МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт цифровых технологий
(наименование института полностью)

Департамент бакалавриата

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка программного обеспечения для оптимального распределения ресурсов в задачах управления проектами»

Обучающийся Ю.Р. Асадуллина
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, О.В. Аникина
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: Разработка программного обеспечения для оптимального распределения ресурсов в задачах управления проектами.

Выпускная квалификационная работа посвящена разработке программного обеспечения для оптимального распределения ресурсов в задачах управления проектами. В результате создан программный продукт, который прошел все этапы тестирования и готов к работе.

В процессе выполнения проекта, был сделан анализ предприятия и предметной области, также проведен сравнительный анализ аналогов, и составлены требования к системе.

Спроектирована и создана клиент -серверная архитектура: клиентская часть разработана на платформе .NET с использованием технологии WPF, серверная - с применением СУБД MySQL Workbench. Проведено тестирование, по результатам которого, подтверждена корректная работа готового продукта.

Пояснительная записка состоит из введения, трех глав, заключения и списка источников. Общий объем работы составляет 51 страниц, среди которых 5 таблицы и 40 рисунков. Список литературы состоит из 19 источников.

Оглавление

Введение

В условиях современного мира с высокой конкуренцией и быстрым темпом жизни эффективность управления ресурсами в проекте становится ключевым фактором успеха любой организации. Проекты становятся все более сложными, многозадачными и часто имеют многие ограничения, например ограничения по времени и бюджету. Одним из наиболее важных аспектов является умение распределять, отслеживать и управлять разными ресурсами: человеческими, техническими, финансовыми и временными.

Для выпускной квалификационной работы была выбрана тема, которая позволит изучить методы оптимального распределения различных ресурсов, а именно: разработка программного обеспечения для оптимального распределения ресурсов в задачах управления проектами.

Актуальность этой работы обусловлена тем, что в современном мире эффективное планирование стало неотъемлемой частью конкурентоспособности бизнеса. С ростом сложности проектов и количества задействованных ресурсов возросла и потребность в точных инструментах для их распределения. Неоптимальное распределение ресурсов может привести к серьезным проблемам, например срыв сроков, превышение бюджета, снижение качества оказываемых услуг или готовой продукции.

Разработка готового продукта, позволит минимизировать эти риски, а также поможет изучить особенности планирования в разнообразных ситуациях, и подготовить программу специально под нужды предприятия. Так как для прохождения преддипломной практики использовалось офтальмологическое медицинское учреждение ГБУЗ СОКОБ им. Т.И. Ерошевского, то в качестве проекта, можно рассмотреть процесс проведения операций, которые регулярно проводятся в этой организации. Но для начала нужно обозначить задачи, цели, объект и предмет исследования:

Объектом исследования является процесс управления ресурсами в проекте.

Целью работы является разработка программного обеспечения для оптимального распределения ресурсов в задачах управления проектами.

Также можно выделить следующие задачи:

- провести анализ предметной области и предприятия;
- проанализировать бизнес процессы, и выявить их слабые стороны;
- провести анализ существующих решений для управления проектами;
- определить функциональные и не функциональные требования к программному продукту;
- спроектировать архитектуру программного обеспечения;
- разработать интерфейс программного продукта;
- разработать программный продукт, который соответствует поставленной теме выпускной квалификационной работы;
- провести тестирование и отладку готового продукта.

В первой главе представлено описание предметной области и предприятия.

Во второй главе представлен процесс подбора необходимых технологий и средств для реализации программы, а также процесс проектирования системы.

В третьей главе описан процесс создания программы, а именно этапы разработки и процесс тестирования.

Глава 1 Анализ предметной области и объекта исследования

1.1 Описание объекта исследования

Основной сферой деятельности организации ГБУЗ СОКОБ им.Т.И. Ерошевского является оказание офтальмологической помощи [5]. На предприятии имеются самые современные технологии для лечения разнообразных офтальмологических болезней, и также, постоянно обновляются технологии, и аппаратура.

Целью данной организации является своевременное оказание офтальмологической помощи, например диагностику, проведение каких-либо хирургических услуг.

К причинам существования этого предприятия, можно отнести тот факт, что с появлением и распространением разнообразных технологий, с каждым годом у большего количества людей замечены разнообразные проблемы со зрением, соответственно имеется спрос на оказание качественных услуг (в данной организации присутствует круглосуточный отдел для оказания экстренных бесплатных услуг, так что делает их очень доступными для людей) кроме этого имеются научные причины, ведь на базе организации регулярно проводятся конференции для изучения новых техник и приемов в офтальмологии [3].

Исходя из выше сказанного можно выделить такие задачи как:

- проведение платных, бесплатных и экстренных услуг;
- консультирование по разнообразным вопросам;
- ведение медицинской документации;
- управление ресурсами;
- подготовка и обслуживание аппаратуры;
- послеоперационные мероприятия;
- внедрение новых технологий и методик;

– проведение обучающих мероприятий для сотрудников.

Упрощенная организационная структура показана на рисунке 1:

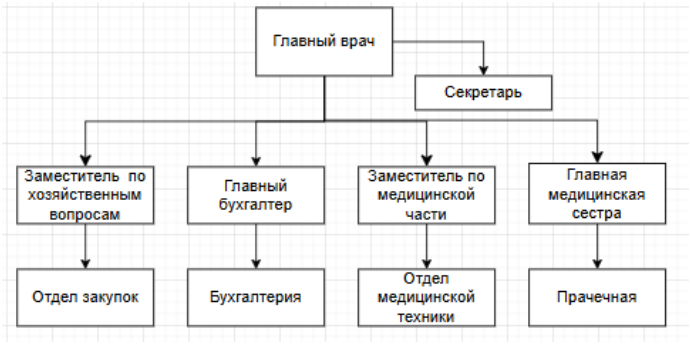


Рисунок 1 – Организационная структура

Для создания программы были выбраны процессы, которые происходят в операционном отделе, целью этого отделения является подготовка и успешное проведение разнообразных операций

Структура подразделения представлена на рисунке 2:

Рисунок 2 – Структура подразделения

За каждым сотрудником закреплены свои должностные обязанности и ответственность, например:

- главный врач отвечает за организацию работы всей больницы, для этого регулярно проводит собрания с заведующими отделениями. Также он отвечает за качество оказываемых услуг и соблюдение всех нормативных актов;
- заведующий отделения отвечает за организацию работы всего отделения, также он распределяет все обязанности и задачи между сотрудниками и отчитывается перед главным врачом;
- старший хирург отвечает за проведение сложных операций, за подготовку новичков и руководит хирургической бригадой;
- анестезиолог отвечает за выбор анестезии для пациента, а также за обеспечение анестезии во время операции, кроме этого, он контролирует состояние пациента до и после анестезии;
- хирург отвечает за проведение операций, а также за пред и после операционное сопровождение пациента. Также в его обязанности входит соблюдения протокола лечения;
- старшая медсестра отвечает за организацию работы среднего и младшего медицинского персонала, следит за соблюдением санитарных норм и ведет учёт медикаментов и материалов;
- медсестра выполняет поручения врачей, ответственна за уход за пациентом;
- санитары отвечают за поддержание чистоты в отделении, транспортировку пациентов и помощь медсёстрам в уходе.

Для начала, нужно понять, как происходит распределение ресурсов данном отделении.

Для этого была создана диаграмма AS-IS [16], с ней можно ознакомиться на рисунке 3.

Рисунок 3 – Диаграмма AS-IS

Далее представлена диаграмма потоков данных [16], в которой показан процесс обмена информацией между отделами.

Это нужно для более наглядного изучения выбранного процесса, и пониманием что стоит учитывать при создании документации и дальнейшей разработки. Диаграмма потоков данных представлена на рисунке 4.

