разработки на JavaScript, HTML и CSS.

1.2.4 Информационное обеспечение

Для обеспечения разработки программного проекта было использовано разнообразное информационное обеспечение.

Для создания диаграмм приложения, от диаграммы данных до алгоритмов веб-приложения использовался сервис DrawIO, он позволяет создать диаграмму любой сложности и экспортировать ее в любой формат.

Для создания логотипа, иконок и прототипирования интерфейса веб-приложения был выбран веб-сервис Figma. Figma предоставляет удобные инструменты для разработки графических элементов приложения, а также позволяет создавать прототипы интерфейса для дальнейшей визуализации и тестирования.

Для работы с документами, такими как таблицы, текстовые файлы и презентации, было использовано приложение LibreOffice. Это бесплатное программное обеспечение предоставляет набор инструментов для создания и редактирования документов различных типов, обеспечивая функциональность, схожую с платными аналогами, такими как Microsoft Office.

Для создания и управления презентациями в процессе защиты проекта было использовано приложение Microsoft PowerPoint. PowerPoint обладает широким спектром возможностей для создания эффектных слайдов, добавления анимации и управления мультимедийным контентом, что способствует наглядному представлению информации перед аудиторией.

Для контроля версий и управления кодовой базой проекта была использована система контроля версий Git, а также платформа GitHub для хранения и совместной работы над репозиторием проекта. Git позволяет отслеживать изменения в коде, создавать ветки для разработки новых функций и объединять изменения от разных разработчиков, обеспечивая целостность и управляемость проекта. GitHub предоставляет удобный интерфейс и инструменты для хостинга репозиториев, совместной разработки и обмена кодом с другими разработчиками.

# Обзор и анализ существующих программных систем

На данный момент в мире насчитывает огромное количество сервисов, которые позволяют планировать свои задачи и работать в команде над своими проектами. Обычно каждая крупная компания разрабатывает подобное приложение для внутреннего пользования, после чего проект выходит под коммерческим началом.

Ниже представлены описание аналогов. Аналоги будут разделены по категории производителя, либо западные компании, либо российские компании.

Западные аналоги приложения.

Notion (Notion Labs) – крупный многофункциональный сервис, для работы с markdown файлами. На рынке с 2011 года, разрабатываемый в США. Достоинства данного приложения, огромная многофункциональность, позволяет создавать огромное множество объектов, в зависимости от требований пользователя, начинаю от планировщика задач, заканчивая полноценными базами данных. Notion отличается простотой дизайна и производительностью, не смотря на огромный функционал, также Notion полностью перенес на все операционные системы, а именно Windows, Linux, MacOS, Android. Недостатки, главным недостатком сервиса, является расположение физический серверов, а именно это страны, США, Великобритания, Франция. То есть данные пользователей могут быть доступны третьим лицам. Сервис, активно блокирует российские корпоративные аккаунты и аккаунты обычных пользователей без возвращения средств, и с полной утратой данных пользователей. Сервис также активно замедлил работу в России. Интерфейс приложения несмотря, на свою простоту, скрывает огромное количество функций приложения, что не позволяет понять весь функционал нового пользователя, на сайте также нет обучения, этим функциям.

Asana ( Asana, Inc ) – американский аналог приложения, представляет собой многофункциональный сервис для планирования задач, был выпущен с 2008 года, по сей день. Достоинства, Asana представляет собой многофункциональность сервиса, конечно она куда меньше, чем у Notion, но сервис предоставляет очень похожий функционал, поддерживает многоязычность, бесплатная подписка пользователей, предоставляет почти половину всех функций сервиса. Недостатки, как ранее отмечалось, сервера с данными пользователей находятся в странах, которые могут использовать их в корыстных целях. Очень высокие региональные цены, и невозможность приобретения продукта, для граждан Российской Федерации, невозможно оплатить подписки, а также были волны удаления данных компаний, без предупреждения с полной утратой данных. Также по функционалу сайта стоит отметить, что Asana значительно проигрывает в производительности Notion, на сайте с большим количество элементов, заметно снижение производительности, а также по интерфейсу, в данном сервисе, интерфейс значительно перегружен, но не скрывает функции, в отличии от Notion.

Trello (Atlassian) – австрийская аналог веб-приложения выпущенный в 2011 году, представляет собой сервис для управления проектами небольших групп, основной функционал это работа с канбан досками. Достоинства, сайт является одним из самых популярных и простых по сравнению с другими аналогами, сервис прекрасно работает на любом устройстве, так как специализация сайта планирование методами канбан, то данный аспект, проработан до самых мельчайших деталей. Недостатки, с момента запуска проекта дизайн сайта перестал соответствовать современным требованиям, интерфейс перегружен, иногда глазу пользователя сложно зацепиться за нужный объект, сайт официально заявил от ухода с российского рынка.

Российские аналоги.

Бипиум – многофункциональное приложения для планирования бизнес-процессов, и задач. Достоинства, крупный сервис по решению проблемы автоматизации бизнес-процессов, предоставляет пользователю возможность формировать графические отчеты, документацию, и многое другое, имеет мобильное приложение, позволяет работать в команде, данные пользователей хранятся на территории страны, сервис имеет свое сервера, и предоставляет доступ к облачному хранилищу до 1 терабайта, хранит резервные копии до 7 дней, и имеет низкие цены за услуги. Недостатки, несмотря на все преимущества, его нельзя назвать полноценным аналогом, во-первых, данное приложение, скорее узконаправленное на средний и малый бизнес, с целью автоматизации документооборота и отчетности, а не для планирования задач пользователя, приложение не может похвастаться скоростью работы, и простотой освоение данного приложения.

WEEEK – сервис для командной работы над проектами. Инструменты — таск-менеджер, база знаний и CRM. Достоинства, отличный дизайн и функционал сайта, тарифные планы имеют очень низкую цену, и взамен предоставляет пользователю огромное количество различных функций. Предоставляет облачное хранилище. Кроссплатформенность, сервис доступен на любых устройствах. Сервис выполнен в современном дизайне, из-за чего не является перегруженным и понятным пользователю, сервис предоставляет возможность кастомизации интерфейса. Недостатки, по сравнению с другими сервисами, данный сервис не имеет большой популярности, из-за чего приложение не имеет поддержки сообщества, облачное хранилище, хоть и предоставляет пользователю место на хранение, но по сравнению, с другими аналогами, хранилище куда меньше. Сервис, также не может похвастаться стабильной работой, на мобильных устройствах сервис вылетает, на стационарных системах, также имеет проблемы с производительностью.

