Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

|  |
| --- |
| *К защите допустить*: |
| Заведующий кафедрой ИСиТ |
| \_\_\_\_\_\_\_\_\_\_\_\_А.И. Парамонов |

Пояснительная записка

к дипломному проекту

на тему

**программное средство автоматизации**

**бизнес-процессов информационного отдела организации**

БГУИР ДП 1-40 01 01 01 040 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А.В. Готовчиц |
| Руководитель |  | А.С. Корзан |
| Консультанты: |  |  |
| *от кафедры ИСиТ* |  | А.Г. Савенко |
| *по экономической части* |  | В.Г. Горовой |
|  |  |  |
| Нормоконтролер |  | А.С. Шелягович |
|  |  |  |
| Рецензент |  |  |

Минск 2024

Решением рабочей комиссии

допущен(а) к защите дипломного проекта

Председатель рабочей комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

Подпись Инициалы и фамилия

«\_\_\_» январь 2025 г.

**РЕФЕРАТ**

Программное средство автоматизации бизнес-процессов информационного отдела организации: дипломный проект / А. В. Готовчиц. – Минск: БГУИР, 2025, – п.з. – 67сс., чертежей (плакатов) – 6 л. формата А1.

Объектом проектирования является программное средство для автоматизации бизнес-процессов информационного отдела организации.

Целью данного дипломного проекта является разработка программного средства, которое позволяет автоматизировать ключевые бизнес-процессы информационного отдела, включая планирование задач, управление видеоконференциями, отправку заявок на доступ к системам, обработку пользовательских запросов и управление доступными сервисами.

Для достижения поставленной цели был реализован комплексный подход, включающий проектирование архитектуры системы, разработку базы данных, создание пользовательского интерфейса и интеграцию с внешними сервисами. Программное средство реализовано с использованием современных технологий, таких как Python, Django, JavaScript, Docker, Kubernetes, React и PostgreSQL, что обеспечивает его масштабируемость, надежность и удобство в эксплуатации.

В процессе выполнения проекта были изучены современные подходы к автоматизации бизнес-процессов, проведен анализ аналогичных решений, а также сформулированы и обоснованы требования к разрабатываемой системе.

Программное средство протестировано с использованием методов белого и черного ящика. Тестирование подтвердило его соответствие заявленным требованиям по функциональности, надежности и производительности. Обнаруженные в процессе тестирования ошибки были исправлены, что повысило стабильность и удобство использования системы.

В разделе технико-экономического обоснования выполнены расчеты затрат на разработку и эксплуатацию программного средства. Результаты расчетов подтверждают экономическую целесообразность разработки и внедрения системы, что позволит значительно сократить время обработки запросов и повысить эффективность работы сотрудников информационного отдела.

Разработанное программное средство полностью готово к внедрению в эксплуатацию и обеспечивает повышение эффективности управления бизнес-процессами, снижая трудозатраты и минимизируя риски, связанные с человеческим фактором.

# СОДЕРЖАНИЕ

[Введение 3](#_Toc167707419)

[1 Описание и анализ объекта автоматизации 3](#_Toc167707420)

[1.1 Структура объекта автоматизации 3](#_Toc167707421)

[1.2 Анализ и описание существующей информационной системы 3](#_Toc167707422)

[1.3 Краткое описание, классификация и анализ существующих информационных систем, аналогичных разрабатываемой 3](#_Toc167707423)

[1.4 Постановка задачи на дипломное проектирование 3](#_Toc167707424)

[2 Проектирование структуры автоматизированной системы мониторинга криптовалютного рынка 3](#_Toc167707425)

[2.1 Организационно-экономическая сущность задачи 3](#_Toc167707426)

[2.2 Информационное обеспечение 3](#_Toc167707427)

[2.3 Структура автоматихированной системы мониторинга криптовалютного рынка 3](#_Toc167707428)

[3 Программная реализация автоматизированной системы мониторинга криптовалютного рынка 3](#_Toc167707429)

[3.1 Выбор средств реализации автоматизированной системы  
мониторинга криптовалютного рынка 3](#_Toc167707430)

[3.2 Реализация базы данных 3](#_Toc167707431)

[3.3 Разработка программного кода системы 3](#_Toc167707432)

[3.4 Руководство пользователя 3](#_Toc167707433)

[4 Экономическое обоснование автоматизированной системы мониторинга криптовалютного рынка 3](#_Toc167707434)

[4.1 Характеристика программного средства 3](#_Toc167707435)

[4.2 Расчет экономической эффективности от реализации программного средства на рынке 3](#_Toc167707436)

[4.3 Расчет показателей эффективности разработки и реализации программного средства на рынке 3](#_Toc167707437)

[Заключение 3](#_Toc167707438)

[Список использованных источников 3](#_Toc167707439)

[Приложение А (обязательное) Программный код алгоритма поиска 3](#_Toc167707440)

[Приложение Б (обязательное) Результаты проверки дипломного проекта   
назаимствования 3](#_Toc167707441)

Ведомость документов 67

**ВВЕДЕНИЕ**

Современные организации сталкиваются с необходимостью автоматизации бизнес-процессов для повышения эффективности работы и оптимизации использования ресурсов. Информационные отделы играют ключевую роль в поддержке таких процессов, обеспечивая доступ к данным и инфраструктуре. В последние годы активно развиваются программные решения для управления задачами, планирования, мониторинга и взаимодействия сотрудников, что способствует снижению человеческого фактора и ускорению выполнения операций.

Целью данного дипломного проектирования является разработка программного средства для автоматизации бизнес-процессов информационного отдела организации. Разрабатываемое решение должно обеспечить удобное планирование задач, управление запросами пользователей, организацию видеоконференций и централизованный доступ ко всем предоставляемым услугам.

Основные принципы проектирования включают использование современных технологий разработки, таких как Python, Django, JavaScript, Docker, Kubernetes и PostgreSQL. Процесс проектирования базируется на принципах модульности, масштабируемости и эргономичности пользовательского интерфейса.

Пояснительная записка к проекту состоит из нескольких разделов, каждый из которых решает конкретные задачи. Первый раздел посвящен анализу предметной области и включает обзор аналогичных программных средств, а также обоснование выбора технологий для разработки. Второй раздел описывает моделирование предметной области, включая информационную модель базы данных и спецификацию требований к системе. В третьем разделе рассматривается проектирование системы: разработаны диаграммы базы данных, классов и развертывания, а также представлены схемы работы приложения. Четвертый раздел содержит результаты тестирования разработанного программного средства. В пятом разделе представлено руководство по эксплуатации системы. Заключительный, шестой раздел включает технико-экономическое обоснование разработки.

Данный дипломный проект выполнен мной лично, проверен на заимствования в системе «Антиплагиат», оригинальность пояснительной записки составляет 85%.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Обзор существующих аналогов**

Для автоматизации бизнес-процессов и управления информационными потоками часто используются системы, предназначенные для планирования, управления проектами, командного взаимодействия и документооборота. Среди них выделяются такие продукты, как Jira, Atlassian Confluence и Taiga, которые широко применяются в различных организациях для управления задачами и проектами. Рассмотрим их более подробно, выявив их преимущества и недостатки.

Jira – это система для управления проектами и задачами, разработанная компанией Atlassian. Она особенно популярна среди команд, использующих методологии Agile (например, Scrum и Kanban). Jira предоставляет множество инструментов для управления рабочими процессами, мониторинга выполнения задач и анализа производительности команд.

Преимущества работы с Jira:

1. Гибкость настройки Jira позволяет настроить рабочие процессы, статусы задач, роли участников и разрешения для различных типов проектов. Это делает её удобной для команд с различными потребностями.
2. Поддержка Agile в Jira предоставляет встроенные инструменты для работы по Scrum и Kanban, такие как доски задач, спринты и отчеты по производительности (burn-down и burn-up диаграммы).
3. Интеграция с другими инструментами Atlassian. Jira легко интегрируется с Confluence (для документации), Bitbucket (для управления репозиториями кода) и другими продуктами Atlassian, что делает её частью комплексной экосистемы для управления разработкой программного обеспечения.
4. Jira поддерживает множество плагинов и расширений, что позволяет добавлять функциональность для специфических нужд, включая отчеты, автоматизацию задач и интеграцию с другими системами.

Недостатки работы с Jira:

1. Сложность для небольших команд. Для небольших организаций или команд с ограниченными ресурсами Jira может оказаться слишком сложной и избыточной по функционалу. Множество настроек и возможностей требуют времени для освоения и внедрения.
2. Стоимость. Jira является платной системой, и при расширении команды или добавлении новых функций затраты на её использование могут значительно возрасти.
3. Потребность в обучении. Интерфейс Jira может быть сложным для новых пользователей, особенно для тех, кто не имеет опыта работы с Agile или подобными системами управления проектами.

Пример работы с Jira представлен на рисунке 1.1.

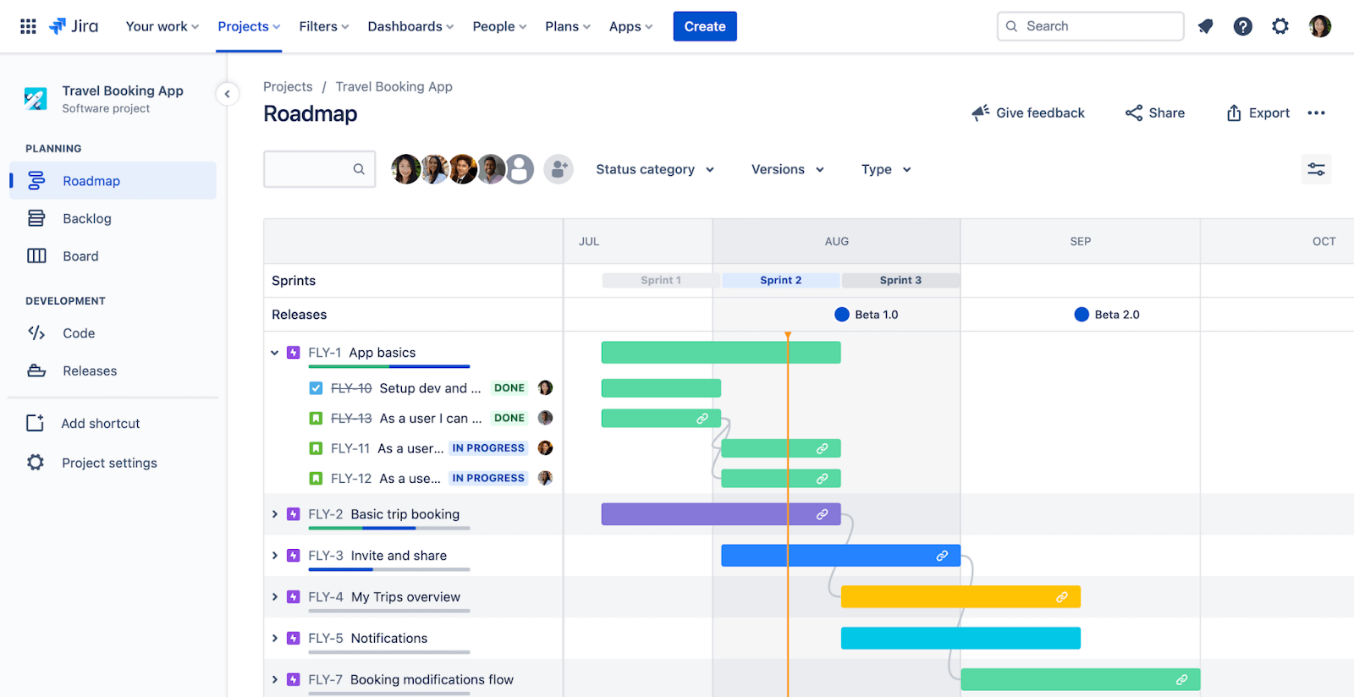


Рисунок 1.1 – Страница с проектами программного средства Jira.

Atlassian Confluence – это система для совместной работы и управления знаниями, предназначенная для создания, организации и управления документацией. Она тесно интегрирована с Jira и используется для ведения проектной документации, разработки спецификаций и совместной работы команд.

Преимущества:

1. Совместное редактирование. Confluence позволяет одновременно редактировать документы нескольким пользователям, что облегчает совместную работу над проектами.
2. Интеграция с Jira. Сильная интеграция с Jira позволяет легко связывать задачи с соответствующей документацией и отслеживать взаимосвязи между различными элементами проекта.
3. Шаблоны документации. Confluence предоставляет множество готовых шаблонов для документации, что упрощает ведение рабочих процессов и структурирование информации.
4. Контроль версий. Confluence поддерживает версионность документов, что позволяет отслеживать изменения и возвращаться к предыдущим версиям, если это необходимо.

Недостатки:

1. Сложность использования для начинающих: Confluence может быть сложным для новых пользователей, особенно для тех, кто не знаком с системами управления знаниями. Освоение всех функций и возможностей может занять время.
2. Отсутствие продвинутых возможностей для совместной работы. хотя Confluence отлично подходит для документации, он не всегда может предложить те же инструменты совместной работы, которые предлагают более специализированные системы управления проектами и коммуникацией.
3. Высокая стоимость продукта. Как и другие продукты Atlassian, Confluence является платным решением, и затраты могут увеличиваться с ростом команды или необходимостью дополнительных функций.

Пример работы с Atlassian Confluence представлен на рисунке 1.2.

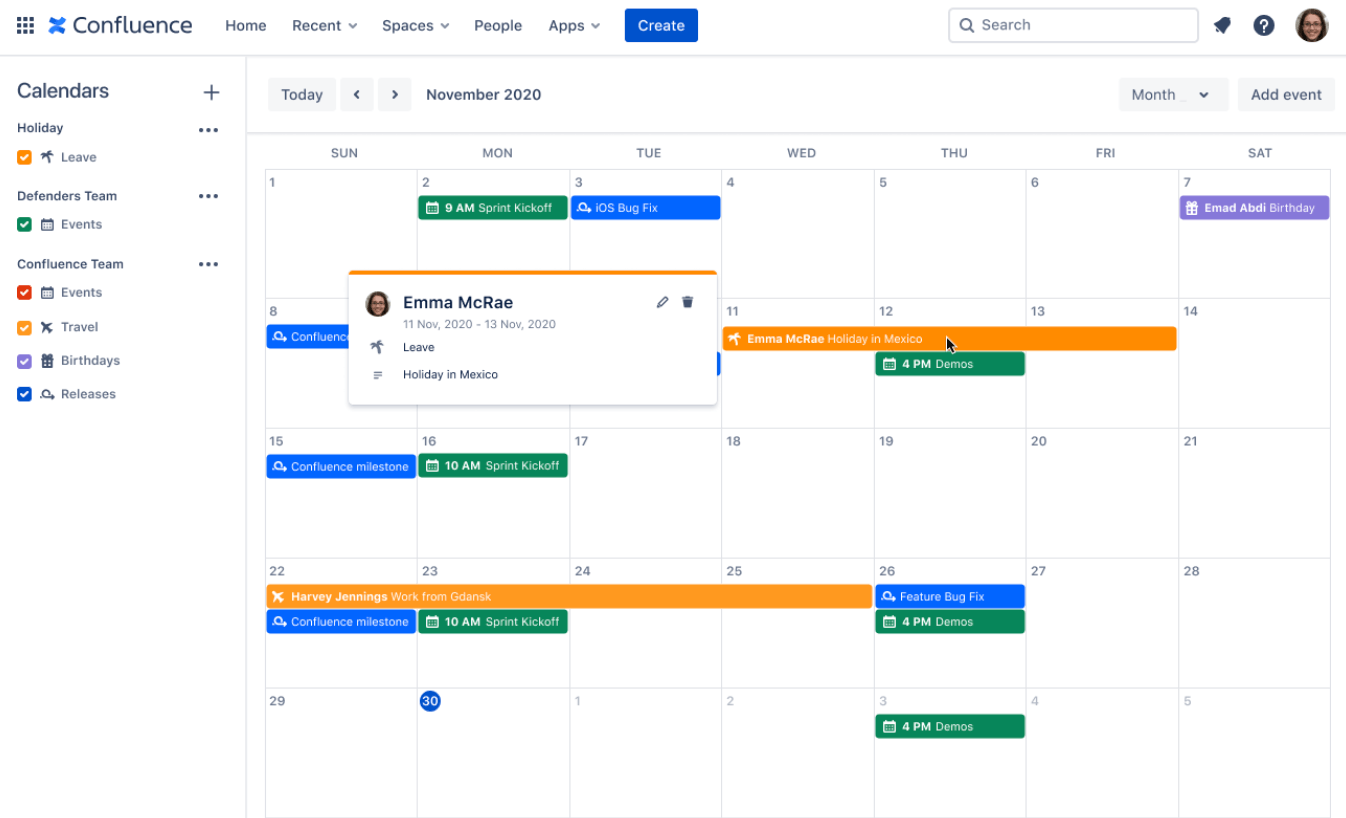


Рисунок 1.2 – Страница работы с календарем программного средства Atlassian Confluence.