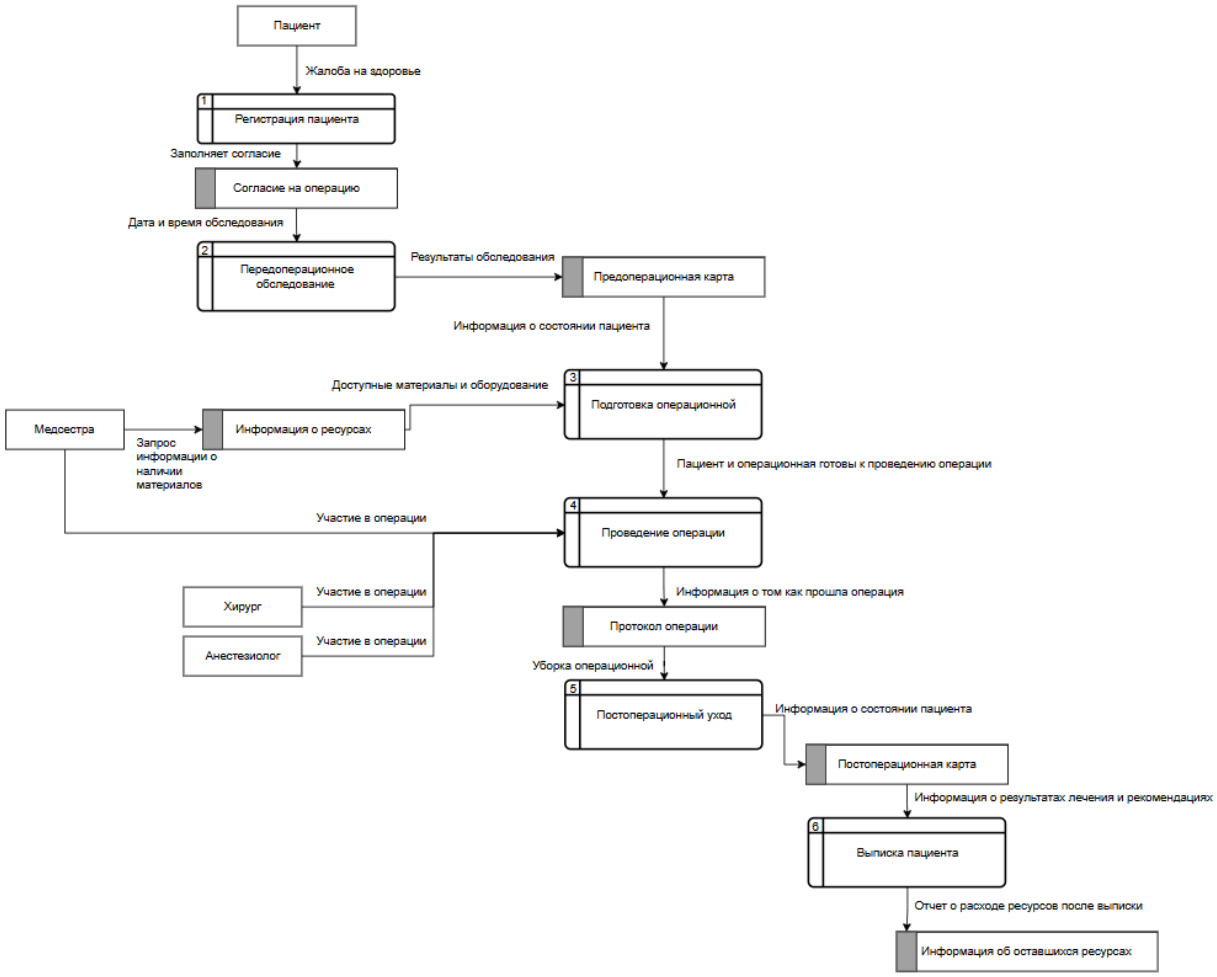


Рисунок 4 – Диаграмма потоков данных

К самой схеме требуются пояснения, с ними можно ознакомиться ниже:

Пациент приходит с жалобой на самочувствие, после чего его регистрируют, затем он заполняет согласие на операцию, ему назначают время и дату предоперационного обследования. Все результаты, полученные во время обследования, вносятся в карту, после чего начинается подготовка операционной, в данном процессе участвует медсестра, она готовит запрос на наличие нужных ресурсов, после чего, они используются в подготовке (к таким материалам можно отнести инструменты, перчатки). После подготовки начинается операция, в ней участвуют медсестра, хирург и анестезиолог, после этого заполняется протокол операции, в котором указывается как все прошло. Следующим этапом идет уборка операционной и постоперационный

уход за пациентом, вся информация о уходе заполняется в постоперационную карту. После этого пациенту дают рекомендации для лечения, и более подробно говорят о результатах операции, после чего, его выписывают. Последним этапом является проверка оставшихся ресурсов и создание отчета о их расходе [7].

При наблюдении было выявлено, что процесс распределения и подготовки ресурсов к операции занимает огромное количество времени, также присутствует возможность ошибки, весь процесс требует огромной усидчивости. Для упрощения данного процесса, можно сделать программный продукт с удаленной базой данных, благодаря которому все будет происходить гораздо быстрее, кроме этого, благодаря базе, можно будет всегда проверить наличие нужного оборудования, или кабинетов.

В результате наблюдения было принято решение создать программное обеспечение для оптимального распределения ресурсов в задачах управления проектами (операциями), готовая программа должна содержать интерфейс и удаленную базу данных.

С учетом этой информации ниже представлена схема TO-BE [16], рисунок 5

Рисунок 5 – Диаграмма TO-BE

В случае успешной автоматизации данного процесса, повыситься производительность сотрудников, ведь им останется только периодически обновлять информацию, которая хранится в базе данных. Кроме этого наличие базы может помочь следить за медицинскими материалами, расходниками и оборудованием, ведь в случае какой либо поломки, или нехватки чего либо, можно сразу позвонить в отдел закупок и договориться с ними, врачи в свою очередь тоже почувствуют облегчение, ведь их расписание станет более наглядным и будет гораздо проще отслеживать какие процедуры прошел пациент. Однако стоит отметить что всегда должна присутствовать возможность ручного редактирования данных, ведь всегда возможны незапланированные изменения (внеплановая смена кабинета, или привлечение дополнительных сотрудников).

1.2 Обзор и анализ аналогов и существующих практик

Для подтверждения или опровержения данной теории стоит проверить какие практики существуют в подобной сфере, и изучить функционал похожих программных решений.

- Surginet – программа для распределения кабинетов и ресурсов для проведения операций, среди плюсов можно отметить богатый функционал, а также наглядное наличие тех или иных ресурсов. Однако у данной программы присутствуют и минусы, например высокая стоимость и огромное потребление мощностей компьютера, также сама программа сложно настраивается и в ней отсутствует русский язык;
- McKesson OR Manager (McKesson Operating Room Management) - программа ориентирована на управление операционными залами и расписанием персонала. Плюсы: наличие готовый списков с материалами, программа учитывает множество факторов, например доступность персонала, оборудования, материалов. К минусам можно

отнести: отсутствие материалов которые требуются в офтальмологии, высокая цена лицензии, отсутствие русского языка, сложный интерфейс.

Результаты сравнительного анализа представлены в виде таблицы 1.

Таблица 1 –Требования к системе

Критерий	Surginet	McKesson OR Manager
Управление материалами	Да	Да
Адаптация под офтальмологию	Нет	Нет
Цена	Высокая стоимость (7 169 р)	Высокая стоимость
Наличие русского языка	Нет	Нет
Сложность интерфейса	Сложный	Сложный
Управление ресурсами (кабинеты, оборудование)	Да	Да
Системные требования	Высокие	Высокие

Таким образом, на основе изучения похожих программ видно, что каждая из них имеет свои проблемы и требует или доработки, или наличие больших денежных ресурсов.

К часто применяемым практикам можно отнести:

- централизованное управление ресурсами - информация о ресурсах, материалах и кабинетах хранится в одном месте, что помогает избегать дублирования;
- автоматическое планирование с учётом ограничений - хоть программа и автоматизирует процесс распределения, для нее должны быть установлены точные ограничения (режим работы сотрудников, точное количество кабинетов);
- вовлечение пользователей - одна из причин по которой перечисленные выше системы не подходят для интеграции, это наличие сложного интерфейса. Интуитивно понятный интерфейс и возможность вручную

корректировать ту или иную информацию ускорит и упростит процесс перехода на нужное ПО.

1.3 Требования к программному продукту

После детального изучения предметной области, и похожих программ, можно описать функциональные требования к программному продукту:

- у программы должен быть интуитивно понятный интерфейс;
- должны быть ограничения для успешного распределения ресурсов;
- наличие удаленной базы данных, для отслеживания ресурсов;
- требуется авторизация, для обеспечения безопасности хранимых данных;
- должна присутствовать возможность ручного редактирования, добавления и удаления данных;
- требуется разграничение пользователей по функционалу, ведь системой будут работать разные пользователи.

Помимо задач при создании программы стоит учитывать бизнес – цели [2], в данном случае к таким целям можно отнести:

- повышение эффективности использования операционных кабинетов и ресурсов;
- сокращение времени на планирование и координацию операций;
- снижение количества конфликтов и ошибок в расписании;
- обеспечение прозрачности и контроля процессов для руководства.

Вывод по главе 1

На основе исследования, представленного в первой главе, были сформулированы цели и задачи проектирования, что является хорошим началом для разработки архитектуры и функциональных модулей системы.

Глава 2 Архитектура проекта и особенности реализации

2.1 Проектирование программного продукта

Перед созданием программы, стоит продумать как все будет работать, для этого будут использоваться UML диаграммы [16]. Это нужно для наглядного понимания работы процесса.

На данном этапе нужно определиться с пользователями и их функционалом. Для данной программы будут выбраны пользователи: врач, админ и менеджер. У каждого из них имеются свои права и ограничения:

- админ является тем, кто полностью управляет системой. Он может добавлять и управлять данными пользователей (в том числе логинами и паролями);
- врач может смотреть назначенное расписание, также он может просматривать списки с оборудованием, кабинетами и сотрудниками;
- менеджер отвечает за составление расписания и распределение ресурсов. Он также может добавлять, редактировать и удалять любые записи кроме данных о пользователей.

