Министерство образования Московской области

ГБПОУ МО «Колледж «Коломна»

09.02.07

К защите допускается

зам. директора по УР

\_\_\_\_\_\_\_Ромашкина Э.Б.

«\_\_\_\_»\_\_\_\_\_\_\_ 20\_\_\_г.

ДИПЛОМНЫЙ ПРОЕКТ

Тема: Разработка корпоративного мессенджера

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ККОО.ДП2178.000

|  |  |  |
| --- | --- | --- |
| Студент | «\_\_\_\_»\_\_\_\_\_\_\_\_\_2025 г. | Городничев Д.А. |
| Руководитель проекта | «\_\_\_\_»\_\_\_\_\_\_\_\_\_2025 г. | Савина Е.Ю. |
| Консультант по экономическому разделу | «\_\_\_\_»\_\_\_\_\_\_\_\_\_2025 г. | Караваев А.В. |
| Нормоконтролер | «\_\_\_\_»\_\_\_\_\_\_\_\_\_2025 г. | Грушникова Т.Н. |
| Рецензент | «\_\_\_\_»\_\_\_\_\_\_\_\_\_2025 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

2025

Министерство образования Московской области

Государственное бюджетное профессиональное образовательное учреждение Московской области «Колледж «Коломна»

|  |  |
| --- | --- |
|  | «Утверждаю»  Зам. директора по УР  \_\_\_\_\_\_\_\_\_\_ Ромашкина Э.Б.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

ЗАДАНИЕ

на дипломный проект по специальности 09.02.07 «Информационные системы и программирование» студенту очного отделения, группы 42112

Городничеву Даниле Алексеевичу

Тема проекта: Разработка корпоративного мессенджера

1. Постановка задачи:

Разработать клиентское приложение и веб-API (далее Web API), реализующие концепцию корпоративного мессенджера для обмена сообщениями внутри организации.

Web API сервис должен решать следующие задачи:

* Управление пользователями: регистрация, аутентификация по логину и паролю;
* Организация личных и групповых чатов: создание и удаление личных и групповых чатов, добавление и исключение участников в групповых чатах;
* Обработка отправки и получения сообщений и файлов;
* Сохранение истории сообщений и файлов;
* Обеспечение безопасности данных, хеширование паролей;

Клиентское приложение должно решать следующие задачи:

* Авторизация пользователей в системе;
* Просмотр списка групповых и личных;
* Отправка и получение сообщений и файлов;
* Настройка персональных данных;

Интерфейс для клиентского приложения должен быть интуитивно понятным и учитывать уровень развития сервисов конкурентов.

В реализованном проекте должны присутствовать следующие формы для просмотра необходимой информации пользователем:

В клиентском приложении:

* Окно авторизации пользователя;
* Окно списка чатов;
* Окно просмотра сообщений чата;
* Окно изменения данных пользователя;
* Окно создания и редактирования групповых чатов;

Для организации данных необходимо хранить и обрабатывать следующую информацию:

О пользователях: логин, пароль, ФИО, аватар, текущая должность, дата регистрации.

О чатах: название, флаг группового чата, аватар, дата создания.

О сообщениях: текст, отправитель, чат, дата и время отправки, прикрепленные файлы.

1. Состав и объем дипломного проекта
   1. Пояснительная записка, содержащая следующие обязательные разделы:

Введение

1. Основная часть
   1. Разработка системного проекта
      1. Описание предметной области
      2. Анализ предметной области
      3. Техническое задание
      4. Описание используемых технологий
   2. Проектирование информационной системы
      1. Функциональное моделирование
      2. Проектирование интерфейсов приложения
   3. Разработка информационной системы
      1. Модель данных
      2. Структура проекта
   4. Оценка качества программного обеспечения
2. Экономический раздел
3. Охрана труда при работе с вычислительной техникой

Выводы и заключение

Список литературы

Приложения

Презентация к докладу.

1. Требования к работе:

Пояснительная записка должна содержать не менее 50 листов формата A4. Шрифт оформления пояснительной записки Times New Roman, 14 пт, 1,5 междустрочный интервал.

Презентация должна содержать не более 15 слайдов. На первом слайде указывается название учебного заведения, тема дипломного проекта, фамилия, имя и отчество студента и руководителя, название специальности, город и год. Фон презентации должен оформляться в светлых, нейтральных тонах с контрастным черным текстом. Размер шрифта – не менее 28. Заголовки нужно выделить, шрифт заголовков – не менее 36.

1. Содержание экономического раздела:

Экономический раздел включает следующие пункты:

* Понятие себестоимости;
* Затраты в составе себестоимости;
* Расчет себестоимости информационной системы.

1. Содержание раздела по технике безопасности.

* Охрана труда при работе с вычислительной техникой

Рекомендованная литература:

1. Беляков Г.И. Охрана труда и техника безопасности: учебник для среднего профессионального образования / Беляков Г.И. – 3-е изд., перераб. и доп. – М.: Издательство Юрайт, 2023. – 404 с.
2. Виноградова Н. А. Научно-исследовательская работа студента: Технология написания и оформления доклада, реферата, курсовой и выпускной квалификационной работы: учеб. пособие для студ. Учреждений сред. Проф. Образования / Н.А. Виноградова, Н.В. Микляева. – 13-е изд., стер. – М.: Издательский центр «Академия», 2023;
3. Кудрина Е. В. Основы алгоритмизации и программирования на языке С#: учеб. пособие для среднего профессионального образования / Е.В. Кудрина, М.В. Огнева. – М.: Издательство Юрайт, 2023;
4. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем: учебник для студ. Учреждений сред. Проф. Образования / Г.Н. Федорова. – 3-е изд., исправленное. – М.: Издательский центр «Академия», 2023;
5. Федорова Г. Н. Осуществление интеграции программных модулей: учебник для студ. учреждений сред. Проф. Образования / Г.Н. Федорова. – 3-е изд., исправленное. – М.: Издательский центр «Академия», 2023;
6. Подбельский В. В. Базовый курс C#: учебник для среднего профессионального образования/ В.В. Подбельский.-М.: Издательство Юрайт, 2023;
7. Рыжко А.Л. Экономика отрасли информационных систем: учебное пособие для среднего профессионального образования / Рыжко А.Л., Рыжко Н.А., Лобанова Н.М., Кучинская Е.О. – 2-е изд., испр. и доп. –М.: Издательство Юрайт, 2024. – 176 с.

Дата выдачи задания «7» апреля 2025 г.

Срок окончания дипломного проекта «14» июня 2025 г.

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель структурного подразделения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Консультант по экономическому разделу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание утверждено на заседании цикловой комиссии

Протокол № 7 от «12» марта 2025г.

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 года.

СОдержание

[1 Основная часть 10](#_Toc200057754)0

[1.1 Разработка системного проекта](#_Toc200057755)

[1.1.1 Описание предметной области](#_Toc200057756)

[1.1.2 Анализ предметной области 11](#_Toc200057757)

[1.1.3 Техническое задание 12](#_Toc200057758)

[1.1.4 Описание используемых технологий 13](#_Toc200057759)

[1.2 Проектирование информационной системы 14](#_Toc200057760)

[1.2.1 Функциональное моделирование](#_Toc200057761)

[1.2.2 Функциональное моделирование 17](#_Toc200057762)

[1.3 Разработка информационной системы 22](#_Toc200057763)

[1.3.1 Модель данных](#_Toc200057764)

[1.3.2 Структура проекта 27](#_Toc200057765)

[1.4 Разработка информационной системы 32](#_Toc200057763)

[2 Экономический раздел 35](#_Toc200057766)

[2.1 Понятие себестоимости](#_Toc200057767)

[2.2 Затраты в составе себестоимости 36](#_Toc200057768)

[2.3 Расчет себестоимости информационной системы 39](#_Toc200057769)

[3 Охрана труда при работе с вычислительной техникой 47](#_Toc200057770)

[Выводы и заключение 51](#_Toc200057771)

[Список литературы 52](#_Toc200057772)

[Приложение А – Код программы 53](#_Toc200057773)

[Приложение Б – Флэш-накопитель с программой 63](#_Toc200057774)

Целью проекта является разработка корпоративного мессенджера для эффективного обмена сообщениями внутри организации. Система призвана решить ключевые проблемы традиционных корпоративных коммуникаций, такие как сложность организации группового взаимодействия и неудобство обмена файлами.

Система предоставляет сотрудникам удобную платформу для обмена сообщениями, организации чатов. Основные функции включают отправку и получение сообщений, поиск чатов, обмен файлами и удобную навигацию между чатами через интуитивно понятный интерфейс.

Для администраторов система предоставляет централизованное управление пользователями, надежное хранение истории сообщений.

Разработанный корпоративный мессенджер представляет собой современное программное решение, направленное на трансформацию внутренних коммуникаций организации. Система создает защищенную цифровую среду для эффективного обмена информацией между сотрудниками, обеспечивая новый уровень взаимодействия между подразделениями компании.

Основу решения составляет технологически совершенная платформа, объединяющая производительный WPF-клиент с мощным серверным компонентом на ASP.NET Core. Использование PostgreSQL в качестве системы хранения данных гарантирует надежность работы и целостность корпоративной информации. Архитектура решения построена с учетом современных требований к масштабируемости и безопасности бизнес-приложений.

Ключевым преимуществом внедрения становится формирование цифровой экосистемы для делового взаимодействия, где технологическая составляющая органично дополняет и усиливает традиционные бизнес-процессы.

Система разработана с минимальным порогом входа для пользователей – интуитивно понятный интерфейс сокращает время адаптации сотрудников.

Проект демонстрирует, как современные технологии могут трансформировать внутренние коммуникации организации, делая их более эффективными и управляемыми. Использование проверенных технологий Microsoft .NET обеспечивает стабильность работы и простоту интеграции с существующей корпоративной инфраструктурой, а продуманный пользовательский интерфейс гарантирует быстрое внедрение и высокий уровень принятия системы сотрудниками.

1. ОСНОВНАЯ ЧАСТЬ
2. Разработка системного проекта
   * 1. Описание предметной области

Разрабатываемая система представляет собой комплексное решение для организации внутреннего общения сотрудников в корпоративной среде. Проект направлен на создание удобного и функционального мессенджера, обеспечивающего передачу текстовых сообщений и файлов, а также управление групповыми чатами. Использование локальной архитектуры позволяет развёртывать систему в рамках конкретной организации, не завися от внешних сервисов и облачных решений. Это особенно актуально для компаний, предпочитающих хранить все данные на своих ресурсах и контролировать каналы взаимодействия между сотрудниками.

Основные компоненты системы включают: централизованную платформу для обмена текстовыми сообщениями, инструменты для создания групповых чатов, возможность обмена файлами различных форматов. Пользователи системы имеют персональные профили с настраиваемыми параметрами, доступ к истории переписки и возможность управления групповыми беседами.

Ключевые функциональные возможности: система обеспечивает удобный поиск по чатам, простой механизм создания групповых чатов и базовые возможности работы с файлами. Интерфейс приложения разработан с учетом потребностей корпоративных пользователей, что обеспечивает быстрое освоение системы сотрудниками.