Битриск24 - огромный сервис, для автоматизации бизнес процессов, имеет в своем функционале, предоставление корпоративной системы, который упрощает коммуникации внутри компании, предоставляет систему CRM – это система управления продажами и коммуникациями с клиентами. Достоинства, на одну лицензию неограниченное количество пользователей, приложение предоставляет, сбор и хранение неограниченного количество данных внутри компании. Сервис имеет огромную поддержку продукта, со стороны разработчиков, в сервисе имеется обратная связь с технической поддержкой. Имеет удобную систему клиент-сотрудник, также сервис предоставляет возможность предоставлять полноценные отчеты о деятельности пользователей. Недостатки, главный недостаток по сравнению с другими аналогами, в первую очередь, это не полноценное приложение для работы с задачами и проектами, это огромная автоматизированная система, для управления бизнес-процессами. Во вторую очередь, Битрикс24 имеет огромную стоимость лицензии, для пользователей, по сравнению с другими аналогами. Второй главный недостаток, проблемы с производительностью, за счет размеров сервиса, производительность находиться на низком уровне и очень часто возникают различные проблемы, с откликом сервиса. Интерфейс, интерфейс данного сервиса, крайне перегружен и сложен, для начала работы пользователя ему необходимо пройти полноценное обучения взаимодействия с продуктом.

По итогу анализа аналогов, можно отметить следующие, достоинства, которые необходимо реализовать в разрабатываемом веб-приложения, простой дизайн, приложение должно не быть перегруженным в функциональном плане, приложение должно эффективно использовать ресурсы вычислительных машин, на которых запускаются.

# 2 Специальная часть

# 2.1 Анализ задачи

2.1.1 Информационное моделирование предметной области

Для начала моделирования предметной области, необходимо выделить основные сущности предметной области. Предметная область содержит следующие сущности:

* пользователи – пользователи приложения;
* тарифный план – тарифный план, который есть у пользователя.
* команды – группа пользователей, имеющий доступ к общим проектами;
* заметки – заметки пользователей;
* задача – задача пользователей;
* подзадача – элемент, который имеет связь с задачей;
* идея – идея пользователя;
* проект – проект команды или пользователя;
* пункт проекта – связан с проектом;
* задача проекта – связана с пунктом проекта.

Ознакомиться с реляционной моделью базы данных можно на рисунке А.1 Описание бизнес процессов графически представлено на рисунке А.2.

2.1.2 Проектирование пользовательского интерфейса

Перед началом разработки интерфейса, сначала необходимо выделить основные аспекты, на которых будет произведен акцент. В первую очередь, это выбор основной цветовой палитры, для этого были выбраны четыре цвета, цвет для темного акцента - # 211621, светлый основной цвет приложения - # EFF4EF, цвет для акцента - # BA3B50, и дополнительный цвет # F4830F, выбор основного шрифта пал, на шрифты Golos и его разновидность Golos UI. Вторым аспектом будет возможность работы в приложении на устройствах с разными разрешениями экрана, то есть дизайн сайта должен отвечать адаптивности под различные устройства. Третий акцент, для коммерческого использования программы необходимо был либо приобрести лицензию на различные иконки, либо разработать самостоятельно, в данном варианте было выбрано решение разработать иконки для приложения.

В приложении разработаны следующие страницы:

* страница ознакомления с продуктом;
* страница регистрации пользователя;
* страница авторизации пользователя;
* страница кабинета пользователя;
* страница вывода недоступного адреса.

Разработаны следующие основные компоненты:

* заметка – компонент позволяющий взаимодействовать с заметками;
* задача – компонент позволяющие компонент взаимодействовать с задачами;
* команда – компонент взаимодействия с командами;
* проект – компонент взаимодействия с проектами;
* блок с контентом – компонент для работы с данными из базы данных.

Дизайн приложения представлен на рисунке А.3, разработанные иконки и логотип приложения представлен на рисунке А.4.

# 2.2 Описание логической структуры

Основной целью разрабатываемого приложения является автоматизация и хранения данных пользователя, следовательно, основными алгоритмами в приложении, являются:

* алгоритм взаимодействия между пользователем и сервером;
* алгоритм взаимодействия пользователя и интерфейса;
* алгоритм вывода данных;
* алгоритм авторизации и регистрации пользователя;
* алгоритм обработки данных на сервере.

Алгоритм взаимодействия между пользователем и сервером предполагает отправления запроса API приложения, и ожидаемого получения ответа, данный алгоритм изображен на рисунке А.5.

Алгоритм взаимодействия пользователя и интерфейса, отвечает за отклик приложения на различные действия пользователя, например, отправка данных на сервер, после нажатия кнопки, или добавления подпункта в задачу. В приложении множество методов, которые позволяют взаимодействовать с данными, но на рисунке А.6 изображен самый используемый шаблон.

Алгоритм вывода данных, отвечает за отображения данных пользователя на странице, в случае корректно получения данных от результата запроса пользователя, необходимо корректно вывести данную информацию, данный алгоритм изображён на рисунке А.6

Алгоритм валидирования данных предполагает, что пользователь не может оставлять обязательные поля пустыми, или вводить не корректные данные в поля, например, поле ввода почвы, останется пустым или пароль будет меньше 6-ти символов. Данный алгоритм изображен на рисунке А.7

Алгоритм авторизации и регистрации пользователя соответствует, алгоритму поиска учетной записи, в случае если запись найдена, и пользователь хочет авторизоваться, то его данные сохраняются в глобальное хранилище, или же если пользователь хочет зарегистрироваться, то происходит добавления учетной записи в базу данных, если ее логин уникален, алгоритм изображен на рисунке А.8.

Алгоритм обработки данных на сервере, данный алгоритм отвечает за проверку корректных данных на сервер, для избегания возникновения ошибок со стороны базы данных. Алгоритм, работы изображен на рисунке А.9.

2.2.2 Используемые методы

Использование JWT-токена для защиты данных из базы данных: JWT (JSON Web Token) - это компактный и самодостаточный формат токена, который может быть использован для аутентификации и авторизации. При использовании JWT-токенов для защиты данных из базы данных, сервер может выдавать токены пользователям после успешной аутентификации. Эти токены содержат информацию о пользователе и могут быть использованы для доступа к определенным данным в базе данных. При получении запроса с JWT-токеном, сервер проверяет подлинность и целостность токена, а затем разрешает доступ к соответствующим данным в базе данных. Таким образом, использование JWT-токенов обеспечивает безопасность данных и контроль доступа.

