**Содержание**

[**Введение** 2](#_Toc195725804)

[**1. Теоретическая часть** 4](#_Toc195725805)

[*1.1 Техническое задание* 4](#_Toc195725806)

[*1.2. Описание инструментария и его обоснования* 8](#_Toc195725807)

[**2. Практическая часть** 13](#_Toc195725808)

[*2.1 Дизайн приложения* 13](#_Toc195725809)

[*2.2 Описание интерфейса* 14](#_Toc195725810)

[*2.3 Процесс разработки* 22](#_Toc195725811)

[*2.4 Тестирование* 42](#_Toc195725812)

[*2.5 Файловая структура* 43](#_Toc195725813)

[**Заключение** 45](#_Toc195725814)

[Приложение А 47](#_Toc195725815)

[Приложение Б 48](#_Toc195725816)

[Приложение В 49](#_Toc195725817)

# **Введение**

В современном мире командная работа стала неотъемлемой частью образовательной и профессиональной деятельности. Эффективное взаимодействие между участниками группы требует удобных инструментов для общения, распределения задач и отслеживания. Однако многие существующие решения либо сложны в освоении для обычных пользователей, либо не обладают достаточной функциональностью для полноценного управления групповой работой.

Программное обеспечение для организации работы в групповых проектах должно обеспечивать удобные средства коммуникации, позволяя пользователям вести обсуждения в общем чате и личных сообщениях. Важной частью является система управления задачами, где каждый участник может фиксировать свою работу и устанавливать дедлайны.

Кроме того, важной функцией такого программного обеспечения является обеспечение прозрачности и подотчётности внутри команды. Возможность отслеживать задачи позволяет значительно улучшить координацию, предотвратить недоразумения и повысить общую продуктивность группы. Такие инструменты также стимулируют участие каждого члена команды в процессе, а уведомления о предстоящих сроках в проекте помогают избежать задержек и пропуска важной информации.

Актуальность данной темы обусловлена возрастающей потребностью в удобных и эффективных инструментах для организации командной работы. Разработка программного обеспечения, учитывающего современные требования пользователей, способствует оптимизации рабочих процессов и повышению качества выполнения групповых проектов.

Цель: Разработка программного обеспечения для организации работы в групповых проектах.

Задачи:

1. Составление технического задание.
2. Проанализировать и обосновать выбор инструментария.
3. Провести анализ предметной области.
4. Создать макет будущего программного обеспечения.
5. Разработать программное обеспечение.
6. Провести тестирование.
7. Написать инструкцию по инсталляции и эксплуатации.

1. Теоретическая часть

*1.1 Техническое задание*

1. Наименование программного продукта или программы:

«GroupTasker».

2. Краткая характеристика области применения: организация и управление работы в групповых проектах.

3. Основание для разработки

3.1 Основания для проведения разработки: Задание курсовой работы от преподавателя ГБПОУ НСО «Новосибирский химико-технологический колледж им. д. и. Менделеева».

3.2 Наименование и условное обозначение разработки: Программное обеспечение для работы в группах «GroupTasker».

4. Назначение разработки

4.1 Функциональное назначение:

Обеспечение доступа к организации и управлении работы в групповых проектах, создавать группы, писать на доске заметок, переписывать в чатах и расписывать свои задачи.

4.2 Эксплуатационное назначение: для использования приложения пользователями.

5. Требования к функциональным характеристикам.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

6. Состав выполняемых функций:

Функции главного меню:

* Функция кнопки "Создать группу" позволяет перейти на страницу создания новой группы.
* Функция кнопки "Войти в группу" позволяет перейти на страницу входа в существующую группу.

Функции окна создания группы:

* Функция кнопки "Back" позволяет вернуться на главную страницу приложения.
* Функция полей ввода позволяет пользователю ввести название группы, свое имя и пароль.
* Функция кнопки "Создать группу" создает новую группу с указанными параметрами и генерирует уникальный код группы.
* Функция автоматического копирования кода группы после создания упрощает обмен кодом между участниками.

Функции окна входа в группу:

* Функция кнопки "Back" позволяет вернуться на главную страницу приложения.
* Функция полей ввода позволяет пользователю ввести код группы, свое имя и пароль.
* Функция кнопки "Присоединиться к группе" проверяет данные и добавляет пользователя в группу.

Функции главного окна приложения:

* Функция кнопки "Back" позволяет вернуться на главную страницу приложения.
* Функция бокового меню позволяет переключаться между разделами приложения.
* Функция кнопки "Группа" открывает панель управления группой.
* Функция кнопки "Мои задачи" открывает доску заметок группы.
* Функция отображения имени пользователя в верхней панели показывает текущего пользователя.
* Функции панели управления группой:
* Функция отображения названия группы показывает информацию о текущей группе.
* Функция списка участников отображает всех членов группы.
* Функция кнопки "Home" позволяет вернуться к доске заметок.
* Функция кнопки "Исключить участника" (для создателя группы) позволяет удалить участника из группы.
* Функция кнопки "Удалить группу" (для создателя группы) позволяет полностью удалить группу.
* Функции доски заметок:
* Функция кнопки "Добавить заметку" открывает диалоговое окно для создания новой заметки.
* Функция отображения заметок показывает все заметки группы в хронологическом порядке.
* Функция подписи автора у каждой заметки указывает, кто создал заметку.
* Функция кнопки удаления позволяет удалить заметку.
* Функция автоматического обновления списка заметок после изменений.

Дополнительные функции:

* Функция валидации ввода проверяет заполнение обязательных полей.
* Функция отображения сообщений об ошибках информирует пользователя о проблемах.

7. Требования к видам обслуживания:

Для поддержки работоспособности приложения GroupTasker требуется системный администратор.

8. Требования к квалификации пользователя:

Для работы с приложением пользователю необходимо базовое знание ПК на уровне пользователя.

9. Привлечение пользователей:

* Простота интерфейса способствует быстрому освоению приложения.
* Возможность совместной работы в группе повышает полезность продукта.
* Минимальные требования к оборудованию делают приложение доступным.
* Система копирования кода группы упрощает приглашение новых участников.

10. Типы поддерживаемых платформ:

Windows 10/11.

11. Минимальные системные требования:

* Процессор: 1 ГГц или выше
* Оперативная память: 2 ГБ
* Дисковое пространство: 1 ГБ
* Наличие интернет-соединения (мин. 5 Мбит/с)

12. Требования к исходным кодам и языкам программирования:

* Python 3.11
* PyQt6 (для графического интерфейса)
* PostgreSQL (для работы с базой данных)
* psycopg2 (для подключения к БД)

13. Требования к программным средствам:

* Интерпретатор Python
* Библиотека PyQt6
* Система управления базами данных PostgreSQL
* Менеджер пакетов pip

14. Стадии разработки:

* Планирование
* Проектирование
* Разработка
* Тестирование
* Внедрение

15. Этапы и содержание работы:

16. Сроки разработки:

Общий срок разработки проекта: 10 недель.

17. Календарный план работ:

18. Виды испытаний:

* Проверка работы всех кнопок и элементов управления
* Тестирование создания и удаления заметок
* Проверка работы с группами (создание, вход, удаление)
* Проверка корректности работы базы данных

19. Общие требования к приёму работы:

Программный продукт считается готовым, если:

* Все запланированные функции реализованы
* Интерфейс работает корректно
* Обеспечена базовая безопасность данных
* Написана сопроводительная документация

20. Порядок контроля и приемки:

* Демонстрация работоспособности основных функций
* Проверка соответствия техническому заданию
* Тестирование на различных платформах
* Проверка документации

*1.2. Описание инструментария и его обоснования*

*1.2.1 Выбор языка программирования*

Язык программирования является фундаментальным инструментом для создания программного обеспечения. Среди множества существующих языков (более 300) для реализации данного проекта требовалось выбрать наиболее подходящий, способный обеспечить выполнение всех поставленных задач. Был проведен сравнительный анализ современных языков программирования.

Python — интерпретируемый высокоуровневый язык общего назначения с динамической строгой типизацией и автоматическим управлением памятью. Основные особенности:

* Ориентирован на повышение производительности разработчика
* Обеспечивает высокую читаемость кода
* Поддерживает несколько парадигм программирования
* Имеет обширную стандартную библиотеку
* Обеспечивает переносимость программ

Для данного проекта Python был выбран благодаря:

* Простоте создания графических интерфейсов (библиотека PyQt6)
* Удобству работы с базами данных (psycopg2 для PostgreSQL)
* Кроссплатформенности
* Быстроте разработки
* Большому количеству специализированных библиотек

Среди альтернативных языков рассматривались:

C++ — компилируемый статически типизированный язык общего назначения. Основные характеристики:

* Поддерживает процедурное, ООП и обобщенное программирование
* Имеет богатую стандартную библиотеку
* Позволяет низкоуровневый доступ к памяти
* Требует ручного управления ресурсами
* Сложный синтаксис по сравнению с Python

Области применения C++:

* Системное программирование
* Разработка игр
* Высоконагруженные серверы
* Встраиваемые системы

Java — компилируемый объектно-ориентированный язык со строгой типизацией. Особенности:

* Исполняется на виртуальной машине JVM
* Автоматическое управление памятью
* Богатая стандартная библиотека
* Менее удобен для создания GUI чем Python+Qt

JavaScript/TypeScript — основные языки веб-разработки. Характеристики:

* Динамическая типизация (статическая в TypeScript)
* Асинхронная модель выполнения
* Огромная экосистема фреймворков
* Оптимальны для веб-приложений
* Менее подходят для десктопных программ

Для данного проекта Python показал себя как оптимальный выбор, сочетающий простоту разработки, богатые возможности для создания GUI и работы с БД, а также кроссплатформенность.

*1.2.2 Выбор среды разработки*

Интегрированная среда разработки (IDE) — важный инструмент, значительно повышающий продуктивность программиста. Для проекта рассматривались несколько популярных IDE.