Преимущества внедрения данной системы заключаются в создании единого пространства для рабочих коммуникаций, сокращении времени на поиск информации и упрощении процессов согласования. Для администраторов системы предусмотрены инструменты управления пользователями, а для сотрудников - интуитивно понятный интерфейс для повседневного общения.

Перспективы развития платформы включают расширение возможностей работы с файлами, улучшение механизмов поиска по истории сообщений и развитие функций администрирования. Это позволит создать удобное решение для внутренних коммуникаций, соответствующее потребностям современного предприятия.

* + 1. Анализ предметной области

В разрабатываемой системе корпоративного мессенджера основными участниками являются пользователи и администраторы. Пользователи обмениваются сообщениями в личных и групповых чатах, отправляют файлы, просматривают переписку, редактируют свои личные данные и создают групповые чаты. Администраторы имеют возможность создавать учетные записи новых пользователей и обладают доступом к базе данных для контроля и обслуживания системы. Взаимодействие между участниками происходит через централизованный сервер, обеспечивающий сохранение данных и корректную маршрутизацию сообщений.

Система решает следующие функциональные задачи: хранение информации о пользователях (логин, ФИО, пароль, аватар, должность), организация и ведение чатов (личных и групповых), передача текстовых сообщений и файлов (документов, фото, видео), отображение истории переписки, а также редактирование данных пользователей.

Администраторы выполняют ограниченный круг задач – создание новых учетных записей и техническое сопровождение системы через доступ к базе данных. Это позволяет централизованно контролировать состав пользователей и обеспечивать стабильную работу приложения.

* + 1. Техническое задание

Требуется разработать программный комплекс для организации внутренней корпоративной коммуникации. Система должна включать клиентское приложение (WPF) и Web API, связанные между собой с использованием REST API. Решение реализуется как локальная корпоративная платформа, ориентированная на обмен сообщениями и файлами между сотрудниками одной организации.

Клиентское приложение должно предусматривать авторизацию пользователей по логину и паролю. Создание учетных записей доступно только администраторам. Каждый пользователь после входа в систему получает доступ к списку личных и групповых чатов, может отправлять текстовые сообщения, а также прикреплять к ним документы, изображения и видеофайлы. В системе предусмотрена возможность редактирования персональных данных, включая имя, должность и аватар.

Каждый чат отображает список сообщений с указанием автора, времени отправки и прикрепленных файлов. Групповые чаты поддерживают добавление и удаление участников. Пользователи могут самостоятельно создавать групповые чаты и управлять составом участников.

Серверная часть должна быть реализована как RESTful Web API, обеспечивающее взаимодействие между клиентским приложением и административной частью системы. Архитектура строится на принципах REST, что позволяет обеспечить простоту масштабирования и расширения функциональности.

* + 1. Описание используемых технологий

Использование трёхслойной архитектуры в проекте позволяет упростить разработку, сопровождение и масштабирование системы за счёт модульности и чёткого разделения ответственности. Это положительно влияет на тестируемость, безопасность и производительность приложения, а также облегчает внедрение новых функций и оптимизацию существующих.

В проекте применяется классическая трёхслойная архитектура, включающая следующие уровни:

* Слой представления – реализуется в клиентском приложении на WPF. Пользователь взаимодействует с графическим интерфейсом, из которого отправляются запросы к серверу через RESTful API;
* Слой логики – реализован в виде сервисов на серверной стороне ASP.NET Core. Здесь сосредоточена бизнес-логика: обработка запросов, управление пользователями, чатами, сообщениями, валидация данных;
* Слой данных – реализуется через репозитории, обеспечивая доступ к базе данных PostgreSQL. Здесь происходит чтение, запись и преобразование данных в доменные модели.

PostgreSQL выбрана как основная система управления базами данных благодаря своей надёжности, безопасности и высокой производительности. СУБД активно используется в корпоративной разработке, обеспечивая поддержку транзакций, индексов, внешних ключей и масштабируемость при работе с большими объёмами информации. В данном проекте PostgreSQL используется для хранения информации о пользователях, чатах, сообщениях, вложениях и другой внутренней информации.

Для работы с базой данных применялось приложение DBeaver — современный инструмент для администрирования баз данных. Приложение позволяет просматривать структуру таблиц, редактировать данные, выполнять SQL-запросы, создавать схемы и визуализировать связи между таблицами. Использование DBeaver значительно упростило настройку, отладку и проверку работы базы данных на всех этапах проекта.

Клиентское приложение реализовано на языке программирования C# с использованием WPF в среде разработки Microsoft Visual Studio 2022. Данный стек технологий выбран на основе уже имеющегося опыта и полной совместимости с остальной частью проекта. Visual Studio 2022 обеспечила мощный инструментарий для разработки, отладки и тестирования, позволив повысить производительность работы и качество создаваемого программного обеспечения.

Серверная часть разработана с использованием ASP.NET Core Web API, реализующего архитектурный стиль REST. Это позволило создать унифицированный и легко расширяемый интерфейс взаимодействия между клиентом и сервером. Все основные действия, включая регистрацию, авторизацию, создание и удаление чатов, отправку сообщений и файлов, реализованы в виде REST-эндпоинтов, что обеспечивает удобство интеграции и возможность дальнейшего расширения .

1. Проектирование информационной системы
   * 1. Функциональное моделирование

Перед разработкой проекта были спроектированы следующие диаграммы, каждая из которых играет важную роль в описании функциональности и структуры системы.

Диаграмма вариантов использования (прецедентов) помогла определить основные действия, которые пользователи могут выполнять в системе. Это включает в себя описание функциональности системы с точки зрения ее пользователей, а также админов и предоставляет общее представление о взаимодействии между системой и ее окружением. Спроектированная диаграмма представлена на рисунке 1.

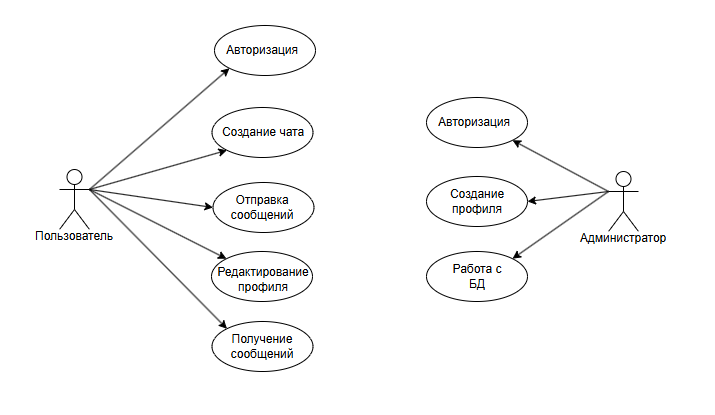


Рисунок 1 – Диаграмма вариантов использования

Диаграмма последовательности предназначена для иллюстрации порядка взаимодействия между объектами системы в рамках конкретного сценария использования. Эта диаграмма помогает подробно описать, как объекты обмениваются сообщениями и вызывают операции друг у друга во времени в пределах выбранного сценария. Разработанная диаграмма представлена на рисунке 2.

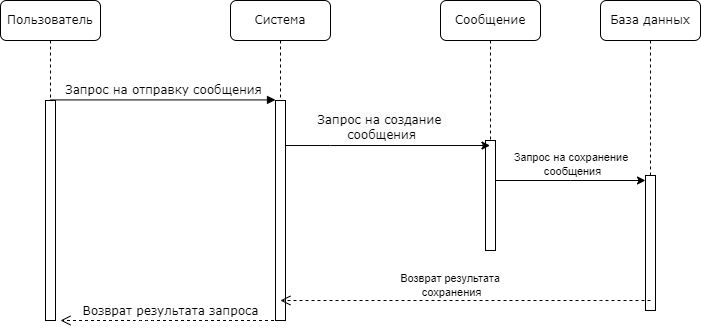
Рисунок 2 – Диаграмма последовательности

Диаграмма деятельности была использована для визуализации последовательности действий или процессов в системе. Это позволило описать порядок выполнения операций или задач в системе и выделить основные этапы взаимодействия пользователей с системой. Разработанная диаграмма представлена на рисунке 3.

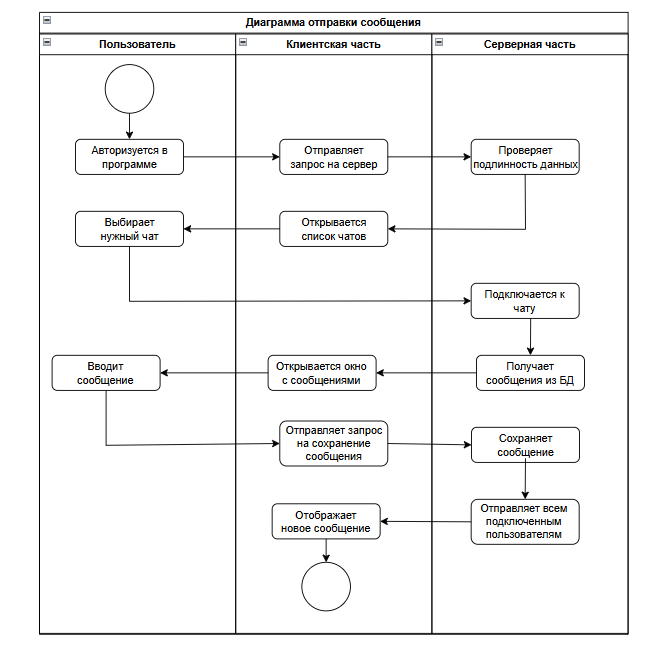


Рисунок 3 – Диаграмма деятельности

* + 1. Функциональное моделирование

Разработка интерфейсов корпоративного мессенджера направлена на создание удобного и продуктивного взаимодействия сотрудников с системой. Интерфейс клиентского WPF-приложения выполнен с учётом корпоративного стиля и требований к эргономике: он позволяет пользователям быстро находить нужные чаты, отправлять сообщения и файлы, а также управлять групповыми чатами. Для администратора в том же приложении предусмотрена отдельная форма для создания новых пользователей. Все интерфейсы спроектированы так, чтобы минимизировать количество действий для выполнения задач и обеспечить максимальное удобство использования.

Клиентское приложение выполнено в цветовой гамме с акцентом на черный и фоновым темным серо-синем оттенком. Интерфейс выдержан в минималистичном стиле — основной акцент сделан на скорости и удобстве работы с чатами и сообщениями.

При проектировании пользовательского интерфейса корпоративного мессенджера был применен комплексный подход, основанный на современных принципах UX/UI-дизайна. Интерфейс разработан с акцентом на функциональную простоту и интуитивную понятность, что позволяет пользователям быстро освоить основные возможности системы без дополнительного обучения. Визуальная согласованность всех элементов обеспечивает целостное восприятие интерфейса, а продуманная иерархия компонентов направляет внимание пользователя на ключевые функции.

Особое внимание уделено оперативной обратной связи - система мгновенно реагирует на действия пользователя визуальными и функциональными изменениями, создавая ощущение непосредственного взаимодействия. Адаптивная верстка гарантирует корректное отображение интерфейса на устройствах с различными характеристиками экранов, сохраняя при этом удобство управления. Все интерактивные элементы имеют оптимальные размеры и четкие зоны клика, что значительно повышает комфорт работы с приложением.