Данное разграничение поможет обеспечить безопасность программы и поможет отслеживать действия, которые будут происходить в программе.

Перед построением диаграммы вариантов использования, стоит определиться с функциями.

По функционалу, можно выделить следующие основные функции: авторизация, добавление записей, удаление записей, редактирование записей, автоматическое распределение ресурсов, вывод данных из базы данных.

Для упрощения разработки и определения функционала, стоит подразделить всю программу на модули.

В данном случае будут такие модули как:

- модуль авторизации;
- модуль по работе с ресурсами;

– модуль по созданию расписания;
– модуль по работе с сотрудниками.

К каждому из этих модулей можно добавить уточняющие требования. С ними можно ознакомиться в таблице 2.

Таблица 2 – Требования к системе

Модуль	Функции	Требования	Примечание
Модуль авторизации	1. Добавление, удаление, редактирование пользователей 2. Возможность входа в систему 3. Разграничение доступа по типу пользователя 4. Возможность восстановления пароля	1. Защита данных 2. Шифрование паролей	1. Доступ к редактированию имеется только у администратора 2. Администратор не должен видеть пароли пользователей
Модуль по работе с ресурсами	1. Добавление, удаление, редактирование ресурсов 2. Учет количества 3. Изменение количества в реальном времени	1. Отслеживание остатков в реальном времени	Нужно учитывать категорию ресурсов
Модуль по созданию расписания	1. Добавление, удаление, редактирование расписания 2. Возможность автоматического создания расписания 3. Назначение кабинетов, сотрудников и ресурсов	1. Учет занятости кабинетов и сотрудников 2. Проверка на конфликты в расписании	Возможность ручной корректировки и автоматического планирования
Модуль по работе с сотрудниками	1. Добавление, удаление, редактирование пользователей 2. Назначение ролей и должностей 3. Просмотр занятости и участия в операциях	1. Хранение актуальных данных сотрудник 2. Разграничение доступа к персональным данным	

Теперь, когда были выяснены и обозначены функции в системе, можно составить диаграмму вариантов использования (рисунок 6):

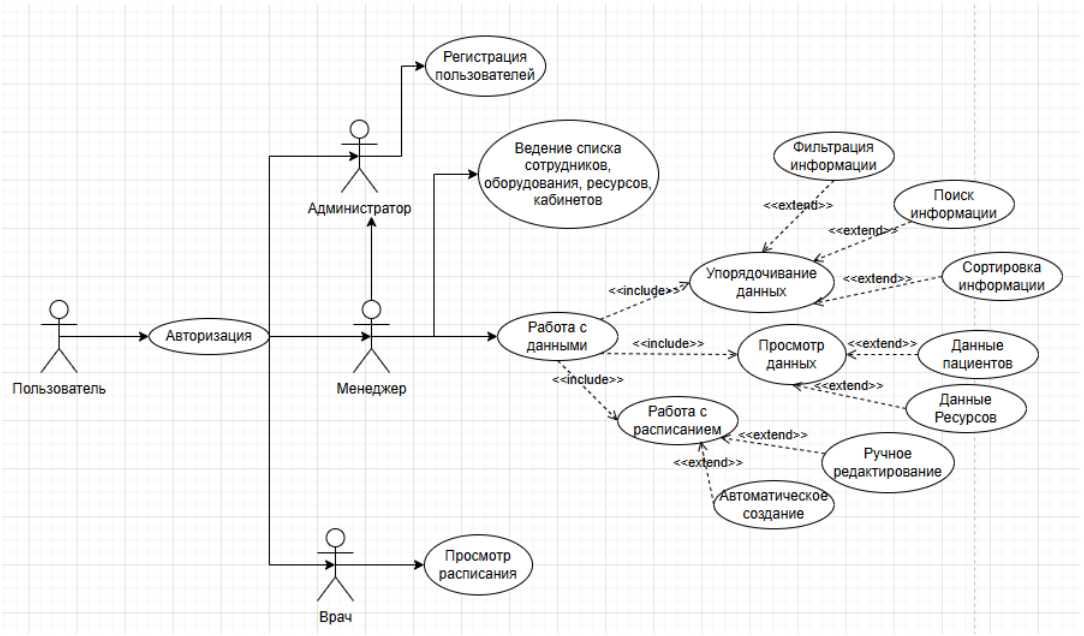


Рисунок 6 – Диаграмма вариантов использования

Далее стоит указать, как объекты системы будут взаимодействовать между собой, для этого наиболее наглядной и подходящей схемой является диаграмма классов (рисунок 7):

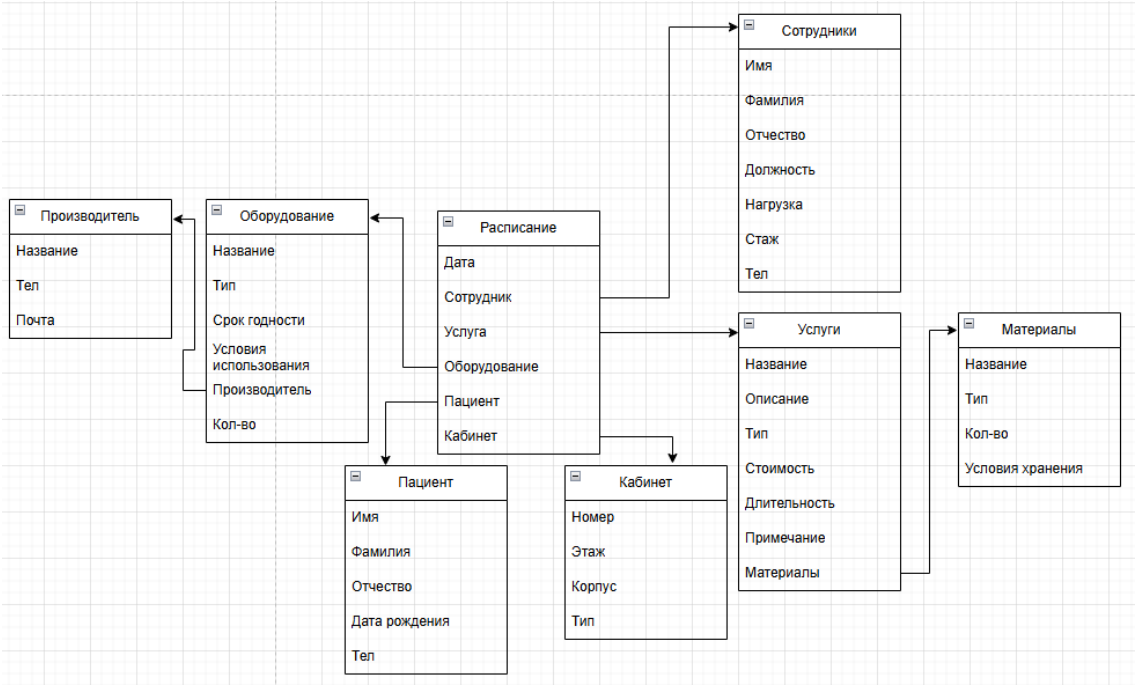


Рисунок 7 – Диаграмма классов

Так как для создания программы будет использоваться база данных, стоит смоделировать логическую модель базы данных (рисунок 8). На данном этапе диаграмма будет сделана в draw.io, а после выбора СУБД уже перенесена в базу. Также на данном этапе стоит определить родительские сущности, на схеме такие сущности выделены голубым цветом.