Taiga – это платформа с открытым исходным кодом для управления проектами, ориентированная на команды, работающие по методологиям Agile. Она поддерживает Kanban и Scrum, а также предоставляет инструменты для отслеживания задач, планирования спринтов и ведения беклогов.

Преимущества работы с Taiga:

1. Открытый исходный код. Taiga является бесплатной для небольших команд и доступной для развертывания на собственных серверах, что позволяет полностью контролировать данные и настройку системы.
2. Простота использования. В отличие от Jira, Taiga предлагает более простой и интуитивный интерфейс, что делает её удобной для небольших команд или стартапов, которые только начинают внедрять Agile.
3. Гибкость и кастомизация. несмотря на простоту, Taiga предлагает достаточный уровень гибкости для настройки рабочих процессов и управления задачами в зависимости от требований команды.
4. Интеграция с Git: Taiga поддерживает интеграцию с системами контроля версий, такими как GitLab и GitHub, что удобно для команд разработчиков программного обеспечения.

Недостатки:

1. Ограниченный функционал. В сравнении с Jira, Taiga предлагает меньше возможностей для кастомизации и расширения. Она больше ориентирована на небольшие команды и может не удовлетворить потребности крупных организаций с комплексными процессами.
2. Ограниченные отчеты и аналитика. Taiga предоставляет ограниченные инструменты для анализа производительности команды и генерации отчетов, что может быть недостатком для менеджеров, ориентированных на метрики.
3. Отсутствие комплексной поддержки. поскольку Taiga является open-source решением, поддержка может быть ограничена, и пользователи должны полагаться на сообщество или свои собственные ресурсы для решения технических проблем.

Пример работы с Taiga представлен на рисунке 1.3.

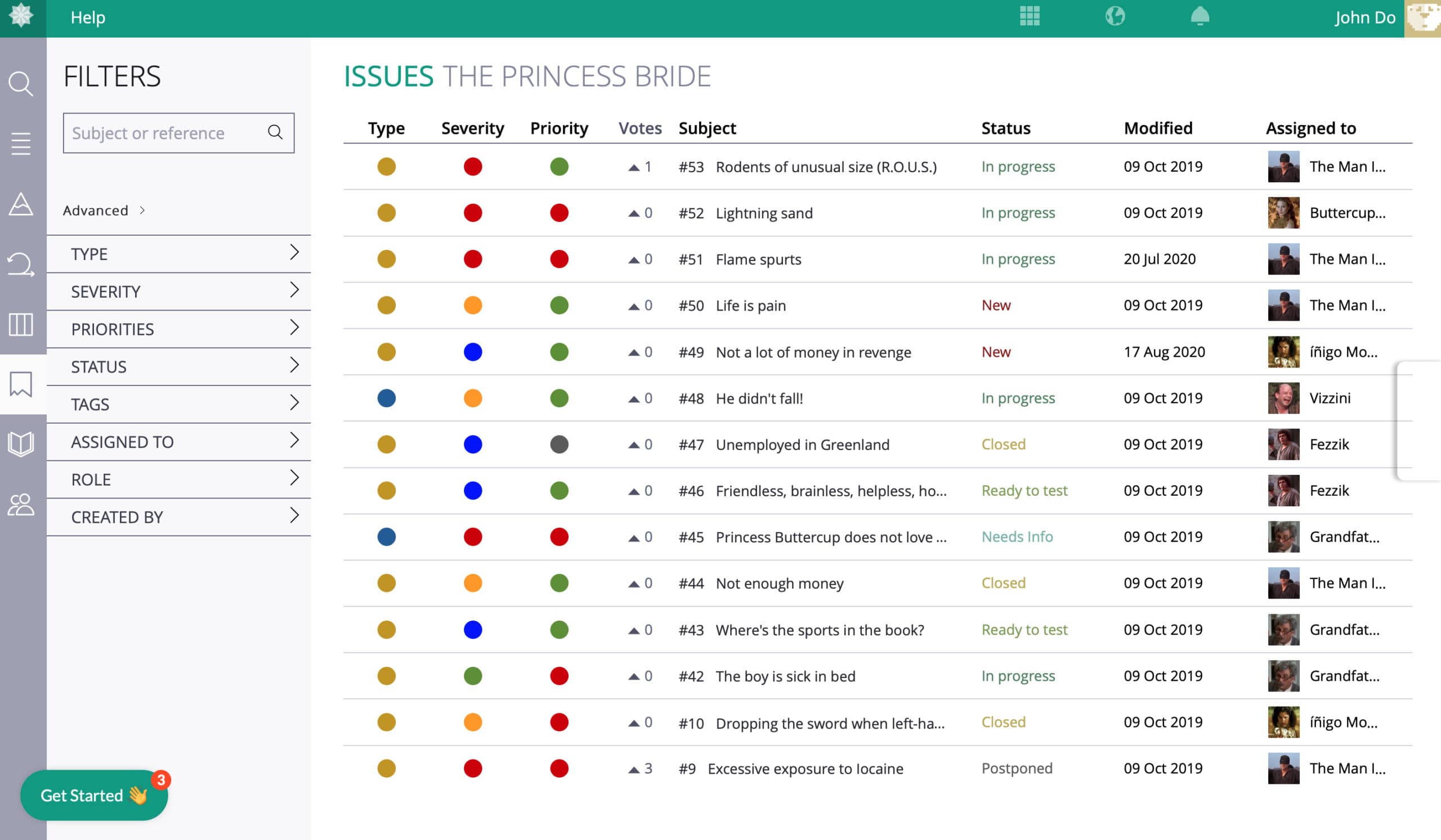


Рисунок 1.3 – Страница работы с проблемами в программном средстве Taiga.

Каждое из рассмотренных решений обладает своими сильными и слабыми сторонами. Jira подходит для крупных команд и сложных проектов, требующих детализированного управления задачами и отчетности. Confluence обеспечивает эффективное управление знаниями и документацией, особенно в связке с Jira. Taiga, с другой стороны, является отличным выбором для небольших команд и стартапов, которые нуждаются в простом и доступном инструменте для управления проектами по методологии Agile.

* 1. **Выбор и обоснование языков программирования, фреймворков, библиотек, СУБД для разработки программного средства**

Для разработки программного средства автоматизации бизнес-процессов информационного отдела организации выбран стек технологий, включающий Python, Django, JavaScript, ReactJS, Bootstrap, PostgreSQL, DBeaver и использование среды разработки Visual Studio Code (VSCode). Рассмотрим обоснование выбора каждого из этих компонентов, исходя из их преимуществ, соответствия поставленным задачам и широкого применения в современном программировании.

Python – это один из самых популярных языков программирования в мире, известный своей простотой, гибкостью и широкими возможностями для разработки различных типов приложений. В рамках данного проекта Python был выбран для реализации серверной части программного средства автоматизации бизнес-процессов информационного отдела. Рассмотрим подробнее причины этого выбора, его преимущества и недостатки, а также особенности работы с языком в контексте проекта.[14]

Python является высокоуровневым интерпретируемым языком программирования с поддержкой нескольких парадигм, таких как объектно-ориентированное программирование (ООП), функциональное программирование и императивное программирование. Его популярность обусловлена легкостью освоения благодаря читаемому синтаксису, который делает код понятным даже для начинающих разработчиков. Это особенно важно для командной разработки и поддержки проектов, так как снижает порог вхождения для новых участников команды и ускоряет процесс разработки.

Преимущества Python:

1. Простота и читаемость кода. Синтаксис Python интуитивно понятен и минималистичен, что делает код легко читаемым и сопровождаемым. Это позволяет разработчикам сосредоточиться на решении задач, а не на технических деталях. Простота языка ускоряет написание программ, что важно при разработке комплексных бизнес-систем, требующих быстрой адаптации и изменений.
2. Широкий набор библиотек и фреймворков. Одной из ключевых причин выбора Python является его мощная экосистема библиотек, которые упрощают разработку и расширяют возможности языка. Для веб-разработки активно используются такие фреймворки, как Django и Flask, которые позволяют быстро создавать веб-приложения и API. В данном проекте Python будет использоваться для обработки запросов от фронтенд-части (написанной на JavaScript с использованием React) и взаимодействия с базой данных PostgreSQL через ORM-библиотеки, такие как SQLAlchemy.
3. Универсальность. Python может использоваться для различных задач: от создания веб-приложений до автоматизации рабочих процессов и анализа данных. Это делает его универсальным инструментом для разработки бизнес-решений, где необходимо решать широкий спектр задач, связанных с автоматизацией и обработкой данных. В контексте данного проекта Python будет отвечать за обработку заявок пользователей, управление календарем видеоконференций и генерацию отчетов.
4. Активное сообщество и документация. Python имеет огромное сообщество разработчиков, что обеспечивает доступ к множеству готовых решений и поддержке. Это помогает в случае возникновения проблем и ускоряет процесс разработки, так как множество задач уже решено и задокументировано.
5. Интеграция с другими системами. Python легко интегрируется с различными базами данных, системами управления доступами и другими внешними сервисами. В данном проекте Python будет взаимодействовать с PostgreSQL для работы с данными, а также обеспечивать интеграцию с электронной почтой и календарями для автоматизации планирования и уведомлений.

Недостатки Python:

1. Скорость выполнения. Python является интерпретируемым языком, что делает его медленнее по сравнению с компилируемыми языками, такими как C++ или Java. Для высоконагруженных приложений, требующих максимальной производительности, Python может быть недостаточно быстрым. Однако для большинства задач, связанных с автоматизацией бизнес-процессов, производительности Python вполне достаточно.
2. Ограниченная многозадачность. Одной из известных проблем Python является наличие глобальной блокировки интерпретатора (Global Interpreter Lock, GIL), которая ограничивает одновременное выполнение потоков. Это может стать узким местом в многопоточных приложениях, где требуется параллельное выполнение задач. Тем не менее, для веб-приложений и API, использующих асинхронную обработку запросов, это ограничение не критично.
3. Необходимость использования сторонних библиотек. Несмотря на богатую экосистему, разработчикам часто приходится полагаться на сторонние библиотеки для реализации определенных функций. Это может создавать сложности с совместимостью при обновлениях или возникновении ошибок в этих библиотеках. Во избежание проблем важно тщательно выбирать библиотеки, активно поддерживаемые сообществом.
4. Ресурсоемкость. Python потребляет больше памяти и ресурсов по сравнению с некоторыми другими языками, особенно при работе с большими объемами данных или сложными операциями. Это может стать проблемой для приложений, развернутых на серверах с ограниченными ресурсами, однако современные серверные решения позволяют минимизировать это влияние.

Особенности использования Python в данном проекте: Python будет использоваться для разработки серверной части системы, которая будет обеспечивать обработку пользовательских запросов, взаимодействие с базой данных и генерацию уведомлений. Серверная часть приложения будет реализована с использованием Flask или Django, что позволит быстро развернуть REST API для связи с фронтенд-частью. Python будет взаимодействовать с PostgreSQL для хранения данных о задачах, видеоконференциях, заявках на доступы и проблемах пользователей. Интеграция с системами электронной почты позволит автоматизировать процесс уведомлений и рассылки приглашений на видеоконференции.

Важной частью проекта станет обработка данных и их представление в виде отчетов. Для этого Python предоставляет множество инструментов, таких как Pandas для анализа данных и Matplotlib для визуализации. Эти библиотеки позволяют гибко обрабатывать и представлять данные, что важно для контроля и оценки эффективности работы информационного отдела.

Django – это высокоуровневый веб-фреймворк на языке Python, который позволяет быстро и эффективно создавать сложные веб-приложения. Django был разработан для того, чтобы автоматизировать рутинные задачи веб-разработки, предоставляя разработчикам инструменты для создания надежных, безопасных и масштабируемых приложений. В рамках данного проекта Django выбран для реализации серверной части системы автоматизации бизнес-процессов информационного отдела. Рассмотрим подробнее особенности Django, его преимущества и недостатки, а также его роль в проекте.

Django является фреймворком с принципом "batteries included" (всё включено), что означает, что он поставляется с набором готовых решений для наиболее распространённых задач веб-разработки. Это позволяет разработчикам сосредоточиться на бизнес-логике приложения, а не на создании базовой инфраструктуры. Django отлично подходит для создания сложных корпоративных систем, таких как автоматизация бизнес-процессов, благодаря своей гибкости и расширяемости.

Преимущества Django:

1. Быстрая разработка. Django изначально разрабатывался для ускорения процесса создания веб-приложений. Он предоставляет мощные встроенные инструменты для работы с базами данных, управления аутентификацией пользователей, администрирования и многого другого. Это значительно ускоряет разработку серверной части приложения. В рамках данного проекта это позволит быстро реализовать функционал обработки заявок, управления пользователями и интеграции с базой данных PostgreSQL.
2. Модульная архитектура. Django поддерживает модель MVC (Model-View-Controller), что делает код структурированным и легко сопровождаемым. Это важно при разработке сложных систем, таких как автоматизация бизнес-процессов, где требуется четкое разделение логики данных, представления и контроля над процессами. В данном проекте модель данных будет отвечать за управление информацией о задачах, заявках, видеоконференциях и пользователях, в то время как контроллеры будут обрабатывать пользовательские запросы и взаимодействие с API.
3. Безопасность. Безопасность является одним из приоритетов Django. Он включает встроенные механизмы защиты от наиболее распространенных веб-угроз, таких как SQL-инъекции, межсайтовый скриптинг (XSS), атаки с подделкой межсайтовых запросов (CSRF) и другие. В данном проекте, где требуется обработка конфиденциальной информации, такой как заявки на доступ и запросы на устранение проблем, это обеспечивает высокий уровень защиты данных.
4. Встроенная панель администратора. Django предоставляет мощную встроенную панель администратора, которая позволяет управлять данными и пользователями через удобный веб-интерфейс. Это значительно упрощает процесс управления системой для сотрудников отдела, так как администраторы смогут контролировать доступы, обрабатывать заявки и просматривать отчеты без необходимости разрабатывать отдельные инструменты для администрирования.
5. Масштабируемость и производительность. Django поддерживает масштабируемость и позволяет работать с большими объемами данных. В проекте, направленном на автоматизацию бизнес-процессов, это важно, так как с ростом организации и увеличением объема заявок система должна быть способна выдерживать возросшую нагрузку. Django также легко интегрируется с такими решениями, как Docker и Kubernetes, что упрощает развертывание и управление масштабируемыми приложениями.
6. Совместимость с различными базами данных. Django поддерживает работу с множеством реляционных баз данных, включая PostgreSQL, MySQL и SQLite. В данном проекте будет использоваться PostgreSQL, которая обеспечивает надежную работу с большими объемами данных и поддерживает сложные транзакции. Django предоставляет ORM (Object-Relational Mapping), что упрощает работу с базой данных, позволяя писать запросы на уровне Python без необходимости напрямую использовать SQL.