PyCharm — специализированная среда для Python-разработки от JetBrains. Основные возможности:

* Полноценная поддержка Python и его экосистемы
* Интеллектуальное автодополнение кода
* Визуальный отладчик с точками останова
* Интеграция с системами контроля версий
* Поддержка научных инструментов
* Инструменты для работы с базами данных
* Возможности рефакторинга

Visual Studio Code — легковесный кроссплатформенный редактор кода. Особенности:

* Поддержка множества языков через расширения
* Встроенный терминал
* Интеграция с Git
* Широкая кастомизация
* Меньшая "из коробки" поддержка Python, чем у PyCharm

IDLE — стандартная среда разработки, поставляемая с Python. Характеристики:

* Минималистичный интерфейс
* Базовые возможности редактирования
* Отсутствие продвинутых инструментов
* Нет интеграции с современными системами разработки

Для данного проекта была выбрана среда PyCharm, так как она:

* Предоставляет наиболее полный набор инструментов именно для Python
* Имеет отличную поддержку PyQt6
* Облегчает работу с базами данных
* Включает мощный отладчик
* Поддерживает все современные практики разработки

PyCharm значительно ускоряет процесс создания приложений за счет интеллектуальных функций автодополнения, навигации по коду и встроенных инструментов тестирования, что делает его оптимальным выбором для данного проекта.

*1.2.3 Выбор библиотек и модулей*

Для реализации всех функций проекта были использованы различные библиотеки и модули, которые значительно расширили возможности программы и позволили реализовать задуманный функционал. К таким модулям и библиотекам относятся:

* PyQt6 - современная библиотека для создания графических интерфейсов. Предоставляет богатый набор виджетов и инструментов для разработки кроссплатформенных приложений с профессиональным интерфейсом. Включает систему сигналов и слотов для обработки событий, поддерживает стилизацию элементов и позволяет создавать сложные многооконные приложения.
* psycopg2 - адаптер PostgreSQL для Python, обеспечивающий надежное взаимодействие с базой данных. Поддерживает все основные функции СУБД, включая транзакции, подготовленные запросы и работу с различными типами данных. Позволяет эффективно выполнять SQL-запросы и обрабатывать результаты.
* random - стандартный модуль Python для работы со случайными числами. Использовался для генерации уникальных кодов групп. Модуль предоставляет криптографически безопасные методы генерации и различные распределения случайных величин.
* pyperclip - простая в использовании библиотека для работы с буфером обмена. Обеспечивает кроссплатформенный доступ к системному буферу, что позволяет удобно копировать и вставлять данные между приложением и другими программами.
* sys - базовый модуль Python, предоставляющий доступ к системным функциям интерпретатора. Использовался для обработки аргументов командной строки и управления завершением работы программы.

1.3 Анализ предметной области

Перед разработкой программного обеспечения для группового управления задачами был проведен анализ существующих решений. Рассмотрим ключевые аналоги, их сильные и слабые стороны:

Trello − Один из самых популярных сервисов для управления задачами по методологии Kanban. Его главное преимущество — визуальная простота: задачи представлены в виде карточек, которые можно перемещать между колонками. Это делает Trello удобным для небольших проектов и личного использования. Однако сервис слабо подходит для сложных проектов из-за ограниченных возможностей хранения данных и отсутствия встроенных коммуникационных инструментов.

Asana − Более профессиональный инструмент с мощной системой планирования. Asana предлагает иерархическую структуру задач, временные линии и аналитику, что делает её привлекательной для командной работы. Однако её интерфейс перегружен функциями, что усложняет освоение для новичков. Кроме того, многие полезные возможности (например, расширенная аналитика) доступны только в платной версии.

Notion − Универсальный инструмент, сочетающий в себе систему управления задачами, базу знаний и среду для совместной работы. Его гибкость позволяет адаптировать его под разные нужды, но это же является и недостатком: для эффективного использования Notion требуется время на настройку. Также новичкам может быть сложно разобраться во всех возможностях сервиса.

Microsoft Teams − Корпоративный инструмент, ориентированный на коммуникацию и совместную работу. Он интегрирован с Office 365, что делает его удобным для компаний, использующих экосистему Microsoft. Однако управление задачами в Teams реализовано неудобно, а интерфейс слишком сложен для небольших команд или личного использования.На основе проведенного анализа можно выделить ключевые требования к разрабатываемому решению:

* Простота и интуитивность интерфейса.
* Возможность групповой работы.
* Гибкие права доступа.
* Минимальные системные требования.
* Локальное хранение данных при необходимости.
* Простота создания и подключения к группам.
* Возможность быстрого обмена информацией.
* Отсутствие сложных настроек.

**2. Практическая часть**

*2.1 Дизайн приложения*

Дизайн приложения выполнен в светлых тонах с акцентами в природных оттенках. Основные элементы:

1. Цветовая гамма:

* Фон: светло-серый (#F0F0F0)
* Основные элементы: светло-бежевый (#E0E2DB)
* Текст и акценты: темно-зеленый (#003C30) и серо-зеленый (#5F7470)
* Ошибки/удаление: красный (#FF6961)

1. Шрифты:

* Используется чистый современный шрифт Inter
* Размеры варьируются для удобного восприятия (заголовки крупные, основной текст читаемый)

1. Компоновка:

* Верхняя панель с навигацией
* Боковое меню для переключения между разделами
* Основная рабочая область с единым стилем всех вкладок

1. Интерактивные элементы:

* Кнопки с эффектами при наведении
* Скругленные углы (15px) для плавности интерфейса

Дизайн минималистичный, удобный и интуитивно понятный, что делает приложение комфортным для пользователя.

*2.2 Описание интерфейса*

Интерфейс приложения разработан с упором на удобство использования и интуитивную навигацию. Он включает несколько ключевых экранов, каждый из которых выполняет определенную функцию и обеспечивает пользователю комфортную работу с задачами, заметками и групповой коммуникацией.

Основные принципы построения интерфейса:

* Простота – минимум лишних элементов, понятное расположение кнопок и меню.
* Единообразие – все экраны выдержаны в едином стиле для быстрого привыкания.
* Функциональность – каждая вкладка решает конкретную задачу без перегруженности опциями.

Меню входа (смотреть Рисунок 1):

* Кнопка «Создать группу» позволяет перейти на страницу создания группы.
* Кнопка «Войти в группу (код группы)» позволяет перейти на страницу входа в группу.



Рисунок 1 – Меню входа

Окна создания группы (смотреть Рисунок 2):

* В поле ввода «Название группы» нужно ввести название группы.
* В поле ввода «Ваше имя» нужно придумать имя пользователя.
* В поле ввода «Пароль» нужно придумать пароль для входа в аккаунт.



Рисунок 2 – Окно создания группы

Окно входа в группу (смотреть Рисунок 3):

* В поле ввода «Код группы» нужно ввести код группы, который получается при создании группы.
* В поле ввода «Ваше имя» нужно придумать или написать существующее имя пользователя.
* В поле ввода «Пароль» нужно придумать или написать существующий пароль для входа в аккаунт.

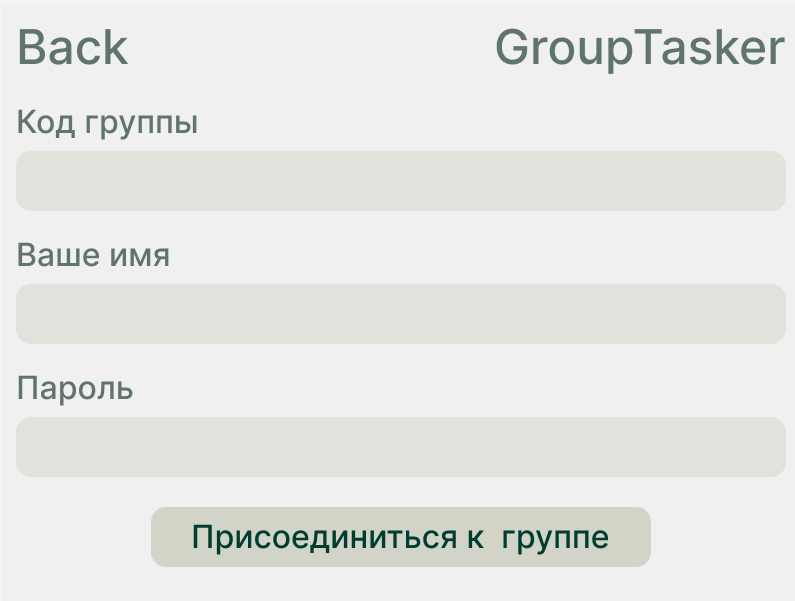


Рисунок 3 – Окно входа в группу

Главное окно приложения (смотреть Рисунок 4):

* Кнопка «Back» позволяет вернуться в меню входа.
* Кнопка «Добавить заметку» открывает окно добавления заметки на доску заметок.
* Кнопка «Группа» позволяет перейти в вкладку с группой.
* Кнопка «Мои задачи» позволяет перейти в вкладку с личными задачами.
* Кнопка «Общий чат» позволяет перейти в чат группы.
* Кнопка «Личные сообщения» позволяет перейти в вкладку, где можно написать сообщение участнику группы.