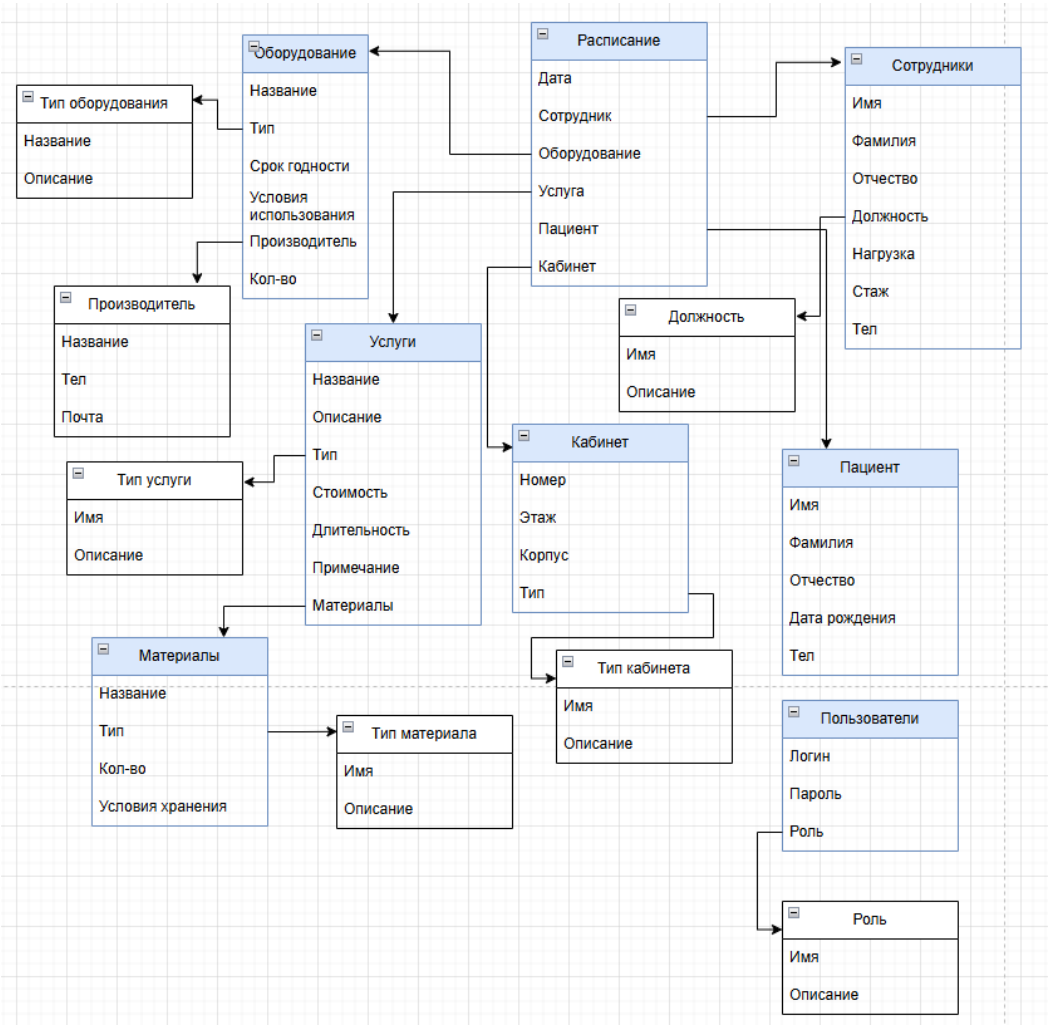


Рисунок 8 – Логическая модель

Для реализации проекта, будет использована клиент - серверная архитектура, она представляет из себя связку из интерфейса программы, приложения (кода и логики) и базы данных.

Примерная структура программы представлена на рисунке 9:

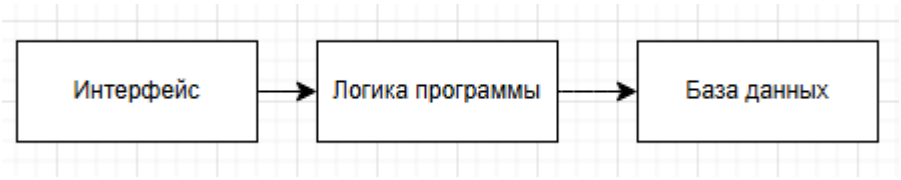


Рисунок 9 – Архитектура работы ПО

Данный вид архитектуры был выбран из-за таких преимуществ как:

- доступность - все сотрудники могут работать с системой с любого компьютера внутри сети больницы или удаленно (в зависимости от настроек сервера);
- масштабируемость - Легко добавлять новые функциональные модули или пользователей;
- централизованное хранение информации - все данные о сотрудниках, ресурсах и расписании хранятся в СУБД, что упрощает резервное копирование, и интеграцию с внешними системами.

2.2 Выбор инструментов и технологий для разработки

Для достижения поставленных целей и создания программного продукта необходимо сформировать набор инструментов разработки. Для создания программы нужно выбрать:

- СУБД (систему управления базами данных) - для создания базы данных;
- инструментальную среду разработки - для создания программы;
- язык программирования.

Выбор будет производиться при помощи сравнительного анализа.

Среди языков программирования выбор производился среди C# [6], Python, Java.

C#. Плюсы [20]:

- имеется кроссплатформенность, что позволяет уменьшить проблемы при переходе на другие операционные системы;
- данный язык является очень популярным, в следствии чего легко можно найти ответы на самые разные вопросы;
- данный язык используют типизацию, из-за чего код является более надежным и в случае возникновения проблем их легче исправить.

Минусы:

- у готовых приложений довольно долгая компиляция, однако стоит отметить что в небольших приложениях этого не заметно;
- для того что бы разобраться в языке может потребоваться много времени, ведь он является довольно сложным.

Python. Плюсы:

- язык является простым для начинающих разработчиков;
- большое количество библиотек и дополнений.

Минусы:

- по сравнению с C#, у язык имеет меньшую производительность;
- все ошибки появляются только во время выполнения, а не компиляции.

Java. Плюсы:

- у языка имеется кроссплатформенность;
- язык является много функциональным, и может быть использован для создания самых разных приложений.

Минусы:

- сложный синтаксис из-за чего требуются постоянные комментарии в коде;
- из-за больших конструкций, приложения являются более требовательными, и занимают больше памяти.

Таким же образом нужно сравнить СУБД, а именно: Workbench, PHP admin, Mongo bd.

Workbench. Плюсы:

- имеется графический интерфейс, что упрощает создание и проектирование базы данных;
- возможность создания пользователей и назначения ролей, что позволяет обезопасить готовую базу;
- имеется кроссплатформенность.

Минусы:

- хоть у СУБД и имеется графический интерфейс, новичкам сложно в нем разобраться;
- некоторые функции нужно изучать самостоятельно, ведь официальной документации недостаточно.

PHP admin. Плюсы:

- имеется удобный веб интерфейс;
- имеются дополнения для расширения функционала.

Минусы:

- обязательное наличие веб сервера для работы;
- слабые функции для проектирования базы.

Mongo bd. Плюсы:

- имеется возможность сохранить данные в виде документа;
- высокая производительность.

Минусы:

- не поддерживает SQL;
- отсутствует кроссплатформенность.

И также среди инструментальных средств сравнивались Visual studio [17], Android studio, XCode

Visual studio. Плюсы:

- имеется поддержка нескольких языков;
- удобный интерфейс;
- имеется много плагинов и дополнений.

Минусы:

– большая требовательность к ресурсам;
– частые обновления, из-за чего некоторые функции могут не работать.

Android studio. Плюсы:

– поддержка Kotlin;
– имеется встроенный эмулятор.

Минусы:

– подходит только для разработки мобильных приложений;
– программа очень требовательная.

XCode - Плюсы:

– имеются встроенные инструменты для тестирования;
– поддержка языка Swift.

Минусы:

– подходит только для разработки мобильных приложений.

Для наглядного сравнения, результаты выбора СУБД, инструментальной среды разработки и языка программирования представлены ниже, в виде таблиц 3-5:

Таблица 3 – Сравнение языков программирования

Название	Порог входа	Основное применение	Производительность
C#	Средний. Строгая типизация	Десктоп - приложения, разработка игр	Высокая
Python	Низкий. Простой и лаконичный синтаксис	Анализ данных, машинное обучение	Средняя или низкая (для "чистого" Python)
Java	Средний. Более многословный синтаксис	Крупные корпоративные системы, Big Data	Высокая

Таблица 4 – Сравнение СУБД

Название	Цена	Тип приложения	Ключевая особенность
Workbench	Бесплатно	Настольное	Мощные инструменты для визуального проектирования реляционных баз данных.
Mongo bd	Бесплатно	Настольное	Лучший инструмент для работы с документами и сложными структурами
PHP admin	Бесплатно	Веб-приложение	Доступность из любой точки, где есть интернет, без установки ПО

Таблица 5 – Сравнение инструментальных средств

Название	Требуемая ОС	Основное назначение	Ключевые языки
Visual studio	Windows	Разработка под Windows	C#, C++, Python, JavaScript
Android studio	Windows, macOS, Linux	Разработка приложений под Android	Kotlin, Java, C++
XCode	macOS	Разработка приложений под ios	Swift

По итогам сравнения, был выбран язык программирования C#, СУБД Workbench, и среда разработки Visual studio. Сам интерфейс будет создан при помощи WPF [19], ведь данные технологии позволяют работать со стилями, что можно использовать для создания красивого интерфейса и его настройки.

Перед созданием интерфейса, нужно сделать его схематический прототип, при этом нужно помнить, что интерфейс должен быть лаконичным, функциональным и иметь приятную цветовую гамму, также стоит уточнить, что расположение элементов и цветовая палитра между прототипом и готовой программой может немного отличаться.

Для примера, будет представлены прототипы 3-х окон

- окно авторизации (рисунок 10);
- меню (рисунок 11);
- окно вывода расписания (рисунок 12).