Недостатки Django:

1. Монотонность и шаблонность. Django предоставляет очень строгую структуру и набор инструментов, что иногда ограничивает гибкость разработчиков. В рамках небольших проектов, где требуется больше свободы в реализации, использование Django может показаться излишним, так как фреймворк навязывает свои подходы к организации кода и процессов. В больших системах это, однако, скорее преимущество, так как строгая структура упрощает поддержание проекта.
2. Производительность для высоконагруженных систем. Хотя Django хорошо масштабируется, его производительность может быть ниже, чем у некоторых более легковесных фреймворков, таких как Flask, если не оптимизировать приложение. В проектах с высокими требованиями к производительности может потребоваться дополнительная настройка и использование дополнительных инструментов для повышения скорости работы.
3. Крутая кривая обучения для новичков. Для начинающих разработчиков Django может показаться сложным из-за большого количества встроенных инструментов и компонентов. Однако в контексте данного проекта, где требуется высокая функциональность, это компенсируется тем, что фреймворк предоставляет готовые решения для большинства задач.

Django будет использоваться для реализации серверной части системы, обеспечивая взаимодействие между клиентской частью, написанной на JavaScript с использованием React, и базой данных PostgreSQL. Django будет отвечать за обработку заявок на доступы, видеоконференции и запросы на устранение проблем, а также за управление пользователями и правами доступа.

С помощью встроенной ORM Django обеспечит удобное взаимодействие с PostgreSQL, автоматизируя процесс создания, чтения, обновления и удаления данных. Это позволит быстро обрабатывать запросы пользователей и генерировать отчеты о выполненных задачах и заявках.

Кроме того, Django обеспечит безопасную работу системы благодаря встроенным механизмам защиты от распространенных атак и уязвимостей. Панель администратора Django предоставит сотрудникам информационного отдела простой инструмент для управления системой, позволяя контролировать заявки, задачи и доступы.

Также Django легко интегрируется с технологиями контейнеризации, такими как Docker, что позволит развернуть приложение в контейнерах, обеспечивая гибкость и простоту управления окружением разработки, тестирования и продакшена.

JavaScript – это основной язык программирования, используемый для разработки клиентской части (frontend) веб-приложений. В рамках данного проекта JavaScript выбран для создания интерактивного и динамичного пользовательского интерфейса, что позволяет эффективно взаимодействовать с сервером, обрабатывать запросы пользователей и обновлять данные на страницах без перезагрузки. Рассмотрим подробнее особенности JavaScript, его преимущества и недостатки, а также причины выбора для автоматизации бизнес-процессов информационного отдела.

JavaScript – это высокоуровневый, динамически типизированный язык программирования, изначально разработанный для работы с веб-страницами в браузере. Со временем его применение значительно расширилось, и теперь JavaScript используется не только на стороне клиента, но и на сервере (благодаря таким средам, как Node.js). В данном проекте JavaScript будет применяться для создания интерактивных элементов пользовательского интерфейса и взаимодействия с серверной частью, написанной на Python, через API.

Преимущества JavaScript:

1. Асинхронность и скорость. Одним из ключевых преимуществ JavaScript является его способность выполнять асинхронные операции, такие как отправка запросов к серверу без блокировки интерфейса пользователя. Это особенно важно для веб-приложений, где нужно быстро обрабатывать запросы и обновлять интерфейс. В контексте данного проекта это позволит пользователям отправлять заявки на доступы, видеоконференции и запросы на устранение проблем без необходимости перезагрузки страницы.
2. Интерактивность и динамичность. JavaScript позволяет создавать динамичные веб-страницы с возможностью изменения содержимого без полной перезагрузки. Это позволяет реализовать удобные формы для отправки заявок и планирования видеоконференций, а также предоставлять пользователям актуальную информацию о состоянии задач и сервисов в режиме реального времени.
3. Кроссбраузерная поддержка. JavaScript поддерживается всеми современными браузерами, что делает его идеальным выбором для создания веб-приложений, которые будут работать на различных устройствах и платформах. Это важно в рамках проекта, так как пользователи могут работать с системой как с настольных компьютеров, так и с мобильных устройств.
4. Компонентная архитектура с React. В данном проекте JavaScript будет использоваться в связке с библиотекой React, которая предоставляет удобную компонентную модель для создания пользовательских интерфейсов. Это позволяет легко управлять состоянием приложения и переиспользовать компоненты, что упрощает разработку и поддержку проекта. React позволяет создавать адаптивные интерфейсы для работы с календарем видеоконференций, страницей с сервисами и заявками на доступы и поддержку.
5. Широкая экосистема. JavaScript обладает огромной экосистемой библиотек и фреймворков, которые могут значительно упростить разработку. Например, использование библиотек для работы с формами, валидацией данных и маршрутизацией помогает ускорить процесс создания интерфейсов. В проекте могут быть использованы такие библиотеки, как Axios для отправки HTTP-запросов или React Router для управления маршрутизацией внутри приложения.[7]

Недостатки JavaScript:

1. Отсутствие строгой типизации. JavaScript является динамически типизированным языком, что означает отсутствие контроля типов на этапе компиляции. Это может привести к ошибкам, которые выявляются только на этапе выполнения программы, что увеличивает риск непредсказуемого поведения кода. Для решения этой проблемы в проекте можно использовать TypeScript, который добавляет статическую типизацию поверх JavaScript и улучшает качество кода.[12]
2. Проблемы с производительностью. При работе с большими объемами данных или сложными операциями JavaScript может замедлять работу интерфейса, особенно если активно используется манипуляция с DOM (Document Object Model). В данном проекте это можно минимизировать за счет использования виртуального DOM, предоставляемого React, который позволяет обновлять только необходимые части интерфейса, а не весь документ целиком.[13]
3. Разнообразие реализаций в браузерах. Несмотря на стандартизацию JavaScript, его реализация в различных браузерах может немного отличаться, что требует дополнительного тестирования и корректировки кода для обеспечения кроссбраузерной совместимости. Однако современные библиотеки и инструменты, такие как Babel, помогают решать эту проблему, обеспечивая поддержку старых версий браузеров.
4. Сложность отладки. В отличие от языков с компиляцией, таких как C++ или Java, JavaScript выполняется в браузере, и отладка может быть сложнее из-за динамической природы языка. Для решения этой проблемы разработчики могут использовать встроенные инструменты отладки браузеров, такие как Chrome DevTools, которые позволяют инспектировать и отслеживать выполнение кода в реальном времени.

Особенности использования JavaScript в данном проекте: JavaScript будет играть ключевую роль в создании динамичного интерфейса для взаимодействия пользователей с системой. Используя React, приложение сможет обновлять данные о задачах, заявках и видеоконференциях в реальном времени, обеспечивая удобный и отзывчивый интерфейс. Асинхронная работа с сервером позволит пользователям отправлять запросы и получать данные без необходимости перезагрузки страницы, что улучшит общую производительность системы и повысит удовлетворенность пользователей.

Также, JavaScript обеспечит интеграцию с внешними сервисами, такими как календарь для планирования видеоконференций, и позволит реализовать уведомления о предстоящих событиях и изменениях в статусах заявок. Для отправки HTTP-запросов к серверу будет использована библиотека Axios, а для управления маршрутизацией в приложении — React Router, что упростит навигацию между страницами и модулями системы.[8]

React – это библиотека JavaScript для построения пользовательских интерфейсов, разработанная компанией Facebook.

Основные причины выбора React:

1. Компонентная структура, что позволяет легко переиспользовать код и улучшать поддерживаемость приложения.
2. Быстрая реакция на изменения данных за счет виртуального DOM, что повышает производительность интерфейса.
3. Широкая поддержка сообществом и наличие множества готовых решений для интеграции с другими инструментами.

Использование React оправдано при разработке приложений, где требуется активное взаимодействие с пользователем и постоянное обновление данных на экране.

Недостатки библиотеки:

1. Сложность обучения: несмотря на популярность React, для новичков изучение его концепций, таких как JSX, Virtual DOM, хуки и управление состоянием, может оказаться сложным и потребовать значительного времени.
2. Быстрое развитие и изменения: React и его экосистема активно развиваются, и новые версии могут требовать пересмотра уже написанного кода. Частые изменения и обновления могут создать трудности в поддержке проекта, если команда разработчиков не успевает следить за нововведениями.
3. Требования к настройке: хотя React сам по себе легковесный, для полного рабочего приложения потребуется настроить множество дополнительных инструментов и библиотек (например, Redux для управления состоянием, React Router для маршрутизации), что увеличивает сложность конфигурации.

Bootstrap – это популярный CSS-фреймворк для разработки адаптивных и современных веб-интерфейсов.

Основные преимущества:

1. Готовые компоненты и стили, что ускоряет разработку и позволяет сосредоточиться на функциональности, а не на дизайне.
2. Поддержка адаптивной верстки, что делает веб-приложения доступными на различных устройствах, включая мобильные.
3. Совместимость с React и JavaScript, что упрощает интеграцию.

Основные недостатки:

1. Однообразие дизайна: Использование стандартных компонентов Bootstrap может привести к однотипности веб-приложений, так как многие сайты используют одни и те же шаблоны и стили. Для уникального дизайна часто требуется значительная кастомизация CSS.
2. Тяжелый CSS-код: По умолчанию Bootstrap включает много неиспользуемого CSS-кода, что может замедлять загрузку страниц. Разработчикам часто приходится вручную удалять ненужные стили или применять специальные инструменты для оптимизации.
3. Ограниченная гибкость: хотя Bootstrap подходит для базового дизайна, при создании сложных пользовательских интерфейсов он может не обладать достаточной гибкостью, требуя больше ручного редактирования стилей.

PostgreSQL – это мощная объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом, которая широко используется для хранения и обработки больших объемов данных в сложных корпоративных приложениях. В рамках данного проекта PostgreSQL выбран в качестве основной базы данных для хранения информации о задачах, заявках на доступ, видеоконференциях и проблемах пользователей. Рассмотрим подробнее причины выбора PostgreSQL, её преимущества и недостатки, а также её роль в системе автоматизации бизнес-процессов информационного отдела. Это одна из самых стабильных и гибких СУБД, которая поддерживает множество расширенных функций, таких как транзакции, индексы, полнотекстовый поиск, хранение и обработка больших данных. Благодаря своей расширяемости и надежности PostgreSQL подходит для сложных бизнес-приложений, где важно не только эффективно управлять данными, но и обеспечивать их целостность, безопасность и производительность.

Преимущества PostgreSQL:

1. Надежность и целостность данных. PostgreSQL полностью соответствует стандарту ACID (Atomicity, Consistency, Isolation, Durability), что гарантирует надежность выполнения транзакций и целостность данных. Это особенно важно в данном проекте, где система будет управлять заявками и данными о видеоконференциях, и требуется, чтобы данные оставались неизменными и корректными даже при сбоях или отказах.
2. Поддержка сложных запросов и индексов. PostgreSQL поддерживает сложные SQL-запросы, включая вложенные запросы, объединения таблиц и агрегатные функции. Это позволяет реализовать сложную логику обработки данных, необходимую для генерации отчетов и аналитики по задачам и заявкам. Кроме того, PostgreSQL предоставляет возможность создания индексов для ускорения выполнения запросов, что позволяет поддерживать высокую производительность даже при работе с большими объемами данных.
3. Масштабируемость. PostgreSQL хорошо масштабируется как по вертикали, так и по горизонтали, что делает её подходящей для систем с увеличивающимися объемами данных и числами пользователей. В данном проекте это особенно важно, поскольку с ростом компании и увеличением количества задач и заявок система должна оставаться производительной и эффективно обрабатывать все запросы.
4. Поддержка JSON и NoSQL-функций. Хотя PostgreSQL является реляционной СУБД, она также поддерживает хранение и обработку данных в формате JSON. Это позволяет использовать её как гибридное решение, объединяя реляционную структуру данных с динамическими JSON-документами. Это может быть полезным в случае, если в системе потребуется хранить нестандартные или полуструктурированные данные, например, для хранения метаданных заявок или результатов видеоконференций.[9]
5. Расширяемость. PostgreSQL поддерживает расширения, которые позволяют добавлять новые функции без изменения исходного кода. Одним из таких популярных расширений является PostGIS, который добавляет поддержку работы с географическими данными. В данном проекте расширяемость PostgreSQL позволяет легко адаптировать систему к новым требованиям, таким как интеграция с внешними сервисами или добавление специфических функций обработки данных.
6. Открытый исходный код. PostgreSQL является полностью бесплатной и открытой системой, что снижает затраты на разработку и развертывание системы. Она активно развивается сообществом, что обеспечивает её актуальность и безопасность.[2]

Недостатки PostgreSQL:

1. Сложность настройки и администрирования. Несмотря на свои возможности, PostgreSQL требует глубоких знаний для настройки и оптимизации производительности. Например, при работе с большими объемами данных необходимо настраивать параметры кэширования, индексации и разбиения таблиц на партиции для повышения скорости обработки запросов. В рамках данного проекта могут потребоваться дополнительные усилия для правильной настройки системы, чтобы обеспечить её максимальную производительность и надежность.
2. Требовательность к ресурсам. PostgreSQL требует значительных ресурсов для работы с большими объемами данных, особенно при выполнении сложных аналитических запросов и обработке транзакций. Это может стать проблемой на серверах с ограниченными вычислительными мощностями.
3. Медлительность при выполнении простых запросов. В некоторых случаях PostgreSQL может уступать другим СУБД, таким как MySQL, при выполнении простых запросов или операций на небольших объемах данных. Это связано с тем, что PostgreSQL изначально ориентирован на более сложные транзакции и обработку данных. Для оптимизации производительности можно использовать индексы и кэширование.

В рамках системы автоматизации бизнес-процессов информационного отдела PostgreSQL будет использоваться для хранения данных о задачах, заявках на доступ, видеоконференциях и проблемах пользователей. Система будет обрабатывать и хранить большие объемы информации, и PostgreSQL обеспечит высокую надежность, целостность и производительность при работе с этими данными.

Одной из ключевых задач PostgreSQL будет обеспечение быстрого и надежного выполнения запросов на обработку заявок и видеоконференций. Пользователи должны иметь возможность отправлять заявки и отслеживать их статус в режиме реального времени, а система — мгновенно обновлять информацию. Для этого будут использоваться индексы и оптимизированные SQL-запросы, которые позволят ускорить обработку данных.

