**КУРСОВАЯ РАБОТА**

**Тема: Разработка модуля «Система управления задачами и проектами»**

**Специальность 09.02.07 Информационные системы и программирование**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 31-ИС21** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **П.С. Сивунов** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **В.Ю. Назаров** |

**Москва 2023**

|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»  КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ |

**Задание**

**на курсовой проект (работу)**

Дисциплина: МДК.01.01 Разработка программных модулей

Тема:

Специальность:09.02.07 Информационные системы и программирование

Группа:

ФИО студента

ФИО руководителя Назаров В.Ю.

1. Проанализировать предметную область
2. Проанализировать готовые решения
3. Подготовить техническое задание
4. Подготовить план тестирования
5. Обосновать выбор инструментов и средств разработки
6. Описать реализацию технического задания
7. Выполнить тестирование
8. Подготовить инструкцию пользователю по работе с модулем

Задание выдано «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Срок выполнения «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Сроки защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

Стр.

[ВВЕДЕНИЕ 4](#_Toc153663315)

[ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc153663316)

[1.1 Основные понятия и объекты 6](#_Toc153663317)

[1.2 Обзор и анализ существующих программных решений 9](#_Toc153663318)

[ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 15](#_Toc153663319)

[2.1 Требования к функциям, выполняемым системой 15](#_Toc153663320)

[2.2 Описание среды разработки 18](#_Toc153663321)

[2.3 Обоснование выбора инструментария по разработке 20](#_Toc153663322)

[2.4 Модуль API 21](#_Toc153663323)

[2.5 Структура приложения 23](#_Toc153663324)

[ГЛАВА 3. РЕАЛИЗАЦИЯ ПРОЕКТА МОДУЛЯ СИСТЕМЫ 24](#_Toc153663325)

[3.1 Описание кода модуля API 24](#_Toc153663326)

[3.2 Описание десктопного приложения 25](#_Toc153663327)

[3.2.1 Запуск 25](#_Toc153663328)

[3.2.2 Окно авторизации 25](#_Toc153663329)

[3.2.3 Главное окно 26](#_Toc153663330)

[3.2.4 Дочерние окна 28](#_Toc153663331)

[ЗАКЛЮЧЕНИЕ 32](#_Toc153663332)

[СПИСОК ИСТОЧНИКОВ 33](#_Toc153663333)

# ВВЕДЕНИЕ

Бизнес-процессы становятся все более сложными и конкурентоспособность компаний напрямую зависит от эффективного управления проектами, системы управления задачами и проектами становятся неотъемлемой частью организационной деятельности. С увеличением объема информации и разнообразия задач возникает потребность в интегрированных и интеллектуальных инструментах, способных эффективно координировать работу команд, управлять ресурсами и повышать общую производительность. В этом контексте возникает необходимость в создании "Системы управления задачами и проектами" — программного продукта, предназначенного для систематизации и автоматизации управленческих процессов.

Основной целью данного проекта является разработка модуля системы управления задачами и проектами. Модуль способен повысить эффективность работы команд, уменьшить временные и ресурсные затраты на управление проектами, а также обеспечить более прозрачную и обоснованную систему принятия управленческих решений.

Актуальность данного проекта обусловлена не только стремительным технологическим прогрессом, но и повышением требований к оперативности и точности управленческих решений. Сложность современных бизнес-процессов требует от организаций эффективных инструментов для планирования, контроля и анализа проектов. Управление задачами и проектами становится ключевым элементом успешного ведения бизнеса, влияя на конечные результаты и обеспечивая гибкость в условиях постоянных изменений рынка.

Для достижения поставленной цели – разработки модуля системы управления задачами и проектами – был выделен ряд ключевых задач:

* Анализ существующих подходов и систем управления проектами.
* Определение требований к системе.
* Разработка архитектуры и дизайна системы
* Реализация и тестирование

Объектом исследования являются современные бизнес-процессы и организационная деятельность компаний, осуществляющих управление проектами. В контексте данного проекта объектом также является создание программного продукта - модуля системы управления задачами и проектами.

Предметом исследования является процесс систематизации и автоматизации управленческих процессов. В рамках предмета исследования анализируются существующие подходы и системы управления проектами, определяются требования к создаваемому модулю, разрабатывается его архитектура и дизайн, а также проводится реализация и тестирование.

# ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## Основные понятия и объекты

Система управления задачами и проектами (далее – Система) предназначена для автоматизации процессов управления задачами и проектами в организации. Система должна обеспечивать следующие цели

1. Улучшение эффективности управления задачами и проектами
2. Повышение гибкости и адаптивности управления задачами и проектами
3. Сокращение затрат на управление задачами и проектами

Эффективность управления задачами и проектами можно повысить за счет следующих факторов:

* Повышение продуктивности менеджеров проектов. Система может автоматизировать такие задачи, как планирование, отслеживание хода выполнения и контроль задач и проектов. Это позволит освободить время менеджерам проектов для более стратегических задач.
* Повышение прозрачности и контроля за выполнением задач и проектов. Система обеспечивает прозрачность и контроль за выполнением задач и проектов. Это позволяет менеджерам проектов своевременно выявлять и устранять отклонения от плана.
* Сокращение времени и ресурсов, затрачиваемых на управление задачами и проектами. Система может помочь сократить время и ресурсы, затрачиваемые на управление задачами и проектами. Это связано с тем, что система автоматизирует рутинные задачи, повышает прозрачность и контроль, а также позволяет менеджерам проектов принимать более эффективные решения.