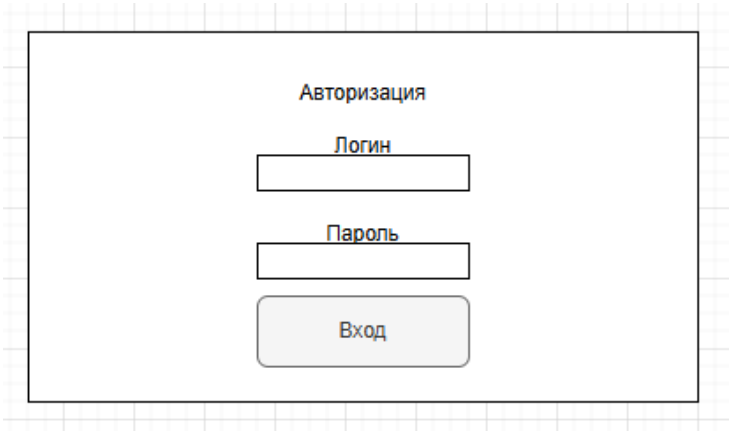


Рисунок 10 – Окно авторизации (прототип)

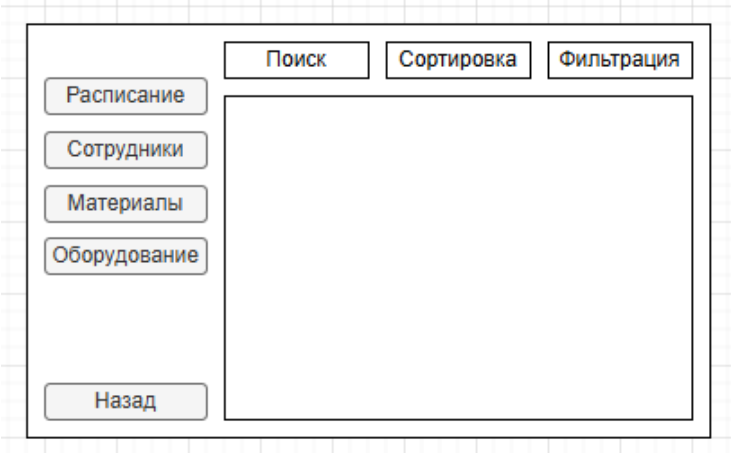


Рисунок 11 – Меню (прототип)

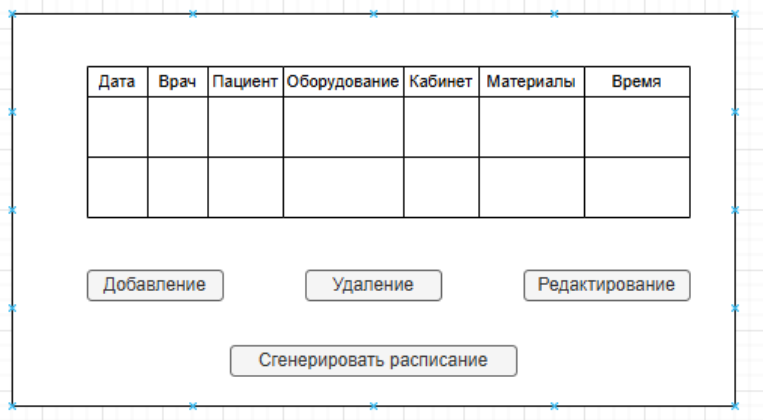


Рисунок 12 – Окно вывода расписания (прототип)

На окне меню, в центре на месте пустой области будет выводиться информация в зависимости от того окна, которое выбрал пользователь. Пример показан на рисунке 13.

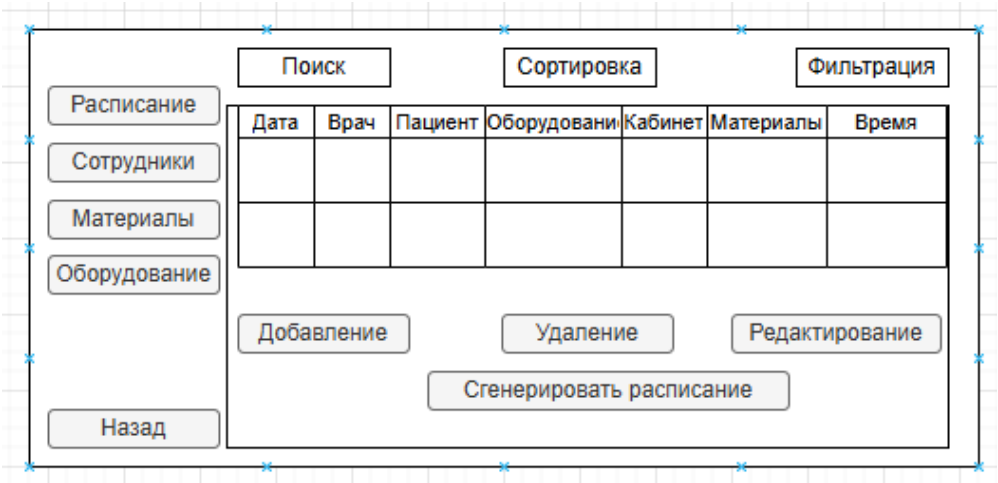


Рисунок 13 – Пример вывода данных из страницы в окно (прототип)

Одной из ключевых особенностей программы является процесс создания расписания и распределения сотрудников и кабинетов, поэтому данный процесс стоит продумать получше.

При распределении, в программе должны присутствовать ограничения, ниже перечислены некоторые их них:

- в одном кабинете, в одно время может быть только один пациент;
- один пациент не может быть записан на 2 услуги одновременно;
- один врач не может производить параллельно 2 процедуры;
- нагрузка врача не может быть превышена;
- врач не может оказывать услуги в свое не рабочее время;
- если какие-то ресурсы для проведения операции закончились, она не может быть проведена.

Исходя из этого можно составить примерный алгоритм подбора ресурсов. Также стоит отметить, что в процессе разработки, алгоритм распределения был переделан несколько раз, объяснение алгоритма и

причины его отмены представлены ниже:

Версия 1

Вводные данные: менеджером указывается пациент, услуга и кабинет остальное заполняется программой на основе данных.

Алгоритм работы: Программа самостоятельно указывает дату, время и врача, однако дата может быть не удобна для пациента.

Итог: требуется доработка

Версия 2

Вводные данные: менеджером указывается пациент, услуга, кабинет и дата, остальное заполняется программой на основе данных.

Алгоритм работы: Программа самостоятельно указывает время и врача, однако менеджер не знает какой кабинет занят, а какой нет, из-за чего увеличивается время создания записи.

Итог: требуется доработка

Версия 3

Вводные данные: менеджером указывается пациент, услуга и дата остальное заполняется программой на основе данных.

Алгоритм работы: Программа самостоятельно указывает время, врача и кабинет, в результате чего получается готовая запись, со всеми нужными данными.

Итог: алгоритм работает, однако стоит добавить возможность отмены записи.

По итогам рассуждений, в программе будет реализована 3 версия.

Для лучшего понимания, была создана блок - схема данного процесса (рисунок 14)

