**Определение целей и объёма проекта**

***Тип ПО для разработки:***веб приложение.

***Основная цель проекта:***создание корпоративного мессенджера, обеспечивающего быструю, надежную и безопасную коммуникацию между сотрудниками предприятия, способствуя повышению эффективности работы и улучшению внутренних бизнес-процессов.

***Задачи проекта***

1. Ознакомление со структурой и функциональностью существующих средств коммуникации: изучение текущих инструментов, используемых для коммуникации между сотрудниками.

2. Разработка архитектуры приложения: клиент-серверная.

3. Выбор технологий:

Разработка серверной части (BackEnd) – Go, Стрекалов Р.П.

База данных – PostgreSQL

Контейнеризация – Docker

Хостинг для репозиторий - GitHub

Разработка клиентской части (FrontEnd) – Typescript, Angular, Ionic, Бондарев Е.С.

Дизайн – Figma, Ковригин Д.В.

Тестирование – Якушенков А.А.

Документирование – Кургузов А.И.

4. Обеспечение функциональности реального времени:

* Реализация поддержки WebSocket для мгновенной передачи сообщений.
* Обеспечение надёжной работы при высоких нагрузках.

5. Тестирование и отладка: проведение модульного и интеграционного тестирования.

Процесс создания корпоративного мессенджера включает:

1. Планирование и проектирование:

* Определение ключевых функций, включая регистрацию пользователей, авторизацию с использованием JWT, отправку и хранение текстовых сообщений, управление чатами, редактирование профиля и обеспечение безопасности данных.

2. Разработка серверной части:

* Сервер на Go с использованием фреймворка Gin обеспечивал аутентификацию, маршрутизацию запросов и взаимодействие с базой данных.
* Внедрение JWT для аутентификации и bcrypt для хеширования паролей.
* Использование Gorilla Mux для WebSocket и сложных маршрутов.
* Подключение к базе данных PostgreSQL с использованием "database/sql" и настройка пула соединений для повышения производительности.
* Реализация мониторинга производительности и логирования для обнаружения проблем.

Разработка клиентской части:

* Интерфейс пользователя (UI): создан удобный интерфейс на Typescript с использованием Angular, ionic, включающий стандартные элементы управления для отправки сообщений, управления контактами и отображения истории чатов.
* Функции отправки сообщений: Реализована мгновенная передача сообщений с использованием WebSocket для обновления чатов в реальном времени.
* Интеграция с серверной частью: организована передача данных между клиентом и сервером через API.
* Обработка и отображение сообщений: сообщения принимаются, обрабатываются и отображаются в реальном времени, обеспечивая непрерывную коммуникацию.
* Отправка файлов и фото

Сроки – система должна быть разработана и внедрена в течении 3 месяцев.

Объём проекта: разработать корпоративный мессенджер, который представляет собой многопользовательскую систему обмена сообщениями, предоставляющая разнообразные функции для обеспечения эффективного взаимодействия между сотрудниками.

Функциональные требования: чат, файловый обмен, уведомления, организация пользователей, история сообщений, интеграция с другими системами.

Нефункциональные требования: высокая производительность системы, безопасность данных.

**Сбор и анализ требований**

**Сбор информации:**

*Интервью с пользователем:* обнаружили, что пользователи хотят улучшить внутреннюю коммуникацию, повысить оперативность принятия решений для сплочённости и информированности команды.

*Опрос:* выявили, что 50% респондентов хотят возможность интеграции с другими инструментами системами.

**Анализ:**

*Проблемы:* необходимо разработать корпоративный мессенджер, который представляет собой многопользовательскую систему обмена сообщениями, предоставляющая разнообразные функции для обеспечения эффективного взаимодействия между сотрудниками.

*Потребности:* пользователи хотят эффективную и организационный обмен сообщениями с возможностью отправки файлов, уведомлений в мессенджере, а также интеграция с другими системами.