PostgreSQL также обеспечит безопасное хранение данных и защиту от потерь. Встроенная поддержка транзакций гарантирует, что все изменения данных будут сохраняться в системе корректно, даже в случае сбоя. Это особенно важно для обработки заявок, где требуется гарантировать целостность данных и их корректную обработку.

Кроме того, PostgreSQL будет интегрирована с серверной частью, реализованной на Django. Django использует встроенную ORM (Object-Relational Mapping), которая позволяет взаимодействовать с базой данных PostgreSQL на уровне Python-кода, что упрощает работу с данными и ускоряет процесс разработки.

VSCode – это современная легкая среда разработки, поддерживающая множество языков программирования и плагинов.

Преимущества:

1. Расширяемость за счет многочисленных плагинов для поддержки различных языков и фреймворков.
2. Поддержка интеграции с системами контроля версий, такими как Git.
3. Легкость в настройке и кроссплатформенность.

VSCode выбран как основная среда разработки из-за его высокой гибкости и удобства для создания проектов любого масштаба, включая проекты с использованием нескольких языков и инструментов.

Недостатки работы в IDE VSCode:

1. Несмотря на то, что VSCode является легковесной средой разработки, с установкой множества расширений он может начать потреблять значительное количество оперативной памяти, что замедляет работу на менее мощных компьютерах.
2. Основная функциональность VSCode может быть недостаточной для некоторых специфических задач, поэтому разработчики сильно зависят от сторонних плагинов, которые могут иметь проблемы с совместимостью или качеством работы.
3. По сравнению с полноценными IDE, такими как JetBrains IntelliJ IDEA или PyCharm, VSCode может не иметь некоторых удобных функций для управления проектами и отладки, что может потребовать дополнительных инструментов.

Таким образом, выбранные языки программирования и технологии обеспечивают максимальную эффективность и производительность при разработке программного средства для автоматизации бизнес-процессов в информационном отделе организации.

## **Постановка задачи на дипломное проектирование**

Целью проекта является создание программного средства для автоматизации бизнес-процессов информационного отдела, которое будет обеспечивать управление задачами, документооборотом, внутренними коммуникациями и предоставлять специализированные инструменты для планирования работы и решения проблем сотрудников. Система должна упростить рутинные операции, улучшить контроль над выполнением задач, а также обеспечить более прозрачное взаимодействие сотрудников отдела.

Назначение разработки заключается в создании платформы, которая обеспечит автоматизацию основных процессов информационного отдела. Это включает сбор и обработку данных, управление задачами и проектами, а также поддержку взаимодействия между сотрудниками через инструменты планирования и контроля.

Основные функции системы:

1. Календарь для планирования видеоконференций. В системе будет интегрирован функционал календаря, позволяющий планировать видеоконференции между сотрудниками и внешними участниками. Календарь должен предоставлять возможность бронирования времени для встреч, интеграции с почтовыми системами для отправки приглашений участникам, а также напоминания о предстоящих мероприятиях.
2. Отправка заявок на получение доступов в системы. Система позволит пользователям отправлять заявки на предоставление или расширение доступа к различным корпоративным системам. Эти заявки будут автоматически отправляться ответственным лицам для обработки. Пользователь сможет отслеживать статус заявки и получать уведомления о её рассмотрении и одобрении.
3. Отправка заявок на решение проблем. В случае возникновения проблем у сотрудников при работе с системами или оборудованием, они смогут быстро отправить заявку в службу поддержки через встроенный интерфейс системы. Программа будет фиксировать все заявки и направлять их соответствующим отделам для устранения. Также будет реализована возможность уведомлений пользователя о ходе решения проблемы.
4. Страница с доступными сервисами. Для удобства сотрудников в системе будет реализована страница, где будут перечислены все доступные корпоративные сервисы и системы. Пользователи смогут видеть, какие из сервисов доступны, и запрашивать доступ к ним через интегрированную систему заявок.

Входные данные системы включают информацию о задачах, заявках на доступы и проблемы, данные о видеоконференциях, а также информацию о сотрудниках и системах организации. Эти данные будут поступать от пользователей системы через соответствующие интерфейсы и формы.

Выходные данные – это отчеты о выполнении задач, напоминания о запланированных мероприятиях, уведомления о статусе заявок, а также список доступных сервисов и их текущий статус.

Требования к временным характеристикам: Система должна обеспечивать работу в режиме реального времени, чтобы пользователи могли моментально получать доступ к информации о задачах, заявках и видеоконференциях. Время отклика при работе с календарем и заявками должно быть минимальным – не более 1-2 секунд.

Программа должна быть устойчива к высоким нагрузкам и обеспечивать бесперебойную работу в условиях одновременного использования большим количеством пользователей. Необходимо предусмотреть регулярное резервное копирование данных для предотвращения их потери.

Среда эксплуатации: Система должна быть веб-приложением, поддерживающим работу в популярных браузерах и совместимым с различными операционными системами (Windows, macOS, Linux). Она должна быть доступна с компьютеров и мобильных устройств, обеспечивая гибкость использования для сотрудников организации.

Информационная и программная совместимость: Программа должна поддерживать интеграцию с почтовыми системами для отправки уведомлений и приглашений, а также возможность подключения к корпоративным системам управления доступами и поддержкой пользователей для автоматизации обработки заявок.

Эти требования и задачи, поставленные перед системой, направлены на повышение эффективности работы информационного отдела и улучшение взаимодействия сотрудников, что сделает процесс управления задачами, доступами и видеоконференциями более удобным и прозрачным.

1. **МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 
   1. **Разработка информационной модели базы данных**

Диаграмма IDEF1X наглядно представляет ключевые сущности и связи в системе, такие как Пользователи, Заявки, Задачи, Проекты, Видеоконференции и другие таблицы, отвечающие за управление заявками, доступом и задачами.

Таблица Users является центральной сущностью и содержит информацию обо всех пользователях системы, включая их роли, контактные данные и статус. Связана с таблицей Roles через таблицу UserRoles, что позволяет одному пользователю иметь несколько ролей и одному типу роли принадлежать множеству пользователей. Users также связана с таблицами AccessRequests, IssueRequests, Tasks, Conferences, ConferenceParticipants, TaskComments, ProjectFiles, Notifications и AuditLogs, обеспечивая взаимодействие пользователя с различными функциями системы.

Таблица Roles хранит типы ролей пользователей (например, администратор, сотрудник) и связана с Users через промежуточную таблицу UserRoles, где ID пользователя и ID роли образуют связь многие-ко-многим.

Таблица AccessRequests хранит данные о заявках пользователей на доступ к корпоративным системам. Она связана с Users (по ID пользователя), который подал заявку, и AuditLogs, которая хранит историю изменений статусов заявки.

Таблица IssueRequests включает данные о проблемах, о которых сообщают пользователи, и связана с таблицей Users (по ID пользователя), а также AuditLogs, что позволяет отслеживать историю и статус решения проблемы.

Таблица Tasks предназначена для управления задачами и связана с Users (по ответственному сотруднику) и Projects, поскольку каждая задача принадлежит определенному проекту. Tasks также связана с TaskComments (комментарии к задачам) и AuditLogs, где фиксируются изменения статусов задачи.

Таблица Projects объединяет задачи в рамках одного проекта и связана с Users (по ID менеджера проекта), Tasks (по ID проекта), ProjectFiles (по ID проекта для хранения связанных файлов) и AuditLogs для записи истории изменений в проекте.

Таблица Conferences хранит информацию о видеоконференциях и связана с Users (по организатору) и ConferenceParticipants, обеспечивая возможность добавления множества участников к одной конференции.

Таблица ConferenceParticipants связывает Users и Conferences для реализации связи многие-ко-многим, отображая список участников каждой конференции.[1]

Таблица Services содержит перечень доступных сервисов для пользователей и связана с ServiceLogs, которая хранит историю доступности сервиса, отражая изменения статуса сервиса с течением времени.

Таблица ServiceLogs хранит логи работы сервисов и связана с Services, предоставляя историю изменений состояния сервисов.

Таблица TaskComments хранит комментарии пользователей к задачам и связана с Tasks (по ID задачи) и Users (по ID пользователя, добавившего комментарий), что позволяет отслеживать, кто добавил комментарий к конкретной задаче.

Таблица ProjectFiles хранит файлы, связанные с проектами, и связана с Projects (по ID проекта) и Users (по ID пользователя, загрузившего файл), позволяя организовать управление документами в рамках проекта.

Таблица Notifications используется для хранения уведомлений, отправленных пользователям. Она связана с Users, которые получают уведомления, и AuditLogs, в которой фиксируются изменения статуса уведомлений.

Таблица AuditLogs служит для записи действий пользователей и системных изменений и связана с Users (по ID пользователя, совершившего действие), AccessRequests, IssueRequests, Tasks, Projects, Notifications, что обеспечивает логирование действий и истории всех ключевых операций в системе.

Диаграмма IDEF1X охватывает все ключевые элементы базы данных, обеспечивая структурированное хранение информации и позволяя эффективно реализовать функции автоматизации бизнес-процессов информационного отдела. Эти связи и ограничения помогают поддерживать целостность данных, их актуальность и упрощают доступ к нужной информации, что критически важно для успешного функционирования системы. Диаграмма IDEF1X показана на рисунке 2.1.

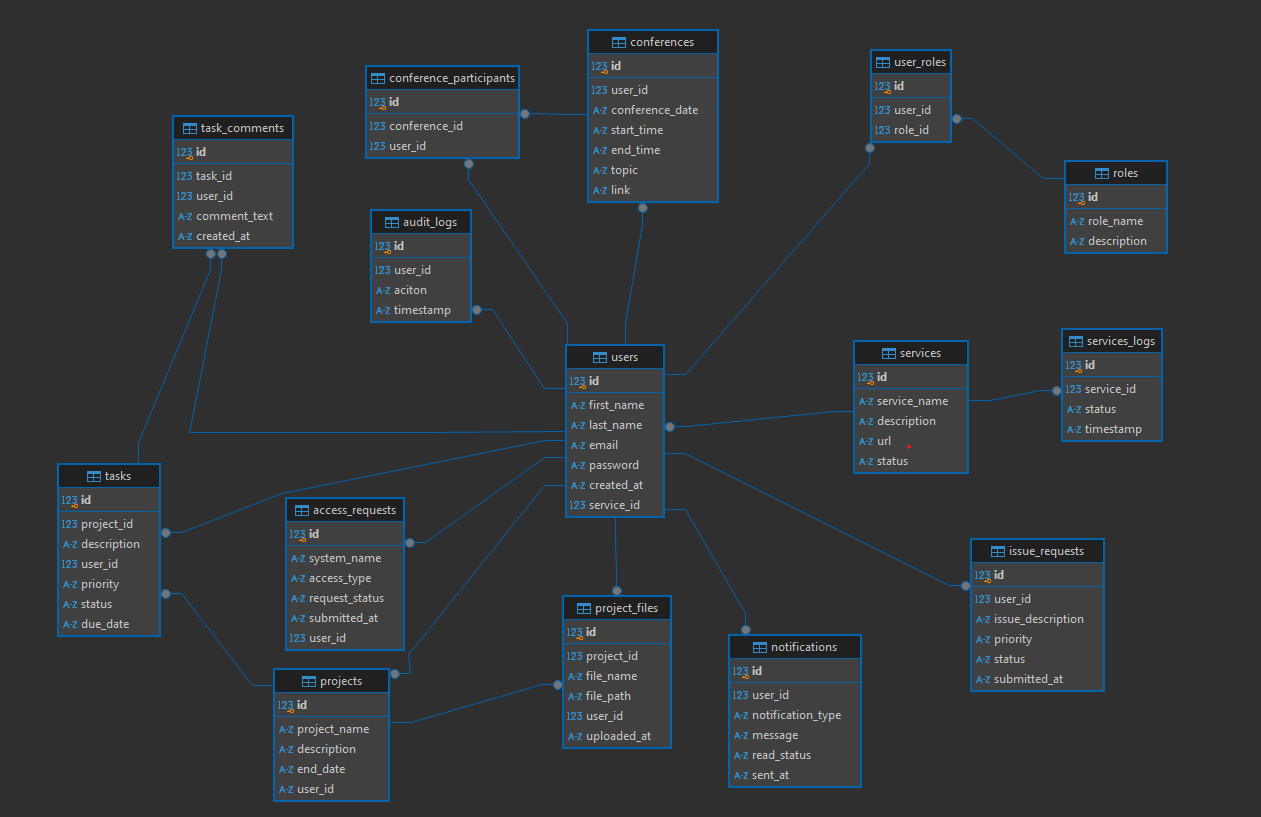


Рисунок 2.1 – Диаграмма IDEF1X

* 1. **Разработка функциональной модели**

Диаграмма A-0 (или верхнего уровня) для системы автоматизации бизнес-процессов информационного отдела представляет общий процесс, который система должна выполнять для достижения поставленных целей. В данной диаграмме отображена основная функция системы – автоматизация бизнес-процессов информационного отдела, и описаны основные входы, выходы, управляющие факторы и механизмы, обеспечивающие выполнение этой функции.

На основе этой верхнего уровня функции диаграммы A-0 строятся дальнейшие, более детализированные уровни диаграмм, где каждая конкретная функция (например, управление заявками или планирование видеоконференций) детализируется и разлагается на подфункции. Диаграмма верхнего уровня показана на рисунке 2.2.