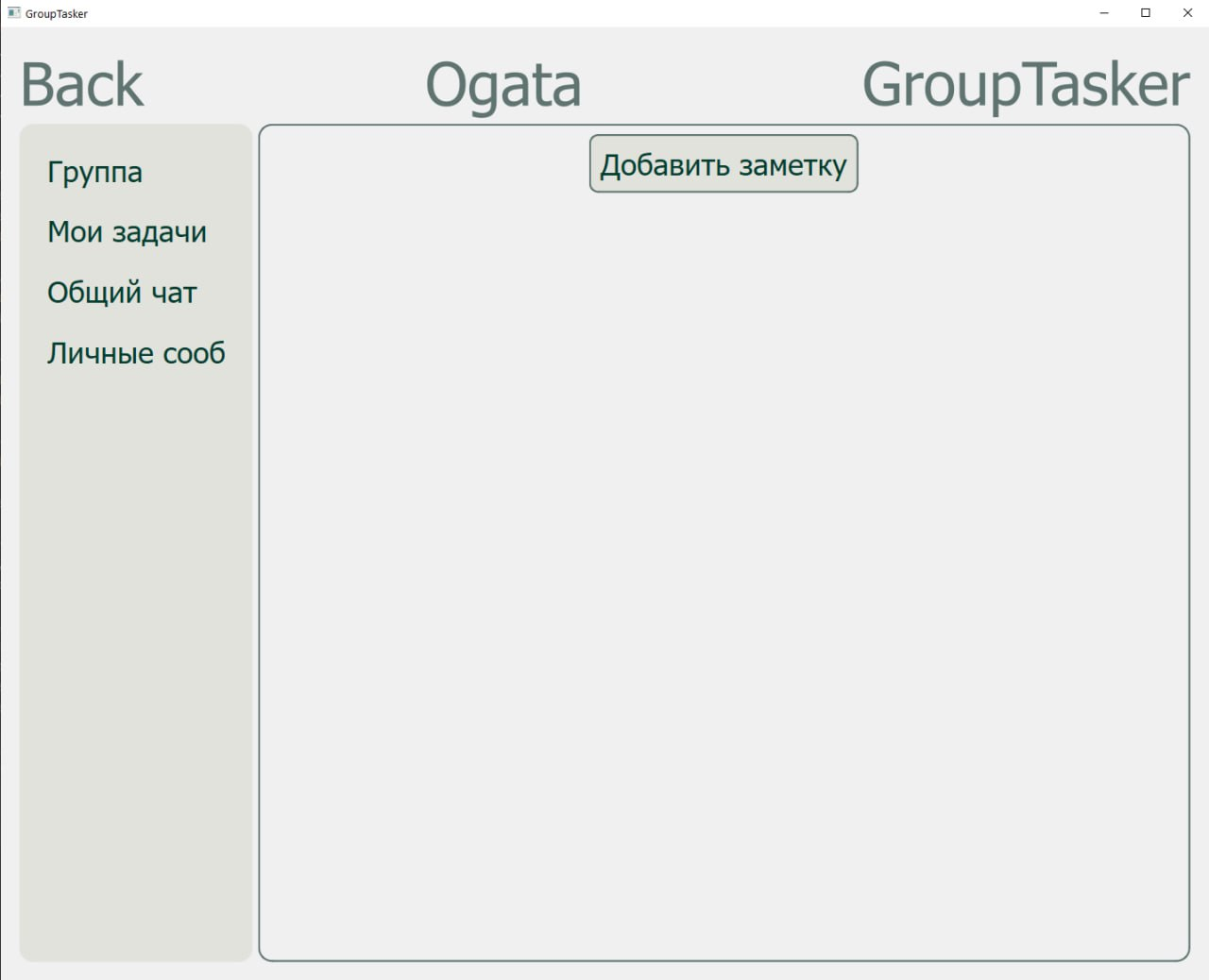


Рисунок 4 – Главное окно приложения

Вкладка «Группа» (смотреть Рисунок 5):

* Кнопка «Home» позволяет вернуться к доске заметок.
* Кнопка «Исключить участника» позволяет исключить выбранного участника.
* Кнопка «Удалить группу» позволяет полностью удалить группу.
* Кнопка «Покинуть группы» позволяет выйти из группы.

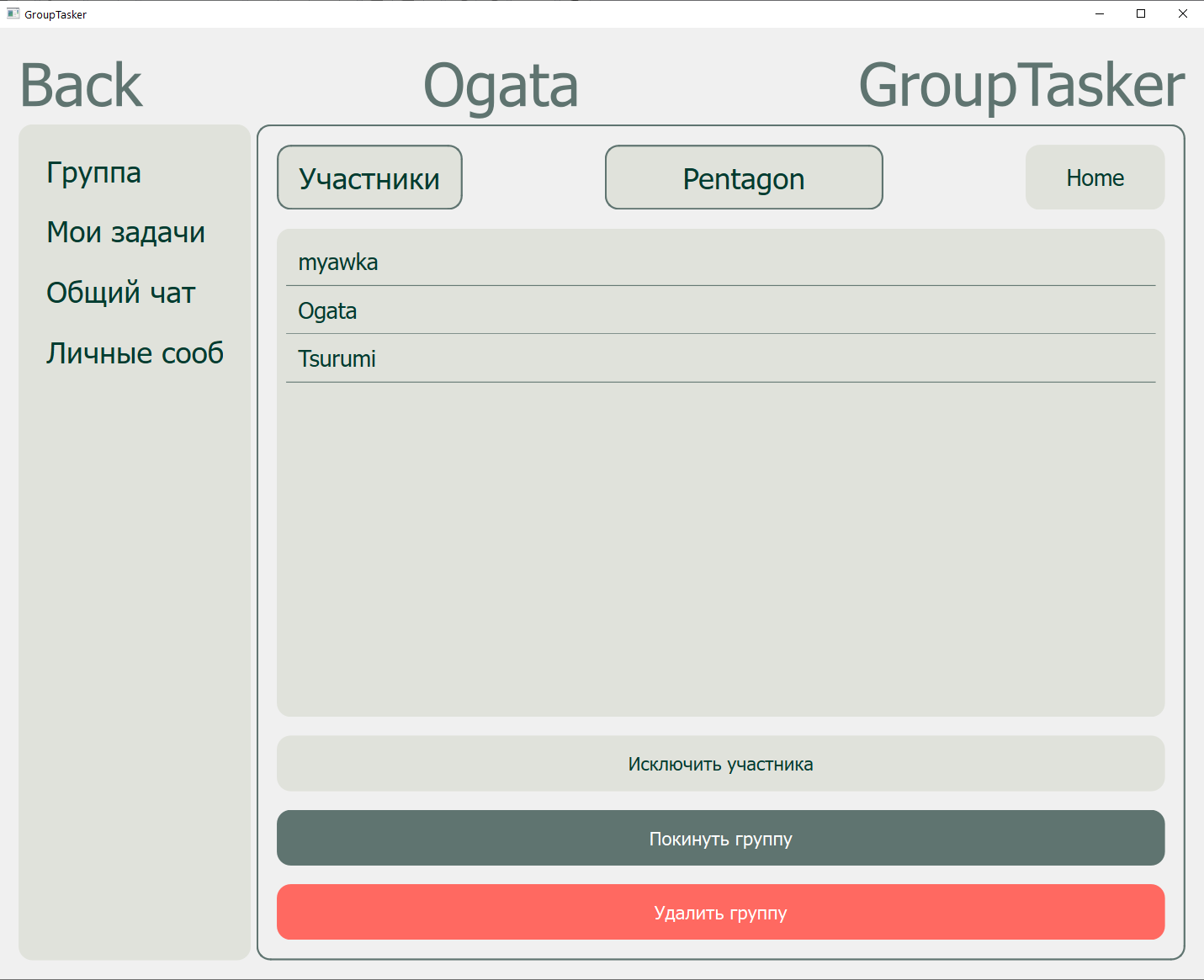


Рисунок 5 – Группа

Вкладка «Мои задачи» (смотреть Рисунок 6):

* Кнопка «Добавить задачу» позволяет добавить расписанную задачу с заданным дедлайном.
* Кнопка «Home» позволяет вернуться к доске заметок.



Рисунок 6 – Мои задачи

Вкладка «Общий чат» (смотреть Рисунок 7):

* Кнопка «Home» позволяет вернуться к доске заметок.
* В поле внизу нужно написать сообщение, после нажатия на кнопку «Отправить» сообщение отправляется в общий чат, которое увидят все участники группы.
* Кнопка «удалить» позволяет удалить сообщение, которое было отправлено в чат.