Гибкость и адаптивность управления задачами и проектами необходимы для своевременного реагирования на изменения в окружающей среде. Система обеспечивает гибкость и адаптивность управления задачами и проектами за счет следующих возможностей:

* Поддержка различных методологий управления проектами. Система должна поддерживать различные методологии управления проектами, что позволяет организациям выбирать наиболее подходящую методологию для своих задач.
* Возможность внесения изменений в план проекта. Система должна предоставлять возможность внесения изменений в план проекта в случае возникновения непредвиденных обстоятельств.
* Поддержка изменений в составе команды проекта. Система должна поддерживать изменения в составе команды проекта, что позволяет организациям быстро адаптироваться к изменениям в структуре организации.

Система может помочь сократить затраты на управление задачами и проектами за счет следующих факторов:

* Автоматизация рутинных задач. Система может автоматизировать такие задачи, как планирование, отслеживание хода выполнения и контроль задач и проектов. Это позволит сократить затраты на оплату труда менеджеров проектов.
* Использование шаблонов. Система позволяет использовать шаблоны для планирования и выполнения задач и проектов. Это позволит сократить затраты на подготовку документов.
* Интеграция с другими системами. Система может быть интегрирована с другими системами, используемыми в организации. Это позволит сократить затраты на ввод и обработку данных.

В предметной области Системы используются следующие основные понятия и объекты:

Задача – это единица работы, которая должна быть выполнена в рамках проекта или вне его. Задача может быть простой или сложной, а также может быть классифицирована по различным признакам, например, по типу, сложности, приоритету, сроку выполнения и т. д.

Проект – это совокупность взаимосвязанных задач, направленных на достижение определенной цели. Проект может быть различным по масштабу, сложности и длительности, а также может быть классифицирован по различным признакам, например, по типу, сложности, длительности, бюджету и т. Д.

Ресурс – это все, что может быть использовано для выполнения задач или проектов, например, люди, оборудование, материалы, деньги. Ресурс может быть материальным или нематериальным, а также может быть классифицирован по различным признакам, например, по типу, стоимости, доступности и т. Д.

Риск – это событие или условие, которое может оказать негативное влияние на достижение целей проекта. Риск может быть внутренним или внешним, а также может быть классифицирован по различным признакам, например, по вероятности наступления, тяжести последствий и т. Д.

Коммуникация – это обмен информацией между участниками проекта. Коммуникация может быть устной, письменной или электронной, а также может быть классифицирована по различным признакам, например, по типу, назначению, направлению и т. Д.

Качество – это степень соответствия продукта или услуги установленным требованиям. Качество может быть оценено с помощью различных показателей, например, надежности, функциональности, удобства использования и т. Д.

## Обзор и анализ существующих программных решений

Существует множество программных продуктов, предназначенных для управления задачами и проектами, каждый из которых ориентирован на определенные потребности и особенности бизнес-процессов. Рассмотрим несколько ключевых программных решений, широко применяемых в предметной области:

Jira

Очень популярная система, которая обычно используется для работы с кодом и как баг‑трекер. Подходит для команд от 100 человек.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Скриншот Jira.

* Модули для разных команд.
* Работа с кодом в задачах. Благодаря интеграции с хостингом исходного кода Bitbucket программисты могут писать код и обсуждать его внутри задач.
* Модуль Confluence. Удобный способ вести документацию. Можно систематизировать информацию, например часто используемые формы и реестры, и контролировать версии.
* Создание задач прямо из чата техподдержки. Из‑за этой функции систему часто используют как баг‑трекер: проблемы легко зафиксировать и сразу начать решать.
* 16 видов отчётов. Диаграммы сгорания задач (чтобы видеть прогресс в достижении цели спринта) и скорости команды, отчёты по спринтам и по загруженности пользователей, контрольный график и так далее.
* Дорожные карты. Позволяют составить подробный план как внутри одной команды, так и для нескольких сразу.

Trello: Очень популярный планировщик и таск‑трекер, который лучше всего подходит для творческих команд и небольших коллективов.

Изображение выглядит как текст, программное обеспечение, снимок экрана, веб-страница

Автоматически созданное описание

Рисунок 2 – Скриншот Trello.

* Автоматизация. Можно настроить автоматические действия с карточками при определённых условиях.
* Шаблоны. Есть готовые варианты под разные отделы и цели: для личных дел, для команды маркетинга, для дизайнеров и т.п.
* Интеграция с почтой. У каждой карточки есть имейл‑адрес, и оставлять комментарии можно прямо из электронного ящика без авторизации в системе.
* Хронология. На временной шкале отображаются все взаимосвязи между членами команды. Можно разбивать работу над проектом на этапы и контролировать каждый из них, чтобы равномерно распределять нагрузку в команде.
* Визуализация данных в «Панели». Ключевые показатели представлены в форме диаграмм и дашбордов. Благодаря этому руководители могут быстро оценить, как продвигаются задачи и насколько загружена команда.

«Битрикс24»

Корпоративный портал для крупных команд. Система заточена под отделы продаж и маркетинга, чаще используется как система управления взаимоотношениями с клиентами.

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Автоматически созданное описание

Рисунок 3 – Скриншот Битрикс24.