**Формулирование:**

*Функциональные потребности:*

* Файловый обмен.
* Организация пользователей.
* История сообщений
* Интеграция с другими системами

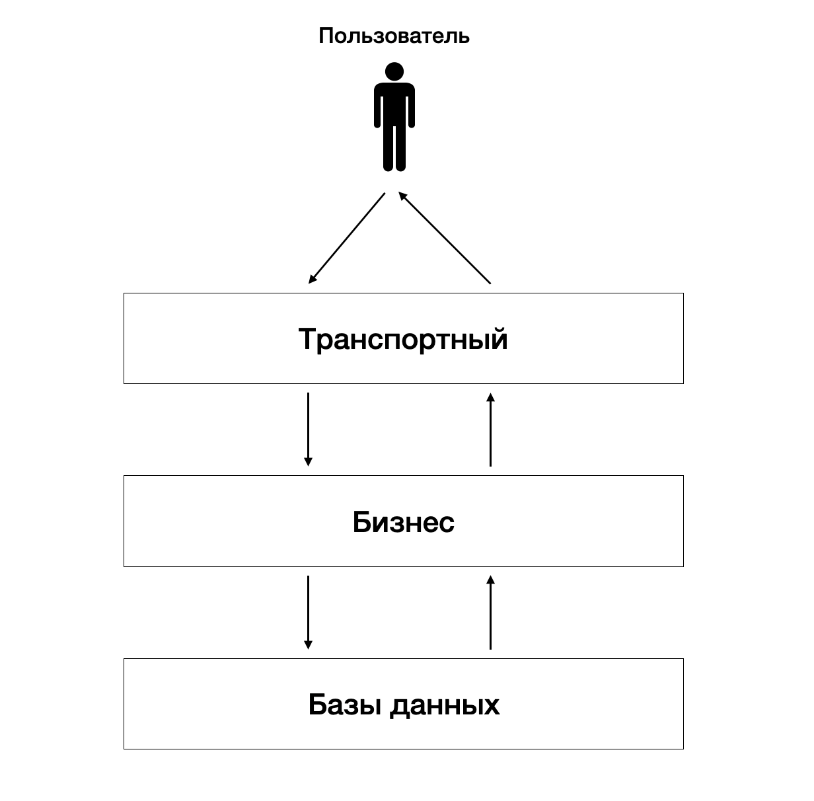
*Нефункциональные потребности:*

* Интерфейс должен быть интуитивно понятным.
* Обеспечение надёжной работы при высоких нагрузках.
* Для предоставления доступа к приложению и api используется технология JWT-token, где токены должны передаватся через заголовок запроса.
* Система должна обрабатывать запросы и запускаться не больше 3 с.

**Проектирование архитектуры и интерфейса**

Состав архитектура корпоративного мессенджера:

Клиентское приложение является основным интерфейсом, через который пользователи общаются в мессенджере, которое реализовано как десктопное приложение. Клиентское приложение отвечает за прием и отправку сообщений, а также за отображение информации о контактах, группах и других аспектах мессенджера.