Хеширование данных пользователей. Хеширование данных пользователей представляет собой процесс преобразования конфиденциальной информации, такой как пароли, в неповторимую строку символов с использованием хэш-функции. Хэш-функция берет входные данные и генерирует фиксированную длину строки, которая служит "отпечатком" входных данных. Хэш-функции являются односторонними, что означает, что исходные данные невозможно восстановить из хэш-значения. При хранении данных пользователей в базе данных, их пароли должны быть хешированы, чтобы предотвратить возможность раскрытия паролей в случае несанкционированного доступа к базе данных.

Использование хранимых процедур для получения данных из базы: Хранимые процедуры - это предварительно скомпилированные программы, которые хранятся и выполняются на сервере базы данных. Они позволяют обеспечить безопасность данных путем ограничения прямого доступа к базе данных и предоставления контролируемого интерфейса для взаимодействия с данными. Путем использования хранимых процедур для получения данных из базы данных, можно контролировать, какие данные доступны для выборки и какие операции могут быть выполнены. Это помогает предотвратить SQL-инъекции и другие атаки на базу данных.

Реализация защиты базы данных с помощью ограничения доступа пользователей. Ограничение доступа пользователей - это процесс определения и регулирования прав доступа пользователей к базе данных. Это может включать определение различных уровней доступа, таких как чтение, запись и изменение данных, а также назначение различных ролей и привилегий пользователям. При реализации защиты базы данных с помощью ограничения доступа пользователей, необходимо убедиться, что каждый пользователь имеет только те права, которые необходимы для выполнения своих задач, и что никакие нежелательные или неавторизованные действия не могут быть выполнены.

Ограничение подключения к внутреннему API методами хостинга. Ограничение подключения к внутреннему API методами хостинга предполагает ограничение доступа к API-методам, только определенным доверенным и авторизованным источникам или хостам. Это может быть достигнуто путем настройки правил брандмауэра или использования функций хостинга, таких как API-ключи или сетевые ограничения. Это помогает предотвратить несанкционированный доступ к внутренним API и защитить данные, которые могут быть переданы через эти методы.

Валидация данных пользователя. Валидация данных пользователя - это процесс проверки и проверки данных, введенных пользователем, на соответствие определенным правилам и форматам. При обработке и хранении данных пользователей важно убедиться, что они соответствуют ожидаемым форматам и не содержат вредоносного или некорректного содержимого. Валидация может включать проверку форматов электронной почты, паролей, длины полей и других требований, а также фильтрацию вводимых данных для предотвращения возможных атак, таких как внедрение кода или скриптов.

Все эти методы служат для обеспечения безопасности данных и предотвращения различных уязвимостей или атак на систему или базу данных. Каждый из этих подходов играет свою роль в обеспечении целостности и конфиденциальности данных, и их комбинированное использование обычно является наилучшей практикой для обеспечения безопасности данных.

Использование библиотеки MobX, как глобального хранилища состояний для ReactJS, MobX - библиотека для управления состоянием в ReactJS. Она упрощает управление состоянием приложения и предоставляет следующие преимущества. Простое управление состоянием позволяет объявить данные как наблюдаемые, и она автоматически отслеживает изменения этих данных, обновляя зависимые компоненты. При изменении данных MobX автоматически обновляет компоненты, которые зависят от этих данных, повышая производительность. MobX можно использовать с другими библиотеками и паттернами управления состоянием, а также легко масштабировать состояние приложения.

Для взаимодействия клиентской и серверной частей приложения был разработан API, средствами веб-фреймворка ExpressJS.

2.2.3 Составные части программы и связи между ними

Принцип взаимодействия компонентов в программе основан на реактивной архитектуре ReactJS, что подразумевает, что компоненты обмениваются сообщениями для взаимодействия. Эта архитектура предусматривает четкую иерархию компонентов, начиная с родительского и до дочерних компонентов, а также использование библиотеки MobX в качестве глобального хранилища состояний.

В реактивной архитектуре ReactJS компоненты взаимодействуют путем передачи данных и событий между собой. Родительские компоненты могут передавать данные и обратные вызовы своим дочерним компонентам через пропсы. Это позволяет обновлять состояние компонентов, передавать им новые данные и реагировать на события.

Использование библиотеки MobX в этой архитектуре обеспечивает удобное управление состоянием приложения. MobX предоставляет глобальное хранилище состояний, которое доступно для всех компонентов. Компоненты могут получать доступ к состоянию, используя MobX, и реагировать на его изменения автоматически. Это позволяет упростить управление состоянием и обновление компонентов, так как они автоматически обновляются при изменении состояния.

Таким образом, реактивная архитектура в ReactJS предлагает эффективное взаимодействие компонентов с помощью обмена сообщениями, основанное на иерархии компонентов. Использование библиотеки MobX дополняет эту архитектуру, обеспечивая глобальное хранилище состояний для удобного управления данными в приложении. Взаимодействия компонентов представлено на рисунке 1.