* Здесь есть карточки, история, контроль сделок, телефония, записи разговоров, и т.д.
* Сквозная аналитика. Благодаря ей маркетологи прямо в «Битрикс24» могут рассчитывать ROMI, эффективность рекламы на разных площадках: от «Яндекс.Директа» и Google Ads до различных социальных сетей.
* Звонки и связь. Есть аудио и видеозвонки, групповые и личные чаты, отслеживание звонков, рассылки SMS‑сообщений. Всё работает без сбоев, не хуже, чем в мессенджерах.
* Облачное хранилище. Файлы и документы легко редактировать прямо в системе.
* Группы. Их можно создавать по разным темам и направлениям работы, делать папки внутри, хранить в них файлы и вести общение.
* Контакт‑центр. «Битрикс24» интегрирован с почтой, мессенджерами и социальными сетями. Тут есть телефония, онлайн‑чат, а также статистика диалогов и звонков — с числом обращений и процентом довольных клиентов.

Asana: Система для планирования, операционного менеджмента и управления задачами. Лучше всего подойдёт небольшим командам: веб‑студиям, агентствам и маркетинговым отделам.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Веб-сайт

Автоматически созданное описание

Рисунок 4 – Скриншот Asana.

* Гибкая настройка задач. Можно помечать их любыми тегами, выделять разными цветами, добавлять картинки на обложку карточки и делать список подзадач.
* Вехи. Они обозначают достижение промежуточного результата, по которому можно судить об успешности проекта в целом. Функция помогает визуально обозначать ключевые контрольные точки, чтобы лучше следить за происходящим.
* Работа на уровне проектов. Их можно копировать, экспортировать, делать из них шаблоны и синхронизировать задачи.
* Панель мониторинга. На ней в режиме реального времени можно видеть все важные данные по проектам.
* Цели. Их можно указать заранее и связать с рабочим процессом, чтобы потом отслеживать прогресс.
* Файлы. В отдельной вкладке можно посмотреть все документы, прикреплённые к проекту.

Microsoft Project: Microsoft Project представляет собой мощный инструмент для управления проектами, разработанный для организаций с разнообразными потребностями в управлении задачами и ресурсами.

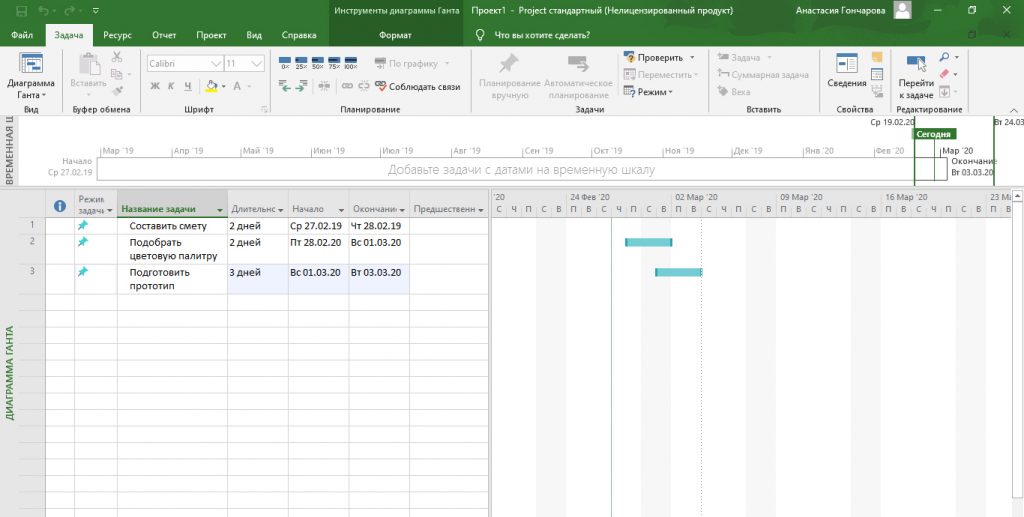


Рисунок 5 – Скриншот Microsoft Project.

* Гибкость планирования: Пользователи могут определять задачи, устанавливать зависимости, назначать ресурсы и определять сроки выполнения.
* Бюджетирование и анализ затрат: Microsoft Project обеспечивает возможность составления бюджетов проектов, а также отслеживание и анализ затрат.
* Отчетность
* Интеграция с другими приложениями Microsoft
* Поддержка различных методологий управления проектами

Рынок программных продуктов для управления задачами и проектами предлагает разнообразные решения, ориентированные на различные потребности и особенности бизнес-процессов. Каждый из рассмотренных инструментов, таких как Jira, Trello, Битрикс24, Asana и Microsoft Project, обладает своими уникальными функциональными возможностями, предназначенными для определенных типов команд и организаций.

Jira выделяется своей широкой функциональностью, особенно в области работы с кодом. Trello, напротив, ориентирован на простоту использования, визуализацию задач и автоматизацию процессов для творческих и небольших коллективов. Битрикс24 предлагает корпоративное решение с широким функционалом, включая систему управления клиентскими отношениями (CRM), облачное хранилище и контакт-центр. Asana предназначена для гибкого планирования задач и работы на уровне проектов, подходя небольшим командам и отделам. Microsoft Project, как мощный инструмент, предоставляет гибкие возможности планирования, бюджетирования и отчетности, а также интеграцию с другими приложениями Microsoft.

Эти преимущества делают их популярными и востребованными у пользователей.

# ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 2.1 Требования к функциям, выполняемым системой

Функции компонента Авторизация

Окно авторизации должно открываться при запуске программы

Должна быть возможность ввести:

* Логин
* Пароль

Должна быть проверка на корректность введенных данных при нажатии на кнопку Войти.