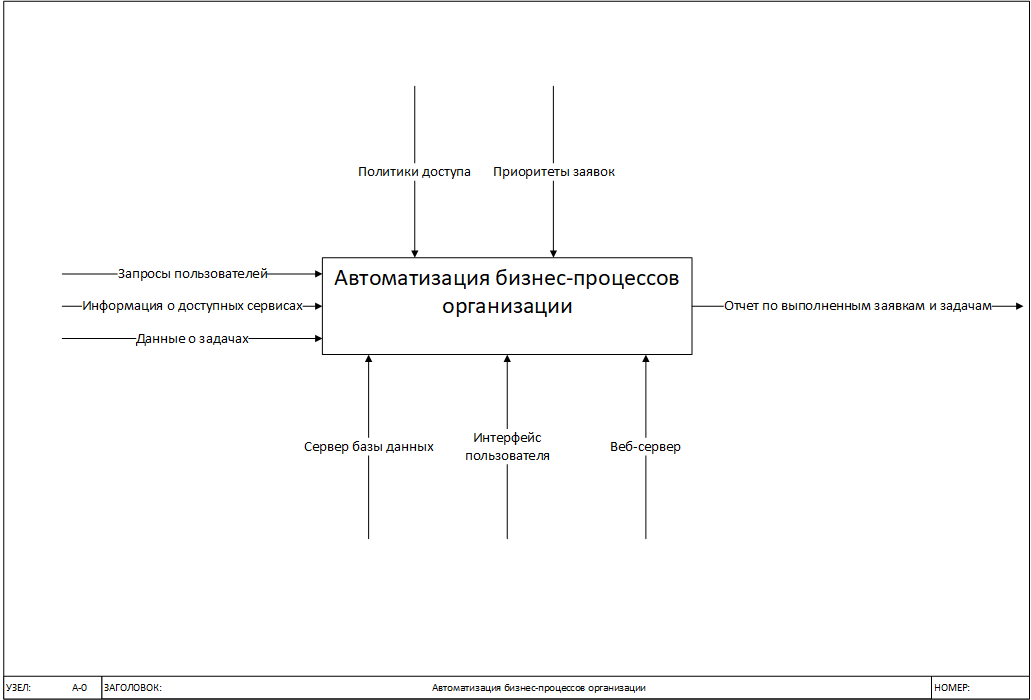


Рисунок 2.2 – Диаграмма *IDEF0*. Верхний уровень

Диаграмма *IDEF0* иллюстрирует основные процессы и функции системы автоматизации бизнес-процессов информационного отдела. Главная функция, обозначенная как A0: Автоматизация бизнес-процессов информационного отдела, отображает ключевые цели системы: обработка заявок, управление доступами, планирование видеоконференций и организация уведомлений для сотрудников. Система направлена на оптимизацию рабочего процесса отдела за счёт автоматизации рутинных задач и улучшения контроля над выполнением заявок и взаимодействием между сотрудниками.[3]

Функции и их описание:

1. Управление заявками на доступ Данная функция отвечает за обработку заявок на предоставление доступа к различным системам и сервисам, используемым сотрудниками. Входными данными для функции являются заявки от пользователей, содержащие информацию о запрашиваемом доступе. На основании политик доступа и правил безопасности заявки проходят проверку и могут быть либо одобрены, либо отклонены. Выходные данные – это обновленный статус каждой заявки, что позволяет сотрудникам оперативно узнавать о результатах их рассмотрения. Механизмами здесь выступают администраторы системы и база данных, где сохраняются все заявки и их статусы.
2. Планирование видеоконференций Эта функция предназначена для организации и планирования видеоконференций между сотрудниками. Входными данными являются запросы на встречи, которые создают сотрудники, указывая дату, время и участников. Управляющим фактором служит информация о графике доступности сотрудников, что позволяет системе избегать накладок во времени. Результатом функции становится подтверждение проведения конференции и напоминания для участников. Механизмы функции включают почтовую систему для отправки уведомлений, а также интерфейс приложения, через который пользователи планируют встречи.
3. Администрирование доступов и управление задачами. В этой функции происходит контроль над задачами и управлением доступом сотрудников к различным системам в рамках их должностных обязанностей. Входные данные включают информацию о задачах и пользователях, а управляющие факторы – политики безопасности и права доступа. Функция позволяет администраторам актуализировать статусы задач, а также управлять доступами, обеспечивая безопасное использование информационных ресурсов. База данных выступает в роли механизма, обеспечивающего хранение всех данных по задачам и доступам.
4. Обработка и отправка уведомлений Эта функция отвечает за формирование и отправку уведомлений для сотрудников и администраторов о статусах заявок и задач. Входные данные включают изменения в статусах заявок, задач и встреч, а управляющими факторами служат настройки уведомлений в системе. Результатом работы функции являются отправленные уведомления, которые позволяют сотрудникам своевременно получать информацию о важных изменениях и обновлениях. Механизмами являются почтовая система и сервер уведомлений.

В качестве управляющих факторов (Controls) для всех функций используются организационные политики, правила доступа и приоритеты обработки заявок. Эти факторы регулируют выполнение каждой функции, обеспечивая безопасность и соответствие внутренним стандартам организации.

Механизмами (Mechanisms) выступают физические и программные ресурсы, такие как сервер приложений, база данных, интерфейс приложения и почтовая система. Совместная работа механизмов позволяет системе обрабатывать запросы, управлять данными и эффективно уведомлять пользователей.

Диаграмма *IDEF0* отражает взаимодействие всех функций в рамках общей системы автоматизации, демонстрируя, как информация поступает, обрабатывается и передается для выполнения задач в информационном отделе. Автоматизация описанных процессов позволяет повысить продуктивность сотрудников и улучшить координацию в рамках отдела. Диаграмма *IDEF0* показана на рисунке 2.3

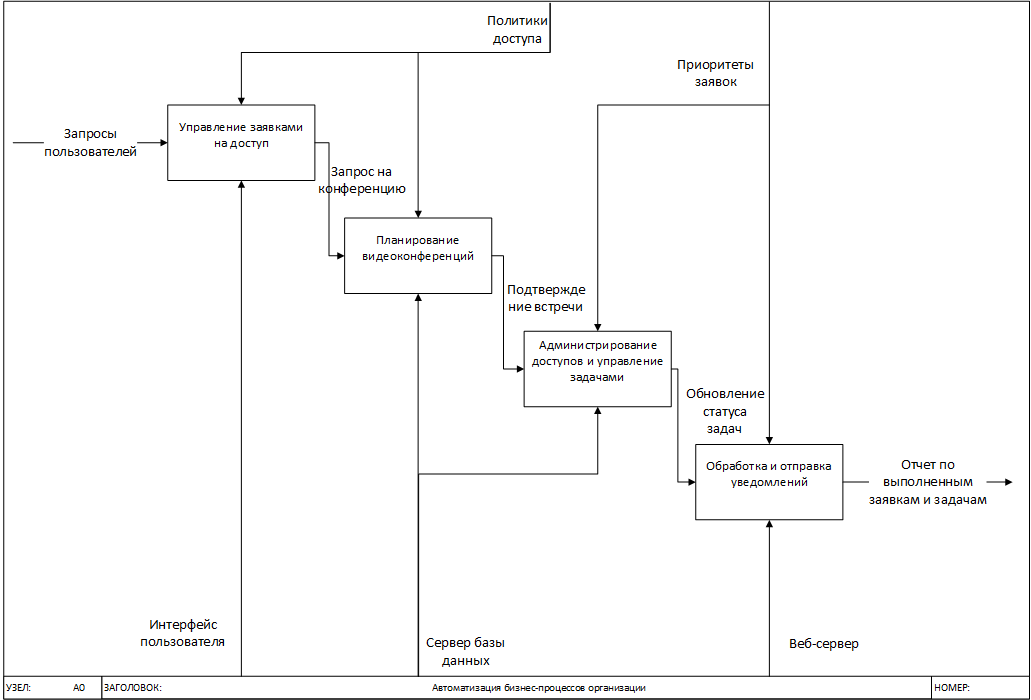


Рисунок 2.3 – Диаграмма декомпозиции

## **Разработка диаграммы развертывания**

Диаграмма развертывания системы автоматизации бизнес-процессов информационного отдела отражает физическую архитектуру системы и её компоненты, распределенные по различным узлам, включая серверы приложений, баз данных, веб-сервер и клиентские устройства. Она иллюстрирует взаимодействие этих компонентов для обеспечения эффективной и безопасной работы системы.

Основным узлом является сервер приложения, на котором работает серверная часть, написанная на Python с использованием Django. Здесь выполняется бизнес-логика и обработка запросов пользователей. Сервер приложения взаимодействует с базой данных PostgreSQL, которая расположена на отдельном сервере и хранит данные о пользователях, заявках, задачах, встречах и логах. Соединение между сервером приложения и сервером базы данных проходит по протоколу TCP/IP, что позволяет передавать запросы и поддерживать целостность данных. Для безопасности соединение может быть зашифровано с использованием SSL/TLS.

Веб-сервер, например, Nginx, обрабатывает все внешние запросы и выполняет роль обратного прокси для приложения, принимая HTTP/HTTPS-запросы от клиентских устройств и передавая их на сервер приложения. Веб-сервер также помогает распределять нагрузку, обеспечивая стабильное и безопасное подключение пользователей к системе.

Клиентские устройства, оснащенные веб-браузерами, позволяют пользователям взаимодействовать с системой через интерфейс, построенный на HTML, CSS и JavaScript с использованием React. Пользователи подключаются к веб-серверу, отправляя запросы на создание заявок, планирование встреч и управление задачами. Сервер приложения обрабатывает запросы, взаимодействует с базой данных для выполнения операций и отправляет результат обратно через веб-сервер.[6]

Для защиты данных на каждом уровне используются средства безопасности, такие как SSL/TLS-сертификаты и firewall, которые защищают серверы от несанкционированного доступа. Вся структура построена таким образом, чтобы обеспечить отказоустойчивость и удобство в обслуживании, а также гибкость при расширении или обновлении компонентов. Диаграмма развертывания показана на рисунке 2.4



Рисунок 2.4 – Диаграмма развертывания

## **Разработка диаграммы классов**

Диаграмма классов системы автоматизации бизнес-процессов информационного отдела представляет собой структурированное отображение основных классов, их атрибутов, методов и связей между ними, что обеспечивает организацию бизнес-логики и взаимодействия компонентов. Основными классами являются User, Role, AccessRequest, IssueRequest, Task, Project, Conference, Service, Notification и AuditLog. Эти классы представляют ключевые сущности системы и определяют её основные функции.

Класс User содержит информацию о пользователях, включая имя, фамилию, электронную почту, роль и дату регистрации. Связан с классом Role через ассоциацию многие-ко-многим, что позволяет каждому пользователю иметь одну или несколько ролей. User также связан с другими классами, такими как AccessRequest и IssueRequest для отслеживания заявок, Task для назначения задач и Conference для участия во встречах.

Класс AccessRequest представляет заявки на доступ к системам и связан с User (пользователь, отправивший заявку) и AuditLog для записи изменений статусов заявки. IssueRequest представляет заявки на устранение проблем и также связан с User и AuditLog. Оба класса содержат атрибуты для хранения типа запроса, статуса и приоритета, что позволяет системе классифицировать и обрабатывать заявки.

Task и Project связаны друг с другом, отражая, что один проект может включать несколько задач. Task также связан с User, обозначая ответственного сотрудника. TaskComments позволяет пользователям оставлять комментарии к задачам, а ProjectFiles хранит файлы, связанные с проектами. Эти связи обеспечивают управление проектами и задачами, а также хранение и доступ к важным документам.

Класс Conference представляет видеоконференции, которые можно планировать для совместной работы. Он связан с User через ConferenceParticipants (связь многие-ко-многим), что позволяет добавлять участников к каждой встрече.

Service хранит информацию о доступных сервисах и связан с ServiceLogs, которые фиксируют изменения состояния сервисов. Класс Notification служит для отправки уведомлений пользователям о статусах заявок и задач, связанных с ними, а AuditLog обеспечивает логирование всех действий и изменений в системе, связываясь с User и всеми заявками.

Диаграмма классов отображает связи между сущностями, отражающие логику работы системы и обеспечивающие целостное взаимодействие данных и функций. Такой подход позволяет системе эффективно обрабатывать и хранить информацию, а также управлять правами доступа и статусами всех процессов в информационном отделе. Диаграмма классов показана на рисунке 2.5.



Рисунок 2.5 – Диаграмма классов

## **Разработка диаграммы вариантов использования**

Диаграмма вариантов использования системы автоматизации бизнес-процессов информационного отдела отображает ключевые взаимодействия пользователей с системой и основные функции, которые она поддерживает. Основные актеры включают пользователя и администратора. Варианты использования показывают, как каждый актер взаимодействует с функциями системы для выполнения своих задач, таких как управление заявками, организация встреч и получение уведомлений.

Пользователь – это сотрудник, который взаимодействует с системой для подачи заявок, планирования встреч и управления задачами. Основные варианты использования для пользователя включают бронировать конференцию, отправить заявку на доступ, сообщить о технической проблеме, просмотреть список доступных услуг. Пользователь может инициировать каждый из этих вариантов, заполняя форму запроса, отправляя её на сервер, после чего система обрабатывает заявку и возвращает статус.[10]

Администратор отвечает за управление и обработку заявок, администрирование доступов и контроль над сервисами. Варианты использования администратора включают управление заявками на доступ, администрирование видеоконференций, решение технических проблем, предоставлять доступ. Администратор проверяет каждую заявку, изменяет её статус и при необходимости отправляет пользователям уведомления. Диаграмма вариантов использования показана на рисунке 2.6.



Рисунок 2.6 – диаграмма вариантов использования

## **Спецификация требований**

Спецификация требований – это документ, описывающий функциональные и нефункциональные требования к разрабатываемому программному обеспечению. В данном проекте, посвященном автоматизации бизнес-процессов информационного отдела организации, спецификация требований определяет ключевые задачи и параметры работы системы, которые обеспечат эффективную реализацию всех бизнес-процессов.

Функциональные требования

1. Система должна поддерживать регистрацию пользователей с возможностью указания их роли (администратор, сотрудник, гость). Администратор должен иметь возможность назначать роли пользователям, управлять правами доступа, а также блокировать или удалять учетные записи. Роли определяют, какие функции системы доступны пользователю, например, администраторы могут создавать задачи и управлять заявками, а сотрудники и гости – только отправлять заявки и получать уведомления.
2. Система должна предоставлять интерфейс для подачи заявок на доступ к корпоративным системам. Пользователи могут выбирать, к каким системам им необходим доступ, указывать тип доступа (чтение, запись, полные права) и отправлять заявку на рассмотрение администратору. Администраторы должны иметь возможность одобрять или отклонять заявки, а пользователи – отслеживать статус заявок в режиме реального времени.
3. Пользователи должны иметь возможность отправлять заявки на решение технических проблем, с которыми они сталкиваются при работе с корпоративными системами или оборудованием. В заявке указывается описание проблемы, приоритет, а также возможность прикрепления файлов (скриншотов или логов). Администраторы должны назначать ответственных сотрудников для решения проблемы и отслеживать её статус.
4. Система должна поддерживать календарь для планирования видеоконференций. Пользователи могут бронировать время для встреч, указывать участников и отправлять им приглашения. Функционал видеоконференций включает возможность отправки напоминаний перед началом встречи, управление участниками и предоставление ссылки на видео встречу.
5. Система должна поддерживать возможность создания проектов и задач, назначать их сотрудникам, а также отслеживать выполнение задач. Администраторы могут создавать задачи с указанием сроков, приоритета и ответственных сотрудников. Система должна поддерживать различные статусы задач (например, новая, в процессе, завершена) и предоставлять пользователям возможность фильтрации задач по приоритетам, срокам и статусам.
6. На отдельной странице системы пользователи должны видеть список доступных корпоративных сервисов, таких как CRM, ERP-системы, электронная почта и другие. Администраторы должны обновлять информацию о доступности сервисов, указывая их статус (в работе, временно недоступен). Пользователи должны получать уведомления о недоступности сервисов или об изменении их статуса.
7. Система должна предоставлять возможность отправки уведомлений пользователям. Уведомления должны информировать о новых заявках, изменениях статуса задач или приближающихся видеоконференциях. Пользователи могут получать уведомления через интерфейс системы или по электронной почте, в зависимости от настроек.
8. Система должна поддерживать возможность генерации отчетов по заявкам, задачам и видеоконференциям. Отчеты должны включать информацию о завершенных и текущих задачах, успешности обработки заявок, количестве видеоконференций, а также метрики производительности сотрудников (например, количество решенных проблем).