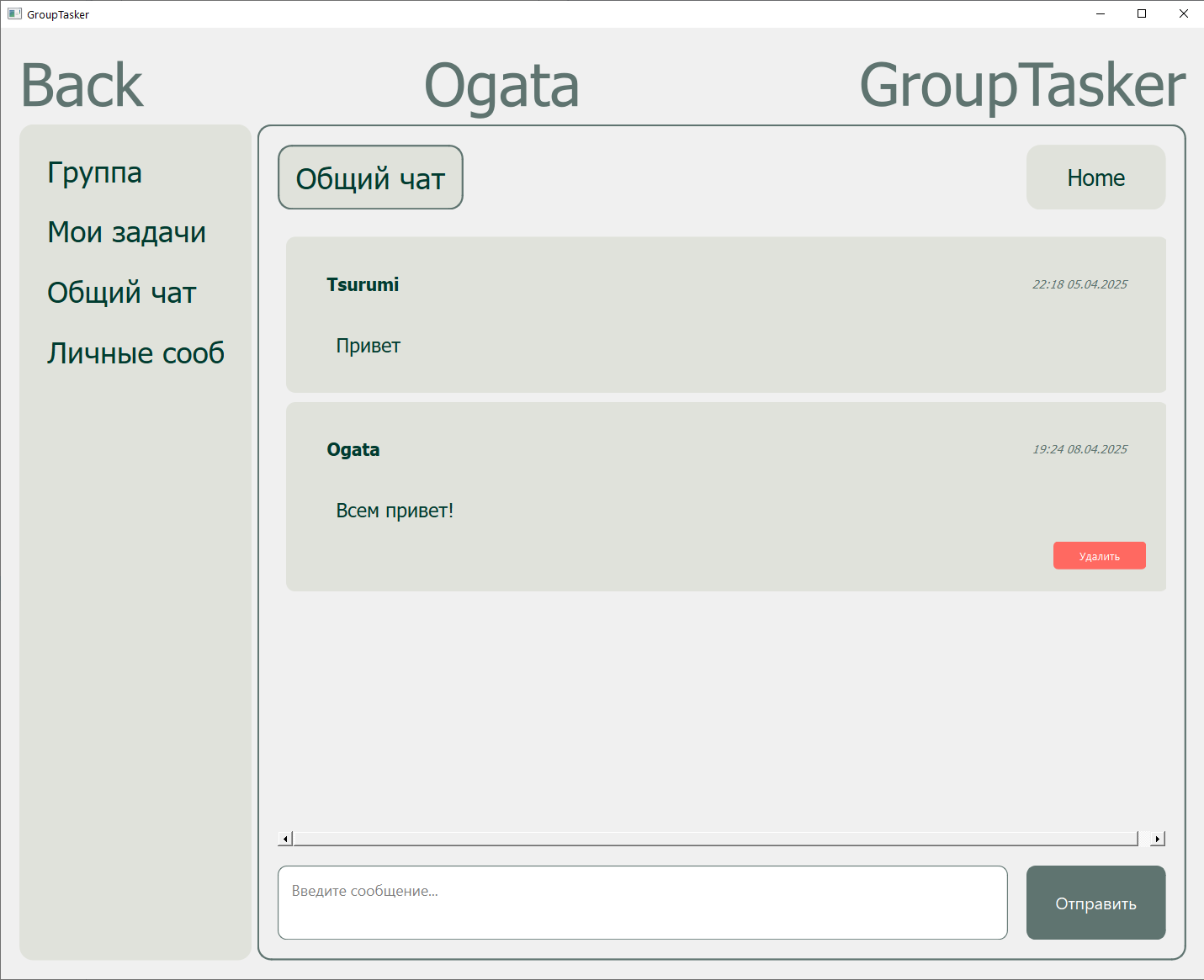


Рисунок 7 – Общий чат

Вкладка «Личные сообщения» (смотреть Рисунок 8):

* Кнопка «Home» позволяет вернуться к доске заметок.
* Кнопка «удалить» позволяет удалить сообщение, которое было отправлено в чат.
* В поле внизу нужно написать сообщение, после нажатия на кнопку «Отправить» сообщение отправляется в личный чат с выбранным пользователем.

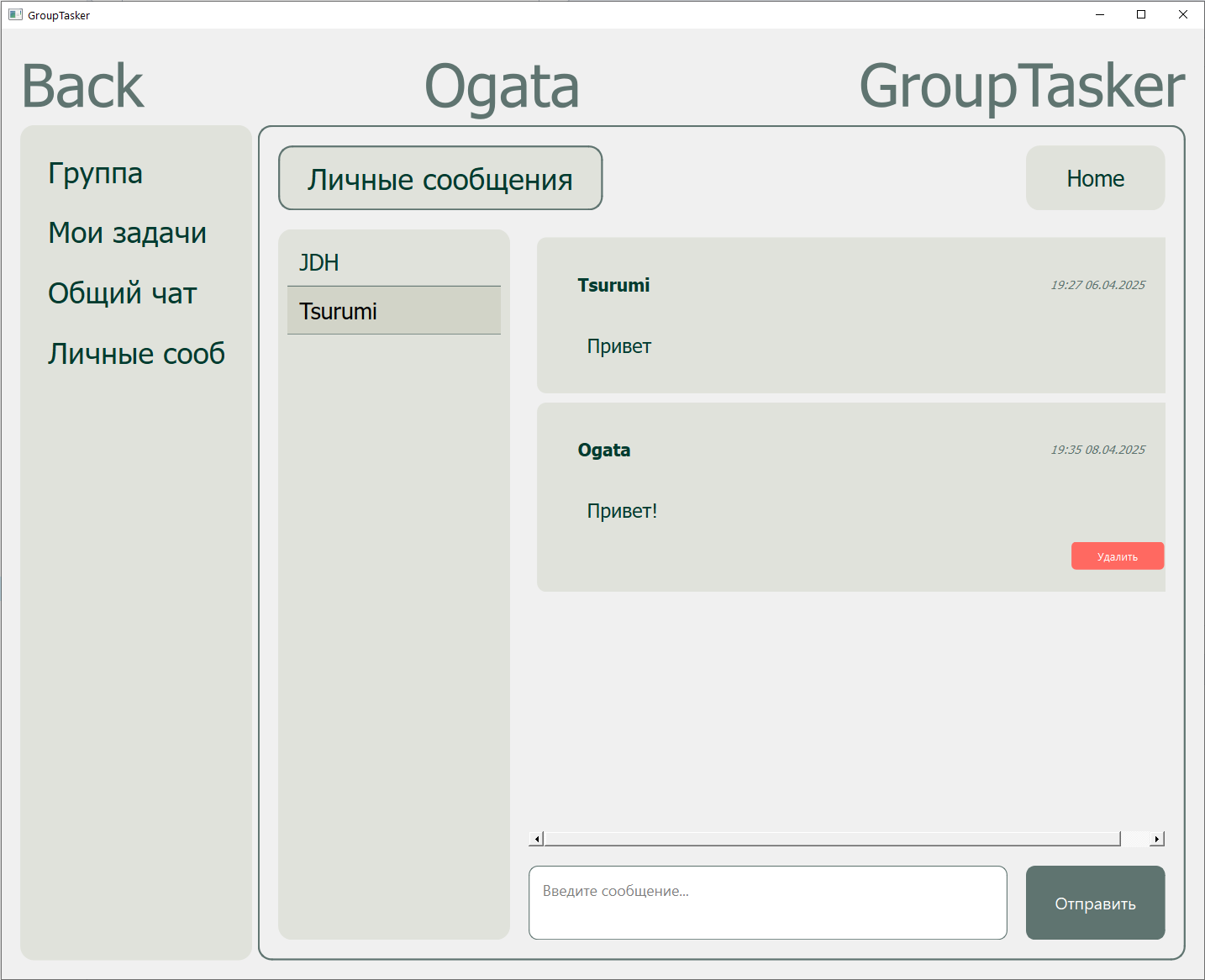


Рисунок 8 – Личные сообщения

*2.3 Процесс разработки*

Разработка приложения началась с создания главного окна приложения и реализации начальной логики перехода между экранами. Интерфейс был реализован с использованием библиотеки PyQt6, которая предоставляет мощный инструментарий для создания графических приложений.

Окно входа в приложение (main.py).

Данный модуль является точкой входа в приложение. В нем создается главное окно программы и определяются основные элементы управления: заголовок, кнопки «Создать группу» и «Войти в группу (код группы)», а также логика переключения экранов.

Метод init\_ui(). Данный метод отвечает за инициализацию пользовательского интерфейса. Здесь задаются размеры окна, стиль, создаются элементы интерфейса и добавляются в вертикальный layout. Также реализуется стек экранов через QStackedWidget, который позволяет в будущем переключаться между разными окнами приложения (смотреть Рисунок 9).

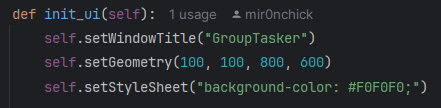


Рисунок 9 − Метод init\_ui()

Элементы интерфейса.

Создан заголовок с названием приложения и две основные кнопки (смотреть Рисунок 10), каждая кнопка оформлена в едином стиле и закреплена в layout так, чтобы интерфейс оставался аккуратным и логичным.



Рисунок 10 − Основные кнопки

Метод create\_group\_window() и join\_group\_window() (смотреть Рисунок 11).

Открывает окно создания новой группы. При нажатии на соответствующую кнопку создается экземпляр окна GroupCreateWindow и происходит переключение на него, аналогично, при нажатии на кнопку входа в существующую группу происходит переход на экран GroupJoinWindow.

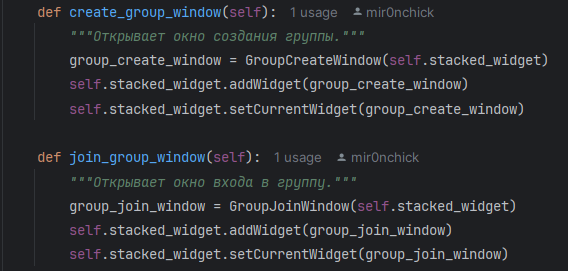


Рисунок 11 − Открытие окон создания группы или входа в неё

Файл groupcreate.py.

Модуль groupcreate.py отвечает за окно, в котором пользователь может создать новую группу. После успешного ввода данных приложение создаёт уникальный код группы, сохраняет информацию в базе данных, копирует код в буфер обмена и открывает главное окно.

Класс GroupCreateWindow (смотреть Рисунок 12) наследуется от QWidget и принимает stacked\_widget − контейнер, управляющий сменой экранов в приложении.

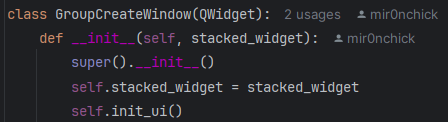


Рисунок 12 − GroupCreateWindow

Метод init\_ui().

Метод init\_ui инициализирует весь графический интерфейс окна. В нём задаются размеры окна, фоновый цвет и структура интерфейса (смотреть Рисунок 13). Верхняя часть состоит из заголовка с надписью "GroupTasker" и кнопки "Back", по которой можно вернуться на главный экран (смотреть Рисунок 14).