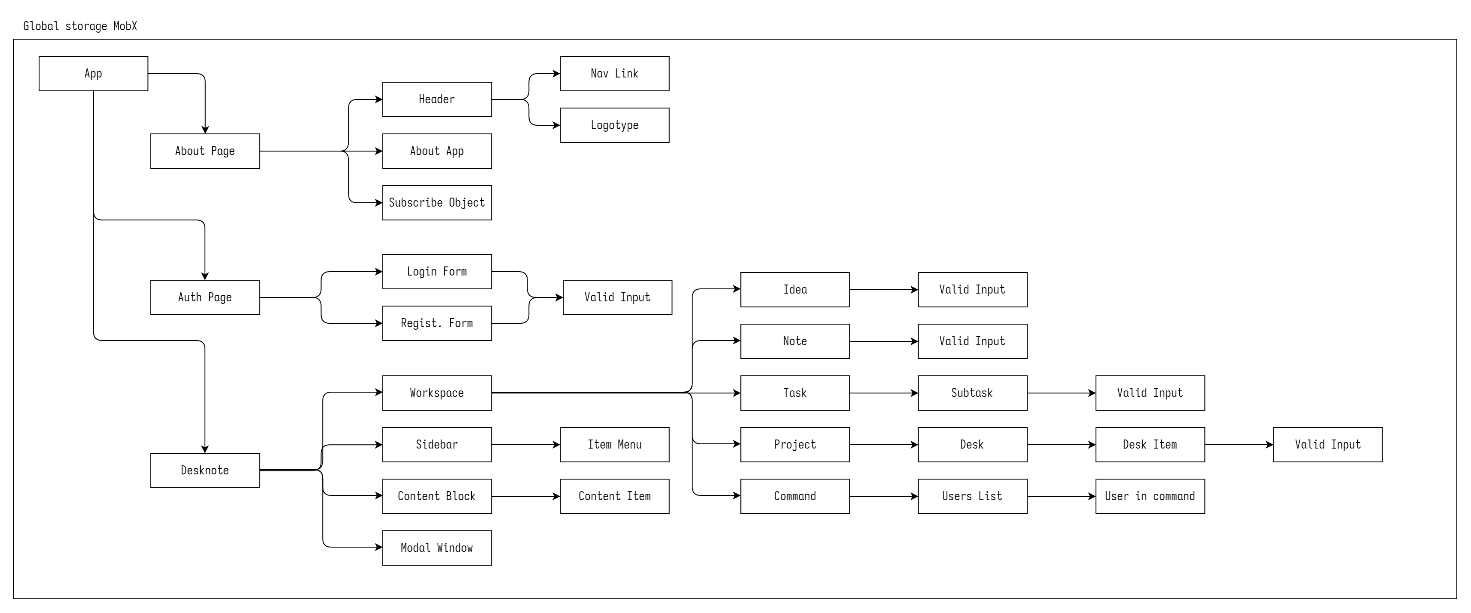


Рисунок 1 – Взаимодействия компонентов клиентского приложения

Для взаимодействия клиента и базы данных был разработан2 сервер, в котором реализована технология API. Взаимодействия происходит следующим образом, компонент обращается к api запросу на сервер, сервер постоянно обрабатывает запросы и прослушивает порт обращения, после получения обращения, на стороне сервера происходит обращения к базе данных, и в случае успешного запроса, возвращает клиенту данные.

# 2.3 Описание работы программы

2.3.1 Общие сведения

«Excellent Note Desk» - это веб-приложение для планирования задач, работой над командными проектами с возможность создавать заметки, иначе можно описать, что это приложение для «управления работой».

2.3.2 Функциональное назначение

Целью разрабатываемого приложения является обеспечение удобного планирования задач, создания заметок и эффективной работы над проектами в командной среде. Оно предоставляет пользователям функциональность, позволяющую легко и систематически управлять своими задачами, вести разработку совместных проектов, и записывать свои идеи или заметки.

2.3.3 Входные данные

Входные данные веб-приложения условно поделены по компонентам, в которые необходимо их ввести.

Страница авторизации пользователей:

* почтовый адрес электронной почты пользователя;
* пароль от учетной записи пользователя.

Страница регистрации пользователя:

* почтовый адрес электронной почты пользователя;
* фамилия пользователя;
* имя пользователя;
* пароль пользователя;
* повторный пароль пользователя.

Страница кабинета пользователя, боковое меню:

* выбор текущего отображение пункта меню.

Страница кабинета пользователя, панель с данным:

* строка поиска данных;
* выбранный пункт данных для отображения.

Страница кабинета пользователя, компонент заметка:

* наименование заметки;
* описание заметки.

Страница кабинета пользователя, компонент задача:

* название задачи;
* описание подзадачи.

Страница кабинет пользователя, компонент команда:

* поисковый запрос или почта добавление участника;
* текущий выбор на удаление пользователя.

Страница кабинет пользователя, компонент проект:

* название проекта;
* название пункта проекта;
* название подпункта проекта;

2.3.4 Выходные данные

Выходными данными приложения являются:

* представленная виде компонентов информация;
* результат запросов пользователей, в виде структурированного компонента.

2.3.5 Тестирование

Тестирование работы веб-приложения является важным этапом разработки, поскольку позволяет обнаружить и исправить ошибки, а также убедиться в правильном функционировании приложения. Для этой цели используются различные методы тестирования, включая ручное тестирование и автоматизированные тесты.

Первый метод - ручное тестирование - заключается в том, что после разработки определенного компонента приложения тестировщик вручную вводит данные и проверяет работу компонента. Этот подход широко применяется для простых компонентов, где не требуется сложных сценариев взаимодействия. Во время ручного тестирования тестировщик активно взаимодействует с приложением, вводит различные комбинации данных, проверяет ответы и убеждается в корректности работы.

Второй метод - тестирование автоматизированными тестами - основан на написании специальных скриптов или классов, которые автоматически проверяют работу приложения. Для этого используются тестовые фреймворки и инструменты, позволяющие создавать и запускать автоматизированные тесты. В данном случае тесты были написаны в отдельном классе UnitTest, который был специально разработан для тестирования функций работы сервера и REST-API. Эти тесты могут проверять различные аспекты работы приложения, включая корректность обработки запросов, валидацию данных, обработку ошибок и другие функциональные возможности. Для тестирования работоспособности компонентов React использовалась библиотека React Test, который позволял проверять автоматизировано производить тестирование.

Тестирование автоматизированными тестами обладает рядом преимуществ перед ручным тестированием. Во-первых, автоматизированные тесты могут быть запущены многократно и с высокой скоростью, что позволяет быстро проверить работу приложения после внесения изменений или исправления ошибок. Во-вторых, автоматизированные тесты могут покрывать широкий спектр сценариев использования и обеспечивать более полное покрытие функциональности приложения. Это помогает выявить потенциальные проблемы и улучшить качество приложения.

Для проведения тестирования были разработаны тест-кейсы, которые описаны в таблице 1.

Таблица 1 – Тест-кейсы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименования | Действия | Ожидаемые результат | Полученный результат |
| Тестирование авторизации пользователя при неполных данных | Оставить пустыми поля для ввода данных пользователя | При попытке входа, в форме появляется текст о том, что поля пустые и являются обязательными | При попытке входа, в форме появляется текст о том, что поля пустые и являются обязательными |
| Тестирование регистрации пользователя при неполных или некорректных данных | Оставить пустыми или некорректно заполненными поля с данными пользователей, и заполнить поле с почтой уже существующей записью в базе данных | При попытке регистрации незаполненные поля помечаются как обязательные поля для заполнения, если же почта пользователя не проходит проверку уникальности, выдает ошибку существования записи | При попытке регистрации незаполненные поля помечаются как обязательные поля для заполнения, если же почта пользователя не проходит проверку уникальности, выдает ошибку |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименования | Действия | Ожидаемые результат | Полученный результат |
| Тестирование работы REST-API без использования JWT токена | Попытка обращения к api приложения без использования JWT токена | Сервер не вернет никаких данных | Сервер не вернет никаких данных |
| Тестирование подключение к удаленному серверу | Подключение по ip адресу к удаленному серверу | Сервер не даст подключиться и ограничить доступ к подключаемому ip | Сервер не даст подключиться и ограничить доступ к подключаемому ip |
| Тестирование работы авто сохраняемых полей в приложении | Попытка быстрого ввода и перехода на другой раздел в меню, после случайного ввода данных в input меню, но не быстрее 300 миллисекунд | Результат ввода пользователя должен сохраниться, и по возвращению к полю, результат не измениться | Результат ввода пользователя должен сохраниться, и по возвращению к полю, результат не измениться |
| Тестирования работы контент-блока попытка уничтожения компонента | Уничтожение компонентов связанных с контентом в случайном порядке | Результатом должно быть корректное удаление компонентов, без каких либо визуальных проблем | Результатом должно быть корректное удаление компонентов, без каких либо визуальных проблем |
| Тестирования работы контент блока попытка добавления записей в зависимости от введенного типа записи | Создание в контент-блоке записи в зависимости от выбранного пункта меню (заметки, проекты, задача, команды) | Результат работы должен создавать записи в базе данных запись, которая совпадает с типом выбранного меню | Результат работы должен создавать записи в базе данных запись, которая совпадает с типом выбранного меню |