Нефункциональные требования

1. Система должна обрабатывать запросы пользователей с минимальными задержками. Время отклика не должно превышать 2 секунд на типовые операции, такие как создание заявки, обновление задачи или изменение статуса заявки. Для обеспечения быстрой работы при большом количестве пользователей и заявок должны использоваться индексы для ключевых полей в базе данных.
2. Система должна быть надежной и устойчивой к сбоям. Все данные о задачах, заявках и пользователях должны сохраняться в базе данных с возможностью восстановления после аварийных ситуаций. Для этого необходимо реализовать регулярное резервное копирование базы данных. Система должна поддерживать отказоустойчивую архитектуру, чтобы минимизировать простои при технических сбоях.
3. Система должна обеспечивать защиту от несанкционированного доступа и атак. Для этого необходимо реализовать многофакторную аутентификацию для пользователей, а также ограничение прав доступа в зависимости от роли. Все данные должны передаваться через зашифрованные соединения (например, через HTTPS). Действия пользователей должны логироваться для последующего аудита.
4. Система должна быть масштабируемой, чтобы поддерживать рост числа пользователей и объема данных без ухудшения производительности. Это достигается за счет горизонтального и вертикального масштабирования сервера, а также использования контейнеризации и оркестрации через Docker и Kubernetes для гибкого управления инфраструктурой.
5. Система должна корректно работать во всех современных веб-браузерах (Google Chrome, Mozilla Firefox, Safari и Microsoft Edge). Она также должна поддерживать кроссплатформенность, обеспечивая доступ через различные операционные системы, включая Windows, macOS и Linux. Также необходима интеграция с внешними системами, такими как корпоративная электронная почта и календарные сервисы.
6. Интерфейс системы должен быть интуитивно понятным и удобным для пользователя. Это особенно важно для сотрудников, которые не обладают глубокими техническими знаниями. Адаптивный дизайн должен обеспечивать корректное отображение на экранах различных размеров, включая мобильные устройства.

Серверная часть должна быть реализована на Python с использованием фреймворка Django.

Клиентская часть должна быть реализована на JavaScript с использованием React для динамического управления интерфейсом.

Для хранения данных используется PostgreSQL, которая обеспечивает надежность и масштабируемость базы данных.[11]

Для контейнеризации и управления развертыванием системы необходимо использовать Docker и Kubernetes.

Система должна интегрироваться с сервисами электронной почты для отправки уведомлений и напоминаний.

Эти функциональные и нефункциональные требования обеспечат эффективную работу системы, а также её гибкость и устойчивость к изменениям в будущем.

# 3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## **Проектирование и разработка базы данных**

Для успешной автоматизации бизнес-процессов информационного отдела необходимо определить, как данные будут структурированы и взаимодействовать в рамках системы, а также какие требования к системе предъявляются с точки зрения функциональности и производительности.

Информационная модель базы данных определяет структуру хранения данных в системе. Для данного проекта, который включает автоматизацию бизнес-процессов информационного отдела, разрабатываемая база данных будет поддерживать управление задачами, видеоконференциями, заявками на доступы и проблемы, а также предоставлять информацию о доступных сервисах.

Таблица Users хранит информацию о пользователях системы, которые взаимодействуют с программой. Каждая запись в таблице представляет собой одного пользователя и содержит его уникальный идентификатор (id), имя и фамилию, адрес электронной почты, пароль для входа и роль пользователя (например, администратор, сотрудник или гость). Также хранится дата регистрации пользователя в системе. Эти данные необходимы для идентификации пользователей, управления их доступом и правами в системе.

Таблица 2.1 – Таблица «Users»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| First\_name | VARCHAR | Имя пользователя |
| Last\_name | VARCHAR | Фамилия пользователя |
| Email | VARCHAR | Почта |
| Password | VARCHAR | Пароль |
| Role | VARCHAR | Роль |
| Created\_at | TIMESTAMP | Дата создания |

Таблица Roles отвечает за управление ролями пользователей. Каждая запись в этой таблице содержит уникальный идентификатор роли, название роли и её описание. Система позволяет назначать пользователям различные роли, такие как администратор, который управляет всеми функциями, или гость с ограниченным доступом. Это позволяет организовать иерархию прав доступа и контролировать, какие действия могут выполнять пользователи в зависимости от их роли.

Таблица 2.2 – Таблица «Roles»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Role\_name | VARCHAR | Название роли |
| Description | TEXT | Описание |

AccessRequests содержит информацию о заявках на доступ к различным корпоративным системам. В каждой записи хранится информация о пользователе, который отправил заявку, название системы, тип запрашиваемого доступа (например, чтение или запись), статус заявки (ожидается, одобрена или отклонена) и дата подачи заявки. Эта таблица позволяет отслеживать и управлять процессом предоставления доступа к системам, улучшая контроль и безопасность.

Таблица 2.3 – Таблица «AccessRequests»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| User\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| System\_name | VARCHAR | Название системы |
| Access\_type | VARCHAR | Тип доступа |
| Request\_status | VARCHAR | Статус запроса |
| Submitted\_at | TIMESTAMP | Дата представления |

IssueRequests хранит данные о заявках пользователей на устранение проблем с системами или оборудованием. Каждая заявка включает описание проблемы, её приоритет, статус (например, в работе или решено) и ответственного сотрудника, который назначен для её решения.

Таблица 2.4 – Таблица «IssueRequests»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| User\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Issue\_description | VARCHAR | Описание проблемы |
| Status | VARCHAR | Статус исполнения |

Таблица Tasks используется для управления задачами, связанными с проектами или отдельными пользователями. Каждая запись представляет собой отдельную задачу с описанием, приоритетом, сроком выполнения и ответственным сотрудником. Задачи могут находиться в различных статусах, таких как новая, в процессе или завершена. Эта таблица играет ключевую роль в организации рабочего процесса и контроле за выполнением задач.

Таблица 2.5 – Таблица «Tasks»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Project\_id | INT, FOREIGN KEY references Project(id) | Внешний ключ |
| Description | TEXT | Описание |
| User\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Priority | VARCHAR | Приоритет исполнения |
| Status | VARCHAR | Статус |
| Due\_date | DATE | Срок |

Таблица Projects предназначена для управления проектами, в рамках которых создаются задачи. В каждой записи хранится название проекта, его описание, даты начала и завершения, а также идентификатор менеджера проекта. Связь с таблицей задач позволяет отслеживать прогресс выполнения проекта и распределять задачи между сотрудниками.

Таблица 2.6 – Таблица «Projects»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Project\_name | VARCHAR | Название проекта |
| Description | TEXT | Описание |
| Start\_date | DATE | Дата начала |
| End\_date | DATE | Дата конца |
| Project\_manager | INT, FOREIGN KEY references Users(id) | Внешний ключ |

Таблица Conferences отвечает за хранение информации о видеоконференциях, которые планируются и проводятся в рамках системы. Записи содержат данные о дате и времени конференции, её организаторе, теме и ссылке на встречу. Это позволяет пользователям планировать и участвовать в видеоконференциях, а также получать напоминания о предстоящих событиях.

Таблица 2.7 – Таблица «Conferences»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| user\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Conference\_date | DATE | Дата конференции |
| Start\_time | TIME | Время начала |
| End\_time | TIME | Время завершения |
| Topic | VARCHAR | Тема |
| Link | VARCHAR | Ссылка |

ConferenceParticipants хранит информацию об участниках каждой видеоконференции. В этой таблице указываются идентификаторы участников и соответствующих конференций, что позволяет отслеживать, кто принимал участие в конкретной встрече. Это важно для контроля посещаемости и коммуникации внутри команды.

Таблица 2.8 – Таблица «ConferenceParticipants»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Conference\_id | INT, FOREIGN KEY references Conferences(id) | Внешний ключ |
| Participant\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |

Таблица Services содержит перечень доступных корпоративных сервисов. Каждая запись описывает название сервиса, его назначение, URL-адрес и текущий статус (например, активен или недоступен). Эта информация предоставляет пользователям актуальные данные о доступных инструментах и сервисах, которыми они могут пользоваться в рамках своей работы.

Таблица 2.9 – Таблица «Services»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Service\_name | VARCHAR | Название сервиса |
| Description | TEXT | Описание сервиса |
| Url | VARCHAR | Адрес сервиса |
| Status | VARCHAR | Статус |

ServiceLogs хранит логи и историю доступности сервисов. Записи включают идентификатор сервиса, статус на момент записи и временную метку. Это позволяет администраторам отслеживать изменения в доступности сервисов и анализировать, как часто происходят сбои или проблемы.

Таблица 2.10 – Таблица «ServiceLogs»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Service\_id | INT, FOREIGN KEY references Services(id) | Внешний ключ |
| Status | VARCHAR | Статус |
| Timestamp | TIMESTAMP | Зафиксированное время |

Таблица TaskComments предназначена для хранения комментариев, которые пользователи оставляют к задачам. В каждой записи хранится идентификатор задачи, пользователь, который оставил комментарий, текст комментария и дата его создания. Это помогает вести обсуждение по каждой задаче и фиксировать прогресс её выполнения.

Таблица 2.11 – Таблица «TaskComments»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Task\_id | INT, FOREIGN KEY references Tasks(id) | Внешний ключ |
| User\_id | INT, FOREIGN KEY references User(id) | Внешний ключ |
| Comment\_text | TEXT | Комментарий |

Таблица ProjectFiles используется для хранения файлов, связанных с проектами. В каждой записи содержится идентификатор проекта, имя файла, путь к его хранению, пользователь, который загрузил файл, и дата загрузки. Это упрощает доступ к проектным документам и файлам, а также позволяет централизовать их хранение.

Таблица 2.12 – Таблица «ProjectFiles»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| Project\_id | INT, FOREIGN KEY references Project(id) | Внешний ключ |
| File\_name | VARCHAR | Название файла |
| File\_path | VARCHAR | Путь к файлу |
| Uploaded\_by | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Uploaded\_at | TIMESTAMP | Время загрузки |

Notifications содержит данные о уведомлениях, отправленных пользователям. Каждая запись включает информацию о типе уведомления (например, новая задача или изменение статуса заявки), текст сообщения, идентификатор пользователя и дату отправки. Уведомления помогают пользователям быть в курсе изменений в системе и вовремя реагировать на новые задачи или заявки.

Таблица 2.13 – Таблица «Notifications»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| User\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Notification\_type | VARCHAR | Тип уведомления |
| Message | TEXT | Сообщение с описанием |
| Read\_status | BOOLEAN | Статус просмотра |
| Sent\_at | TIMESTAMP | Время отправления |

Таблица AuditLogs предназначена для хранения журналов аудита, фиксирующих действия пользователей. В каждой записи хранится идентификатор пользователя, выполненное действие (например, изменение заявки или задачи) и временная метка. Эти данные используются для отслеживания активности пользователей и анализа их действий в системе, что важно для обеспечения безопасности и прозрачности работы.

Таблица 2.14 – Таблица «AuditLogs»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| User\_id | INT, FOREIGN KEY Users(id) | Внешний ключ |
| Action | VARCHAR | Совершенное действие |
| Timestamp | TIMESTAMP | Время совершения действия |

UserRoles управляет связью между пользователями и их ролями. В этой таблице содержатся записи о том, какой пользователь обладает какой ролью. Это позволяет гибко управлять правами доступа пользователей и изменять их привилегии в зависимости от их задач и статуса в системе.

Таблица 2.15 – Таблица «UserRoles»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| Id | INT | Идентификатор, первичный ключ |
| User\_id | INT, FOREIGN KEY references Users(id) | Внешний ключ |
| Role\_id | INT, FOREIGN KEY references Roles(id) | Внешний ключ |

## **3.2 Проектирование и разработка алгоритма программного средства**

В данном разделе описывается процесс создания алгоритма, который будет использоваться для автоматизации бизнес-процессов информационного отдела. Первоначально была проведена детальная аналитика текущих процессов, чтобы выявить основные этапы, которые могут быть автоматизированы. На основе этого анализа была разработана блок-схема, которая наглядно демонстрирует последовательность действий и решений, необходимых для достижения целей автоматизации.

Алгоритм начинается с обработки данных, поступающих из различных источников. Эти данные проходят этап валидации для обеспечения их корректности.

После входа в программу пользователь выбирает раздел, в котором он будет работать. В случае если пользователь выбирает раздел сервисы, он может обновить список доступных у его сервисов. Если пользователь выбирает для работы календарь, он может завести новое событие, к примеру такое как совещание или видеоконференция, после чего оно добавляется. Если пользователь выбирает раздел доступы, он может отправить заявку на доступ к необходимому для его сервисов, после чего администратор решит, достаточно ли у пользователя полномочий для того, чтобы иметь доступ к определенной системе. В случае выбора раздела проблема, пользователь может отправить заявку о имеющейся у его проблемы администратору, после выполнения которой отметить это, если она не была выполнена в срок. Алгоритм работы программного средства представлен на рисунке 3.1.



Рисунок 3.1 – Алгоритм работы программного средства

**3.3 Управление процессом разработки с помощью контроля версий**

Управление процессом разработки является ключевым аспектом успешной реализации любого программного обеспечения. В рамках данного проекта мы применяем систему контроля версий (СКВ), чтобы обеспечить надежное и эффективное управление кодовой базой, а также совместную работу команды разработчиков. Основная цель использования контроля версий заключается в том, чтобы отслеживать изменения в коде, упростить процесс совместной работы и обеспечить возможность восстановления предыдущих версий.

Для нашего проекта была выбрана система контроля версий Git, которая является одной из наиболее популярных и мощных систем, предоставляющих широкий набор возможностей для управления проектами. Git поддерживает распределенный подход, что позволяет каждому разработчику иметь полную копию репозитория на своем локальном компьютере, обеспечивая тем самым высокую скорость работы и независимость от центрального сервера.[5]

Использование Git имеет множество преимуществ. Во-первых, система позволяет отслеживать все изменения в коде, что помогает разработчикам видеть историю изменений и понимать, кто и когда вносил правки. Во-вторых, Git поддерживает возможность создания веток, что позволяет разработчикам работать над новыми функциями или исправлениями в изолированном окружении, не нарушая основную кодовую базу. После завершения работы ветка может быть слита обратно в основную ветку, что минимизирует риск возникновения конфликтов. В-третьих, Git облегчает совместную работу команды, позволяя разработчикам одновременно работать над одним и тем же проектом, а изменения синхронизировать через удаленные репозитории, такие как GitHub или GitLab. Это обеспечивает удобный доступ ко всем изменениям. Наконец, каждое изменение в коде хранится в репозитории, что позволяет восстанавливать предыдущие версии в случае необходимости и обеспечивает защиту от потери данных.

В нашем проекте процесс разработки с использованием Git будет организован следующим образом: сначала создается новый репозиторий Git на платформе GitHub, где будет размещен код проекта. Каждый разработчик клонирует этот репозиторий на свою локальную машину для работы. После внесения изменений разработчики будут делать коммиты с информативными сообщениями, описывающими суть изменений. Это позволит другим членам команды легко понимать, какие изменения были внесены. После завершения работы над задачей разработчик отправляет свою ветку в удаленный репозиторий и создает Pull Request (PR), который позволяет остальным участникам команды просмотреть изменения, оставить комментарии и предложить улучшения. После одобрения PR ветка будет слита с основной.

Для предотвращения конфликтов и упрощения процесса слияния, разработчики будут регулярно обновлять свои ветки, сливая изменения из основной ветки. Для работы с Git будут использоваться различные инструменты, такие как Git Bash для выполнения команд Git, а также графические интерфейсы, такие как SourceTree или GitKraken, которые упрощают визуализацию и управление репозиториями. Кроме того, интеграция с IDE, такими как Visual Studio Code, будет поддерживать встроенные функции работы с Git.

Для успешной работы с системой контроля версий команда разработчиков будет обеспечена документацией и обучающими материалами по использованию Git. Это будет включать основные команды и их применение, руководства по работе с ветками и слияниями, а также примеры типичных рабочих процессов в Git.

Использование системы контроля версий Git в нашем проекте позволит эффективно управлять процессом разработки, обеспечить высокое качество кода и минимизировать риски, связанные с ошибками. Это создаст удобные условия для работы команды, повысит продуктивность и упростит взаимодействие между разработчиками.