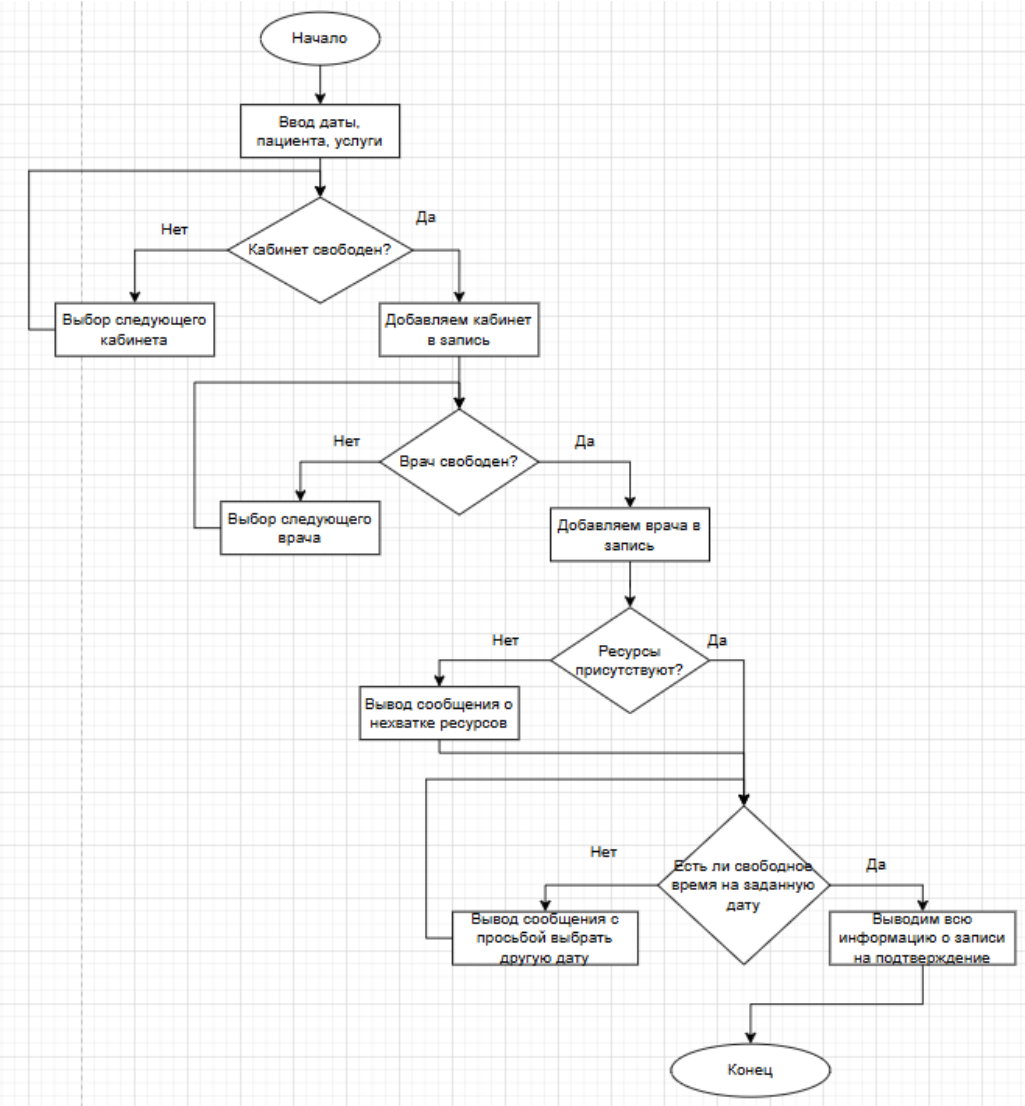


Рисунок 14 – Блок схема алгоритма распределения ресурсов

Выводы по главе 2

Во второй главе созданы необходимые модели, выбраны технологии для реализации программы и продуман алгоритм для реализации программы.

Глава 3 Реализация проекта

3.1 Создание базы данных

После проектирования и выбора технологий, можно приступать к созданию программы.

Для начала нужно спроектировать базу данных, при этом стоит учитывать несколько требований:

- база должна быть в 3 нормальной форме [12];
- должно отсутствовать дублирование информации;
- нельзя допустить появления круговых связей;
- нельзя называть поля и таблицы на русском языке;
- по возможности, нужно оставлять только связи один ко многим.

С учетом этих ограничений, была создана база в MySQL Workbench [1].

Структура базы (со списком всех таблиц представлена на рисунке 15).

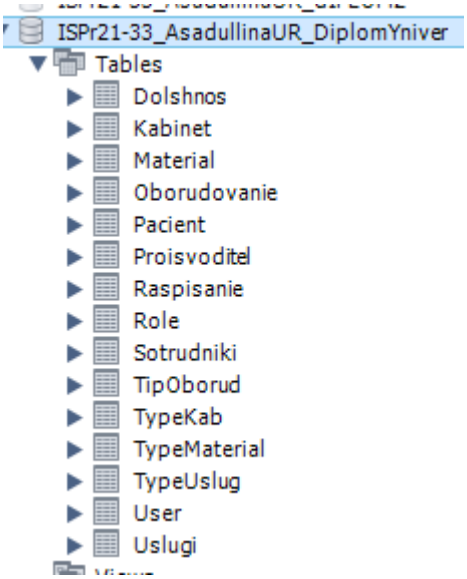


Рисунок 15 – Структура базы данных

На рисунке 16 представлена er-диаграмма [15], где можно ознакомиться

с ограничениями, типами данных и связями.

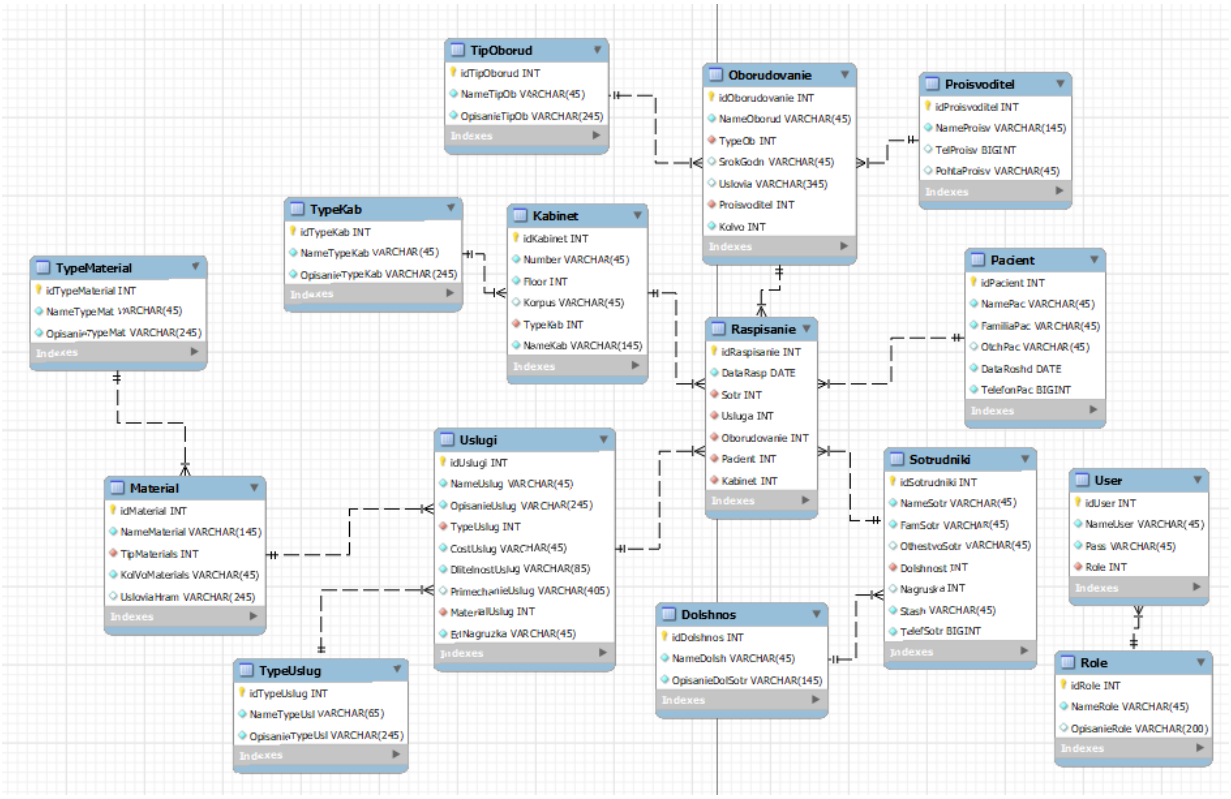


Рисунок 16 – Er-диаграмма

Для каждой таблицы были выставлены свои ограничения, разберем их на примере таблицы пользователей, которая показана на рисунке 17.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	De
idUser	INT	☑	☑	☐	☐	☐	☐	☑	☐	
NameUser	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
Pass	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
Role	INT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 17 – Таблица пользователи с ограничениями

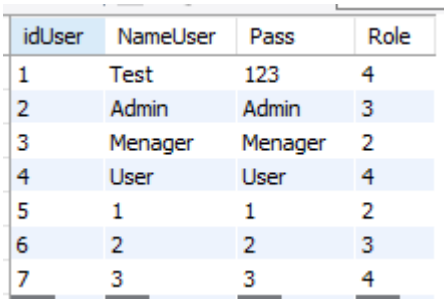
В каждой таблице есть Id, уникальный номер каждой записи, это же поле является первичным ключом, который нужен для создания связей.

В некоторых полях стоят ограничении NN или же не нулевое поле, если в данном ограничении стоит галочка, поле не может быть пустым.

В поле с первичным ключом стоит ограничение AI- это ограничение обозначает, что значение в этом поле должно заполняться автоматически (например, ограничение стоит для поля id, это поле будет автоматически заполняться при добавлении записи, и при этом это поле не может дублироваться).

Для каждого поля стоит выбрать тип данных. В данной таблице у нас 2 типа: int - в поле может храниться только числовое значение, и varchar (45) - в поле может храниться текстовые значения (буквы, числа, знаки).

После выставления ограничений, можно заполнять таблицы, для примера на рисунке 18 показана таблица пользователи.



idUser	NameUser	Pass	Role
1	Test	123	4
2	Admin	Admin	3
3	Menager	Menager	2
4	User	User	4
5	1	1	2
6	2	2	3
7	3	3	4

Рисунок 18 – Заполненная таблица пользователи

Остальные таблицы заполнены по такому же принципу.

3.2 Разработка программы

После проектировки базы, можно приступать к программе. Для начала нужно соединить базу и программу, для этого сначала нужно подключить библиотеки. Со списком библиотек [14] можно ознакомиться на рисунке 19.