Окончание таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименования | Действия | Ожидаемые результат | Полученный результат |
| Тестирование работоспобности поиска в контент блоке | Ввод случайных данных в поисковое поле ввода в контекст блоке | Результат должен выводить элементы, которые соответствуют результата запроса, | Результат должен выводить элементы, которые соответствуют результата запроса, |
| Тестирование отказоустойчивости приложения | Отключение, сервера, и попытка взаимодействия с сайтом | Статичность сайта, сайт показывает лишь экран с последней активности | Статичность сайта, сайт показывает лишь экран с последней активности |
| Тестирование работоспособности компонента задачи и подзадачи | Создание случайных заметок, со случайным текстом, а также создание случайных подзадач | Приложение, сохраняет результат добавления задачи, и подзадачи, и корректно выводит результат | Приложение, сохраняет результат добавления задачи, и подзадачи, и корректно выводит результат |

2.3.6 Вызов и загрузка

Чтобы воспользоваться веб-приложением, необходимо запустить браузер и перейти по ссылке на сайт, после чего приложение будет работоспособно. Запуск веб-приложения не зависит от операционной системы, и устройства, главное, чтобы устройство поддерживало минимальные системные требования приложения.

Для запуска сервера необходимо, что на запускаемом компьютере имелась операционная система, которая поддерживает NodeJS версии 18.16 и младше, а также NPM для установки всех зависимостей проекта. После установки всех зависимостей пользователь может выполнить команду запуска, и сервер запуститься, предупредив пользователя об этом сообщением в консольном приложении. В случае если база данных находиться в удаленном доступе необходимо стабильное подключение к интернету.

# 2.4 Руководство пользователя

2.4.1 Назначение программы

Целевая аудитория веб-приложения является широкой и многоцелевой. Оно предназначено для пользователей, которым требуется эффективное планирование задач, составление заметок и совместная работа над проектами. В эту аудиторию входят как студенты и школьники, нуждающиеся в организации своего учебного процесса, так и работники различных сфер деятельности, которым необходимо эффективное управление проектами и задачами.

2.4.2 Условия выполнения

Для запуска веб-приложения требуется всего лишь современный веб-браузер версии 7.0 и выше, а также устройство, способное поддерживать эту версию браузера. После входа в браузер пользователю нужно просто перейти по ссылке на сайт, где размещено веб-приложение, для этого необходимо иметь стабильное интернет подключение. После этого пользователю становятся доступными все необходимые функции и действия, для реализации поставленных задач.

Этот простой и удобный способ доступа позволяет пользователям быстро начать пользоваться веб-приложением без необходимости установки дополнительных программ или расширений. Вся необходимая функциональность предоставляется непосредственно через веб-браузер, что делает процесс использования максимально простым и удобным для пользователей.

Для выполнения серверной части, необходимо воспользоваться услугами компаний предоставляющий доступ к удаленным серверам, после чего необходимо выполнить установку всех зависимостей для работы программы, и запустить серверную часть.

2.4.3 Выполнение программы

2.4.3.1 Работа пользователя

При первом посещении сайта пользователь попадает на информационную страницу, где представлена основная информация о продукте. Однако, чтобы воспользоваться полным функционалом приложения, пользователь должен пройти процесс регистрации. Для этого ему необходимо нажать на кнопку "Регистрация" и корректно заполнить все необходимые поля, включая личные данные и учетные данные. После успешной регистрации пользователь будет перенаправлен на страницу авторизации, где пользователь должен указать введенные данные, и нажать кнопку «Войти», после чего он попадет в пользовательский кабинет.

В личном кабинете пользователь получает доступ к удобной боковой панели меню, которая расположена слева. При наведении на нее она автоматически раскрывается, отображая полный список доступных пунктов меню. Это позволяет пользователям легко найти необходимые функции и совершать нужные действия.

Например, для создания заметки пользователю достаточно выбрать соответствующий пункт меню "Заметки". При выборе этого пункта открывается список существующих заметок, где пользователь может выбрать нужную ему заметку для редактирования или создать новую заметку. При создании новой заметки в рабочей области открывается пустой шаблон, где пользователь может свободно вводить свои данные и информацию. После завершения редактирования пользователь должен просто нажать кнопку "Сохранить", чтобы сохранить свою заметку.

Аналогично, для создания задачи пользователю нужно перейти в пункт меню "Задачи". Здесь он может выбрать существующую задачу для просмотра и редактирования, либо создать новую задачу. При создании новой задачи в рабочей области открывается пустой шаблон задачи, где пользователь может указать название, сроки выполнения, приоритет и другие необходимые детали. При необходимости пользователь может также добавить подзадачи к основной задаче, чтобы более подробно структурировать свою работу. Для создания команды, пользователю необходимо выбрать пункт меню, «Команды», после чего появиться список команд, в которых состоит пользователь, пользователь может создать свою команду, и после того, как задаст имя команды, может добавлять других пользователь, введя их почту, после чего пользователь должен принять приглашение и в команде добавиться новый участник.

Для управления командами и совместной работы над проектами пользователь может воспользоваться пунктом меню "Команды". Здесь пользователь может просмотреть список команд, в которых он состоит, а также создать новую команду. При создании команды пользователь может указать ее название и добавить других участников, используя их электронную почту. После отправки приглашения участникам, они должны подтвердить свое участие, и затем они становятся членами команды, где могут взаимодействовать, делиться информацией и работать вместе над проектами.

Наконец, для создания проектов пользователю следует перейти в пункт меню "Проекты". Здесь он может просмотреть список уже существующих проектов или создать новый проект, важно отметить, что пользователь должен находится в какой-либо команде, после чего нажатии на добавление нового проекта, пользователю покажется модальное окно, в котором необходимо выбрать команду, в которой вы создаете проект. При создании нового проекта пользователь получает доступ к канбан-доске, где может создавать и организовывать задачи, добавлять пункты.