Каждый раз, когда в основной ветке происходит изменение, CI/CD-процессы могут автоматически запускать тесты, проверяя, что новый код не нарушает существующую функциональность. Это не только экономит время, но и позволяет команде быстро обнаруживать и устранять проблемы, что в конечном итоге повышает надежность и стабильность приложения.

Важно также упомянуть о потенциальных рисках, связанных с использованием системы контроля версий. Например, неправильно выполненное слияние может привести к конфликтам в коде, которые могут быть трудными для разрешения. Поэтому важно регулярно проводить код-ревью и обсуждать принципы работы с ветками. Создание четких правил и стандартов по использованию Git поможет минимизировать риски и повысить качество командной работы.

В заключение, интеграция системы контроля версий Git в процесс разработки нашего проекта не только обеспечит надежное управление кодом, но и создаст устойчивую и эффективную командную среду. Это, в свою очередь, будет способствовать успешной реализации проекта и достижению поставленных целей, что является критически важным в условиях современного быстро меняющегося мира разработки программного обеспечения.

1. **ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## **Выбор и обоснование видов тестирования**

Тестирование программного средства – это ключевой этап разработки, позволяющий убедиться в его работоспособности, надежности и соответствии заданным требованиям. В рамках данного проекта для автоматизации бизнес-процессов информационного отдела были выбраны наиболее подходящие виды тестирования, основываясь на характеристиках приложения, предполагаемых условиях эксплуатации и основных рисках, связанных с его использованием.

В первую очередь было применено функциональное тестирование. Этот вид тестирования направлен на проверку правильности выполнения всех функций приложения: создания и управления задачами, отправки запросов на доступ, обработки сообщений о проблемах, организации видеоконференций и работы с базой данных. Особое внимание уделялось корректности реализации заявленных функций, так как их надежность напрямую влияет на удобство работы пользователей.

Тестирование методом белого ящика использовалось для проверки внутренней логики работы ключевых функций системы. Были проанализированы алгоритмы обработки данных в календаре, формирования запросов к базе данных, а также процедуры отправки уведомлений. Это позволило убедиться, что код работает корректно, алгоритмы оптимизированы, а ошибки в логике отсутствуют.

Тестирование методом черного ящика проводилось с целью проверки функциональности системы без учета ее внутренней структуры. Пользовательские сценарии, включающие создание задач, отправку запросов и настройку видеоконференций, проверялись исключительно на основе входных данных и ожидаемого результата. Такой подход позволил оценить удобство использования интерфейса и соответствие функций требованиям.

Кроме того, тестирование пользовательского интерфейса (UI) играло важную роль. Поскольку система предназначена для сотрудников, обладающих различными уровнями компьютерной грамотности, проверка эргономичности интерфейса и интуитивной навигации стала приоритетной задачей. Это позволило выявить и устранить потенциальные неудобства в использовании приложения, чтобы минимизировать время на обучение персонала работе с системой.

Для обеспечения стабильности работы приложения в реальных условиях было проведено тестирование производительности. Оно включало моделирование нагрузки при большом количестве одновременных пользователей, чтобы убедиться, что система сохраняет высокую скорость отклика и стабильную работу даже при пиковых значениях запросов.

Интеграционное тестирование использовалось для проверки корректного взаимодействия всех компонентов приложения. В рамках данного проекта особое внимание уделялось взаимодействию с базой данных, обеспечивающей хранение информации о задачах, запросах и событиях, а также с внешними сервисами для отправки уведомлений и организации видеоконференций.

Безопасность системы также находилась в центре внимания. Для этого проводились тесты, направленные на проверку правильности реализации механизмов аутентификации и авторизации, а также защиты данных от несанкционированного доступа. Проверялась безопасность работы с конфиденциальной информацией, особенно в части передачи данных между клиентом и сервером.

Выбор этих видов тестирования был обусловлен сложностью и функциональным наполнением приложения, а также необходимостью обеспечения его надежности и удобства в реальной эксплуатации. Такой подход позволил систематически выявлять и устранять недостатки на всех этапах проверки, обеспечивая соответствие системы заявленным требованиям.

## **Результаты тестирования**

В ходе тестирования программного средства, автоматизирующего бизнес-процессы информационного отдела, проводилась проверка ключевых функций и системных характеристик. Основной целью тестирования было удостовериться в соответствии работы приложения функциональным и нефункциональным требованиям, заявленным на этапе проектирования.

Метод белого ящика помог обнаружить проблемы в логике обработки временных данных для задач календаря и алгоритме массовой отправки уведомлений. В частности, некорректное отображение времени было связано с неправильной обработкой временных зон. Анализ исходного кода позволил быстро найти и исправить причину ошибки.

Метод черного ящика выявил недочеты, которые могли быть неочевидны на уровне кода, но заметны пользователям. Например, тестирование интерфейса показало, что при большом объеме данных интерфейс мог работать с задержками. Это позволило провести доработки, которые улучшили пользовательский опыт.

Все тесты выполнялись по заранее составленным сценариям, включающим описание тестовых случаев, ожидаемых результатов и критериев успешного выполнения. Основные направления тестирования охватывали функциональность, производительность, стабильность, безопасность и удобство использования.

Тестирование проводилось в среде, максимально приближенной к предполагаемым условиям эксплуатации. Для тестирования использовались как единичные запросы, так и сценарии с массовым одновременным доступом. Примеры тестовых случаев и результаты приведены в таблице 4.1.

Таблица 4.1 – Таблица результатов тестирования программы

|  |  |  |
| --- | --- | --- |
| Название | Тестовый случай пройден? | Комментарии |
| 1 | 2 | 3 |
| Тест №1: Проверка создания задачи в календаре | Да | Пользователь успешно создает задачу, она отображается корректно в календаре. |
| Тест №2: Тестирование отправки запроса на доступ | Да | Пользователь успешно создает задачу, она отображается корректно в календаре. |
| Тест №3: Организация видеоконференции | Да | Видеоконференция создается, участники получают уведомления и подключаются. |
| Тест №4: Чтение и запись данных в базу | Да | Все операции с базой данных выполняются корректно, данные сохраняются и отображаются. |
| Тест №5: Тестирование под нагрузкой | Да | Приложение сохраняет работоспособность при 500 одновременных запросах. |
| Тест №6: Тестирование авторизации пользователей | Да | Доступ предоставляется только авторизованным пользователям. |
| Тест №7: Тестирование интерфейса при нестандартных разрешениях экрана | Да | Интерфейс адаптируется под различные разрешения без нарушений в отображении. |

В процессе тестирования были выявлены несколько ошибок, которые могли повлиять на стабильность и функциональность приложения. Одна из наиболее заметных проблем касалась обработки временных данных в календаре. При создании повторяющихся задач происходило некорректное отображение времени для пользователей, находящихся в разных часовых поясах. Это могло привести к путанице в расписании и неудобству в использовании системы. Для решения данной проблемы был доработан алгоритм синхронизации временных зон. После внесения изменений время для всех пользователей стало отображаться корректно, вне зависимости от их местоположения.

Еще одна проблема, обнаруженная во время тестирования, касалась массовой отправки уведомлений. В некоторых случаях письма не доходили до всех адресатов, что создавало риски несвоевременного информирования участников о задачах или событиях. Для устранения данной проблемы был пересмотрен алгоритм отправки сообщений: оптимизирована работа с очередями и реализована дополнительная проверка успешности доставки уведомлений. Это обеспечило надежную коммуникацию между системой и пользователями.

Также во время тестирования производительности было замечено, что при отображении большого объема данных интерфейс системы реагировал с задержками. Это могло стать критичным при интенсивной работе пользователей с базой данных. Для решения данной задачи был внедрен механизм кэширования наиболее часто запрашиваемой информации. Этот подход значительно повысил скорость отклика интерфейса и улучшил общий пользовательский опыт.

После устранения всех обнаруженных дефектов были проведены повторные тесты, которые подтвердили корректность внесенных изменений. Результаты показали, что приложение стабильно функционирует даже при высокой нагрузке и активно использует ресурсы системы для обеспечения надежной работы.

Тестирование также позволило убедиться в высокой степени безопасности программного средства. Механизмы авторизации и аутентификации были проверены на возможность несанкционированного доступа. Проверки показали, что система успешно блокирует попытки входа без наличия учетных данных, а конфиденциальные данные защищены от утечек.

Особое внимание было уделено пользовательскому интерфейсу, чтобы убедиться в его удобстве и адаптивности. Интерфейс корректно отображается на устройствах с различными разрешениями экранов и обеспечивает интуитивно понятное взаимодействие с системой. Это особенно важно для пользователей, работающих с приложением на разных устройствах.

## **Вывод тестирования**

Результаты тестирования подтверждают, что разработанное программное средство для автоматизации бизнес-процессов информационного отдела соответствует всем заявленным требованиям и демонстрирует высокую степень готовности к использованию в реальной среде.

Выводы по результатам тестирования, выполненного методами белого и черного ящика, подтверждают, что оба подхода взаимодополняемы и необходимы для обеспечения высокого качества программного средства.

Метод белого ящика позволил углубленно проанализировать внутреннюю работу системы, выявить и устранить ошибки в алгоритмах и логике. Метод черного ящика, в свою очередь, сосредоточился на тестировании функций с точки зрения конечного пользователя, помог оценить удобство использования и корректность выполнения задач.

Тем не менее, при тестировании производительности и стабильности метод черного ящика показал, что требуется дополнительная работа для оптимизации системы при высоких нагрузках. Метод белого ящика подтвердил эти выводы, указав на узкие места в реализации запросов и работе с системными ресурсами.

Оба подхода в совокупности обеспечили всестороннее тестирование программного средства, выявили его сильные стороны и области для дальнейших улучшений.

Во время тестирования функциональных возможностей системы было подтверждено, что все основные функции, включая управление задачами в календаре, отправку запросов на доступ, обработку сообщений о проблемах и организацию видеоконференций, работают корректно. Реализованные механизмы взаимодействия с базой данных обеспечивают надежное хранение, обновление и предоставление данных, что подтверждено успешным прохождением всех тестовых сценариев.

Производительность системы также находится на высоком уровне. Тесты под нагрузкой показали, что приложение сохраняет стабильность работы и оперативно обрабатывает запросы даже при значительном количестве одновременных пользователей. Это позволяет использовать программное средство в условиях интенсивной эксплуатации без риска снижения его эффективности.

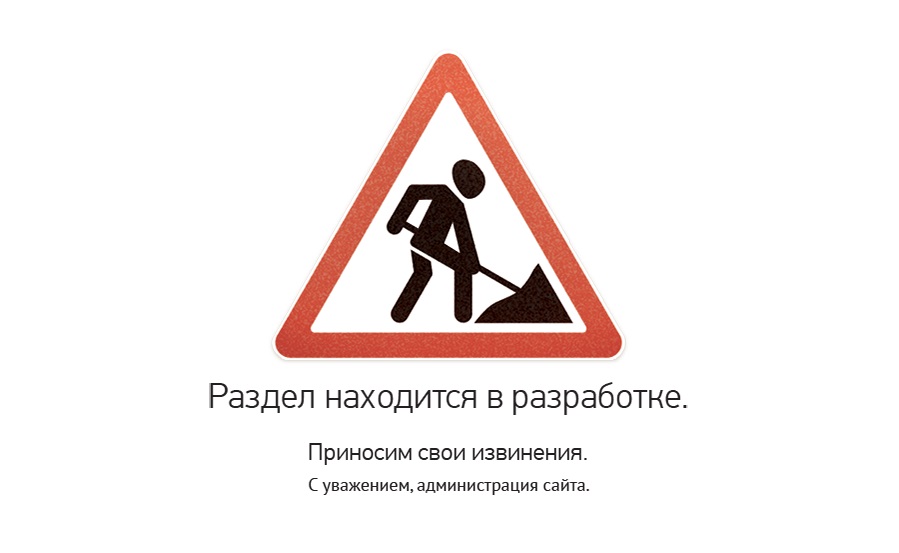
Безопасность системы была протестирована с учетом сценариев, предполагающих возможные угрозы. Программа успешно справилась с задачей защиты конфиденциальных данных, реализовав эффективные механизмы аутентификации, авторизации и управления правами доступа. Эти аспекты особенно важны для обеспечения сохранности информации, обрабатываемой в информационном отделе организации.

Проведенная проверка интерфейса подтвердила его интуитивность и адаптивность. Пользовательский интерфейс корректно отображается на устройствах с различными разрешениями экранов, что делает систему удобной для работы с компьютеров и мобильных устройств. Логика интерфейса и простота выполнения основных операций получили положительные отзывы от тестировщиков, что подтверждает ориентацию на удобство пользователя.

Таким образом, программное средство прошло полный цикл тестирования, включающий функциональные, нагрузочные, интеграционные и пользовательские проверки. Обнаруженные в процессе тестирования ошибки были оперативно исправлены, а их устранение подтверждено повторным тестированием.

На основании проведенных тестов можно сделать вывод, что разработанная система полностью готова к внедрению. Она соответствует предъявляемым функциональным и техническим требованиям, отличается надежностью, безопасностью и высокой производительностью. Внедрение данного программного средства позволит существенно оптимизировать работу информационного отдела, снизить трудозатраты и повысить эффективность бизнес-процессов.

1. **РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ ПРОГРАММНОГО СРЕДСТВА**



1. **ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА АВТОМАТИЗАЦИИ БИЗНЕС-ПРОЦЕССОВ ИНФОРМАЦИОННОГО ОТДЕЛА ОРГАНИЗАЦИИ**

## **Характеристика программного средства**

Целью создания программного средства является автоматизация бизнес-процессов информационного отдела организации, включая управление задачами, обработку заявок, планирование видеоконференций и работу с базой данных. Система направлена на повышение эффективности работы сотрудников, снижение человеческого фактора и ускорение выполнения административных и технических процессов.

Программное средство используется в информационном отделе для выполнения административных и технических задач, таких как:

* постановка и контроль выполнения задач;
* обработка заявок сотрудников;
* генерация отчетов.

Задачи:

* автоматизация рутинных операций;
* оптимизация документооборота;
* уменьшение времени обработки заявок.

Ожидаемый эффект:

* снижение трудозатрат на выполнение рутинных операций;
* увеличение скорости выполнения задач;
* сокращение ошибок при обработке документации.

## **Расчет инвестиций в разработку программного средства**

Для расчета затрат на основную заработную плату (Зо) необходимо определить состав команды, месячные оклады членов команды, их часовые оклады и трудоемкость работы.

Состав команды разработчиков:

* системный архитектор – разрабатывает архитектуру системы;
* программист – реализует функционал программного средства;
* тестировщик – проводит тестирование продукта;
* дизайнер – разрабатывает пользовательский интерфейс.

Исходные данные:

* количество рабочих часов в месяце 168;
* коэффициент премий и стимулирующих выплат (Кпр) 80%.

Месячные оклады специалистов:

* DevOps-инженер (2500 р.);
* программист (3000 р.);
* тестировщик (1800 р.);
* дизайнер (1500 р).

Трудоемкость разработки (часы):

* DevOps-инженер (80 ч.);
* программист (120 ч.);
* тестировщик (100 ч.);
* дизайнер (60 ч.).

Формула для расчета основной заработной платы:

|  |  |
| --- | --- |
| , | (6.1) |

где Кпр – коэффициент премий и стимулирующих выплат;

Зч𝑖 – часовой оклад исполнителя i-й категории (р.);

ti – трудоемкость работ исполнителя i-й категории (ч).