Изображение выглядит как снимок экрана, Прямоугольник, текст, линия

Автоматически созданное описание

Рисунок 6 – Набросок окна авторизации

Функции главного окна

Главное окно для администратора проекта должно обеспечить автоматизацию следующих функций платформы:

* Вывод задач по дате в текущем проекте
* Смена текущего проекта
* Просмотр описания задачи
* Добавление задачи себе, другим участникам проекта
* Редактирование задачи
* Удаление задачи

Главное окно для обычного пользователя должно обеспечить автоматизацию следующих функций платформы:

* Вывод задач по дате в текущем проекте
* Смена текущего проекта
* Просмотр описания задачи
* Изменение статуса задачи

Профиль

Личный профиль пользователя должен быть доступен всем пользователям платформы после авторизации.

Личный профиль пользователя должен содержать блок общих сведений, заполняемых при регистрации пользователя:

* ФИО пользователя;
* Логин
* Пароль

Пользователь должен иметь возможность редактирования общих сведений, а также сведений, необходимых для авторизации пользователя.

Уведомление о задаче

В Платформе должны быть реализованы следующие функции в части уведомления пользователя о поступившей задаче:

* Получение заявки исполнителем, после создания задачи администратором
* Возможность принять заявку

Проекты

В платформе должны быть реализованы следующие функции в части информирования пользователя о его проектах:

* Просмотр текущих проектов

Чат

В платформе должны быть реализованы следующие функции в части коммуникации между пользователями – участниками проекта. Должно быть доступно окно Чат. Оно должно выполнять функции:

* Отображение истории сообщений
* Отправка и получение сообщений

Изображение выглядит как Прямоугольник, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 7 – набросок главного окна

Функции компонента Регистрация

Окно регистрации должно быть доступно администратору, для того чтобы зарегистрировать человека в программе на определенный проект. После чего работник входит в систему с помощью учетных данных, переданных ему администратором.

Должна быть возможность ввести:

* Логин
* Пароль
* ФИО
* Проект, на который назначается человек

Должна быть проверка на корректность введенных данных при нажатии на кнопку зарегистрировать.

Изображение выглядит как снимок экрана, Прямоугольник, линия, текст

Автоматически созданное описание

Рисунок 8 – Набросок окна регистрации

Приложение должно состоять из 2 модулей:

* API – модуль обменивающийся данными с БД на локальном сервере
* Десктопное приложение с пользовательским интерфейсом. Обмен данных с БД через API с помощью http-запросов.

**2.2 Описание среды разработки**

Для разработки программного модуля “Система управления задачами и проектами” были выбраны следующие программные инструменты:

1. Среда разработки: PyCharm 2023.2.4
   * Описание: PyCharm – это интегрированная среда разработки (IDE) для языка программирования Python. Она предоставляет разработчикам удобный интерфейс, обширные инструменты отладки, автоматическое завершение кода, анализ кода в реальном времени и другие функции, содействующие эффективному и комфортному процессу разработки.
2. Язык программирования: Python
   * Описание: Python – высокоуровневый язык программирования, известный своей простотой и читаемостью кода. Он предоставляет разработчикам мощный инструментарий для разнообразных задач и широко используется в веб-разработке, научных исследованиях, анализе данных и создании прикладных программ.
3. Платформа: PyQt6
   * Описание: PyQt6 представляет собой набор Python-оберток для библиотеки Qt, предназначенной для создания графических пользовательских интерфейсов. PyQt6 обеспечивает разработчиков высокопроизводительными и гибкими инструментами для создания современных и эстетичных интерфейсов приложений. Поддержка PyQt6 обеспечивает широкий спектр функциональности для проекта.
4. FastAPI
   * Описание: FastAPI представляет собой современный фреймворк для создания веб-приложений на языке Python с акцентом на быстродействие и автоматическую генерацию документации API. Он обеспечивает простоту использования, поддержку стандарта OpenAPI и встроенные средства валидации запросов и ответов, что делает его отличным выбором для создания веб-сервисов и API.
5. Другие инструменты и библиотеки:
   * Requests: Библиотека Requests предоставляет простой и удобный способ отправки HTTP-запросов и работы с ответами. Она широко используется для взаимодействия с веб-сервисами, API и другими удаленными ресурсами, предоставляя удобный интерфейс для выполнения различных HTTP-запросов.

Выбранные программные инструменты предоставляют широкий набор функциональности и поддерживают разработку приложений с использованием необходимых технологий и библиотек.

## 2.3 Обоснование выбора инструментария по разработке

Выбранный инструментарий обладает несколькими преимуществами:

* PyCharm предоставляет удобный и интуитивно понятный интерфейс, что упрощает процесс разработки и улучшает продуктивность. Обладает широким функционалом, включая интеграцию с системами контроля версий, предоставляет продвинутые средства отладки и профилирования кода.
* Язык программирования Python обладает простым и лаконичным синтаксисом, поставляется с обширной стандартной библиотекой, предоставляя множество готовых модулей и инструментов для различных задач.
* PyQt6 обеспечивает гибкий инструментарий для создания кроссплатформенных графических интерфейсов, что позволяет модулю работать на различных операционных системах.
* Фреймворк FastAPI обеспечивает высокую производительность благодаря использованию современных асинхронных технологий, что важно для обработки большого количества запросов.
* Библиотека Requests предоставляет простой и интуитивно понятный интерфейс для отправки HTTP-запросов, что упрощает взаимодействие с внешними ресурсами.

Этот выбранный инструментарий лучше всего соответствует требованиям курсовой работы, так как обладает необходимыми функциональными возможностями и предоставляет удобный способ разработки модуля “Система управления задачами и проектами”.

## 2.4 Модуль API

API разделен на модуль с классами (back), и модуль описывающий сервер и http-запросы (FastApi).