Для начала работы с моим клиентским приложением необходимо открыть стартовое окно авторизации, изображенное на рисунке 4.

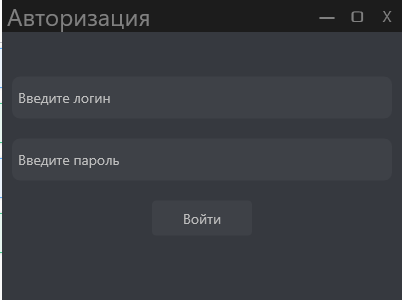


Рисунок 4 – Окно авторизации

После прохождение авторизации пользователь будет перенаправлен на главную страницу. (Рисунок 5)

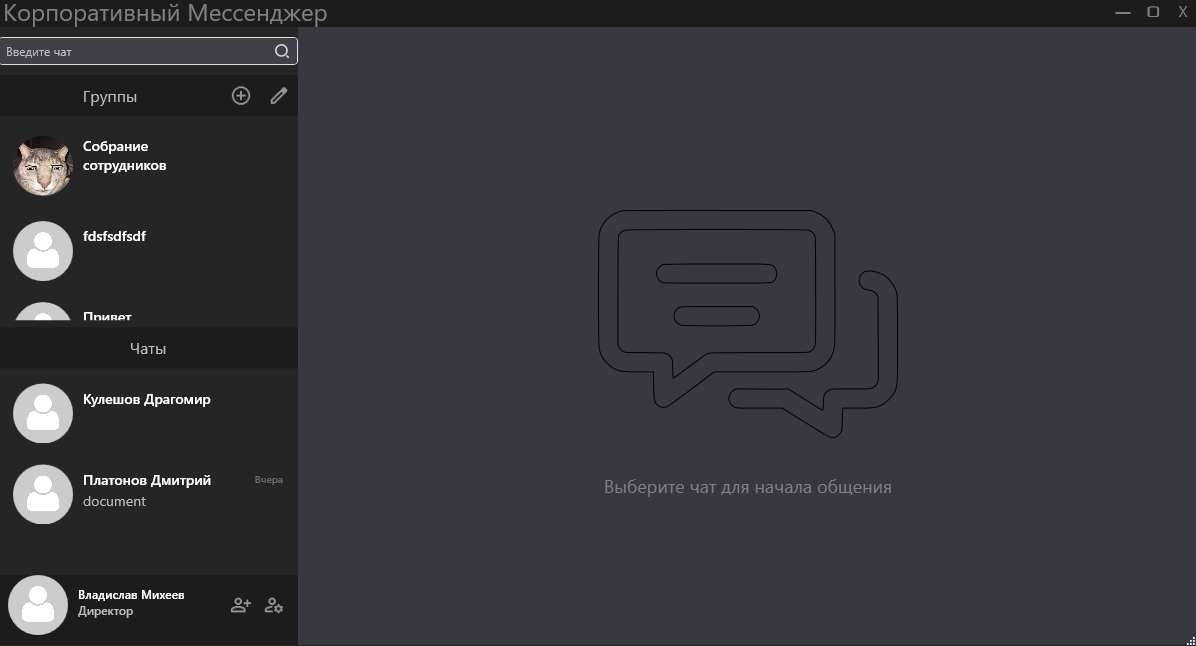


Рисунок 5 – Главное окно

Пользователь может перейти на окно с сообщениями, нажав на любой из доступных чатов. (Рисунок 6)

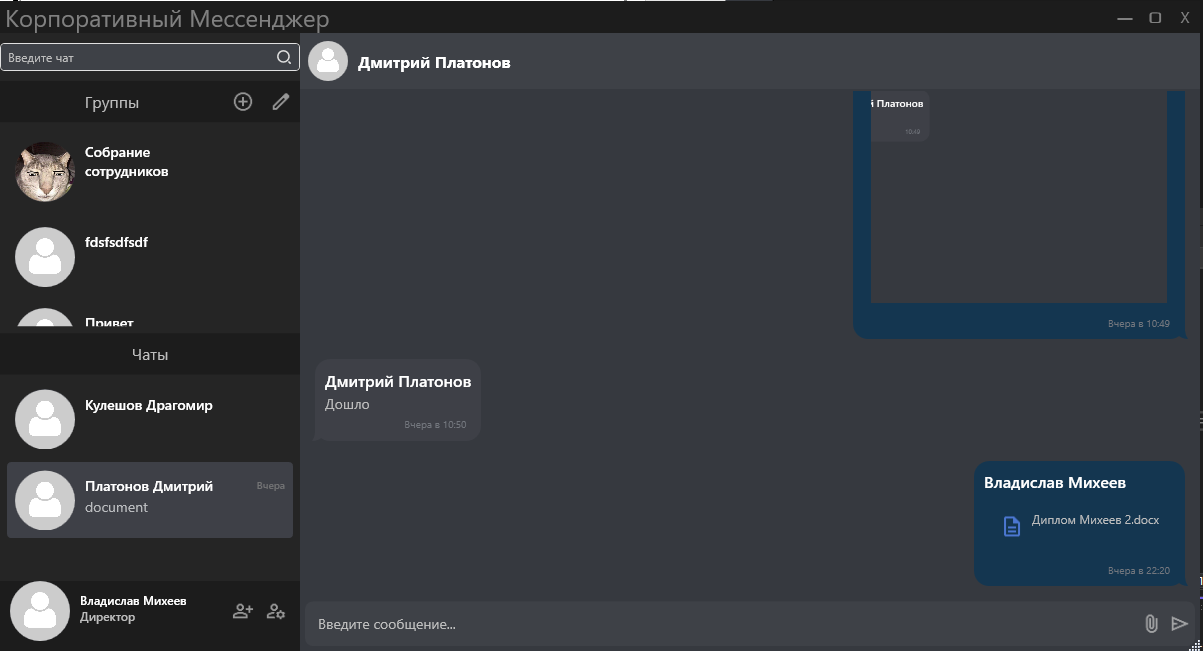


Рисунок 6 – Окно с сообщениями

Администратор может перейти на окно добавления нового пользователя. (Рисунок 7)

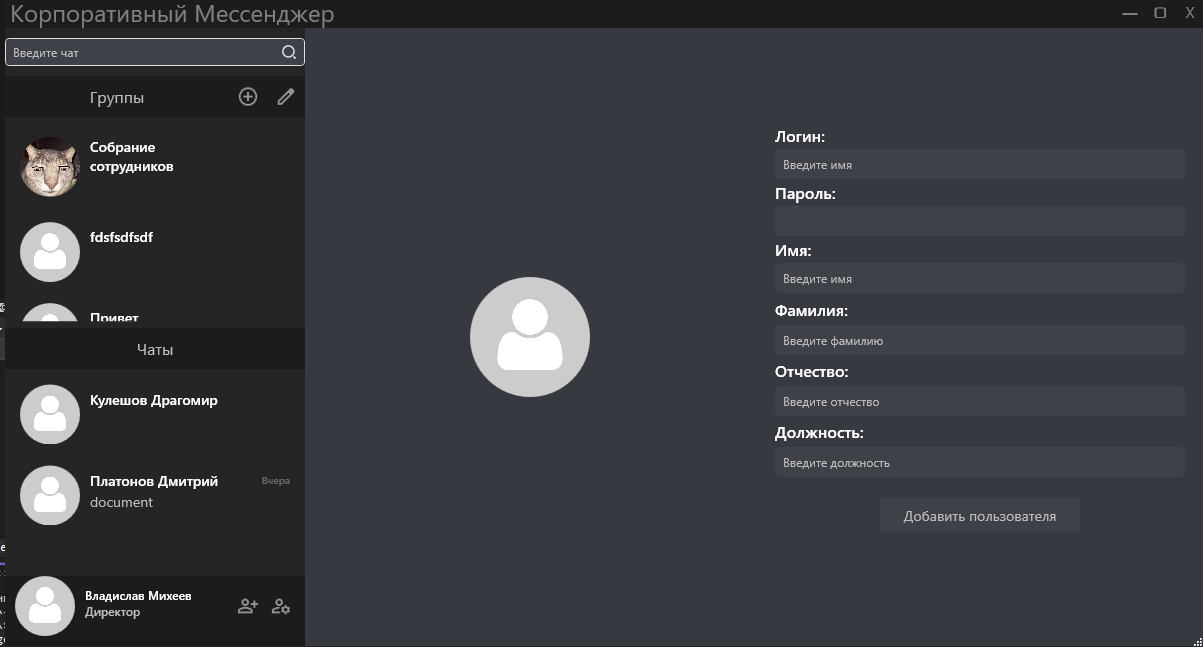


Рисунок 7 – Окно с добавлением нового пользователя

Пользователь может перейти на окно с созданием и редактированием группы. (Рисунок 8)

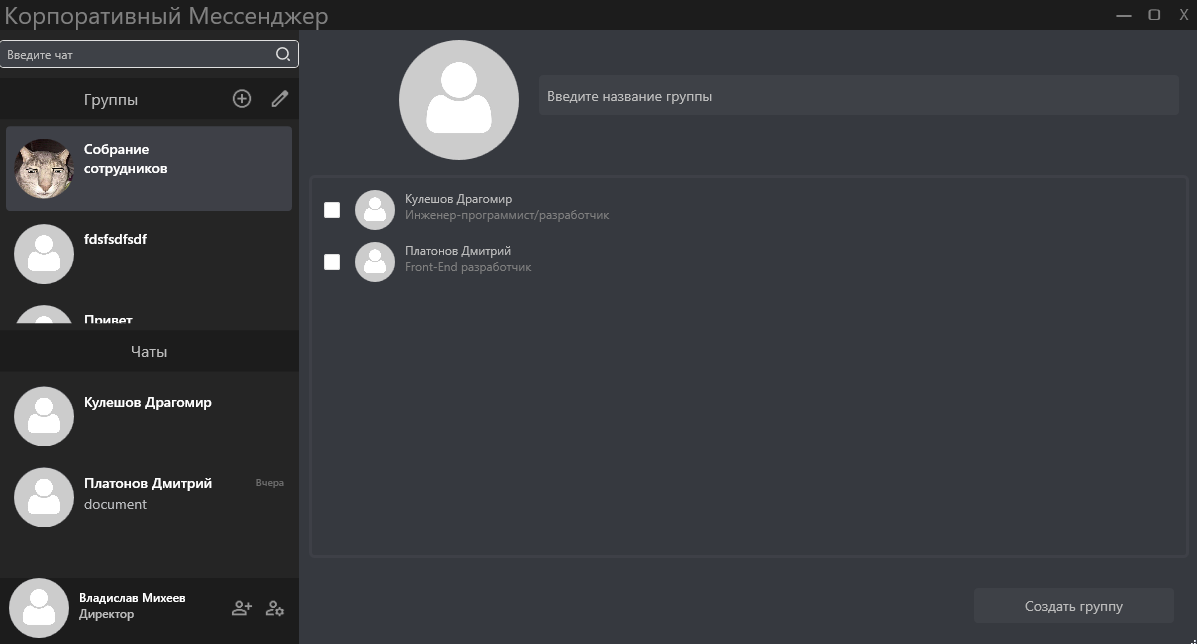


Рисунок 8 – Окно с добавлением нового пользователя

Разработка пользовательского интерфейса велась с ориентацией на максимальное упрощение рабочих коммуникаций в организации. Система предлагает лаконичный и функциональный дизайн, где каждый элемент служит конкретной деловой задаче.