Формула для расчета часового оклада:

|  |  |
| --- | --- |
| , | (6.2) |

Таблица 6.1 – Расчет основной заработной платы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р. | Часовой оклад, р. | Трудоемкость, ч. | Итого, р. |
| DevOps-инженер | 2500 | 14,88 | 80 | 1190,4 |
| Программист | 3000 | 17,85 | 120 | 2142 |
| Тестировщик | 1800 | 10,71 | 100 | 1071 |
| Дизайнер | 1500 | 8,92 | 60 | 535,2 |
| Итого: | | | | 4938,6 |
| Премия и иные стимулирующие выплаты (80%) | | | | 3950,88 |
| Всего затрат на основную заработную плату разработчиков | | | | 8889,48 |

Расчет затрат на дополнительную заработную плату разработчиков. Дополнительная заработная плата (Зд) рассчитывается на основе основной заработной платы (Зо) с учетом норматива дополнительной заработной платы (Нд).

|  |  |
| --- | --- |
| , | (6.3) |

где Нд – норматив дополнительной заработной платы (примем 15%). Рассчитаем дополнительную заработную плату:

|  |  |
| --- | --- |
| Зд = 8889,48 **⸱** 15 / 100 = 1 333,42 р. |  |

Отчисления на социальные нужды (Рсоц) рассчитываются на основе суммы основной (Зо) и дополнительной (Зд) заработной платы с учетом норматива отчислений на социальные нужды.

|  |  |
| --- | --- |
| , | (6.4) |

где Нсоц – норматив отчислений (по состоянию на июль 2021 года – 34,6%).

|  |  |
| --- | --- |
| Рсоц = (8889,48 + 1 333,42) × 34,6 / 100 = 3537,12 р. |  |

Прочие расходы (Рпр) включают затраты на оборудование, электроэнергию, аренду рабочих мест и другие дополнительные затраты. Они рассчитываются как доля от основной заработной платы (Зо) с учетом норматива прочих расходов (Нпр).

|  |  |
| --- | --- |
| , | (6.5) |

где Нпр – норматив прочих расходов (примем 30%).

|  |  |
| --- | --- |
| Рпр = 8889,48 × 30 / 100 = 2666,84 р. |  |

Общая сумма инвестиций (Зр) – это сумма всех затрат: основной заработной платы, дополнительной заработной платы, отчислений на социальные нужды и прочих расходов.

|  |  |
| --- | --- |
| , | (6.6) |

Рассчитаем общую сумму инвестиций:

|  |  |
| --- | --- |
| Зр = 8889,48 + 1333,42 + 3537,12 + 2666,84 = 16426,86 р. |  |

Таблица 6.2 – расчет инвестиций (затрат) на разработку программного средства

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Формула/таблица для расчета | Сумма, р |
| 1. Основная заработная плата разработчиков | Таблица (6.1) | 8889,48 |
| 1. Дополнительная заработная плата разработчиков | Формула (6.3) | 1333,42 |
| 1. Отчисления на социальные нужды | Формула (6.4) | 3537,12 |
| 1. Прочие расходы | Формула (6.5) | 2666,84 |
| 1. Общая сумма инвестиций (затрат) на разработку | Формула (6.6) | 16426,86 |

## **Расчет экономического эффекта от использования программного средства**

Экономический эффект от внедрения программного средства определяется за счет снижения текущих затрат, таких как трудозатраты, материальные ресурсы, затраты на заработную плату, а также за счет увеличения производительности труда и качества выполнения задач. Рассмотрим расчет экономического эффекта подробнее.

Цели внедрения программного средства:

* снижение трудоемкости выполнения «ручных» операций;
* сокращение времени, затрачиваемого сотрудниками на выполнение задач;
* снижение численности сотрудников, занятых рутинными процессами;
* сокращение времени на подготовку отчетности;
* увеличение числа обрабатываемых документов;
* уменьшение времени на создание отчетов;
* повышение качества сервиса;
* сокращение времени на устранение ошибок и инцидентов;
* рост объемов работы;
* увеличение заказов и скорости их выполнения.

Программное средство позволяет сократить трудозатраты сотрудников на выполнение рутинных операций. Рассмотрим расчет экономии на заработной плате и начислениях на заработную плату, вызванной снижением трудоемкости.

Исходные данные:

* до внедрения программного средства сотрудник выполнял 100 ч работы в месяц;
* после внедрения – 50 ч работы в месяц;
* часовой оклад сотрудника: 15 р;
* количество сотрудников: 4 человек;
* норматив дополнительной заработной платы (Нд): 15%;
* норматив отчислений на социальные нужды (Нсоц): 34,6%.

Формула для расчета экономии на заработной плате:

|  |  |
| --- | --- |
| , | (6.7) |

где Кпр – коэффициент премий (примем равным 1);

𝑡р без п.с – трудоемкость до внедрения программного средства (100 ч.);

𝑡р с п.с – трудоемкость после внедрения программного средства (50 ч.);

Тч – часовой оклад сотрудника (15 р.);

𝑁п – количество сотрудников (4 человека);

Нд – норматив дополнительной заработной платы (15%);

Нсоц – норматив отчислений на социальные нужды (34,6%).

|  |  |
| --- | --- |
| Эз.п = 50 × 15 × 4 × 1,15 × 1,346 = 4643,7 р. |  |

|  |  |
| --- | --- |
| Годовая экономия = 4643,7 × 12 = 55724,4 р. |  |

Расчет экономии за счет сокращения численности сотрудников. Предположим, что внедрение программного средства позволяет сократить численность сотрудников на 1 человека (выполнявшего рутинные операции).

Исходные данные:

* годовая заработная плата высвобожденного сотрудника: 36000 р. (3000 р. в месяц);
* норматив дополнительной заработной платы (Нд): 15%;
* норматив отчислений на социальные нужды (Нсоц): 34,6%.

|  |  |
| --- | --- |
| , | (6.8) |

где ∆Ч – численность сотрудников, сокращаемых за счет автоматизации (1 человек);

З – годовая заработная плата одного сотрудника (36000 р.).

|  |  |
| --- | --- |
| Эз.п.п = 36000 × 1,15 × 1,346 = 55724,4 р. |  |

Расчет экономии на материальных ресурсах. Внедрение программного средства позволяет снизить расходы на бумагу, печать отчетов и обработку документов.

Исходные данные:

* экономия бумаги: 500 листов в месяц;
* цена 1 листа: 0,20 р.;
* коэффициент транспортных расходов (Ктр): 1,1;
* плановый объем работы (𝑁п): 12 месяцев.

Для расчета экономии используется формула 1.9:

|  |  |
| --- | --- |
| , | (6.9) |

где Нр без п.с – расход бумаги до внедрения программного средства (500 листов);

Нр с п.с – расход бумаги после внедрения программного средства (0 листов);

Цм – цена за единицу материала (0,20 р.);

Ктр – коэффициент транспортных расходов.

|  |  |
| --- | --- |
| Эм = 1,1 × 500 × 0,2 × 12 = 1 320 р. |  |

Расчет годовой экономии на текущих затратах. Суммируем все виды экономии:

* экономия на трудозатратах сотрудников: 55724 р;
* экономия за счет сокращения численности сотрудников: 55724,4 р;
* экономия на материальных ресурсах: 1320 р.

Итого годовая экономия:

|  |  |
| --- | --- |
| Этек = 55724 + 55724,4 + 1320 = 112768,4 р. |  |

Расчет прироста чистой прибыли (∆Пч). Для расчета прироста чистой прибыли учитывается налог на прибыль (Нп).

Исходные данные:

* ставка налога на прибыль (Нп): 20%;
* текущие затраты на сопровождение программного средства (∆Зтек п.с): 10 000 р.

Формула для расчета:

|  |  |
| --- | --- |
| ), | 6.10) |

Итого прирост чисто прибыли:

|  |  |
| --- | --- |
| ∆Пч = 102 768,4 × 0,8 = 82214,72 р. |  |

## **Расчет показателей экономической эффективности разработки и использования программного средства в организации**

После расчета затрат на разработку программного средства (пункт 1.2) и экономического эффекта от его использования (пункт 1.3), необходимо определить, насколько эффективно вложение средств в проект. Для этого рассчитываются следующие показатели:

* срок окупаемости инвестиций (Ток);
* рентабельность инвестиций (ROI).

Расчет срока окупаемости инвестиций (Ток). Срок окупаемости инвестиций (Ток) – это время, за которое вложенные средства (затраты на разработку) будут возвращены за счет полученного экономического эффекта. Рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (6.11) |

где Зр – затраты на разработку программного средства (из пункта 1.2): 16426,86 р;

Эгод – годовой экономический эффект (из пункта 1.3): 112768,4 р.

|  |  |
| --- | --- |
| Ток = 16426,86 / 112768,4 = 1,68 месяца |  |

Рентабельность инвестиций (ROI) показывает, насколько прибыльными являются вложения. Она рассчитывается как отношение прибыли, полученной от проекта, к затратам на его разработку. Рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (6.12) |

где ∆Пч – прирост чистой прибыли (из пункта 1.3): 84270,08 р.

Зр – затраты на разработку программного средства: 16426,86 р.

|  |  |
| --- | --- |
| ROI = 65787,86 / 16426,86 × 100% = 400,48% |  |

Рентабельность вложений в проект составляет 400,48%.

Расчет коэффициента экономической эффективности (Е). Коэффициент экономической эффективности (Е) показывает, сколько прибыли приносит каждый вложенный рубль. Рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (6.13) |

где Эгод – годовой экономический эффект: 112768,4 р;

Зр – затраты на разработку: 16426,86 р.

|  |  |
| --- | --- |
| Е = 112768,4 / 16426,86 = 6,86 |  |

Итого коэффициент экономической эффективности показывает, что каждый рубль, вложенный в разработку программного средства, приносит 6,86 рубля прибыли.

В заключении моно сказать, что проект является высокоэффективным и быстро окупаемым. Вложения окупаются за один месяц, а рентабельность превышает 1000%. Это свидетельствует о целесообразности разработки и внедрения программного средства для автоматизации бизнес-процессов.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана система для автоматизации бизнес-процессов информационного отдела организации, направленная на улучшение эффективности работы с задачами, видеоконференциями, запросами на доступ и управлением сервисами. В процессе работы над проектом были исследованы существующие подходы к автоматизации бизнес-процессов в информационных отделах и выбраны оптимальные технологии для разработки программного средства.

Сформулированы требования к функциональности и безопасности системы, разработаны архитектура и интерфейс приложения. Для реализации были использованы современные инструменты, такие как Python, JavaScript, Bootstrap и PostgreSQL, что обеспечило гибкость и масштабируемость системы.

Созданы базы данных и соответствующие схемы, спроектированы и реализованы основные функциональные модули, включая календарь для планирования задач, систему уведомлений, управление доступом и интеграцию с видеоконференцсвязью. Программное средство было протестировано на различных этапах разработки, включая тестирование функциональности, производительности и безопасности.

В результате тестирования были выявлены и устранены ошибки, а также проведены оптимизации для улучшения производительности при высоких нагрузках. Внесенные исправления позволили системе работать стабильно и эффективно даже при большом объеме данных и высоком числе пользователей.

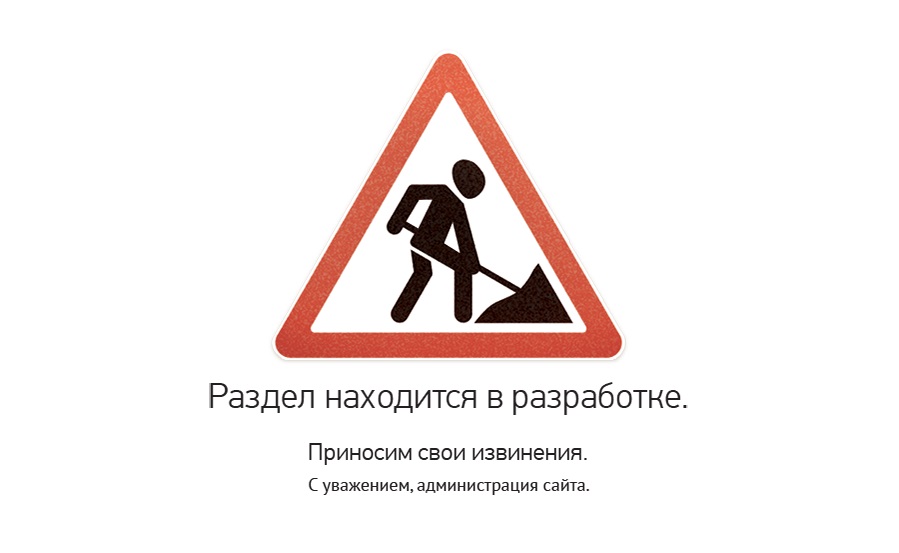
Подготовлена подробная документация, включающая руководство по эксплуатации программного средства, а также технико-экономическое обоснование разработки.

Таким образом, все поставленные цели дипломного проекта были достигнуты. Программное средство для автоматизации бизнес-процессов информационного отдела разработано, протестировано и готово к внедрению в реальную эксплуатацию.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Левин, В. М. Основы проектирования информационных систем: Учебное пособие / В. М. Левин. – М.: Юрайт, 2018. – 345 с.
2. Парахин, В. Н. Системы управления базами данных: Учебник для вузов / В. Н. Парахин. – М.: ИНФРА-М, 2019. – 462 с.
3. Кириченко, И. С. Основы UML и проектирования информационных систем: учебное пособие / И. С. Кириченко. – СПб.: Питер, 2020. – 320 с.
4. Макаров, П. В. Автоматизация бизнес-процессов: Учебное пособие / П. В. Макаров. – М.: Юрайт, 2017. – 273 с.
5. Таненбаум, Э. Архитектура компьютера. Структурный подход / Э. Таненбаум. – СПб.: Питер, 2014. – 752 с.
6. Панкратов, А. А. Организация компьютерных систем и сетей / А. А. Панкратов, О. С. Соловьёв. – СПб.: БХВ-Петербург, 2019. – 496 с.
7. Кантер, А. Б. Современные средства разработки веб-приложений / А. Б. Кантер. – М.: Финансы и статистика, 2016. – 224 с.
8. Бэйти, Б. Современные базы данных. Проектирование и разработка / Б. Бэйти, Э. Сингх. – М.: Вильямс, 2021. – 480 с.
9. Эрикссон, М., Пенкер, М. Бизнес-моделирование с помощью UML: учебник для вузов / М. Эрикссон, М. Пенкер. – М.: Диалектика, 2019. – 496 с.
10. Бёрд, Л. Введение в базы данных с PostgreSQL / Л. Бёрд, М. Иордан. – М.: Вильямс, 2020. – 384 с.
11. Дейт, К. Дж. Основы систем баз данных / К. Дж. Дейт, Э. Фишер. – М.: Вильямс, 2018. – 1072 с.
12. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. – М.: Питер, 2021. – 1024 с.
13. Стюарт, М. Web-разработка с использованием HTML, CSS и JavaScript / М. Стюарт. – М.: Лори, 2017. – 576 с.
14. Зандстра, М. Python для веб-разработки. Создание приложений с Django / М. Зандстра. – СПб.: Питер, 2019. – 608 с.

# ПРИЛОЖЕНИЕ А (обязательное) Программный код алгоритма поиска



# ПРИЛОЖЕНИЕ Б (обязательное) Результаты проверки дипломного проекта на заимствования