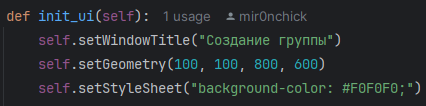


Рисунок 13 − Метод init\_ui()

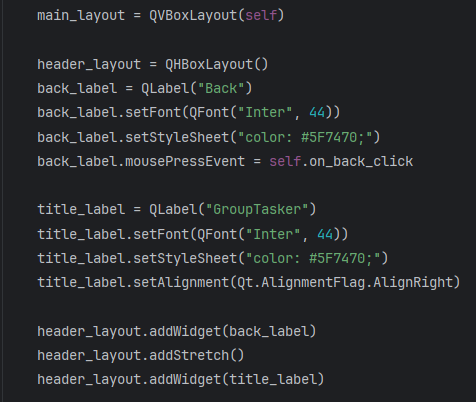


Рисунок 14 − Заголовок и кнопка Back

Далее идёт создание полей для ввода (смотреть Рисунок 15):

* Название группы
* Имя пользователя
* Пароль

Для упрощения кода была добавлена вспомогательная функция create\_input\_field().

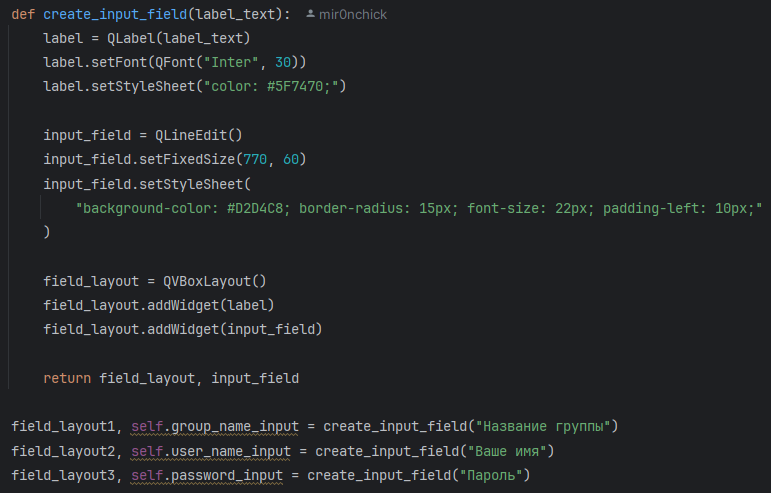


Рисунок 15 − Создание полей для ввода

После делается кнопка для создания группы (смотреть Рисунок 16).

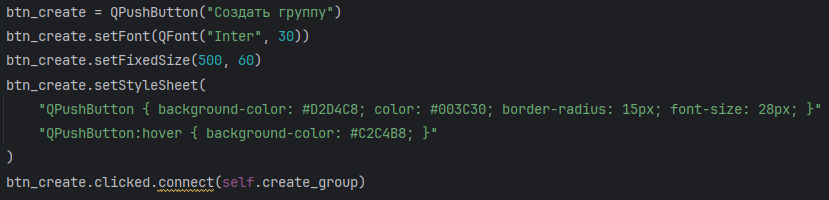


Рисунок 16 − Кнопка для создания группы

Метод create\_group() (смотреть Рисунок 17).

Этот метод срабатывает при нажатии кнопки "Создать группу". Его задача получить данные из полей, проверить их заполненность, сгенерировать уникальный код группы, сохранить информацию в БД и открыть главное окно приложения.

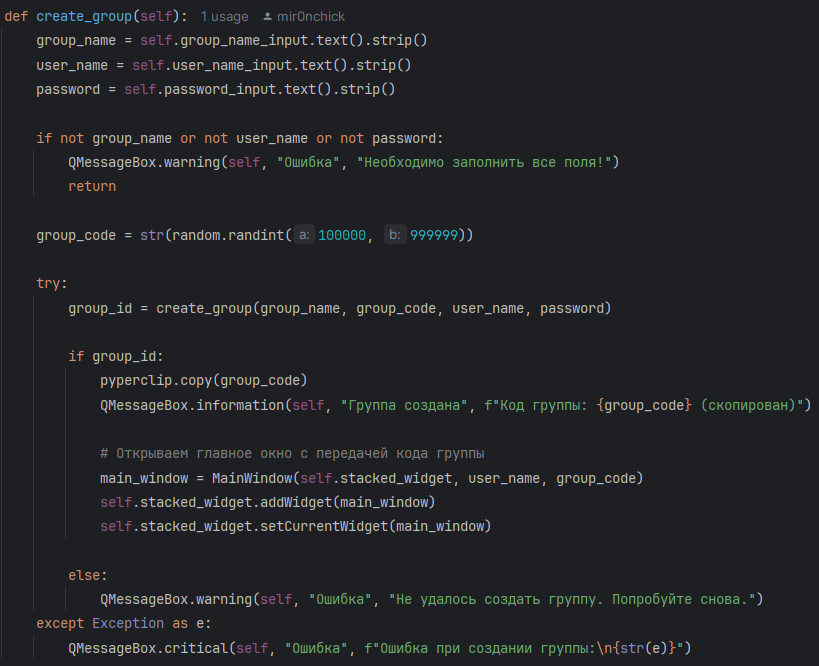


Рисунок 17 − Создание группы (метод create\_group())

Файл groupjoin.py.

Модуль groupjoin.py реализует окно входа в уже существующую группу. Пользователь вводит код группы, имя и пароль. Если данных достаточно, происходит проверка на существование группы и пользователя. При необходимости пользователь добавляется в базу и открывается MainWindow.

Метод init\_ui().

Метод строит форму, аналогичную по дизайну форме создания группы. Используются всё те же элементы:

* Заголовок
* Кнопка "Back"
* Три поля: код группы, имя, пароль
* Кнопка "Присоединиться к группе"

Метод join\_group() (смотреть Рисунок 18).

Проверяет правильность введённых данных, наличие группы и пользователя. Добавляет пользователя, если его не было.

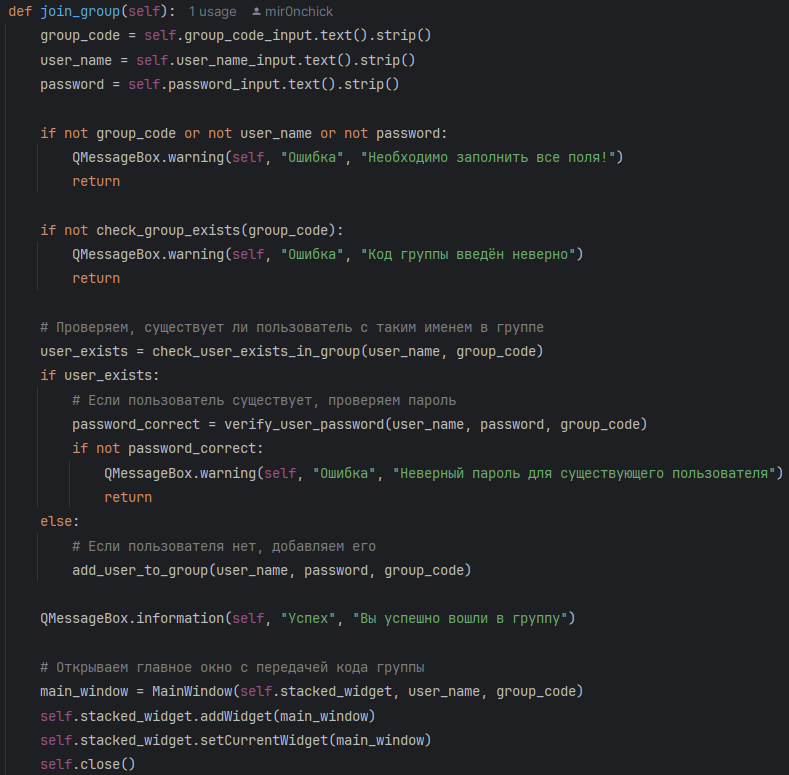


Рисунок 18 − Метод join\_group()

Файл main\_window.py.

Данный модуль реализует основное окно приложения, которое отображается после успешного входа или создания группы. Это центральный интерфейс, предоставляющий доступ ко всем основным функциям: доске заметок, задачам, общему и личным чатам, а также информации о группе.

Верхняя панель управления (смотреть Рисунок 19).

Содержит три элемента: кнопку "Back", имя пользователя и название приложения. При клике на "Back" пользователь возвращается в главное меню.

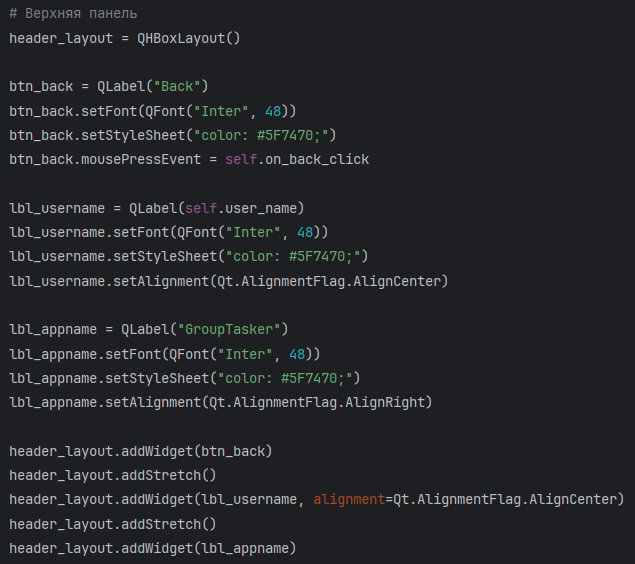


Рисунок 19 − Верхняя панель управления

Боковое меню.

Создано с использованием QFrame и QVBoxLayout. В нём размещены кнопки переключения между разделами (смотреть Рисунок 20), кнопки переключают содержимое QStackedWidget (смотреть Рисунок 21).