Библиотека SQLAlchemy позволяет использовать объектно-ориентированный подход к проектированию БД, что упрощает понимание и реализацию схемы БД. Это означает, что каждая таблица в БД представляется как класс в Python. Поля таблицы представляются атрибутами класса, а связи между таблицами представляются отношениями между классами.

Рассмотрим диаграмму БД (см приложение 1)

Согласно диаграмме, в коде описываются классы, определяющие сущности БД.

Для обмена данными между приложением и сервером используется API.

Рассмотрим диаграмму классов. (см приложение 2)

Модуль FastApi представляет класс, состоящий из методов – реализующих http запросы к серверу.

В классе DataRequests хранятся запросы, посылаемые приложением к серверу, и ссылки на классы. Запрос содержит ссылку на класс – таблицу в БД.

## 2.5 Структура приложения

Функционал, который предоставляет интерфейс, ограничивается в зависимости от уровня прав пользователя.

Предусмотрено 2 уровня доступа:

Admin: Администратор. Может создавать проекты, регистрировать на них работников, создавать, редактировать, удалять задачи для работников и себя. Видит статусы выполнения задач работниками. Имеет доступ в чат с каждым работником.

User: Сотрудник. Сотрудник, который видит заявки на выполнение задач от Администратора, может принимать их. Имеет доступ к чату с другими сотрудниками и администратором.

Изображение выглядит как диаграмма, линия, круг, рисунок

Автоматически созданное описание

Рисунок 9 – Диаграмма вариантов использования

**ГЛАВА 3.** **РЕАЛИЗАЦИЯ ПРОЕКТА МОДУЛЯ СИСТЕМЫ**

## 3.1 Описание кода модуля API

API – фактически локальный сервер базы данных. В нем хранится БД.

Код состоит из 2 частей: файла классов и главного файла.

Файл классов отвечает за подключение к БД, создание таблиц, создание сессии.

from sqlalchemy import \*  
from sqlalchemy.orm import sessionmaker, declarative\_base  
  
\_\_all\_\_ = ['Tasks', 'User', 'session', 'Chat', 'Project', 'Request', 'User\_project']  
engine = create\_engine('sqlite:///db.db')  
Base = declarative\_base()  
  
class Tasks(Base)   
class User(Base)  
class Chat(Base)

class Project(Base)  
class Request(Base)  
class User\_project(Base)

Base.metadata.create\_all(engine)  
Sessionlocal = sessionmaker(bind=engine)  
session = Sessionlocal()

Здесь происходит создание объекта engine для взаимодействия с базой данных SQLite. Определение моделей таблиц. Создание таблиц в базе данных с помощью метаданных моделей. Создание сессии для взаимодействия с данными в таблицах.

Главный файл инициализирует приложение FastApi

from fastapi import FastAPI, Body, HTTPException  
from sqlalchemy.exc import NoResultFound  
  
from back import \*  
from sqlalchemy import and\_, or\_  
from datetime import datetime  
  
app = FastAPI()

class DataRequests  
  
session.close()

DataRequests - описывает http-запросы к серверу. По окончании сессия закрывается. Запуск локального сервера осуществляется командой терминала uvicorn main:app –reload.

## 3.2 Описание десктопного приложения

### 3.2.1 Запуск

Программа начинает выполнение с файла main.py:

def main():  
 import sys  
 from app import app  
 from auth import Login  
  
 auth\_window = Login()  
 auth\_window.show()  
  
 def close\_all\_windows():  
 for widget in app.topLevelWidgets():  
 widget.close()  
  
 app.aboutToQuit.connect(close\_all\_windows)  
  
 sys.exit(app.exec())

Здесь импортируетcя app – инициализированный экземпляр приложения из файла app.py

import sys  
from PyQt6 import QtWidgets  
app = QtWidgets.QApplication(sys.argv)

Затем импортируется окно авторизации, и запускается приложение. Также прописана пользовательская функция close\_all\_windows, отвечающая за закрытие всех окон при закрытии главного окна приложения

### 3.2.2 Окно авторизации

Представляет собой окно с 2 полями ввода логина и пароля соответственно и кнопкой войти.

При инициализации создается дополнительный поток, отвечающий за проверку соединения с сервером.

Спроектированный пользовательский интерфейс окна авторизации выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 10 – Окно авторизации

Данные считываются с полей ввода, и в зависимости от ответа сервера происходит авторизация, или сообщение об ошибке. Функция авторизации:

def login(self):  
 *"""Авторизация пользователя"""* if self.st\_label.text() != "Сервер в сети":  
 return  
 if self.loginEdit.text() != '' and self.passEdit.text() != '':  
 response = requests.get( f"{BASE\_URL}/auth?login={str(self.loginEdit.text())}&password={str(self.passEdit.text())}")  
 if response.status\_code == 404:  
 self.error\_label.setText('Неправильное имя пользователя\nили пароль')  
 elif response.status\_code == 200:  
 user\_id = int(response.json()['id'])  
 permission = response.json()['root']  
  
 from todo import MainWindow  
  
 self.mw = MainWindow(user\_id, permission)  
 self.mw.show()  
  
 self.close()  
 else:  
 self.error\_label.setText('Что-то пошло не так')  
 else:  
 self.error\_label.setText('Поля не могут быть пустыми')

### 3.2.3 Главное окно

Главное окно содержит календарь, таблицу вывода дел, поля вывода описаний дел.