Визуальная концепция построена на сочетании темных тонов с акцентными элементами корпоративного стиля, что создает профессиональную атмосферу без излишней строгости. Особое внимание уделено плавности переходов между экранами и мгновенной реакции на действия пользователя.

Административная часть интерфейса органично интегрирована в общую структуру, сохраняя при этом необходимый уровень контроля. Простота управления сочетается с достаточной функциональностью для решения повседневных задач корпоративного общения.

Получившийся интерфейс превращает процесс обмена сообщениями в естественный рабочий инструмент, который не требует специального обучения. Четкая визуальная иерархия направляет внимание пользователя на текущие задачи, а продуманная организация пространства исключает информационную перегрузку.

Система демонстрирует, как современные подходы к проектированию интерфейсов могут сделать корпоративные коммуникации более эффективными. Удобство использования достигается за счет продуманной компоновки элементов и ориентации на реальные рабочие сценарии сотрудников.

1. Разработка информационной системы
   * 1. Модель данных

Разработка структуры базы данных осуществлялась с целью создания надежного фундамента для хранения и обработки корпоративных коммуникаций. Архитектура базы построена с учетом требований к производительности и безопасности, обеспечивая эффективное управление пользователями, чатами и сообщениями.

Модель «Пользователь»:

Модель «Пользователь» представляет собой ключевую сущность системы корпоративного мессенджера, отвечающую за хранение учетных и персональных данных сотрудников. Она обеспечивает возможность аутентификации, авторизации и управления профилем. Содержит сведения о полном имени, должности и логине пользователя. Для персонализации предусмотрена загрузка аватара. В целях безопасности реализовано хранение пароля в защищённом виде. Контроль активности обеспечивается через дату регистрации, а также возможность отслеживания текущего состояния учетной записи.

Модель «Чат»:

Модель «Чат» описывает структуру коммуникационных каналов между участниками системы. Поддерживает как личные, так и групповые беседы, что позволяет организовать гибкое взаимодействие внутри коллектива. Для групповых чатов предусмотрена возможность задания визуального представления, управления участниками и определения прав доступа. Модель также фиксирует дату создания, что дает возможность отслеживать активность и управлять архивами общения.

Модель «Сообщение»:

Модель «Сообщение» является ключевым элементом системы, предназначенной для хранения и отображения коммуникации между пользователями. Каждое сообщение привязано к определённому чату и содержит данные об отправителе, что обеспечивает контекст общения и позволяет точно идентифицировать автора. Хронология переписки обеспечивается благодаря метке времени отправки, что особенно важно для анализа активности и логирования событий. Поддержка текстового содержания делает модель универсальной для обмена информацией в личных и групповых чатах.

Модель «Файл»:

Модель «Файл сообщения» реализует хранение вложений, прикрепляемых к сообщениям в чатах. Каждая запись содержит информацию о пути к файлу на сервере, его имени и типе, а также уникальный идентификатор, связывающий файл с конкретным сообщением. Такая структура позволяет управлять файлами независимо от основного текста сообщений, повышая гибкость системы и упрощая реализацию функций скачивания, фильтрации и анализа переданных данных.

Все поля спроектированной базы данных описаны в таблице 1 – в словаре данных.

Словарь данных – структурированный набор терминов и их определений, используемых для описания данных, используемых в системе или проекте.

Таблица 1 – Словарь данных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Users |  | |  | |  | |  | |
| Ключ | Поле | | Тип данных | | Обязательное | | Примечание | |
| Первичный | id | | uuid | | Да | | Уникальный идентификатор | |
|  | login | | varchar | | Да | | Логин | |
|  | password | | varchar | | Да | | Хеш пароля | |
|  | created\_at | | timestamptz | | Да | | Дата создания | |
|  | name | | varchar | | Да | | Имя | |
|  | username | | varchar | | Да | | Фамилия | |
|  | patronymic | | varchar | | Нет | | Отчество | |
|  | avatar | | bytea | | Да | | Фото | |
|  | current\_position | | varchar | | Да | | Должность | |
|  | role | | varchar | | Да | | Роль в системе | |
| Chats | |  | |  | |  | |  |
| Ключ | | Поле | | Тип данных | | Обязательное | | Примечание |
| Первичный | | Id | | uuid | | Да | | Уникальный идентификатор |
|  | | name | | varchar | | Да | | Название |
|  | | is\_group | | bool | | Нет | | Групповой чат |
|  | | created\_at | | timestamptz | | Да | | Дата создания |
|  | | avatar | | bytea | | Да | | Фото |
|  | | creator | | uuid | | Нет | | Создатель |
| messages | |  | |  | |  | |  |
| Ключ | | Поле | | Тип данных | | Обязательное | | Примечание |
| Первичный | | id | | Uuid | | Да | | Уникальный идентификатор |
|  | | text | | varchar | | Нет | | Текст сообщения |
|  | | sender\_id | | uuid | | Да | | идентификатор отправителя |

Продолжение таблицы 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ключ | | Поле | | Тип данных | | Обязательное | | Примечание | |
|  | | chat\_id | | uuid | | Да | | идентификатор чата | |
|  | | sent\_at | | timestamptz | | Да | | Дата отправления | |
|  | | file\_url | | varchar | | Нет | | Путь файла | |
|  | | file\_name | | varchar | | Нет | | Имя файла | |
|  | | file\_type | | varchar | | Нет | | Тип файла | |
|  | | file\_size | | bytea | | Нет | | Байты файла | |
| chat\_user | | | | | | | | |
| Ключ | Поле | | Тип данных | | Обязательное | | Примечание | |
|  | user\_id | | Uuid | | Да | | идентификатор отправителя | |
|  | chat\_id | | Uuid | | Да | | идентификатор чата | |

Исходя из предметной области была построена ER-диаграмма для базы данных, предназначенной для эффективного управления арендами автомобилей.

ER-диаграмма (сокращение от Entity-Relationship diagram) – это графическое представление структуры базы данных, которое помогает моделировать и анализировать взаимосвязи между сущностями в системе. Этот метод дает возможность визуализировать основные сущности, их атрибуты и связи между ними. На рисунке 23 изображена ER-диаграмма для базы данных.

База данных спроектирована с использованием первичных ключей (Primary Key) и вторичных ключей (Foreign Key) на уровне схемы. Связи между таблицами реализованы через логические поля (например, user\_id в таблице Chats), что позволяет гибко управлять данными, но требует дополнительного контроля целостности на уровне приложения. ER диаграмма представлена на рисунке 9.

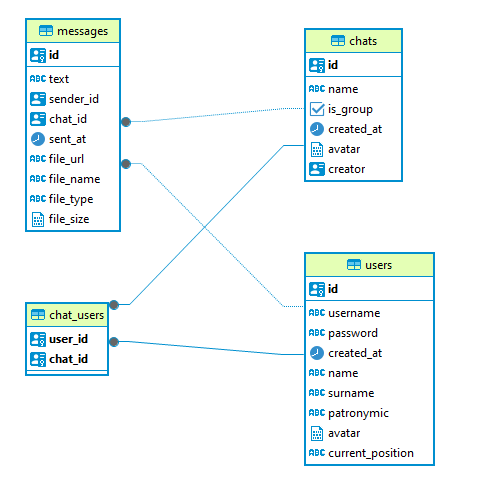


Рисунок 9 – ER диаграмма

* + 1. Структура проекта

Клиентское приложение (CorparateMessenger):

Архитектура проекта организована с соблюдением принципов модульности и четкого разделения ответственности между компонентами. В основе структуры лежит паттерн MVVM, который обеспечивает логическое разделение пользовательского интерфейса, бизнес-логики и данных.

Проект включает несколько ключевых директорий, каждая из которых выполняет строго определенную роль в общей архитектуре приложения. В корневой директории сосредоточены основные конфигурационные файлы, определяющие параметры сборки и запуска приложения. Здесь же располагаются глобальные ресурсы, используемые во всех модулях системы.

Директория ViewModels содержит классы, отвечающие за представление данных и бизнес-логику приложения. Эти компоненты выступают посредниками между пользовательским интерфейсом и моделями данных, обеспечивая гибкость и тестируемость кода. Views включают XAML-разметку, определяющую визуальное представление всех экранов приложения, с четким разделением на отдельные окна и пользовательские элементы управления.

Особое внимание уделено единообразию визуального стиля, которое достигается за счет централизованного хранения ресурсов в директории Themes. Здесь собраны все стили, шаблоны и цветовые схемы, используемые в приложении. Графические элементы и иконки вынесены в отдельную папку Icons, что упрощает их повторное использование и обновление.

Для обработки специфических преобразований данных в интерфейсе предназначена директория Converters, содержащая реализации интерфейса IValueConverter. Вспомогательные утилиты и общие инструменты вынесены в папку Tools, где собраны различные хелперы, расширения и сервисные классы, используемые в разных частях приложения.

Структура проекта разработана с учетом принципов модульности, удобства поддержки и масштабируемости. Каждая директория имеет четкое назначение и содержит файлы или ресурсы, которые отвечают за определенный аспект работы информационной системы.

Структура клиентского приложения «CorparateMessenger» представлена на рисунке 10.

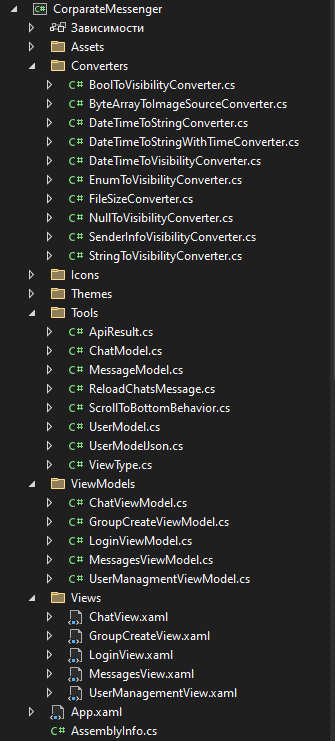


Рисунок 10 – Структура клиентского приложения

Web API (Server):

В корне проекта размещаются базовые конфигурационные файлы и точки входа в приложение. Здесь задаются параметры запуска, настройки подключения к базе данных и общие зависимости, необходимые для работы всех модулей системы.

Директория Controllers содержит контроллеры, отвечающие за обработку входящих HTTP-запросов. Каждый контроллер связан с конкретной областью функциональности – управление пользователями, чатами, сообщениями и т.д. Контроллеры обеспечивают маршрутизацию запросов, выполняют валидацию входных данных и формируют стандартизированные HTTP-ответы для клиента.