Важно отметить, что данные пользователя сохраняются автоматически, то есть нет необходимости устанавливать, какие-либо кнопки на сохранение данных, все данные сохраняются автоматически после введения пользователя.

Таким образом, веб-приложение предоставляет удобный и интуитивно понятный интерфейс, таким образом предполагается простотой приложения расширить рамки целевой аудитории пользователей.

2.4.3.2 Работа администратора

В целях обеспечения конфиденциальности данных пользователей, администратор не имеет прямого доступа к этим данным. В связи с этим, веб-приложение «Excellent Note Desk» было разработано без учета создания административной панели. Вместо этого, для получения информации о активности пользователей можно использовать различные встроенные метрики, которые предоставляют данные о взаимодействии пользователей с приложением.

Для удаления неактивных пользователей может быть использован механизм триггеров в базе данных. Это позволяет автоматически удалять пользователей, которые не проявляют активности в течение определенного временного периода. Такой подход обеспечивает автоматическую очистку базы данных от неактивных аккаунтов, минимизирую ручное вмешательство и облегчая управление пользователями.

В итоге, веб-приложение «Excellent Note Desk» было разработано с учетом конфиденциальности данных пользователей, исключая прямой доступ администратора к этим данным.

2.4.3.3 Техника безопасности при работе на компьютере

При работе за компьютером необходимо соблюдать определенные меры безопасности, чтобы обеспечить свою физическую и информационную безопасность. В данном тексте представлена набор основных техник безопасности, которые следует применять при работе за компьютером:

* ограничьте прием пищи и питье: когда вы работаете за компьютером, рекомендуется ограничить прием пищи и питье рядом с рабочим местом. разлитие напитка или попадание пищи на клавиатуру или другие компоненты компьютера может привести к повреждению оборудования и потере данных;
* регулярно выполняйте резервное копирование данных: регулярное резервное копирование важных данных является важной мерой безопасности. это позволяет восстановить данные в случае их потери или повреждения. используйте надежные внешние накопители или облачные сервисы для хранения резервных копий;
* закройте сеанс или заблокируйте компьютер при отсутствии: важно закрывать сеанс работы или блокировать компьютер, когда вы временно покидаете рабочее место. это предотвращает несанкционированный доступ к вашим данным и программам;
* используйте надежные пароли: создание и использование надежных паролей является важной мерой безопасности. используйте пароли, состоящие из комбинации заглавных и строчных букв, цифр и специальных символов. избегайте использования личной информации или простых словарных слов. кроме того, рекомендуется использовать уникальные пароли для каждого аккаунта и регулярно их менять;
* обновляйте программное обеспечение: регулярное обновление программного обеспечения, включая операционную систему и приложения, является важным шагом для обеспечения безопасности компьютера. обновления содержат исправления уязвимостей и помогают защитить вашу систему от известных угроз;
* остерегайтесь фишинговых атак: будьте осторожны при получении подозрительных электронных писем или переходе по непроверенным ссылкам. фишинговые атаки могут пытаться получить ваши личные данные или пароли путем подделки электронных писем или веб-страниц. не раскрывайте конфиденциальную информацию, если вы не уверены в подлинности источника;
* используйте надежное антивирусное программное обеспечение: установка и регулярное обновление надежного антивирусного программного обеспечения поможет защитить ваш компьютер от вредоносных программ и вирусов. выберите известные и проверенные антивирусные программы и регулярно сканируйте систему для выявления и удаления потенциальных угроз;
* используйте защищенное подключение к интернету: при работе за компьютером убедитесь, что вы используете защищенное подключение к интернету, особенно при передаче конфиденциальной информации. используйте надежные wi-fi сети с шифрованием или виртуальные частные сети (vpn) для защиты своих данных от несанкционированного доступа.

Соблюдение этих техник безопасности при работе за компьютером поможет вам минимизировать риски и обеспечить сохранность вашей информации. Будьте внимательны и ответственны в отношении безопасности при работе за компьютером, и ваша работа будет более защищена.

# 3 Экономическая часть

# 3.1 Расчет основной и дополнительной заработной платы

Смета затрат на производство — это общий свод плановых затрат (в стоимостном выражении) на производство продукции, выполнение работ и услуг в соответствии с производственной программой предприятия. Расчет сметы затрат на производство осуществляется на основе бизнес-плана в части производственной программы на будущий финансовый год, а также анализа фактических данных прошедшего периода и расчетных нормативов по статьям затрат, принятых в организации.

Расчетные нормативы по статьям затрат планируются на основе анализа действующих рыночных цен на аналогичную выпускаемую продукцию, фактических цен реализации продукции прошедшего финансового года.

По этим данным определяется предельный норматив затрат на производство при существующем уровне рыночной (договорной) цены. Рассчитанный таким образом предельный норматив затрат на единицу продукции с учетом принятых на предприятии нормативов по статьям затрат позволяет определить плановую смету в разрезе статей и элементов.

С учетом специфики отраслевой деятельности организации составление сметы затрат на производство ведется по сегментам (видам) деятельности, видам продукции, работ, услуг; дальнейшая группировка возможна по структурным подразделениям.

На заключительном этапе планирования сметы затрат на производство составляется сводная смета под плановую производственную программу с учетом объемов выпуска.

На первом этапе разработки программного продукта необходимо рассчитать смету затрат, а также определить этапы разработки программного продукта и трудоёмкость выполнения каждого этапа. Все показатели представлены в таблице 2.

Таблица 2 – Этапы и трудоемкость выполнения работ в часах

|  |  |  |
| --- | --- | --- |
| Этапы разработки | Трудоемкость выполнения работ | |
| Руководитель | Техник |
| Постановка задачи | 3 | 18 |
| Выбор инструментальных средств | 1 | 4 |
| Разработка математической модели | 2 | 35 |
| Построение алгоритма | 2 | 30 |
| Программирование | 1 | 50 |
| Тестирование | 1 | 5 |
| Отладка | 1 | 4 |
| Написание пояснительной записки | 1 | 14 |
| Итого | 13 | 160 |

Месячный оклад руководителя составляет 38250 рублей, месячный оклад техника составляет 19788 рублей. Общие итоговое время работы составило 173 часа.

Основная зарабатывая плата работника – это зарплата работника, умноженная на объем выполнения работы и разделенная на количество отработанных часов, 38250 \* 13 / 173 = 2874,27 рублей, основная заработная плата руководителя. Основная заработная плата техника составляет 19788 \* 160 / 173 = 18301,04 рублей. В сумме основная заработная плата руководителя и техника составила 21175,31 рублей.