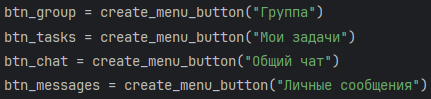


Рисунок 20 − Кнопки переключения между разделами

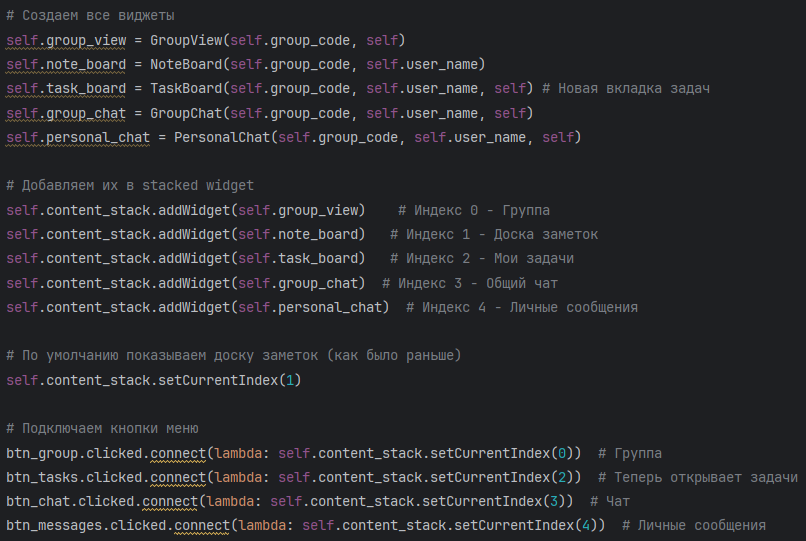


Рисунок 21 − Создание виджетов и их переключение

Файл note\_board.py.

Модуль note\_board.py реализует функциональность доски с заметками, доступной всем участникам группы. Каждый участник может оставить запись, просматривать чужие, а также удалять свои. Заметки хранятся в базе данных и синхронизируются при каждом обновлении окна.

Кнопка добавления новой заметки (смотреть Рисунок 22)

При нажатии на кнопку "Добавить заметку" появляется всплывающее диалоговое окно, где пользователь может ввести текст заметки. Визуально кнопка оформлена в стиле приложения и размещается в верхней части интерфейса.



Рисунок 22 − Кнопка добавления заметки

Метод add\_note\_to\_board() (смотреть Рисунок 23).

Добавляет на экран новый блок заметки. Каждая заметка содержит:

* Основной текст;
* Имя автора;
* Кнопку удаления.

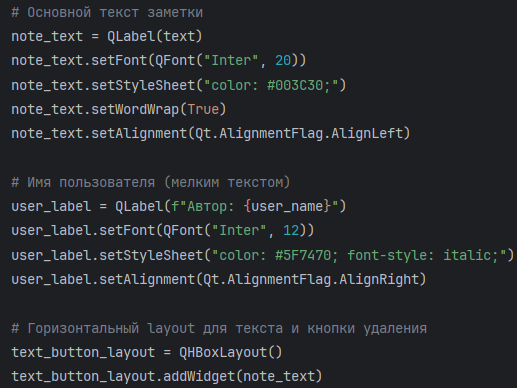


Рисунок 23 − Блок заметок

Метод add\_note() (смотреть Рисунок 24).

Метод отвечает за добавление новой заметки как на доску, так и в базу данных. Если текст был успешно введён, вызывается функция save\_note() из модуля database, после чего заметка отображается на экране.

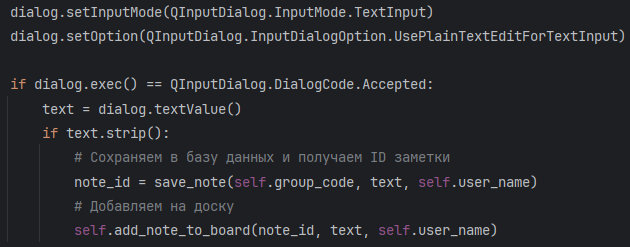


Рисунок 24 − Добавление новой заметки

Метод load\_notes() (смотреть Рисунок 25).

При открытии доски происходит загрузка всех сохранённых заметок группы из базы данных. Для каждой записи вызывается add\_note\_to\_board(), перед загрузкой все текущие виджеты очищаются.

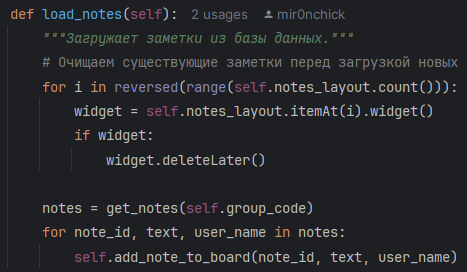


Рисунок 25 − Загрузка сохранённых заметок

Метод remove\_note(note\_id) (смотреть Рисунок 26).

Удаляет заметку по идентификатору с помощью функции delete\_note() и обновляет список.

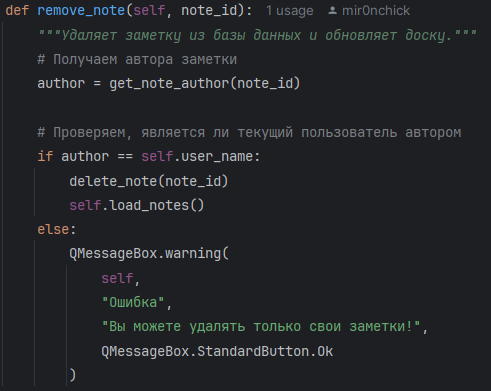


Рисунок 26 − Удаления заметок

Файл group\_view.py.

Модуль group\_view.py реализует интерфейс для просмотра и управления группой. Это окно отображает название группы, список её участников, а также позволяет создателю группы удалять участников и саму группу.

Верхняя панель.

В верхней части размещаются:

* Заголовок "Участники";
* Название группы (получается из базы данных);
* Кнопка Home, возвращающая пользователя на доску заметок (смотреть Рисунок 27).



Рисунок 27 − Кнопка Home

Загрузка данных о группе.

Метод load\_group\_data() выполняет SQL-запросы (смотреть Рисунок 28) к базе данных PostgreSQL и:

* Получает название группы;
* Определяет, является ли текущий пользователь создателем;
* Загружает список участников.

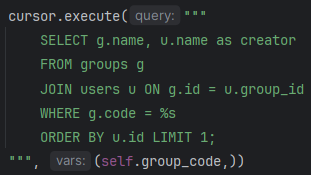


Рисунок 28 − SQL-запросы для получения информации

В случае, если текущий пользователь является создателем группы, активируются кнопки "Удалить участника" и "Удалить группу" (смотреть Рисунок 29).

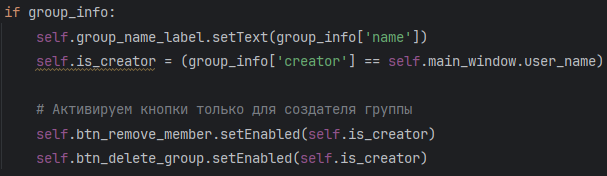


Рисунок 29 − Активация кнопок для админа

Исключение участника.

Кнопка "Исключить участника" активна только для создателя группы. При её нажатии вызывается метод remove\_member() (смотреть Рисунок 30).



Рисунок 30 − Кнопка для исключения участника

Пользователь выбирается из списка, и после подтверждения удаляется SQL-запросом (смотреть Рисунок 31).

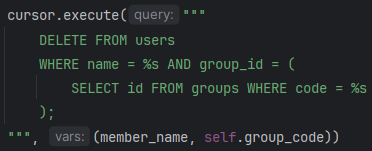


Рисунок 31 − SQL-запрос для исключения участника

Удаление группы.

Если создатель решит удалить группу, вызывается метод delete\_group(). Действие подтверждается через диалог QMessageBox (смотреть Рисунок 32).

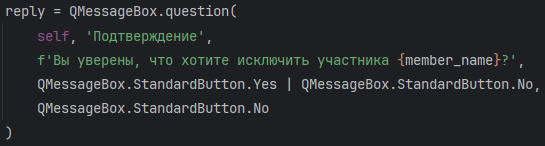


Рисунок 32 − Подтверждение удаления группы

Если пользователь подтвердит, группа будет удалена каскадно (вместе с участниками, задачами, заметками и пр.) (смотреть Рисунок 33).



Рисунок 33 − SQL-запрос для удаления группы

Файл task\_board.py.

Модуль task\_board.py реализует интерфейс управления личными задачами пользователя внутри группы. Здесь отображаются активные задачи, реализована возможность их создания, удаления и отслеживания дедлайна.

Верхняя панель и добавление задачи.

Верхняя панель содержит заголовок, кнопку «Добавить задачу» (смотреть Рисунок 34), а также кнопку Home, которая возвращает доску заметок (смотреть Рисунок 35).



Рисунок 34 − Добавление задачи



Рисунок 35 − Кнопка Home

Метод add\_task() (смотреть Рисунок 36).

Открывает диалоговое окно TaskDialog, где пользователь может ввести:

* Название;
* Описание;
* Дедлайн (с использованием QDateEdit с календарём).

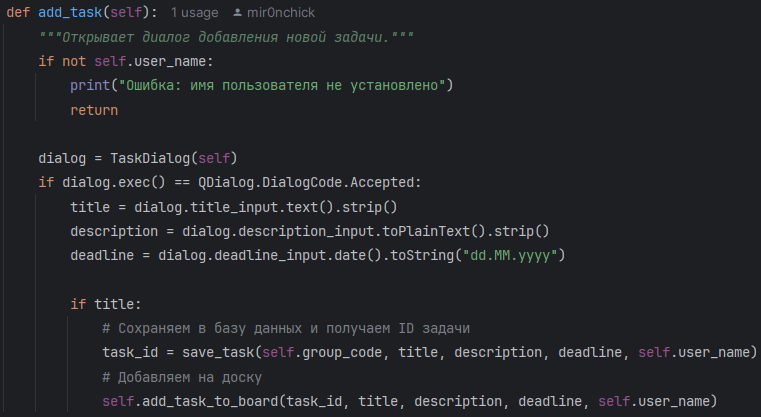


Рисунок 36 − Окно добавления задачи

Метод load\_tasks().