Директория Hubs включает компоненты SignalR, реализующие взаимодействие в реальном времени. Основной элемент – ChatHub, который обрабатывает события чатов, доставляет сообщения и синхронизирует действия между всеми активными подключениями. Это позволяет системе обеспечивать мгновенную реакцию на действия пользователей без использования опроса сервера.

Директория Models делится на два уровня: модели базы данных (DB модели) и модели передачи данных (DTO). DB-модели отражают структуру таблиц в базе данных и используются в работе с хранилищем, а DTO-модели адаптированы для безопасной передачи данных между сервером и клиентом. Такое разделение позволяет избежать утечки лишней или чувствительной информации во внешние интерфейсы.

В Repositories сосредоточена логика доступа к данным. Этот слой инкапсулирует все взаимодействия с базой данных и предоставляет контроллерам и сервисам необходимые методы для получения, обновления или удаления информации. Репозитории абстрагируют реализацию хранилища, что упрощает замену источника данных или его масштабирование.

В Services реализована бизнес-логика приложения. Сервисы координируют работу репозиториев, выполняют преобразование данных, а также реализуют сложные процессы, такие как регистрация пользователей, отправка сообщений или управление сессиями. Они служат связующим звеном между контроллерами и уровнем доступа к данным.

Директория Tools содержит вспомогательные классы и общие механизмы, такие как обработка результатов (Result, DataResult) и MainConnector, обеспечивающий стандартизированное взаимодействие между слоями. Эти компоненты помогают унифицировать поведение приложения и минимизировать дублирование кода.

Архитектура серверной части проекта реализована с соблюдением современных принципов разработки API, обеспечивая надежную и масштабируемую основу для работы клиентского приложения. Основной акцент сделан на четком разделении слоев приложения и эффективной организации потока данных.

Структура Web API «Server» представлена на рисунке 11.

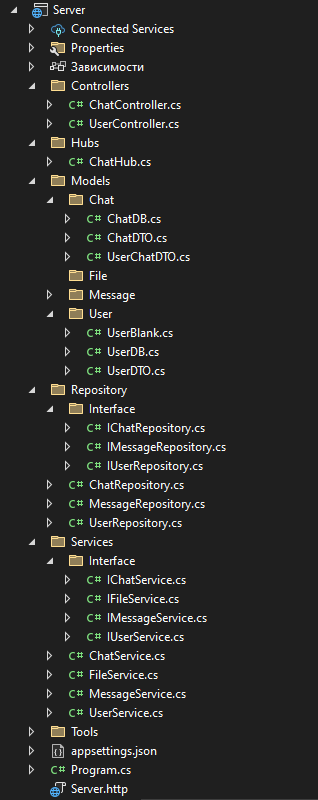


Рисунок 10 – Структура Web API

* 1. Оценка качества программного обеспечения

Оценка качества программного обеспечения является важным этапом разработки информационной системы управления арендами и пользователями. Целью данного процесса является убедиться, что система соответствует всем требованиям, работает стабильно и надежно в различных сценариях использования.

Тест-кейсы играют центральную роль в процессе оценки качества программного обеспечения. Они представляют собой подробные сценарии проверки.

Процесс добавления нового клиента является одним из ключевых функционалов информационной системы управления арендами и пользователями.

Тест добавления нового клиента направлен на проверку корректности работы модуля управления клиентами, который позволяет администратору создавать новые записи о клиентах. Успешное выполнение данного теста гарантирует, что система способна эффективно расширять базу данных клиентов, обеспечивая удобство работы с информацией о пользователях. Тест-кейс представлен в таблице 2.

Таблица 2 – Тест-кейс добавления нового клиента

|  |  |
| --- | --- |
| Статус | Пройден |
| Название | Регистрация нового клиента |
| Цель | Убедиться, что система корректно добавляет нового клиента в базу данных. |
| Функция | AddUser |
| Предусловия | * Окно регистрации открыто. |
| Входные данные | Логин: «string»  Пароль: «string»  Имя: «Иван»  Фамилия: «Иванов»  Отчество: «Иванович»  Должность: «Сотрудник» |

Продолжение таблицы 2.

|  |  |  |
| --- | --- | --- |
| Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1. Нажать кнопку «Добавить пользователя»; 2. Заполнить формы; 3. Проверить, что пользователь добавился. | * Клиент успешно добавлен в базу данных; * Чат с клиентом отображается в главном окне. | * Клиент успешно добавлен в базу данных; * Чат с клиентом отображается в главном окне. |
| Комментарий | Тест выполнен успешно. Все шаги выполнены в соответствии с ожидаемым результатом. | |

Таким образом, тестирование процесса добавления нового клиента подтвердило корректность работы модуля управления клиентами. Все шаги выполнения теста были успешно реализованы, а фактические результаты совпали с ожидаемыми. Это гарантирует, что система способна эффективно обрабатывать новые записи о клиентах, обеспечивая точность и удобство работы с базой данных.

Тест создания новой группы направлен на проверку корректности работы модуля создания группы, который позволяет пользователям создавать новые группы с участниками. Успешное выполнение данного теста гарантирует, что система способна эффективно расширять базу данных клиентов, обеспечивая удобство работы с информацией о пользователях. Тест-кейс представлен в таблице 3.

Таблица 3 – Тест-кейс создания новой группы

|  |  |
| --- | --- |
| Статус | Пройден |
| Название | Создание группового чата |
| Цель | Убедиться, что система корректно создает групповой чат. |
| Функция | CreateGroup |
| Предусловия | * Пользователь авторизован в системе; * Главное окно открыто; |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| Предусловия | * Окно с создание группы открыто; | |
| Входные данные | Название: «string»  Фото: «null»  Пользователи: «Дима, Алексей» | |
| Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1. Нажать кнопку «Создать группу»; 2. Заполнить формы; 3. Проверить, что группы была создана. | * Группа успешно добавлен в базу данных; * Группа отображается в главном окне. | * Группа успешно добавлен в базу данных; * Группа отображается в главном окне. |
| Комментарий | Тест выполнен успешно. Все шаги выполнены в соответствии с ожидаемым результатом. | |

Таким образом, тестирование процесса создания новой группы подтвердило корректность работы системы в части работы с данными. Все шаги выполнения теста соответствуют ожидаемым результатам, что свидетельствует о надежности и точности данного функционала.

Тестирование проводилось с использованием тест-кейсов, которые позволили систематизировать проверку операций системы. Результаты тестирования показали, что система корректно выполняет заявленные функции.

В целом, проведенная оценка качества программного обеспечения подтвердила, что информационная система для автоматизации операций по аренде автотранспорта является надежным, функциональным и удобным инструментом для автоматизации бизнес-процессов. Система готова к внедрению и способна обеспечить хороший уровень комфорта для администраторов и клиентов, а также стать основой для дальнейшего развития и расширения функционала.

1. ЭКОНОМИЧЕСКИЙ РАЗДЕЛ
   1. Понятие себестоимости

Себестоимость (товаров, работ, услуг) — стоимостная оценка используемых в процессе производства природных ресурсов, сырья, материалов, основных фондов, трудовых ресурсов и других затрат на ее производство и реализацию.

В себестоимости находят отражение стоимостная оценка природных ресурсов, сырья, материалов, топлива, энергии, трудовых ресурсов.

В себестоимость включается часть стоимости основных фондов, используемых для производства продукции, а также расходы, связанные с обслуживанием и управлением производством.

Себестоимость продукции образует основу цены товаров и является одним из качественных показателей работы предприятия.

От уровня себестоимости зависят прибыль, получаемая предприятиями и рентабельность продукции.

Себестоимость может быть фактической и нормативной.

При расчете фактической себестоимости берутся реальные данные, т.е. исходя из фактических издержек, формируется цена товара, работы или услуги. Производить такой расчет очень неудобно, т.к. зачастую узнать себестоимость необходимо раньше, чем производится конкретное производительное действие. От этого зависит рентабельность бизнеса.

Рассчитывая нормативную себестоимость, данные берутся согласно производственных норм.

Благодаря этому можно жестко контролировать расход материалов, что минимизирует возникновение неоправданных расходов.

В следующем вопросе конкретизируются затраты, включаемые в состав себестоимости.

* 1. Затраты в составе себестоимости

Перечень расходов, включаемых в себестоимость продукции (работ, услуг) в целях расчета налогооблагаемой прибыли, определяется в соответствии с законодательством Российской Федерации, в частности с главой 25 НК РФ «Налог на прибыль организаций».

Расходами признаются обоснованные и документально подтвержденные затраты, осуществленные предприятием.

Под обоснованными расходами понимаются экономически оправданные затраты, оценка которых выражена в денежной форме.

Под документально подтвержденными расходами понимаются затраты, подтвержденные документами, оформленными в соответствии с законодательством Российской Федерации. Расходами признаются любые затраты при условии, что они произведены для осуществления деятельности, направленной на получение дохода.

Расходы в зависимости от их характера, а также условий осуществления и направлений деятельности предприятия подразделяются на расходы, связанные с производством и реализацией, и внереализационные расходы.

Расходы, связанные с производством и реализацией, включают в себя:

* расходы, связанные с изготовлением (производством) или реализацией товаров (работ, услуг, имущественных прав);
* расходы на содержание и эксплуатацию, ремонт и техническое обслуживание основных средств и иного имущества, а также на поддержание их в исправном состоянии;
* расходы на обязательное и добровольное страхование;
* прочие расходы, связанные с производством и (или) реализацией.
* расходы, связанные с производством и реализацией и образующие себестоимость продукции, группируются в соответствии с их экономическим содержанием по следующим элементам:
* материальные расходы;
* расходы на оплату труда;
* суммы начисленной амортизации;
* прочие расходы.

К материальным расходам, в частности, относятся следующие затраты предприятия:

* на приобретение сырья и (или) материалов, которые входят в состав вырабатываемой продукции (основные материалы);
* приобретение материалов для обеспечения технологического процесса и для упаковки продукции или расходуемых на другие производственные и хозяйственные нужды;
* приобретение запасных частей и расходных материалов, используемых для ремонта оборудования, инструментов, приспособлений и др.;
* приобретение топлива, воды и энергии всех видов, расходуемых на технологические цели, выработку всех видов энергии, отопление зданий, а также расходы на трансформацию и передачу энергии;
* расходы, связанные с содержанием и эксплуатацией фондов природоохранного назначения, а также платежи за предельно допустимые выбросы загрязняющих веществ в природную среду.

Под возвратными отходами понимаются остатки сырья, материалов, полуфабрикатов, теплоносителей и других видов материальных ресурсов, полностью или частично утратившие потребительские качества исходного ресурса и в силу этого используемые с повышенными затратами (с понижением выхода продукции) или вовсе не используемые по прямому назначению.

К расходам на оплату труда относятся затраты предприятия на оплату труда основного производственного персонала, а также не состоящих в штате работников, участвующих в основной деятельности предприятия.

Затраты на оплату труда включают в себя заработную плату, начисляемую по сдельным расценкам, тарифным ставкам и должностным окладам в соответствии с системами оплаты труда, принятыми на предприятии: надбавки о доплаты, премии за производственные результаты, оплату очередных и дополнительных отпусков, единовременные вознаграждения за выслугу лет и другие расходы, предусмотренные законодательством Российской Федерации.