Спроектированный пользовательский интерфейс главного окна выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 11 – Главное окно

При инициализации создаются объекты дочерних окон: профиль, чат, проекты, регистрация, добавления дела, изменение дела. Выполняется запрос для отображения списка дел на текущую дату. Часть функции отображения списка дел:

for row in range(len(self.lst\_do)):  
 for col in range(self.listWidget.columnCount()):  
 time = self.lst\_do[row]['time'].split(':')  
 if self.lst\_do[row]["status"] != 2 and self.permission == 'user' and self.lst\_do[row]["user\_id"] == self.user\_id:  
 item = None  
 if col == 0:  
 item = QtWidgets.QTableWidgetItem(f'{time[0]}:{time[1]}')  
 elif col == 1:  
 item = QtWidgets.QTableWidgetItem(f'{self.lst\_do[row]["title"]}')  
 elif col == 2:  
 item = QtWidgets.QTableWidgetItem(  
 f'{"Сделано" if self.lst\_do[row]["status"] == True else "На выполнении"}')  
 self.listWidget.setItem(row, col, item)  
 elif self.permission == 'admin':  
 item = None  
 user = requests.get(f"{BASE\_URL}/list\_users?user\_id={self.lst\_do[row]['user\_id']}").json()  
 if col == 0:  
 item = QtWidgets.QTableWidgetItem(f'{time[0]}:{time[1]}')  
 elif col == 1:  
 item = QtWidgets.QTableWidgetItem(f'{self.lst\_do[row]["title"]}')  
 elif col == 2:  
 item = QtWidgets.QTableWidgetItem(f'{user['FIO']}')  
 elif col == 3:  
 item = QtWidgets.QTableWidgetItem(  
 f'{"Сделано" if self.lst\_do[row]["status"] == True else "На выполнении"}')  
 self.listWidget.setItem(row, col, item)

Здесь в зависимости от прав человека формируется и заполняется таблица дел.

### 3.2.4 Дочерние окна

**3.2.4.1 Чат.**

Спроектированный пользовательский интерфейс окна чата выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 12 – Окно чата

При появлении запрашивает и формирует выпадающий список участников проекта. При выборе человека отображает историю переписки с ним. Функция отображения истории чата:

def render\_ch(self):  
 self.listWidget.clear()  
 r\_id = 0  
 for i in self.query:  
 if self.combo.currentText() == i['FIO']:  
 r\_id = i['id']  
 query = requests.get(f'{BASE\_URL}/history\_chat?sender\_id={self.user\_id}&reciever\_id={r\_id}')  
 self.sort\_hist = sorted(query.json(), key=lambda x: (x['date'], x['time']))  
  
 for i in self.sort\_hist:  
 time = i['time'].split(':')  
 date = datetime.strptime(i['date'], '%Y-%m-%d').date().strftime("%d.%m.%Y")  
 self.listWidget.addItem(f'{date} {time[0]}:{time[1]} {self.get\_sender(i['sender\_id'])}\n {i['message']}')

**3.2.4.2 Профиль.**

Спроектированный пользовательский интерфейс окна профиля выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – Окно профиля

При инициализации вызывается функция, которая запрашивает и отображает данные пользователя:

def render(self):  
 self.query = requests.get(f"{BASE\_URL}/profile/{self.user\_id}").json()  
 fio, login, password = str(self.query['FIO']), str(self.query['login']), str(self.query['password'])  
 self.lineEdit\_2.setText(fio)  
 self.lineEdit.setText(login)  
 self.lineEdit\_3.setText(password)

**3.2.4.3 Добавление дела**

Спроектированный пользовательский интерфейс окна добавления дела выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 14 – Окно добавления дела

При отображении окно формирует выпадающий список пользователей, которым можно назначит дело. Часть функции добавление дела:

data = {  
 'date': str(self.dateEdit.date().toPyDate()),  
 'time': str(self.timeEdit.time().currentTime().toPyTime().strftime('%H:%M:%S')),  
 'title': self.titleLineEdit.text(),  
 'description': str(self.descEdit.toPlainText()),  
 'status': 0 if self.user\_id == u\_id else 2,  
 'user\_id': u\_id,  
 'project\_id': self.proj\_id,  
}  
query = requests.post(f'{BASE\_URL}/add', json=data)  
if self.user\_id != u\_id:  
 data2 = {  
 'sender\_id': self.user\_id,  
 'task\_id': query.json()['id'],  
 }  
 requests.post(f'{BASE\_URL}/add2', json=data2)

Здесь формируются 2 словаря данных, один для информации о деле, второй для записи о заявке о деле. После чего отправляется запрос с данными на сервер.

**3.2.4.4 Добавление пользователя**

Спроектированный пользовательский интерфейс окна регистрации пользователя выглядит следующим образом:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 15 – Окно регистрации пользователя

При инициализации окна заполняется список проектов на который можно зарегистрировать пользователя. Часть функции регистрации:

data = {  
 'FIO': self.fio\_label.text(),  
 'login': str(self.login\_edit.text()),  
 'password': self.password\_edit.text(),  
 'root': "user",  
}  
q = requests.post(f'{BASE\_URL}/add\_user', json=data).json()По  
pr\_id = 0  
for i in self.pr:  
 if self.combo.currentText() == i['name']:  
 pr\_id = i['id']  
data2 = {  
 'user\_id': q['id'],  
 'project\_id': pr\_id  
}  
requests.post(f'{BASE\_URL}/user\_to\_project', json=data2)

Здесь также формируются 2 словаря данных, один для информации о человеке, второй для записи о человека на выбранный проект. После чего отправляется запрос с данными на сервер.

**3.2.4.5 Окно работы с проектами**

Позволяет администратору просматривать, добавлять, удалять проекты.