Загружает список задач конкретного пользователя из базы данных с помощью функции get\_tasks() (смотреть Рисунок 37), перед загрузкой все текущие задачи очищаются.

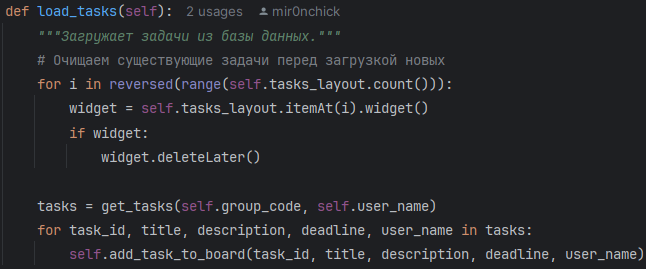


Рисунок 37 − Загрузка задач из базы данных

Метод add\_task\_to\_board().

Создаёт визуальный блок. Содержит:

* заголовок;
* описание;
* дедлайн;
* оценку оставшегося времени (дни, сегодня, просрочено);
* кнопку удаления.

Метод remove\_task() (смотреть Рисунок 38)

Удаляет задачу из базы данных и перезагружает интерфейс.

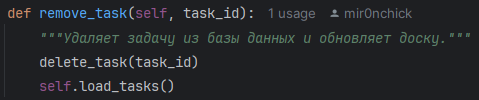


Рисунок 38 − Удаление задачи

Файл group\_chat.py.

Модуль group\_chat.py реализует общий чат, в котором все участники группы могут обмениваться сообщениями в реальном времени. Сообщения сохраняются в базе данных и отображаются с авторством, временем, и возможностью удаления своих сообщений. Обновление чата происходит как по таймеру, так и принудительно при отправке.

Верхняя панель содержит:

* Заголовок "Общий чат";
* Кнопку Home, возвращающую пользователя на доску заметок.

Поле ввода сообщения и кнопка «Отправить» размещены внизу.

Загрузка и отображение сообщений

Метод load\_all\_messages() загружает сообщения из базы и сохраняет их идентификаторы для последующего отслеживания изменений (смотреть Рисунок 39).

Метод clear\_messages() очищает удалённые сообщения (смотреть Рисунок 40).

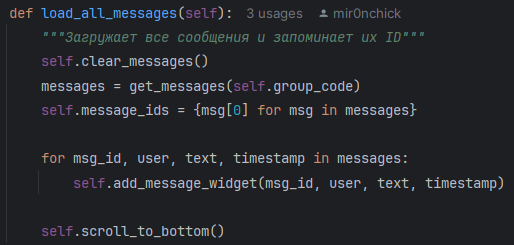


Рисунок 39 − Загрузка сообщений из базы данных

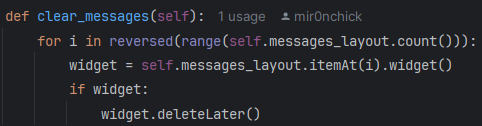


Рисунок 40 − Чистка сообщений

Метод send\_message().

Отправляет сообщение в базу данных, очищает поле ввода и инициирует обновление интерфейса (смотреть Рисунок 41).

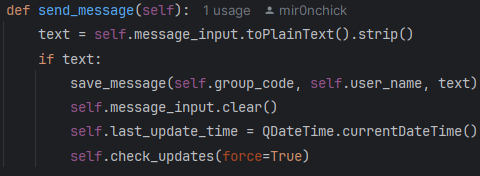


Рисунок 41 − Отправление сообщений

Метод delete\_message().

Удаляет сообщение по ID после подтверждения от пользователя (смотреть Рисунок 42).

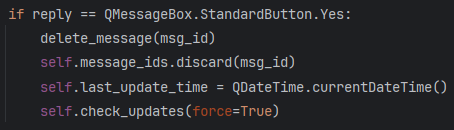


Рисунок 42 − Удаление сообщений

Обновление чата происходит автоматически по таймеру (смотреть Рисунок 43). После метод check\_updates() сравнивает количество сообщений и ID, чтобы определить, были ли изменения (смотреть Рисунок 44).

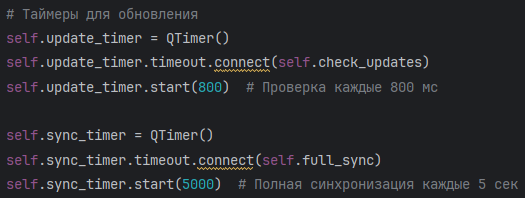


Рисунок 43 − Таймер обновления чата

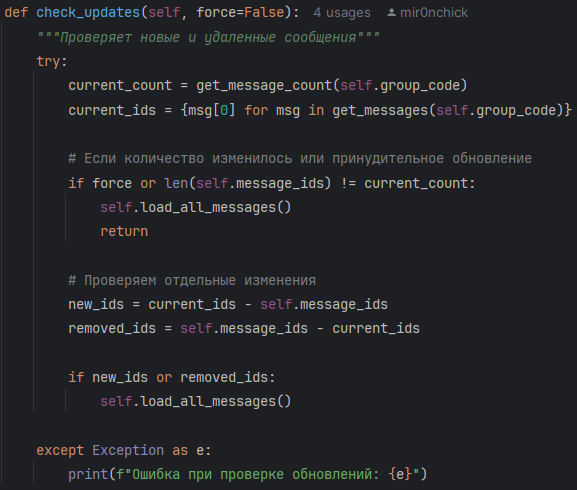


Рисунок 44 − Проверка новых и удалённых сообщений

Файл personal\_chat.py.

Модуль personal\_chat.py реализует вкладку с личными сообщениями. Он позволяет каждому участнику группы вести приватную переписку с другими участниками. Интерфейс чата делится на две части: список пользователей и окно сообщений. Также реализована возможность отправки, получения и удаления сообщений.

Интерфейс и структура

Главная часть интерфейса делится на два блока:

* Слева список пользователей группы;
* Справа чат с выбранным пользователем.

Поле ввода сообщения и кнопка «Отправить» размещены внизу правой части, в верхней части присутствует кнопка «Home», возвращающая пользователя на доску заметок.

Загрузка списка пользователей (смотреть Рисунок 45).

Метод load\_group\_users() запрашивает список всех участников группы, кроме текущего пользователя.

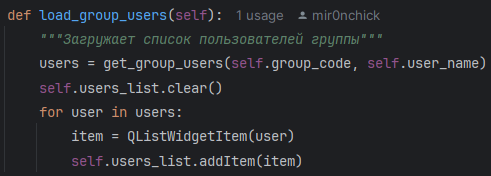


Рисунок 45 − Загрузка списка пользователей

Выбор собеседника (смотреть Рисунок 46).

При выборе пользователя в списке срабатывает метод on\_user\_selected(), который вызывает загрузку сообщений.

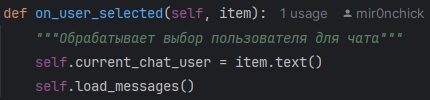


Рисунок 46 − Выбор собеседеника

Загрузка сообщений (смотреть Рисунок 47).

Метод load\_messages() получает все личные сообщения между текущим пользователем и выбранным собеседником, идентификаторы сообщений сохраняются в self.message\_ids, чтобы отслеживать изменения.

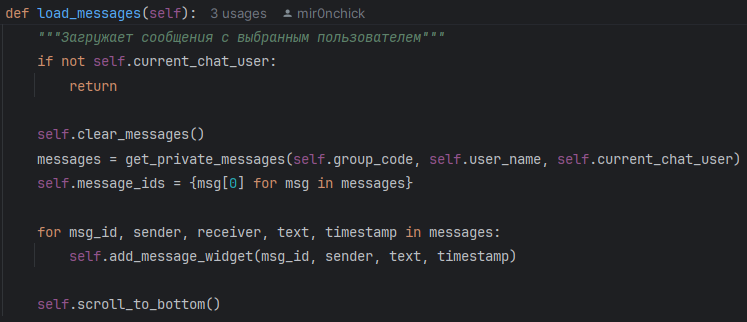


Рисунок 47 − Загрузка сообщений

Отправка сообщений (смотреть Рисунок 48).

Метод send\_message() обрабатывает нажатие кнопки «Отправить». Сообщение сохраняется в базе и запускается проверка обновлений:

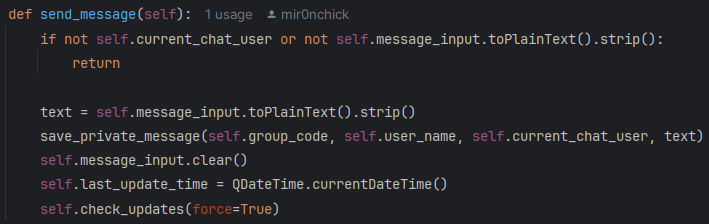


Рисунок 48 − Отправка сообщений

Удаление сообщений и обновление чата.

Метод delete\_message() удаляет сообщение после подтверждения пользователя (смотреть Рисунок 42). Обновление чата происходит автоматически по таймеру (смотреть Рисунок 43). После метод check\_updates() сравнивает количество сообщений и ID, чтобы определить, были ли изменения (смотреть Рисунок 44).

Файл database.py.

Модуль database.py реализует всю серверную логику для взаимодействия с базой данных PostgreSQL. Он содержит функции для создания и проверки групп, управления пользователями, заметками, задачами, общими и личными сообщениями. Все операции обернуты в контексты with, что гарантирует корректное закрытие соединений и безопасность транзакций.

*2.4 Тестирование*

Тестирование приложения проводилось на этапе разработки каждого функционального блока и заключалось в проверке корректности работы пользовательского интерфейса, базы данных и логики взаимодействия между ними.