К суммам начисленной амортизации относятся суммы амортизационных отчислений по амортизируемому имуществу (основным средствам и нематериальным активам).

К прочим расходам относятся:

* расходы на сертификацию продукции и услуг;
* расходы на оплату услуг по охране имущества, обслуживанию охранной пожарной сигнализации, услуг пожарной охраны и иных услуг охранной деятельности;
* расходы по обеспечению нормальных условий труда и техники безопасности, предусмотренных законодательством РФ;
* арендные (лизинговые) платежи за арендуемое имущество;
* расходы на содержание служебного автотранспорта;
* расходы на командировки в пределах норм, утвержденных
* правительством РФ;
* расходы на канцелярские товары;
* расходы на подготовку и освоение новых производств, цехов и агрегатов;
* расходы на ремонт основных средств.
* В состав внереализационных расходов включают:
* расходы на содержание переданного по договору аренды имущества;
* проценты по долговым обязательствам любого вида, включая проценты по кредитам и ценные бумаги, выпущенным предприятием;
* расходы, связанные с организацией выпуска и обслуживанием собственных ценных бумаг;
* судебные расходы и арбитражные сборы и др.

К внереализационным расходам в целях определения налогооблагаемой прибыли приравниваются убытки, связанные с потерями от брака, потерями от стихийных бедствий, пожаров, аварий и других чрезвычайных ситуаций, а также убытки прошлых периодов, выявленные в текущем периоде.

* 1. Расчет себестоимости информационной системы

Расчет себестоимости разрабатываемой информационной системы включает определенную последовательность действий. Результаты расчетных действий приведены в таблицах, использованы при необходимости соответствующие формулы.

Расчет трудоемкости произведен в таблице 4.

Таблица 4 – Расчет трудоёмкости

|  |  |
| --- | --- |
| Виды работ | Трудоёмкость, дн. |
| Получение ТЗ | 1 |
| Сбор информации и ознакомление с предметной | 2 |
| Разработка основного алгоритма | 2 |
| Разработка программы | 7 |
| Отладка | 1 |

Продолжение таблицы 4

|  |  |
| --- | --- |
| Виды работ | Трудоёмкость, дн. |
| Подготовка технической документации | 2 |
| Сдача продукта | 1 |
| Подготовка технической документации | 2 |
| Сдача продукта | 1 |
| Итого | 16 |

Расчет затрат на разработку информационной системы.

Величина себестоимости разрабатываемого программного продукта рассчитывается по формуле 1.

С = Сзп + Сэл + Сам + См + Сн + Ссоц.ф , где: (1)

С – величина себестоимости разрабатываемого программного продукта;

Сзп – заработная плата;

СЭл – расходы на электроэнергию;

САм – сумма амортизационных отчислений;

См – затраты на расходные материалы;

Сн – накладные расходы;

Ссоц.ф – отчисления в социальные фонды;

Численность – 2 человека (студент-разработчик и преподаватель).

Зарплата персонала (ЗП) показана в таблице 5. За основу заработной платы студента, разрабатывающего программный продукт, взята величина его заработной платы на должности «Слесарь механосборочных работ» в АО «НПК «КБМ».

Таблица 5 – Расчет основной заработной платы

|  |  |  |
| --- | --- | --- |
| Персонал | Количество  (чел.) | ЗП за время разработки  (руб.) |
| Руководитель диплома | 1 | 3 200 |
| Студент | 1 | 12 283,84 |
| Всего | 2 | 15 483,84 |

Расчет заработной платы.

Заработная плата руководителя диплома за час = 400 руб.

Время работы руководителя диплома = 8 ч.

Заработная плата руководителя диплома на время разработки дипломного проекта = 8 \* 400= 3 200 руб.

Величина оплаты труда студента на своей рабочей должности составит 23 800 руб. В расчетах исходим из того, что заработная плата начисляется за 31 день. Посуточная заработная плата студента за один месяц работы в мае составит 23 800 / 31 = 767,74 руб.

Время работы студента – 128 часов за 16 дней (с учетом 8 часов работы в день) осуществления проекта. Тогда зарплата за 16 дней работы составит 767,74 \* 16 = 12 283,84 рублей.

Затраты на электроэнергию рассчитываются по формуле 2:

|  |  |
| --- | --- |
| Сэл=Р\*Т\* Z, | (2) |

где: Р – общая мощность оборудования (КВт);

Т – общее время работы оборудования (р.ч.);

Z – цена одного КВт (В 2025 г. – 7 руб. 33 коп).

Затраты на электроэнергию показаны в таблице 6.

Таблица 6 – Расчет затрат на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оборудование | Количество,  шт. | Общая  стоимость,  руб. | Общая  мощность,  КВт | Количество часов работы, р.ч. |
| Компьютер | 1 | 750,59 | 0,8 | 128 |
| Струйный принтер | 1 | 13,9 | 0,6 | 3 |
| Итого | 2 | 763,78 | 1,4 | 131 |

Амортизационные отчисления.

Расчет амортизационных отчислений на рабочий период производится по формуле 3.

|  |  |
| --- | --- |
| Сам = (Спер\*Н a) / (365/16), | (3) |

где: Спер – первоначальная стоимость оборудования;

Нa – годовая норма амортизации (20%);

365 – количество дней в году;

16 – количество дней на разработку проекта.

Амортизационные отчисления рассчитываются на основании таблицы 7.

Таблица 7 – Амортизационные отчисления

|  |  |  |  |
| --- | --- | --- | --- |
| Оборудование | Количество,  шт. | Первоначальная стоимость оборудования, руб. | Стоимость амортизационных отчислений, руб. |
| Ноутбук | 1 | 99 000 | 867,95 |
| Струйный  принтер | 1 | 15 600 | 136,77 |
| Итого | 2 | 114 600 | 1 004,72 |

Затраты на материалы.

Расчет затрат на материалы осуществляется согласно фактическим данным в год, которые представлены в таблице 8.

Таблица 8 – Расчет затрат на материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  материала | Количество шт. | Стоимость за 1 шт., руб. | Общая стоимость, руб. |
| Бумага | 1 | 330 | 330 |
| Картридж для принтера | 1 | 1 562 | 1 562 |

Продолжение таблицы 8

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  материала | Количество шт. | Стоимость за 1 шт., руб. | Общая стоимость, руб. |
| Флэш-накопитель 4 GB | 1 | 219 | 219 |
| Итого | | | 2 111 |

Носителем информации был выбран флэш-накопитель, так как этот вариант является наиболее экономичным и современным.

Был произведен расчет накладных расходов для выбранного проекта и показан в таблице 9.

В нашей стране законодательно закреплены накладные расходы только в таких сферах как строительство и медицина и в ряде других видах деятельности. Остальные предприниматели и компании устанавливают самостоятельно перечень подобных затрат. Установим величину накладных расходов на уровне 31%.

Накладные расходы считаются по формуле 4 (Общепроизводственные и общехозяйственные расходы):

|  |  |
| --- | --- |
| Сн = ЗПосн \* % накл. расх. / 100, | (4) |

где: % накл. расх. – процент накладных расходов – 31.

Таблица 9 – Расчет накладных расходов

|  |  |  |  |
| --- | --- | --- | --- |
| Персонал | Количество | Заработная | Накладные |
|  | (чел) | плата, руб. | расходы, руб. |
| Руководитель диплома | 1 | 3 200 | 992 |
| Студент | 1 | 12 283,84 | 3 808 |
| Итого | | | 4 800 |

Расчет отчислений в социальные фонды по формуле 5 показаны в таблице 10:

|  |  |
| --- | --- |
| Ссоцф = ЗПосн \* % отч. / 100, | (5) |

где: % отч. – процент отчислений (В 2025 г. – 30).

Изменения в тарифах в 2025 г. касаются части субъектов малого бизнеса. Для субъектов малого предпринимательства в обрабатывающей промышленности тарифы будут снижены: 30% с суммы 1,5 МРОТ и 7,6% с суммы, превышающей 1,5 МРОТ.

В наших расчетах изменений в соответствии с формулой 5 не будет, так как 1.5 МРОТ составляет в 2025 году 33660 рублей, а обозначенная заработная плата – 30 968 рублей. Итоги расчетов представлены в таблице 12.

Таблица 10 – Расчет отчислений в социальный фонд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Персонал | Количество  (чел) | Заработная плата, руб. | | Сумма отчислений, руб. |
| Руководитель диплома | 1 | | 3 200 | 960 |
| Студент | 1 | | 12 283,84 | 3 685,15 |
| Итого | | | | 4 645,15 |

Затраты на разработку программного продукта рассчитываются по формуле (6):

С = 15 483,84+ 763,78+ 1 004,72+ 2 111+ 4 800+ 4 645,15 = 28 808,49 руб. (6)

Удельный вес по элементам затрат представлен в таблице 11.

Таблица 11 – Удельный вес по элементам затрат

|  |  |  |
| --- | --- | --- |
| Элементы затрат | Сумма (руб.) | Удельный вес (%) |
| Зарплата персонала | 15 483,84 | 53,7 |
| Расходы на электроэнергию | 763,78 | 2,7 |
| Сумма амортизационных отчислений | 1 004,72 | 3,5 |

Продолжение таблицы 11.

|  |  |  |
| --- | --- | --- |
| Элементы затрат | Сумма (руб.) | Удельный вес (%) |
| Сумма затрат на материалы | 2 111 | 7,3 |
| Накладные расходы | 4 800 | 16,7 |
| Отчисления в соц. фонды | 4 645,15 | 16,1 |
| Итого | 28 808,49 | 100 |

На рисунке 25 представлена диаграмма себестоимости программного продукта. Из диаграммы следует, что наибольший удельный вес поэлементных затрат характерен для зарплаты персонала, накладных расходов и отчислений в социальные фонды.

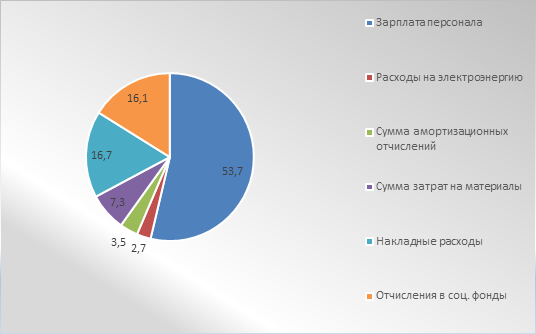


Рисунок 25 – Себестоимость программного продукта, в процентах (%)

В ходе выполнения экономических расчетов был проведен анализ издержек, возникающих при разработке информационной системы.

В результате проведенного анализа удалось выявить, что основная сумма себестоимости складывается из заработной платы (58%), расходов на электроэнергию (1,3%), отчислений в социальные фонды (17,4%), амортизационных отчислений (1,8%), накладных расходов (17,9%) и затрат на материалы (3,6%).

Общая сумма себестоимости разработанной информационной системы составляет 58 887,68 рублей.

Окупаемость полученной разработки возможна только в случае ее коммерческого применения или реализации на основе соответствующего бюджетного заказа с учетом ожидаемой нормы прибыли.

Проведенное исследование позволило получить навык рассчитывать затраты на разработку информационной системы в целях последующего внедрения в системы автоматизации процессов на предприятии в соответствии с освоением профильной специальности.