Код см. приложение 3

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 16 – Окно работы с проектами

**3.2.4.6 Окно уведомлений**

Позволяет работнику просматривать уведомления о задачах, и принимать их к исполнению. Код см. приложение 4.

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 17 – Окно уведомлений сотрудника

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы были проанализированы существующие подходы и системы управления проектами, определены требования к Системе, была разработана и протестирована архитектура и дизайн Системы.

Разработанная Система является мощным инструментом, который может значительно повысить эффективность организационной деятельности.

Система обеспечивает следующие преимущества:

* Повышение эффективности работы команд: система позволяет сотрудникам легко координировать свои действия, отслеживать прогресс и вносить изменения в планы по мере необходимости.
* Уменьшение временных и ресурсных затрат: система автоматизирует многие задачи, связанные с управлением проектами, что позволяет сотрудникам сосредоточиться на более важных аспектах своей работы.
* Улучшение прозрачности и обоснованности принятия решений: система предоставляет руководителям подробную информацию о ходе выполнения проектов.

Система была протестирована с использованием различных тестов и показала, что она работает корректно и соответствует требованиям.

Можно доработать систему, добавив следующие функции:

* Возможность отслеживать время, затрачиваемое на задачи и проекты. Эта функция позволит руководителям оценивать эффективность использования рабочего времени сотрудников.
* Возможность создавать отчеты о выполнении задач и проектов. Эти отчеты будут полезны для руководителей, которые хотят отслеживать прогресс проектов и оценивать их результаты.
* Возможность интеграции с другими системами. Эта функция позволит объединить систему управления задачами и проектами с другими системами.

# СПИСОК ИСТОЧНИКОВ

1. ГОСТ 7.32–2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления: Стандартинформ, 2017
2. ГОСТ 7.1—2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления
3. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования
4. ГОСТ 7.11—2004 (ИСО 832:1994) Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках
5. ГОСТ 7.12—93 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила
6. ГОСТ 7.80—2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления
7. ГОСТ 7.82—2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления
8. Добрый, добрый Python - обучающий курс от Сергея Балакирева // https://stepik.org/course/100707/syllabus
9. Добрый, добрый Python ООП - обучающий курс от Сергея Балакирева // https://stepik.org/course/116336/syllabus
10. Чернышев, С. А. Основы программирования на Python: учебное пособие для среднего профессионального образования / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. https://urait.ru/bcode/532292 (дата обращения: 16.12.2023)
11. Программирование на SQL Маркин, А. В. Программирование на SQL: учебное пособие для среднего профессионального образования / А. В. Маркин. — Москва: Издательство Юрайт // https://urait.ru/bcode/518166 (дата обращения: 16.12.2023).
12. Qt for Python // https://doc.qt.io/qtforpython-6/ (дата обращения: 16.12.2023).
13. Работа с SQLite в Python (для чайников) / Хабр // https://habr.com/ru/articles/754400/ (дата обращения: 16.12.2023).
14. Python и FastAPI | Подключение к базе данных и создание таблиц //

https://metanit.com/python/fastapi/2.1.php (дата обращения: 16.12.2023).

1. Знакомство с FastAPI / Хабр // https://habr.com/ru/articles/488468/
2. FastAPI // https://fastapi.tiangolo.com/ (дата обращения: 16.12.2023).
3. Как сделать из Python-скрипта исполняемый файл / Хабр // https://habr.com/ru/companies/slurm/articles/746622/ (дата обращения: 16.12.2023).
4. Работа с датой и временем в Python // https://python-scripts.com/datetime-time-python (дата обращения: 16.12.2023).
5. Jira | Программное обеспечение для отслеживания задач // https://www.atlassian.com/ru/software/jira
6. Manage Your Team’s Projects From Anywhere | Trello // https://trello.com
7. Битрикс24 + AI. Бесплатный онлайн-сервис для бизнеса! // https://www.bitrix24.ru
8. Asana. Работайте над большими идеями без лишней суеты. // https://asana.com/ru
9. ПО для управления проектами | Microsoft Project // https://www.microsoft.com/ru-ru/microsoft-365/project/project-management-software

Приложение 1

Диаграмма БД

Изображение выглядит как текст, диаграмма, План, Параллельный

Автоматически созданное описание

Приложение 2

Диаграмма классов

Изображение выглядит как текст, диаграмма, Параллельный, линия

Автоматически созданное описание

Приложение 3

Код окна работы с проектами

import requests  
from PyQt6 import QtCore, QtWidgets  
from connection import BASE\_URL  
  
  
class Project:  
 def \_\_init\_\_(self):  
 self.permission = None  
 self.user\_id = 0  
 self.\_proj\_win = QtWidgets.QWidget()  
 self.\_proj\_win.resize(400, 243)  
 self.\_proj\_win.setFixedSize(400, 243)  
  
 self.verticalLayout = QtWidgets.QVBoxLayout(self.\_proj\_win)  
 self.verticalLayout.setContentsMargins(0, 0, 0, 0)  
 self.listWidget = QtWidgets.QTableWidget(parent=self.\_proj\_win)  
 self.listWidget.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.ResizeMode.Stretch)  
 self.listWidget.setSelectionBehavior(QtWidgets.QTableWidget.SelectionBehavior.SelectRows)  
 self.listWidget.setEditTriggers(QtWidgets.QAbstractItemView.EditTrigger.NoEditTriggers)  
 self.verticalLayout.addWidget(self.listWidget)  
 self.horizontalLayout = QtWidgets.QHBoxLayout()  
 self.pushButton = QtWidgets.QPushButton(parent=self.\_proj\_win)  
 self.horizontalLayout.addWidget(self.pushButton)  
 self.pushButton\_2 = QtWidgets.QPushButton(parent=self.\_proj\_win)  
 self.horizontalLayout.addWidget(self.pushButton\_2)  
 self.verticalLayout.addLayout(self.horizontalLayout)  
  