1. Проверка подключения к базе данных

На начальном этапе было протестировано подключение к базе данных PostgreSQL. Проверка осуществлялась вызовом функции «get\_connection()»

Также проверялась корректность выполнения SQL-запросов и автоматическое закрытие соединений при помощи конструкции with.

2. Тестирование регистрации и входа

Функции были проверены на:

* Создание новой группы;
* Повторный вход под существующим пользователем;
* Обработку некорректных паролей и несуществующих кодов группы.

3. Проверка логики добавления заметок

Проводилось ручное тестирование:

* Создания заметки;
* Отображения нескольких заметок;
* Корректности автора;
* Удаления.

Все изменения немедленно отражались в интерфейсе и таблице notes.

4. Тестирование задач

Проверялась корректность ввода дедлайна, отображение задач и цветовая индикация по сроку.

5. Чаты: общий и личный

Были протестированы:

* Отправка и удаление сообщений;
* Автоматическое обновление чата;
* Обработка сообщений от нескольких пользователей в группе.

По итогам тестирования ошибок обнаружено не было. Все функции работают, интерфейс остаётся стабильным при работе с базой данных, а обновление данных в реальном времени подтверждено.

*2.5 Файловая структура*

В процессе разработки приложения была сформирована логичная и модульная структура проекта (смотреть Рисунок 49).

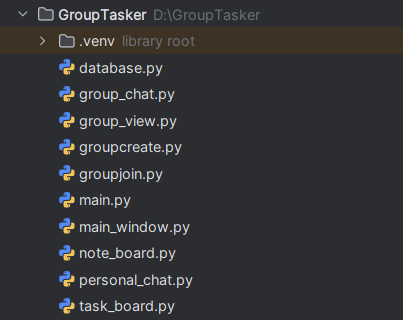


Рисунок 49 − Файловая структура проекта

Описание ключевых файлов:

* main.py — основной вход в приложение. Содержит интерфейс главного меню с выбором: создание новой группы или подключение к существующей.
* database.py — централизованный модуль работы с СУБД PostgreSQL. Содержит более 30 функций для создания, чтения, обновления и удаления данных (CRUD-операции) по всем сущностям проекта.
* groupcreate.py / groupjoin.py — модули регистрации новых групп и подключения пользователей к существующим. Обеспечивают простую и понятную навигацию.
* main\_window.py — основной интерфейс взаимодействия внутри группы. Содержит навигационное меню и загрузку всех вкладок: чатов, задач, доски и информации о группе.
* note\_board.py — доска с коллективными заметками. Реализована прокрутка, поддержка авторства и возможность удаления.
* task\_board.py — раздел индивидуальных задач пользователя. Реализованы дедлайны, статус задач, а также визуальные элементы приоритетов.
* group\_chat.py — общий чат группы. Поддерживает удаление сообщений, автоматическую синхронизацию и отображение времени.
* personal\_chat.py — реализация интерфейса личных сообщений между пользователями. Содержит выбор собеседника и отдельное отображение переписки.

**Заключение**

Подводя итоги, можно сказать, что разработка программного обеспечения для организации работы в групповых проектах является актуальной и весьма интересной задачей.

Целью данной курсовой работы являлась разработка десктопного приложения, позволяющего участникам команды эффективно взаимодействовать между собой, организовывать задачи, делиться заметками и вести внутреннюю коммуникацию. В процессе работы было создано полнофункциональное приложение с удобным пользовательским интерфейсом и стабильной работой с базой данных PostgreSQL.

В ходе реализации проекта был использован современный стек технологий: язык программирования Python, библиотека PyQt6 для графического интерфейса и система управления базами данных PostgreSQL. Такой выбор обеспечил высокую скорость разработки, хорошую масштабируемость проекта и удобство в сопровождении кода. Логика приложения построена по модульному принципу, каждая функция вынесена в отдельный файл, что позволило добиться структурированности и гибкости архитектуры.

Одним из главных достоинств является простота взаимодействия пользователя с системой. Интерфейс приложения интуитивно понятен, а такие функции, как общий чат, личные переписки, доска задач и заметок, позволяют команде вести совместную деятельность централизованно и с минимальными затратами времени на организационные процессы.

В ходе выполнения курсовой работы были реализованы все поставленные задачи:

* Составлено техническое задание с описанием функционала и требований к системе.
* Произведён выбор подходящего инструментария, включая язык программирования, графическую библиотеку и СУБД.
* Проведён анализ существующих решений и аналогов в области систем для командной работы.
* Разработано настольное приложение с полным графическим интерфейсом, модулями общения и отслеживания задач.
* Проведено тестирование основных функций, взаимодействия с базой данных и интерфейса.
* Составлено краткое руководство пользователя для описания установки и эксплуатации приложения.

Приложение А

Ссылка для скачивания программного обеспечения «GroupTasker»:

<https://github.com/mironchick/GroupTasker>

Приложение Б

Руководство программиста

1. Введение

Данное руководство предназначено для разработчиков, которые будут работать с исходным кодом приложения "GroupTasker". Оно содержит описание архитектуры, используемых технологий и структуры проекта.

2. Технологический стек

* Язык программирования: Python 3.11
* Графический интерфейс: PyQt6
* База данных: PostgreSQL

Дополнительные библиотеки:

* + psycopg2 — для работы с PostgreSQL.
  + random — генерация уникальных кодов групп.
  + pyperclip — работа с буфером обмена.
  + sys — системные функции.

3. Структура проекта

Проект имеет модульную структуру:

main.py.

Точка входа в приложение. Содержит главное меню и логику переключения между окнами.

database.py.

Центральный модуль для работы с базой данных. Включает функции для CRUD-операций (создание, чтение, обновление, удаление данных).

groupcreate.py / groupjoin.py.

Окна для создания новой группы и входа в существующую.

main\_window.py.

Основное окно приложения с навигационным меню и загрузкой вкладок:

* + Доска заметок (note\_board.py).
  + Личные задачи (task\_board.py).
  + Общий чат (group\_chat.py).
  + Личные сообщения (personal\_chat.py).
  + Управление группой (group\_view.py).
* note\_board.py

Реализация доски с коллективными заметками.

task\_board.py.

Управление личными задачами пользователя, включая дедлайны и приоритеты.

group\_chat.py / personal\_chat.py.

Общий и приватный чаты с поддержкой отправки, удаления и автоматического обновления сообщений.

4. Настройка окружения

* Установите Python 3.11.
* Установите PostgreSQL и создайте базу данных для приложения.
* Установите библиотеки.
* Настройте подключение к БД в файле database.py (функция get\_connection()).

5. Работа с базой данных

Таблицы:

* + groups — информация о группах.
  + users — данные пользователей.
  + notes — заметки на доске.
  + tasks — личные задачи.
  + messages — сообщения общего чата.
  + private\_messages — личные переписки.

Транзакции: все операции выполняются в контексте with для безопасного управления соединениями.

Приложение В

Руководство пользователя

1. Начало работы.

2. Меню входа.

После запуска откроется главное окно с двумя кнопками:

* "Создать группу" – для создания новой группы.
* "Войти в группу" – для подключения к существующей.

3. Создание группы.

3.1 Нажмите "Создать группу".

3.2. Заполните поля:

* Название группы – произвольное имя (например, "Учебный проект").
* Ваше имя – ваше имя или никнейм в системе.
* Пароль – пароль для входа в группу.

3.3. Нажмите "Создать группу".

3.4. После успешного создания появится уникальный код группы (он автоматически скопируется в буфер обмена). Сохраните этот код – он нужен для приглашения участников.

4. Вход в существующую группу.

4.1. Нажмите "Войти в группу".

4.2. Введите:

* Код группы (полученный от создателя).
* Ваше имя (может отличаться от имени при регистрации).
* Пароль (если вы не первый участник, введите пароль, который использовал создатель).

4.3. Нажмите "Присоединиться к группе".

5. Основной интерфейс

После входа откроется главное окно с доской заметок и боковым меню:

* Группа – управление участниками.
* Мои задачи – личный список задач.
* Общий чат – групповое общение.
* Личные сообщения – приватные переписки.

5.1. Доска заметок.

Добавить заметку:

* + Нажмите кнопку "Добавить заметку".
  + Введите текст и нажмите "Сохранить".

Просмотр заметок:

* + Все заметки отображаются в хронологическом порядке.
  + Под каждой заметкой указан автор.

Удаление заметок:

* + Нажмите "Удалить" на своей заметке (удалять чужие нельзя).

5.2. Раздел "Мои задачи".

Добавление задачи:

1. Нажмите "Добавить задачу".

2. Заполните:

* + - Название (например, "Подготовить отчёт").
    - Описание (дополнительные детали).
    - Дедлайн (укажите дату выполнения).

3. Нажмите "Сохранить".

Просмотр задач:

* + Активные задачи отображаются списком.
  + Просроченные задачи выделяются красным.

Удаление задач:

* + Нажмите "Удалить" на выполненной задаче.

5.3. Общий чат.

Отправка сообщения:

* + Введите текст в нижнее поле.
  + Нажмите "Отправить" (или Enter).

Удаление сообщений:

* + Нажмите "Удалить" на своём сообщении.

Автоматическое обновление:

* + Чат обновляется каждые несколько секунд.

5.4. Личные сообщения.

* В левой части экрана выберите участника из списка.
* В правой части откроется история переписки.
* Введите сообщение и нажмите "Отправить".
* Удалять можно только свои сообщения.

5.5. Раздел группа.

Для администратора группы:

Исключение участника:

* + - Перейдите в раздел "Группа".
    - Выберите участника и нажмите "Исключить участника".

Удаление группы:

* + - Нажмите "Удалить группу" (все данные будут удалены безвозвратно).

Для участника:

Выйти из группы:

* Нажмите "Покинуть группу"