1. ОХРАНА ТРУДА ПРИ РАБОТЕ С ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКОЙ

Только длительная работа за компьютером может оказать существенное влияние на здоровье человека.

Рассмотрим основные аспекты длительной работы за компьютером:

Работающий за компьютером человек длительное время должен сохранять относительно неподвижное положение, что негативно сказывается на позвоночнике и циркуляции крови во всем организме (застой крови). При длительных нарушениях циркуляции крови нарушается питание тканей и повреждаются стенки сосудов, что в свою очередь приводит к их необратимому расширению. Такое расширение сосудов наблюдается, например, при геморрое.

Чтение информации с монитора вызывает перенапряжение глаз. Возникает это главным образом потому, что во время чтения с монитора расстояние от текста до глаз постоянно остается одним и тем же, из-за этого мышцы глаз, регулирующие аккомодацию, находятся в постоянном напряжении. Со временем это может привести к нарушению аккомодативной способности глаз и, следовательно, к нарушениям зрения.

Дисплейная болезнь, характеризуется нарушением аккомодации глаз из-за длительного перенапряжения ресничного тела. Синдром сухого глаза – собирательное название заболевания, вызванного нарушением увлажнения передней поверхности глаза (роговицы) слезной жидкостью. Также длительная работа за компьютером может увеличить риск таких глазных заболеваний как миопия (близорукость), дальнозоркость, глаукома.

Длительная работа на клавиатуре приводит к перенапряжению суставов кисти и мышц предплечья, что вызывает развитие туннельного синдрома запястья.

Работа за компьютером предполагает переработку большого массива информации и постоянную концентрацию внимания, поэтому при длительной работе за компьютером нередко развивается умственная усталость и нарушение внимания.

Человек, работающий за компьютером, вынужден все время принимать решения, от которых зависит эффективность его работы. Поэтому, длительная работа за компьютером, часто является причиной хронического стресса.

Все чаще появляются сообщения о возникновении компьютерной зависимости. Действительно, длительная работа за компьютером, работа в Интернете и компьютерные игры могут вызвать подобные психические расстройства.

Работа за компьютером нередко поглощает все внимание работающего человека и потому, такие люди часто пренебрегают нормальным питанием и работают впроголодь весь день. Неправильное питание приводит к снижению интеллектуальных способностей человека.

Гиподинамия, стресс, вредные привычки и неправильное питание являются основными причинами сердечно-сосудистых заболеваний и диабета. Таким образом, человек длительное время, работающий за компьютером подвергается реальному риску сердечно-сосудистых заболеваний, различных заболеваний глаз, двигательного аппарата, органов желудочно-кишечного тракта, психических расстройств.

Лечебная гимнастика.

Для развития мышц спины существует множество упражнений, в основном это наклоны в разные стороны. Очень полезно плавание и упражнения на турнике. В том случае, если вы работаете в офисе, старайтесь почаще потягивайтесь и вообще, делайте как можно больше движений, даже без надобности.

Упражнение для рук.

Чем чаще вы будете прерываться для выполнения упражнений, тем больше они принесут пользы.

Встряхните руки;

Сжимайте пальцы в кулаки (10 раз);

Вращайте кулаки вокруг своей оси;

Надавливая одной рукой на пальцы другой руки со стороны ладони, как бы выворачивая ладонь и запястье наружу.

Положения рук при работе за компьютером.

В процессе работы старайтесь удерживать туловище в вертикальном положении, опираясь на спинку стула или кресла. Ступни ног должны прочно стоять на полу или на специальной подставке, плечи следует держать в расслабленном состоянии. В оптимальной позиции руки должны быть согнуты в локте под углом 90°. Локти должны располагаться вблизи туловища.

Профилактика заболеваний органов дыхания.

Как можно чаще делайте влажную уборку помещения и проветривайте его;

Для увеличения влажности можно ставить открытую емкость с водой.

Профилактика нервных расстройств.

Постарайтесь сделать так, чтобы при работе компьютер давал как можно меньше сбоев и меньше раздражал вас;

Как можно чаще прерывайте работу с компьютером.

Рабочее место и правильная посадка.

Перед тем как приступить к работе, уделите немного времени организации своего рабочего места. Клавиатура и экран монитора должны быть расположены прямо перед вами. Если конструкция стула позволяет регулировать высоту сиденья, настройте ее таким образом, чтобы в положении сидя ваши ноги прочно опирались на пол или на специальную подставку, а поверхность, на которой расположены клавиатура и мышь, находились на уровне локтя или немного ниже. Разместите клавиатуру и мышь таким образом, чтобы при работе с ними не возникало необходимости отдалять локти от туловища.

Экран монитора должен располагаться на расстоянии вытянутой руки сидящего за столом пользователя. Отрегулируйте высоту подставки монитора таким образом, чтобы центр экрана находился чуть ниже уровня глаз. При отсутствии такой возможности желательно установить монитор на специальную полку или подставку.

При работе с бумажными документами рекомендуется установить на корпус монитора специальный держатель для бумаг.

Упражнения для глаз.

Зажмурьте глаза на ~ 10 секунд;

Быстро моргайте в течение ~5-10 сек;

Сделайте комплекс упражнений:

ВЫВОДЫ И ЗАКЛЮЧЕНИЕ

В рамках реализации проекта был разработан современный корпоративный мессенджер, который кардинально преобразует систему внутренних коммуникаций организации. Созданное решение успешно преодолевает ключевые ограничения традиционных средств корпоративного общения, предлагая сотрудникам удобный и безопасный инструмент для эффективного взаимодействия.

Система обеспечивает качественно новый уровень деловых коммуникаций, сочетая интуитивно понятный интерфейс для пользователей с мощными административными возможностями управления. Использование надежных технологий Microsoft .NET и продуманной архитектуры гарантирует стабильность работы и простоту интеграции с существующей ИТ-инфраструктурой компании.

Разработанный мессенджер представляет собой не просто инструмент для обмена сообщениями, а целостную цифровую экосистему, которая органично встраивается в бизнес-процессы организации. Минимальный порог входа и продуманный UX-дизайн обеспечивают быстрое внедрение и высокую степень адаптации сотрудников.

Данное решение демонстрирует, как современные технологии могут оптимизировать внутренние коммуникации, делая их более структурированными, безопасными и эффективными. Внедрение системы позволит организации вывести корпоративное взаимодействие на качественно новый уровень, что в конечном итоге положительно скажется на общей продуктивности работы.

СПИСОК ЛИТЕРАТУРЫ

1. Беляков Г. И. Охрана труда и техника безопасности: учебник для среднего профессионального образования / Беляков Г.И. — 3-е изд., перераб. и доп. — М: Издательство Юрайт, 2022. — 404 с.
2. Ганенко А.П. Оформление текстовых и графических материалов при подготовке дипломных проектов, курсовых и письменных экзаменационных работ (требования ЕСКД) : учебно-метод. Пособие для студ. Учреждения сред. Проф. Образования / Ганенко А.П., Лапсарь М.И. – 9-е изд., стер. – М.: Издательский центр «Академия», 2022.
3. Федорова Г.Н. Разработка, администрирование и защита баз данных: учебник для студ. Учреждений сред. проф. образования / Федорова Г.Н. – 2-е изд., стер. – М. Петкович Д., 2022
4. Кудрина Е. В. Основы алгоритмизации и программирования на языке С#: учеб. пособие для среднего профессионального образования / Е.В. Кудрина, М.В. Огнева. – М.: Издательство Юрайт, 2023;
5. Рыжко А.Л. Экономика отрасли информационных систем: учебное пособие для среднего профессионального образования / Рыжко А.Л., Рыжко Н.А., Лобанова Н.М., Кучинская Е.О. – 2-е изд., испр. и доп. –М.: Издательство Юрайт, 2024. – 176 с.

Электронные ресурсы:

1. MSDN – сеть разработчиков Microsoft [Электронный ресурс] Режим доступа: свободный <https://msdn.microsoft.com/ru-ru/dn308572.aspx> (Дата обращения 30.05.2025)
2. Metanit – язык программирования С# и платформа .NET [Электронный ресурс] Режим доступа: свободный <https://metanit.com/sharp/> (Дата обращения 20.05.2025)

ПРИЛОЖЕНИЕ А – Код программы

Весь проект содержит много строчек кода, поэтому прикрепить весь код не представляется возможным. Для этого приложение загружено в публичный репозиторий системы GitHub.

Ссылка на GitHub: <https://github.com/Taxic27/CorparateMessengerProject>

Обработка данных:

public class MessageRepository : IMessageRepository

{

private readonly IMainConnector \_mainConnector;

public MessageRepository(IMainConnector mainConnector)

{

\_mainConnector = mainConnector;

}

public MessageDTO SaveMessageAndReturning(MessageDB message)

{

const string request = @"

WITH inserted\_message AS (

INSERT INTO messages (id, chat\_id, sender\_id, text, sent\_at)

VALUES (@Id, @ChatId, @SenderId, @Text, @SentAt)

RETURNING \*

)

SELECT

m.id,

m.chat\_id as ChatId,

m.text,

m.sent\_at as SentAt,

m.file\_url as FileUrl,

m.file\_name as FileName,

m.file\_type as FileType,

m.file\_size as FileSize,

u.username as Username,

(u.name || ' ' || u.surname) AS SenderName

FROM inserted\_message m

JOIN users u ON m.sender\_id = u.id";

var parameters = new DynamicParameters();

parameters.Add("@Id", Guid.NewGuid());

parameters.Add("@ChatId", message.ChatId);

parameters.Add("@SenderId", message.SenderId);

parameters.Add("@Text", message.Text);

parameters.Add("@SentAt", DateTime.UtcNow);

return \_mainConnector.Get<MessageDTO>(request, parameters);

}

public MessageDTO SaveMessageFileAndReturning(MessageDB message)

{

const string request = @"

WITH inserted\_message AS (

INSERT INTO messages

(id, chat\_id, sender\_id, text, sent\_at, file\_url, file\_name, file\_type, file\_size)

VALUES

(@Id, @ChatId, @SenderId, @Text, @SentAt, @FileUrl, @FileName, @FileType, @FileSize)

RETURNING \*

)

SELECT

m.id,

m.chat\_id as ChatId,

m.text,

m.sent\_at as SentAt,

m.file\_url as FileUrl,

m.file\_name as FileName,

m.file\_type as FileType,

m.file\_size as FileSize,

u.username as Username,

(u.name || ' ' || u.surname) AS SenderName

FROM inserted\_message m

JOIN users u ON m.sender\_id = u.id";

var parameters = new DynamicParameters();

parameters.Add("@Id", message.Id);

parameters.Add("@ChatId", message.ChatId);

parameters.Add("@SenderId", message.SenderId);

parameters.Add("@Text", message.Text ?? string.Empty);

parameters.Add("@SentAt", message.SentAt);

parameters.Add("@FileUrl", message.FileUrl);

parameters.Add("@FileName", message.FileName);

parameters.Add("@FileType", message.FileType);

parameters.Add("@FileSize", message.FileSize);

return \_mainConnector.Get<MessageDTO>(request, parameters);