 \_translate = QtCore.QCoreApplication.translate  
 self.\_proj\_win.setWindowTitle(\_translate("Form", "Проекты"))  
 self.pushButton.setText(\_translate("Form", "Удалить"))  
 self.pushButton\_2.setText(\_translate("Form", "Добавить"))  
  
 self.add\_pr = Add\_proj()  
 self.add\_pr.pushButton.clicked.disconnect(self.add\_pr.execute)  
 self.add\_pr.pushButton.clicked.connect(self.render)  
  
 self.pushButton\_2.clicked.connect(self.add)  
 self.pushButton.clicked.connect(self.remove\_pr)  
 self.for\_permission()  
  
 def for\_permission(self):  
 if self.permission == 'user':  
 self.pushButton.close()  
 self.pushButton\_2.close()  
  
 def render(self):  
 query = requests.get(f"{BASE\_URL}/project\_for\_admin").json() if self.permission == 'admin' else requests.get(  
 f"{BASE\_URL}/project/{self.user\_id}").json()  
 self.listWidget.clear()  
 self.listWidget.setColumnCount(2)  
 self.listWidget.setHorizontalHeaderLabels(['Название', 'Описание'])  
 self.listWidget.setRowCount(len(query))  
 for row in range(len(query)):  
 for col in range(self.listWidget.columnCount()):  
 item = None  
 if col == 0:  
 item = QtWidgets.QTableWidgetItem(f'{query[row]['name']}')  
 elif col == 1:  
 item = QtWidgets.QTableWidgetItem(f'{query[row]['desc']}')  
 self.listWidget.setItem(row, col, item)  
  
 def show(self, user\_id, permission):  
 self.permission = permission  
 self.user\_id = user\_id  
 self.render()  
 self.\_proj\_win.show()  
  
 def add(self):  
 self.add\_pr.show\_()  
  
 def remove\_pr(self):  
 title = self.listWidget.item(self.listWidget.currentRow(), 0).text()  
 requests.delete(f"{BASE\_URL}/project/{title}").json()  
 self.render()  
  
  
  
from add import Add  
  
  
class Add\_proj(Add):  
 def \_\_init\_\_(self):  
 Add.\_\_init\_\_(self)  
  
 self.dateEdit.close()  
 self.timeEdit.close()  
 self.date\_label.close()  
 self.time\_label.close()  
 self.for\_label.close()  
 self.combobox.close()  
  
 \_translate = QtCore.QCoreApplication.translate  
  
 self.\_winAdd.setWindowTitle(\_translate("Form", "Добавить проект"))  
 self.pushButton.clicked.connect(self.add\_pr)  
  
 def add\_pr(self):  
 *""" Удаление дела"""* data = {  
 'name': self.titleLineEdit.text(),  
 'desc': self.descEdit.toPlainText(),  
 }  
 requests.post(f'{BASE\_URL}/add\_project', json=data)  
 self.\_winAdd.close()  
  
 def show\_(self):  
 self.\_winAdd.show()

Приложение 4

Код окна работы с уведомлениями

import requests  
from PyQt6 import QtCore, QtWidgets  
from project import Project  
from connection import BASE\_URL  
  
  
class Note(Project):  
 def \_\_init\_\_(self):  
 Project.\_\_init\_\_(self)  
  
 self.notes = None  
 self.query = None  
 self.proj\_id = None  
 \_translate = QtCore.QCoreApplication.translate  
 self.\_proj\_win.setWindowTitle(\_translate("Form", "Уведомления"))  
 self.pushButton.setText(\_translate("Form", "Принять"))  
 self.pushButton\_2.close()  
 self.pushButton.clicked.disconnect(self.remove\_pr)  
 self.pushButton.clicked.connect(self.accept)  
  
 def render(self):  
 self.notes = requests.get(f"{BASE\_URL}/list\_notes?u\_id={self.user\_id}").json()  
 self.listWidget.clear()  
 self.listWidget.setColumnCount(4)  
 self.listWidget.setHorizontalHeaderLabels(['Название', 'Дедлайн', 'От', 'Статус'])  
 self.listWidget.setRowCount(len(self.notes))  
 for row in range(len(self.notes)):  
 for col in range(self.listWidget.columnCount()):  
 item = None  
 if col == 0:  
 item = QtWidgets.QTableWidgetItem(f'{self.notes[row]['title']}')  
 elif col == 1:  
 item = QtWidgets.QTableWidgetItem(f'{self.notes[row]['date']}')  
 elif col == 2:  
 item = QtWidgets.QTableWidgetItem(f'{requests.get(f"{BASE\_URL}/get\_name\_for\_notes?task\_id={self.notes[row]['id']}").json()}')  
 elif col == 3:  
 item = QtWidgets.QTableWidgetItem(f'{'Не принято' if self.notes[row]['status'] == 2 else 'Принятно'}')  
 self.listWidget.setItem(row, col, item)  
 def show(self, user\_id, proj\_id=0):  
 self.user\_id = user\_id  
 self.proj\_id = proj\_id  
 self.query = requests.get(f"{BASE\_URL}/list\_proj\_users/{self.proj\_id}").json()  
  
 self.fr = requests.get(f"{BASE\_URL}/get\_name\_for\_notes/").json()  
 self.render()  
  
 self.\_proj\_win.show()