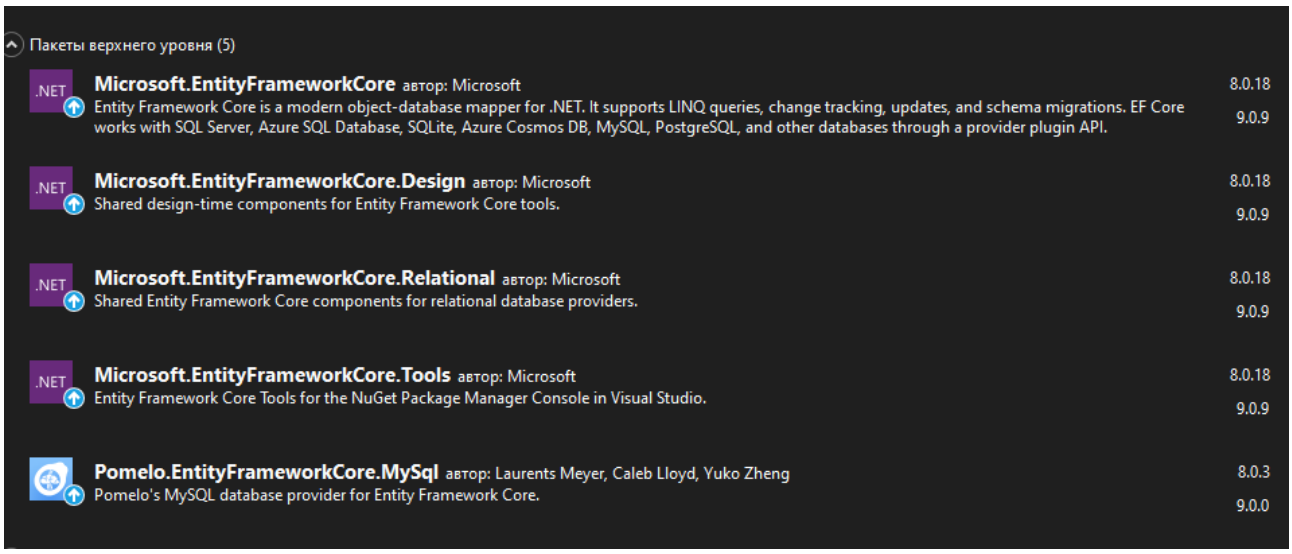


Рисунок 19 – Подключенные библиотеки

Пояснения к библиотекам:

- Microsoft.EntityFrameworkCore - пакет предоставляет основные классы и интерфейсы для работы с базой данных;
- Microsoft.EntityFrameworkCore.Design - пакет содержит вспомогательные компоненты, необходимые для работы с инструментами разработки, такими как миграции базы данных, генерация кода, обновление схемы;
- Microsoft.EntityFrameworkCore.Relational - Включает компоненты, необходимые для взаимодействия с реляционными базами данных (такими как MSSQL, PostgreSQL, MySQL). Реализует логику построения SQL-запросов, транзакций, ограничений, индексов и других особенностей реляционных СУБД;
- Microsoft.EntityFrameworkCore.Tools - Содержит команды для работы из командной строки и из Visual Studio (например, Add-Migration, Update-Database), которые позволяют управлять миграциями данных, scaffold-кодом и прочими функциями прямо из IDE или CLI;

– Pomelo.EntityFrameworkCore.MySql - Это сторонний пакет-провайдер, который реализует поддержку работы Entity Framework Core с базами данных MySQL,

Если все установлено верно, в боковом поле появится вкладка с библиотеками. Данная вкладка представлена на рисунке 20:

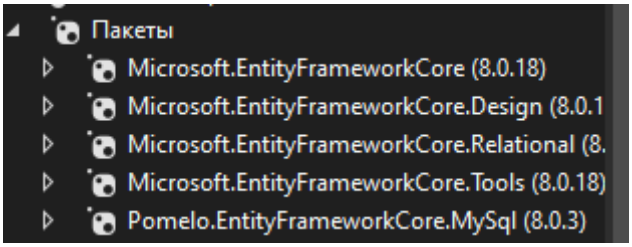


Рисунок 20 – Подключенные библиотеки в структуре проекта

После подключения библиотек, можно приступить к интегрированию базы. Для этого прописываем данный код:

```
FrameStart.Navigate(new Autorisation_Page());

MySqlConnectionStringBuilder bild = new MySqlConnectionStringBuilder
{
   Server = "cfif31.ru",
   Port = 3306,
   UserID = "ISPr21-33_AsadullinaUR",
   Password = "ISPr21-33_AsadullinaUR",
   Database = "ISPr21-33_AsadullinaUR_DiplomYniver",
   CharacterSet = "utf8"
};
        Trace.WriteLine(bild.ConnectionString);
```

В данном коде нужно указать все данные для подключения, сервер, имя пользователя, пароль, название базы и кодировку данных. После этого в поле вывода сообщений системы получим данную строку:

Server=cfif31.ru;Port=3306;UserID=ISPr21-33_AsadullinaUR;Password=ISPr21-33_AsadullinaUR;Database=ISPr21-33_AsadullinaUR_DiplomYniver;Character Set=utf8

Следующим шагом, открываем консоль диспетчера пакетов и прописываем код импорта данных из базы данных.

PM> scaffold - dbContext "Server=cfif31.ru;Port=3306;User ID=ISPr21-33_AsadullinaUR;Password=ISPr21-33_AsadullinaUR;Database=ISPr21-33_AsadullinaUR_DiplomYniver;Character Set=utf8" Pomelo.EntityFrameworkCore.MySql

После этого, если все сделано верно, должны появиться классы, которые дублируют таблицы из базы данных. Итог подключения показан на рисунке 21.

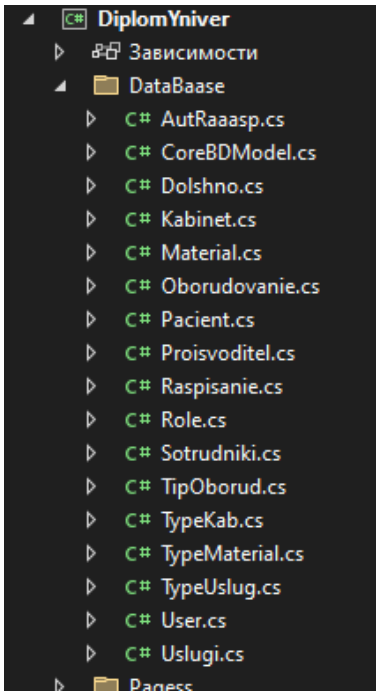


Рисунок 21 – Интеграция таблиц в программу

Каждый класс представляет собой таблицу из базы данных, при этом один из классов (CoreBDModel) [8] отвечает за построение связей. Для упрощения работы стоит написать в нем код для проверки подключения, данный код нужен на случай проблем с подключением. Код представлен на рисунке 22.

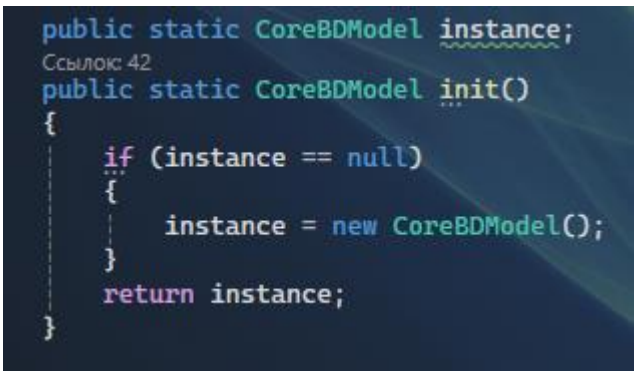


Рисунок 22 – Код проверки подключения

После этого можно приступать к созданию интерфейса. Сначала было сделано окно авторизации. Для интерфейса используется WPF [9], соответственно он тоже представляет из себя код.

Код окна авторизации:

```
<Grid Background="#FFD6E8F7">
    <Grid.RowDefinitions>
        <RowDefinition Height="50"/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <TextBlock Text="Авторизация" HorizontalAlignment="Center" VerticalAlignment="Bottom"  FontSize="25"/>
    <StackPanel Grid.Row="1" HorizontalAlignment="Center">
        <TextBlock Text="Логин" FontSize="15"/>
        <TextBox Name="TBLogin" Width="250" FontSize="15"/>
        <TextBlock Text="Пароль"  FontSize="15"/>
        <TextBox Name="TBParol"  Width="250" FontSize="15"/>
        <Button Content="Вход" FontSize="20" Margin="0,20,0,0" Background="#FFADCCFF" Click="Bt_Vxod_Click"/>
        <Button Content="Регистрация" FontSize="20" Margin="0,20,0,0" Background="#FFADCCFF" Click="Bt_Registr_Click"/>
    </StackPanel>
```

</Grid>

Готовый интерфейс представлен на рисунке 23.