}

public List<MessageDTO> GetMessagesForChat(Guid chatId, int skip, int take)

{

const string request = @"

SELECT

m.id,

m.chat\_id AS ChatId,

m.text AS Text,

m.sent\_at AS SentAt,

m.file\_url as FileUrl,

m.file\_name as FileName,

m.file\_type as FileType,

m.file\_size as FileSize,

u.username AS Username,

(u.name || ' ' || u.surname) AS SenderName

FROM messages m

JOIN users u ON m.sender\_id = u.id

WHERE m.chat\_id = @ChatId

ORDER BY m.sent\_at DESC

OFFSET @Skip ROWS

FETCH NEXT @Take ROWS ONLY";

var parameters = new DynamicParameters();

parameters.Add("@ChatId", chatId);

parameters.Add("@Skip", skip);

parameters.Add("@Take", take);

return \_mainConnector.GetList<MessageDTO>(request, parameters);

}

}

}

Работа с сообщениями:

public partial class MessagesViewModel : ObservableObject

{

[ObservableProperty]

private bool \_isLoadingMessages;

[ObservableProperty]

private string \_messageText = string.Empty;

private UserDB \_currentUser;

public ObservableCollection<MessageModel> Messages { get; } = new();

private int \_loadedMessagesCount = 10;

private const int MessagesBatchSize = 20;

private HubConnection \_hubConnection;

[ObservableProperty]

private ChatDB? \_currentChat;

public MessagesViewModel(HubConnection hubConnection, UserDB currentUser)

{

\_hubConnection = hubConnection;

\_currentUser = currentUser;

InitializeHubMethods();

}

private void InitializeHubMethods()

{

\_hubConnection.On<MessageDTO>("ReceiveMessage", (message) =>

{

Application.Current.Dispatcher.Invoke(() =>

{

var newMessage = new MessageModel

{

Id = message.Id,

Role = message.Username == \_currentUser.Username ? ChatRoleType.Sender : ChatRoleType.Receiver,

Username = message.Username,

SenderName = message.SenderName,

Text = message.Text,

SentAt = message.SentAt,

FileUrl = message.FileUrl,

FileName = message.FileName,

FileType = message.FileType,

FileSize = message.FileSize

};

var lastMessage = Messages.LastOrDefault();

newMessage.IsSenderInfoVisible = lastMessage == null || lastMessage.SenderName != newMessage.SenderName;

Messages.Add(newMessage);

});

});

}

public async Task SetCurrentChat(ChatDB chat)

{

CurrentChat = chat;

Messages.Clear();

\_loadedMessagesCount = 0;

if (chat != null)

{

await \_hubConnection.InvokeAsync("JoinChat", chat.Id);

await LoadChatMessages();

}

}

private async Task LoadChatMessages()

{

Messages.Clear();

\_loadedMessagesCount = 0;

try

{

if (CurrentChat != null)

{

var initialMessages = await \_hubConnection.InvokeAsync<List<MessageModel>>(

"RequestMessageHistory",

CurrentChat.Id,

0,

10

);

var processedMessages = initialMessages.Select(m => new MessageModel

{

Id = m.Id,

Role = m.Username == \_currentUser.Username ? ChatRoleType.Sender :ChatRoleType.Receiver,

Username = m.Username,

SenderName = m.SenderName,

Text = m.Text,

SentAt = m.SentAt,

FileUrl = m.FileUrl,

FileName = m.FileName,

FileType = m.FileType,

FileSize = m.FileSize

}).ToList();

if (processedMessages?.Count > 0)

{

Application.Current.Dispatcher.Invoke(() =>

{

var reversedMessages = processedMessages.AsEnumerable().Reverse().ToList();

ProcessMessagesHeaders(reversedMessages, Messages.FirstOrDefault());

foreach (var message in reversedMessages)

{

Messages.Add(message);

}

\_loadedMessagesCount = processedMessages.Count;

});

}

}

}

catch (Exception ex)

{

Growl.WarningGlobal($"Ошибка загрузки чата: {ex.Message}");

}

}

[RelayCommand]

private async Task LoadMoreMessages()

{

if (IsLoadingMessages || CurrentChat == null) return;

IsLoadingMessages = true;

try

{

var additionalMessages = await \_hubConnection.InvokeAsync<List<MessageModel>>(

"RequestMessageHistory",

CurrentChat.Id,

\_loadedMessagesCount,

MessagesBatchSize);

if (additionalMessages?.Count > 0)

{

var reversedMessages = additionalMessages.AsEnumerable().ToList();

foreach (var message in reversedMessages)

{

Messages.Insert(0, message);

}

\_loadedMessagesCount += additionalMessages.Count;

}

}

finally

{

IsLoadingMessages = false;

}

}

private void ProcessMessagesHeaders(List<MessageModel> messages, MessageModel previousMessage = null)

{

string lastSender = previousMessage?.SenderName;

for (int i = 0; i < messages.Count; i++)

{

messages[i].IsSenderInfoVisible = messages[i].SenderName != lastSender;

lastSender = messages[i].SenderName;

}

}

[RelayCommand]

private async Task SendMessage()

{

if (string.IsNullOrWhiteSpace(MessageText)) return;

if (CurrentChat == null) return;

try

{

await \_hubConnection.InvokeAsync("SendMessage",

CurrentChat.Id,

\_currentUser.Id,

MessageText);

MessageText = string.Empty;

}

catch (Exception ex)

{

Growl.WarningGlobal($"Ошибка отправки: {ex.Message}");

}

}

[RelayCommand]

private async Task SendFile()

{

if (CurrentChat == null) return;

var openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

try

{

string filePath = openFileDialog.FileName;

string fileName = Path.GetFileName(filePath);

byte[] fileBytes = File.ReadAllBytes(filePath);

string fileType = GetFileType(fileName);

await \_hubConnection.InvokeAsync("SendFile",

CurrentChat.Id,

\_currentUser.Id,

fileName,

fileType,

fileBytes);

}

catch (Exception ex)

{

Growl.WarningGlobal($"Ошибка отправки файла: {ex.Message}");

}

}

}

[RelayCommand]

public async Task DownloadFile(MessageModel message)

{

if (message?.FileUrl == null) return;

try

{

string fileName = message.FileName;

string fileExtension = Path.GetExtension(fileName);

var saveDialog = new SaveFileDialog

{

FileName = fileName,

Filter = $"{fileExtension} files|\*{fileExtension}|All files|\*.\*"

};

if (saveDialog.ShowDialog() == true)

{

byte[] fileData = await \_hubConnection.InvokeAsync<byte[]>(

"DownloadFile",

message.FileUrl);

await File.WriteAllBytesAsync(saveDialog.FileName, fileData);

Growl.Success("Скачен");

}

}

catch (Exception ex)

{

Growl.WarningGlobal($"Ошибка скачивания: {ex.Message}");

}

}

private string GetFileType(string filename)

{

string extension = Path.GetExtension(filename).ToLower();

return extension switch

{

".jpg" or ".png" or ".jpeg" or ".gif" => "image",

".mp4" or ".mov" or ".avi" => "video",

".mp3" or ".wav" => "audio",

\_ => "document"

};

}

[RelayCommand]

private void ScrollChanged(ScrollChangedEventArgs e)

{

if (e.VerticalChange < 0 && e.VerticalOffset <= 10 && !IsLoadingMessages)

{

LoadMoreMessagesCommand.Execute(null);

}

}

[RelayCommand]

private void OpenImage(MessageModel message)

{

if (message?.FileUrl == null || message.FileType != "image")

return;

try

{

new ImageBrowser(new Uri(message.FileUrl))

{

Owner = WindowHelper.GetActiveWindow()

}.Show();

}

catch (Exception ex)

{

Growl.WarningGlobal($"Не удалось открыть изображение: {ex.Message}");

}

}

[RelayCommand]

private void AddNewLine(System.Windows.Controls.TextBox textBox)

{

if (textBox == null) return;

var caretPos = textBox.CaretIndex;

MessageText = MessageText.Insert(caretPos, Environment.NewLine);

textBox.CaretIndex = caretPos + Environment.NewLine.Length;

}

}

Работа с чатами:

private async void InitializeHubConnection()

{

\_hubConnection = new HubConnectionBuilder()

.WithUrl("https://localhost:7274/chatHub")

.WithAutomaticReconnect()

.Build();

\_hubConnection.Reconnecting += \_ =>

{

ConnectionState = \_hubConnection.State;

return Task.CompletedTask;

};

\_hubConnection.Reconnected += \_ =>

{

ConnectionState = \_hubConnection.State;

return Task.CompletedTask;

};

\_hubConnection.Closed += \_ =>

{

ConnectionState = \_hubConnection.State;

return Task.CompletedTask;

};

try

{

await \_hubConnection.StartAsync();

ConnectionState = \_hubConnection.State;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка подключения: {ex.Message}");

}

}

private async void LoadAvailableChats()

{

try

{

using var httpClient = new HttpClient();

var response = await httpClient.GetAsync($"https://localhost:7274/api/chats/userchats-group?currentUserId={\_currentUser.Id}");

if (response.IsSuccessStatusCode)

{

var result = await response.Content.ReadFromJsonAsync<ApiResult<List<ChatDTO>>>();

if (result?.IsSuccess == true)

{

Application.Current.Dispatcher.Invoke(() =>

{

Chats.Clear();

foreach (var chat in result.Data)

Chats.Add(chat);

});

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

}

private async void LoadAvailableUsers()

{

try

{

using var httpClient = new HttpClient();

var userResponse = await httpClient.GetAsync("https://localhost:7274/api/users/id");

if (userResponse.IsSuccessStatusCode)

{

var userResult = await userResponse.Content.ReadFromJsonAsync<ApiResult<List<Guid>>>();

if (userResult?.IsSuccess == true)

{

var request = new

{

UsersId = userResult.Data.Where(id => id != \_currentUser.Id).ToList(),

CurrentUserId = \_currentUser.Id

};

var chatResponse = await httpClient.PostAsJsonAsync(

"https://localhost:7274/api/chats/userchats-private",

request);

if (chatResponse.IsSuccessStatusCode)

{

var chatResult = await chatResponse.Content.ReadFromJsonAsync<ApiResult<List<ChatDTO>>>();

if (chatResult.IsSuccess)

{

Users.Clear();

foreach (var userChats in chatResult.Data)

Users.Add(userChats);

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

}

[RelayCommand]

private async Task SelectUser()

{

if (SelectedUser == null) return;

\_selectedChatDB = await CreateOrGetPrivateChat(SelectedUser, \_currentUser.Id);

SelectedChat = null;

CurrentView = ViewType.Messages;

await MessagesVM.SetCurrentChat(\_selectedChatDB);

}

[RelayCommand]

private async Task SelectChat()

{

if (SelectedChat == null) return;

if (SelectedChat.IsGroup)

{

SelectedUser = null;

}

\_selectedChatDB = ConvertToDB(SelectedChat);

CurrentView = ViewType.Messages;

await MessagesVM.SetCurrentChat(\_selectedChatDB);

}

ПРИЛОЖЕНИЕ Б – Флэш-накопитель с программой