Таблица 3 – Основная заработная плата работников в рублях

|  |  |
| --- | --- |
| Должность | Заработная палата |
| Руководитель проекта | 2874,27 |
| Техника | 18301,04 |
| Итого | 21175,31 |

3.1.2 Расчет дополнительной заработной платы и отчислений на социальное страхование

Заработная плата зависит от количества сотрудников, их квалификации и времени, затраченного на разработку продукта и его этапов. В дополнение к основной зарплате сотрудники получают премии за различные виды работ, включая сверхурочные, специализированные навыки и работу в праздничные дни.

Доплата в виде компенсации положена сотрудникам в определенных случаях, когда он работает:

* в особых климатических условиях;
* на территориях с радиоактивным загрязнением;
* во вредных, тяжелых или опасных условиях труда;
* со сведениями, которые составляют государственную тайну;
* в условиях, которые отклоняются от нормальных, например, работа в ночное время, совмещение должностей;
* в других случаях, которые установлены системой оплатой труда.

Дополнительная заработная плата составляет 10% от основной заработной платы и составляет 2117,53 рублей.

Взносы в Социальный фонд Российской Федерации и Фонд обязательного медицинского страхования составляют 30% от основной и дополнительной заработной платы. Таким образом, они эквивалентны 6352,5 рубля.

Общая сумма основной и дополнительной заработной платы, и взносов в Российский социальный фонд и Фонд обязательного медицинского страхования составляет 29645,11 рубля.

# 3.2 Расчёт стоимости материалов и лицензионного обеспечения

Исходя из нормы потребления материальных ресурсов и их оптовых цен, используя формулу 1 для расчета стоимости материалов, необходимых для разработки программного обеспечения.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

где – норма расхода материальных ресурсов в натуральных единицах;

Ц – цена приобретения за единицу, руб.

Расчёт стоимости материалов представлен в таблице 4.

Таблица 4 – Расчет стоимости материалов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование материала | Норма расхода, шт. | Цены за единицу, рублей | Сумма в рублях |
| Упаковка бумаги | 1 | 386,99 | 386,99 |
| Папка | 1 | 84,55 | 84,55 |
| Флешка на 32 гб. | 1 | 1245,00 | 1245,00 |
| Картридж | 1 | 420,00 | 420,00 |
| Ручка | 1 | 98,99 | 98,99 |
| Итого | | | 2235,53 |

Общая стоимость материалов, необходимых для реализации проекта составляет 2235,53 рублей.

3.2.2 Расчет стоимости лицензионного обеспечения

При разработке программного продукта учитывается стоимость используемого в процессе лицензионного программного обеспечения. Это также учитывается при расчете стоимости создания продукта.

В таблице 5 определен перечень лицензионного программного обеспечения, срок его действия и цена.

Таблица 5 – Перечень программного обеспечение для выполнения работ

|  |  |  |
| --- | --- | --- |
| Наименование лицензионных программ | Норма установки, шт. | Цена, руб |
| Microsoft Windows 10 Pro | 1 | 20980,00 |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| Наименование лицензионных программ | Норма установки, шт. | Цена, руб |
| VPS хостинг Beget | 1 | 19800,00 |
| HeidiSQL | 1 | 0,00 |
| Удаленная база данных Beget | 1 | 32400,00 |
| Microsoft Office 365 | 1 | 6500,00 |
| Visual Studo Code | 1 | 0,00 |
| Итого | | 58700,00 |

Стоимость лицензионного программного обеспечения для разработки программного продукта составит 58700,00 рублей.

Общие операционные расходы составили 90580,64 рублей и были направлены на покрытие основной дополнительной заработной платы, взносов в Российский социальный фонд, и фонд обязательного медицинского страхования, расходов на материалы и программное обеспечение.

# 3.3 Расчет накладных расходов

Накладные расходы — это затраты, используемые для организации, обслуживания производства, продаж и администрирования. Они не связаны напрямую с основным производством товаров или оказанием услуг и не включаются в стоимость материалов или труда.

Накладные расходы включаются в издержки производства, изготовления и распределения, пропорционально стоимости материалов и сырья, размеру заработной платы и т.д.

Исходя из рассчитанных выше данных, себестоимость программного продукта представлена в таблице 6.

Таблицы 6 – Смета затрат на программный продукт

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Сумма, рублей | Структура затрат, % |
| Основная и дополнительная заработная плата | 23292,84 | 23,3 |
| Отчисления на социальное страхование | 6352,5 | 6,6 |
| Стоимость материалов | 2235,53 | 2,2 |
| Стоимость лицензионного обеспечения | 58700 | 58,9 |
| Накладные расходы | 9058,06 | 9,0 |
| Итого | 99638,06 | 100 |

Накладные расходы составляют 10% от расходов на эксплуатацию. и рассчитываются по формуле 2.

|  |  |
| --- | --- |
|  | (2) |

где Рэксп –величина расходов на эксплуатацию, рублей.

рублей

Таким образом, общая сумма затрат на разработку программного продукта составит 90580,64 + = 99638,06 рублей.

Под структурой затрат понимается удельный вес (в процентах) каждой статьи в общей величине затрат.

По результатам расчета построим диаграмму структуры затрат на программный продукт и представим её на рисунке В.1

# 3.4 Составление и расчет цены реализации программного продукта

Цена — это важный экономический показатель, который приносит доход от продажи программных продуктов и влияет на прибыль и финансовую стабильность компании. Она является важным фактором выживания компании. Поскольку целью бизнеса является получение прибыли, продажная цена программного продукта представляет собой сумму затрат на разработку и ожидаемой прибыли. Здесь уровень нормы прибыли принимается равным P = 14%. Значение общих затрат на разработку программного продукта будет взято из таблицы 5, а значение материальных затрат - из таблицы 4.

Расчет оптовой цены производится по формуле 3.

|  |  |
| --- | --- |
| , | (3) |

где З – общие затраты на разработку программного продукта, рублей;

Р – уровень рентабельности проекта, коэффициент

|  |  |
| --- | --- |
| = 103261,92 рублей |  |

Расчёт прибыли от продажи продукта производится по формуле 4.

|  |  |
| --- | --- |
| , | (4) |
|  |  |

Таким образом, можно сделать вывод, что данный продукт прибыльный и может быть реализован. Прибыль от продажи программного продукта составит рублей.