  
Рисунок 1 – модель трёхуровневой архитектуры

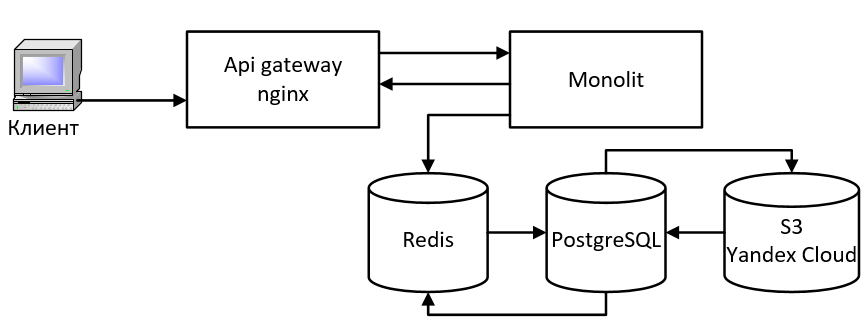


Рисунок 2 – диаграмма архитектуры приложения

1. Сервер приложения – является основным компонентом, который обрабатывает все запросы и отвечает за управление данными мессенджера. Он принимает сообщения от клиентских приложений, обрабатывает их и передает получателям. Сервер приложения также отвечает за хранение данных пользователей, таких как контакты, группы и настройки профиля.
2. База данных - является хранилищем данных мессенджера. Она содержит информацию о пользователях, контактах, группах, сообщениях и других данных, необходимых для работы мессенджера. База данных реализована с использованием PostgreSQL-базы данных.
3. Система аутентификации и авторизации - отвечает за обеспечение безопасности доступа к мессенджеру. Она проверяет подлинность пользователей при входе в систему и управляет правами доступа к различным функциям и данным мессенджера.
4. Система шифрования - отвечает за обеспечение безопасности передачи данных в мессенджере. Она шифрует сообщения перед их отправкой и расшифровывает их после получения. Это гарантирует, что сообщения останутся конфиденциальными и защищенными от несанкционированного доступа.
5. Система уведомлений - отвечает за отправку уведомлений пользователям о новых сообщениях, входящих вызовах и других событиях в мессенджере. Она может отправлять уведомления в виде уведомлений на устройстве, электронных писем или SMS-сообщений.
6. Система интеграции - позволяет мессенджеру интегрироваться с другими приложениями и сервисами, такими как календарь, почта или сервисы управления проектами. Это позволяет пользователям получать доступ к функциям мессенджера из других приложений и использовать его в рамках своих рабочих процессов.

Взаимодействие этих компонентов может быть реализовано с использованием различных протоколов и технологий, таких как HTTP, WebSocket вместе с Gorilla Mux. Клиентское приложение отправляет запросы на сервер приложения, который обрабатывает их и взаимодействует с базой данных, системой аутентификации и авторизации, системой шифрования, системой уведомлений и системой интеграции для обеспечения полной функциональности мессенджера.

**Создание дорожной карты (Roadmap)**

**Фаза 1: Планирование**

* *Задача 1:* сбор и анализ требований (1-13 сентября 2024)
* *Задача 2:* разработка проектного плана и дорожной карты (14-25 сентября 2024)

**Фаза 2: Проектирование**

* *Задача 1:* Создание прототипов и макетов (13-25 сентября 2024)
* *Задача 2:* Утверждение дизайна (26 сентября 2024)

**Фаза 3: Разработка**

* *Задача 1:* Разработка клиентской части, фронтенда (27 сентября – 11 ноября 2024)
* *Задача 2:* Разработка серверной части, базы данных, бэкенда (11 октября – 11 ноября 2024)

**Фаза 4: Тестирование**

* *Задача 1:* Проведение функционального тестирования (11 ноября – 1 декабря 2024)
* *Задача 2:* Проведение нагрузочного тестирования (11 ноября – 1 декабря 2024)

**Фаза 5: Внедрение**

* *Задача 1:* Подготовка к запуску (1 декабря 2024)
* *Задача 2:* Запуск приложения (2 декабря 2024)

**Фаза 6: Поддержка**

* *Задача 1:* Техническая поддержка и исправление ошибок (3 декабря – 15 декабря 2024)

**Оценка рисков и план управления рисками**

*Оценка рисков:*

* Риски: задержка в разработке, технические проблемы, недостаточная пользовательская поддержка.
* План управления рисками: создание резервного плана, привлечение дополнительных ресурсов.

*Управление рисками:*

* Риски: задержка в разработке, проблема с интеграцией.
* План управления: регулярные проверки статуса, резервация времени для решения проблем.

**Оценка бюджета и ресурсов**

**Бюджет**

*Оценка затрат:* зарплата разработчиков, лицензии, серверы, среди них – 150 – 200 р. На хостинг.

*Управление бюджетом:* мониторинг затрат еженедельно, отчётность.

**Ресурсы**

*Основные этапы:* анализ и планирование, разработка, тестирование, внедрение, обучение.

*Ресурсы:* команда из 5 разработчиков, сервер