# Заключение

В результате выполнения работы по дипломному проектированию, были достигнуты поставленные цели, а именно разработка веб-приложения для планирования задач пользователей, с возможностью командной работы, были закреплены и улучшены навыки программирования в области веб-технологий, изучены методы создания REST-API и работы с ним, были также подкреплены навыки создания информационных систем, с помощью MySQL, и улучшены навыки тестирования веб-приложений.

На этапе формирования требований к системе были сформулированы цеди и задачи дипломного проекта, которые были полностью выполнены в ходе разработки.

На этапе проектирования была разработана реляционная модель базы данных, сначала в виде диаграммы с помощью сервиса по созданию диаграмм DrawIO, а также с помощью данного сервиса, разрабатывалась предположительная архитектура веб-приложения.

В этап реализации приложения была реализована база данных средствами MySQL и СУБД HeidiSQL. После чего был разработан api взаимодействия клиента и базы данных. И конечным этапом реализации была разработка клиентской части веб-приложения.

После завершения разработки приложения была написана пояснительная документация.

Данный комплекс приложений позволит организовывать пользователям, удобную организацию своей работы.

# Список литературы

1. ГОСТ 2.701—2008 Единая система конструкторской документации. Схемы. Виды и типы. Общие требования к выполнению. – М.: Стандартинформ, 2009. – 15 с.
2. ГОСТ 2.743—91 Единая система конструкторской документации. Обозначения условные графические в схемах. Элементы цифровой техники. – М.: Стандартинформ, 2009. – 62 с.
3. Баранов А.С., Леснов Д.В. "React и Redux. Функциональная веб-разработка" - Питер, 2018 - 448 страниц.
4. Волков М.А. "JavaScript. Основы языка" - БХВ-Петербург, 2017 - 400 страниц.
5. Гафаров А. "MySQL. Оптимизация производительности" - ДМК Пресс, 2016 - 352 страницы.
6. Головачева А., Макаров С. "React. Быстрый старт" - ДМК Пресс, 2019 - 272 страницы.
7. Дамм А., Мисра Р. "Выразительный JavaScript" - ДМК Пресс, 2019 - 472 страницы.
8. Кантор Я. "JavaScript для детей" - ДМК Пресс, 2016 - 336 страниц.
9. Кузнецов Д. "React+Redux. Разработка приложений" - Питер, 2019 - 336 страниц.
10. Павловская Н. "JavaScript. Уровень 1. Основы" - Лань, 2020 - 304 страницы.
11. Розенберг Е., Райан Т. "Изучаем MySQL" - ДМК Пресс, 2016 - 832 страницы.
12. Шакирзянов Р., Исмаилов Ф. "JavaScript. Основы и применение" - БХВ-Петербург, 2019 - 480 страниц.

# Приложение А (Обязательное)

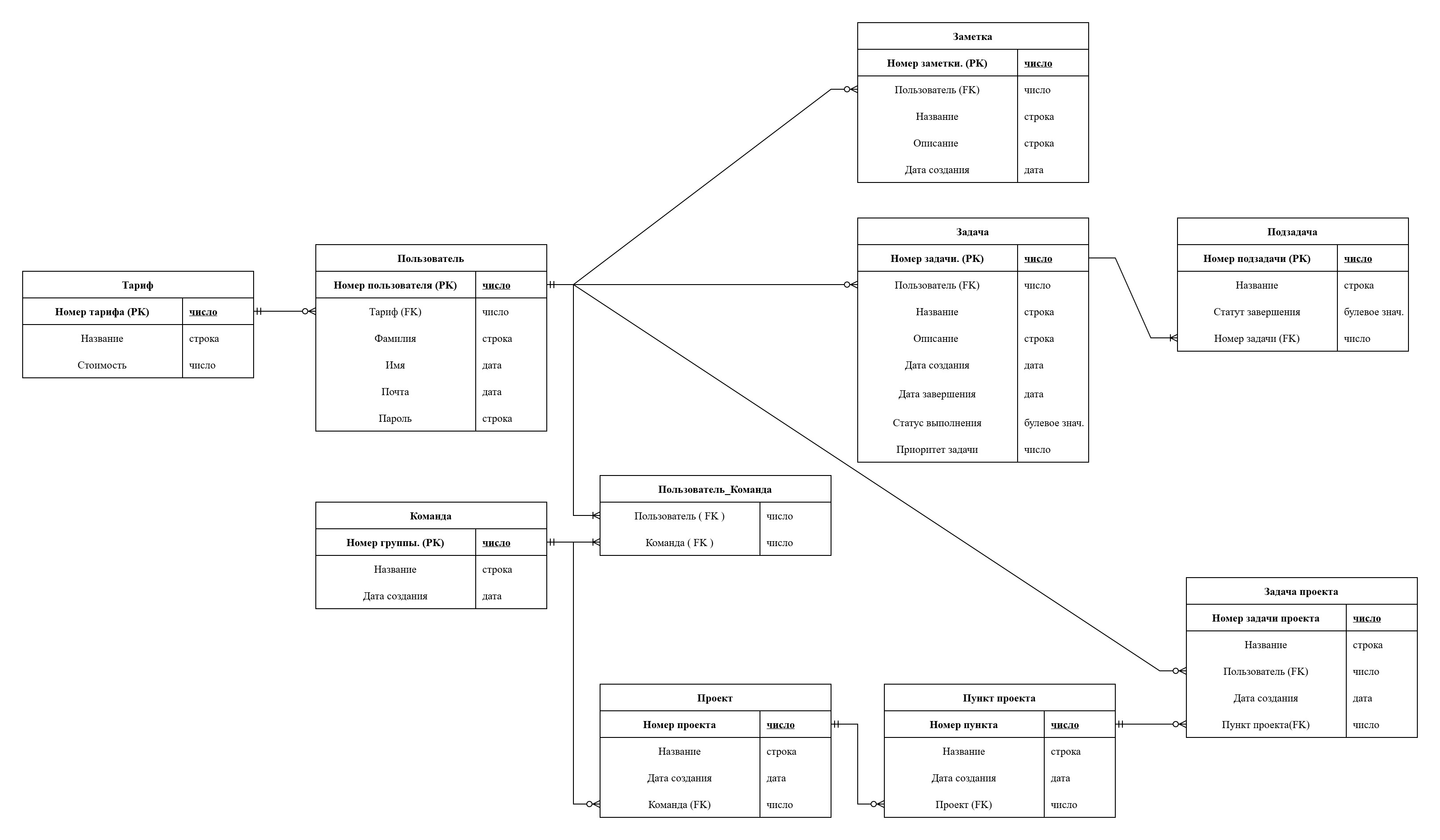


Рисунок А1 – Реляционная модель данных