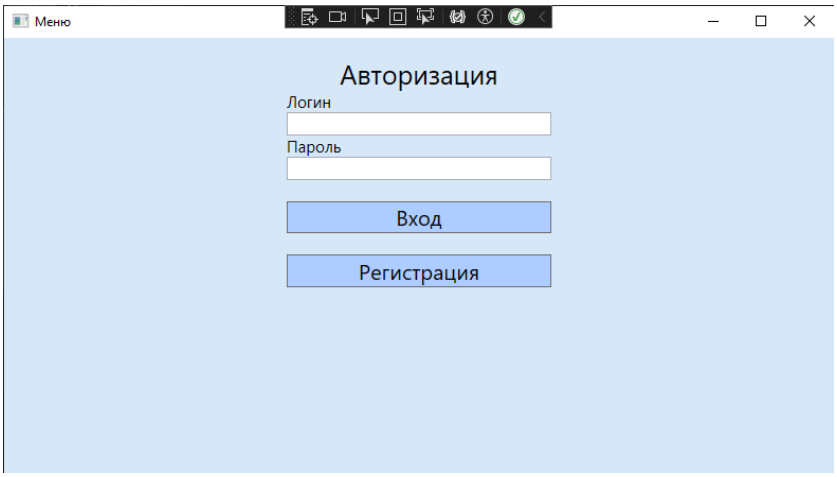


Рисунок 23 – Интерфейс окна авторизации

Далее приступаем к написанию функции авторизации. Для этого используется связь с таблицей пользователи. Введенные в поля логин и пароль должны сравниваться с записями в базе и в случае нахождения совпадения авторизация должна проходить успешно. Код авторизации:

```
User polzovatel = CoreBDModel.init().Users.FirstOrDefault(
        p => p.NameUser == TBLogin.Text &&
        p.Pass == TBParol.Text);
if (polzovatel != null)
{
    if (polzovatel.Role == 1)
    {
        Admin_Start_Window ad = new Admin_Start_Window();
        ad.Show();
        Window mainWindow = Window.GetWindow(this);
        mainWindow.Close();
    }
    else if (polzovatel.Role == 2)
```

```
{
    Meneger_Start_Window me = new Meneger_Start_Window();
    me.Show();
    Window mainWindow = Window.GetWindow(this);
    mainWindow.Close();
}
else
{
    Sotrudnik_Start_Window so = new Sotrudnik_Start_Window();
    so.Show();
    Window mainWindow = Window.GetWindow(this);
    mainWindow.Close();
        }
```

Не стоит забывать и про обратную связь с пользователем, ведь если будет вылезать ошибка с кодом, он не поймет что именно произошло, для этого нужно выводить сообщения, например если поля будут пустые то пользователь увидит сообщение, представленное на рисунке 24.

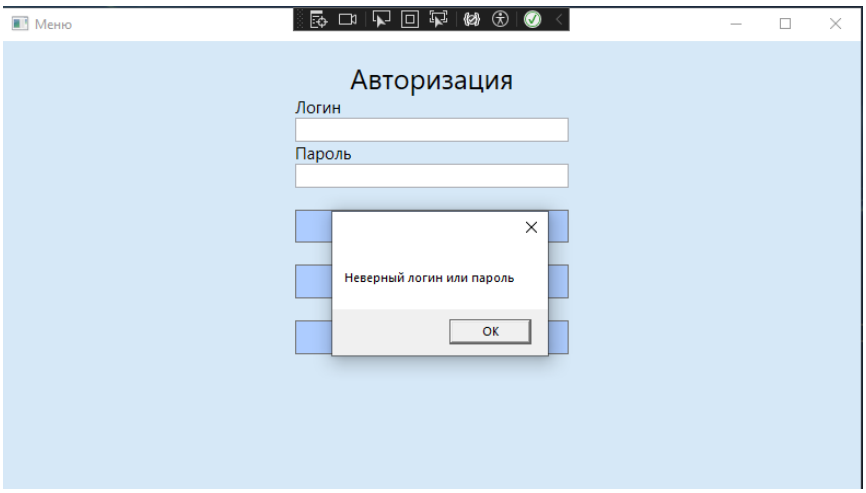


Рисунок 24 – Сообщение об ошибке

Код для вывода сообщения представлен на рисунке 25.

```
if (TBLogin.Text == null || TBParol.Text == null)
{
    MessageBox.Show("Заполните все поля");
}

else
{
```

Рисунок 25 – Код вывода сообщения

После успешно пройденной авторизации пользователь попадает главное меню, которое отличается в зависимости от его учетной записи. Для примера, на рисунке 26 показано меню администратора.

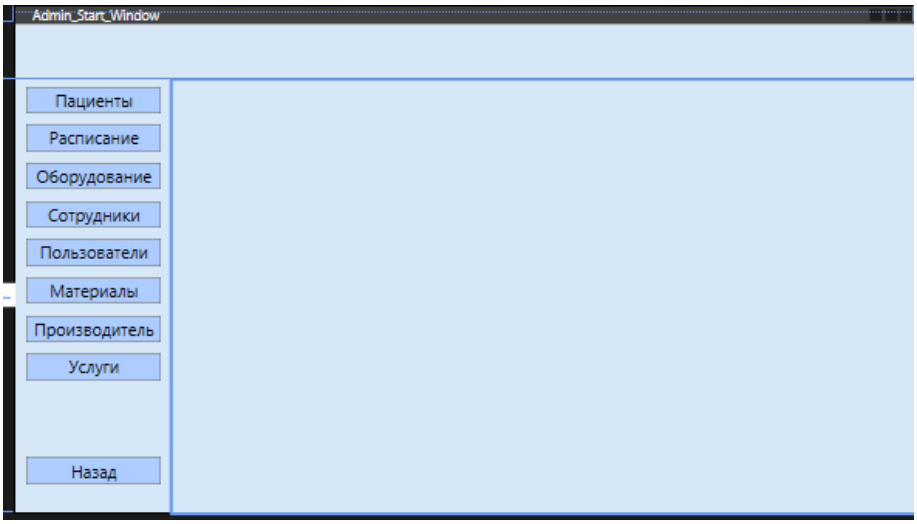


Рисунок 26 – Интерфейс главного меню

При нажатии на нужную кнопку откроется соответствующее окно (страница).

Код интерфейса данного окна представлен ниже:

```
<Grid  Background="#FFD6E8F7">
    <Grid.RowDefinitions>
        <RowDefinition Height="50"/>
        <RowDefinition/>
    </Grid.RowDefinitions>
```

```
    <Grid Grid.Row="1">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="140"/>
        <ColumnDefinition/>
      </Grid.ColumnDefinitions>
      <Frame Name="Admin_Frame" NavigationUIVisibility="Hidden"
Grid.Column="1"/>
      <StackPanel>
        <Button Content="Пациенты" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Pac_Click"/>
        <Button Content="Расписание" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Rasp_Click"/>
        <Button Content="Оборудование" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Oborud_Click"/>
        <Button Content="Сотрудники" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Sotr_Click"/>
        <Button Content="Пользователи" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Pols_Click"/>
        <Button Content="Материалы" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Material_Click"/>
        <Button Content="Производитель" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Proisv_Click"/>
        <Button Content="Услуги" FontSize="15" Margin="10,5"
Background="#FFADCCFF" Click="Bt_Uslugi_Click"/>
      </StackPanel>
      <Button Content="Назад" FontSize="15" Margin="10,25"
Background="#FFADCCFF" Click="Bt_Back_Click"
VerticalAlignment="Bottom"/>
    </Grid>
  </Grid>
```

Код самой страницы представлен далее, здесь стоит только отметить функционал кнопки назад, ведь остальные кнопки просто выводят нужные окна.

При нажатии кнопки проверяется что нужно сделать, вывести предыдущую страницу, или если страниц не осталось то вернуться на окно с авторизацией.

```
private void Bt_Pac_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Pacient_Page());
}

private void Bt_Rasp_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Raspisanie_Page());
}

private void Bt_Sotr_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Sotrudnik_Page());
}

private void Bt_Pols_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new User_Page());
}

private void Bt_Material_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Material_Page());
}

private void Bt_Proisv_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Proisvoditel_Page());
}

private void Bt_Uslugi_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Uslugi_Page());
}
```

```
private void Bt_Back_Click(object sender, RoutedEventArgs e)
{
    if (Admin_Frame.CanGoBack)
    {
        Admin_Frame.GoBack();
    }
    else
    {
        MainWindow mainWindow = new MainWindow();
        mainWindow.Show();
        Window ad = Window.GetWindow(this);
        if (ad != null)
            ad.Close();
    }
}

private void Bt_Oborud_Click(object sender, RoutedEventArgs e)
{
    Admin_Frame.Navigate(new Oborudovanie_Page());

}
```

Также стоит отметить возможность вывода данных из таблицы. В качестве примера, на рисунке 27 показана страница, с выводом информации из таблицы пользователи.

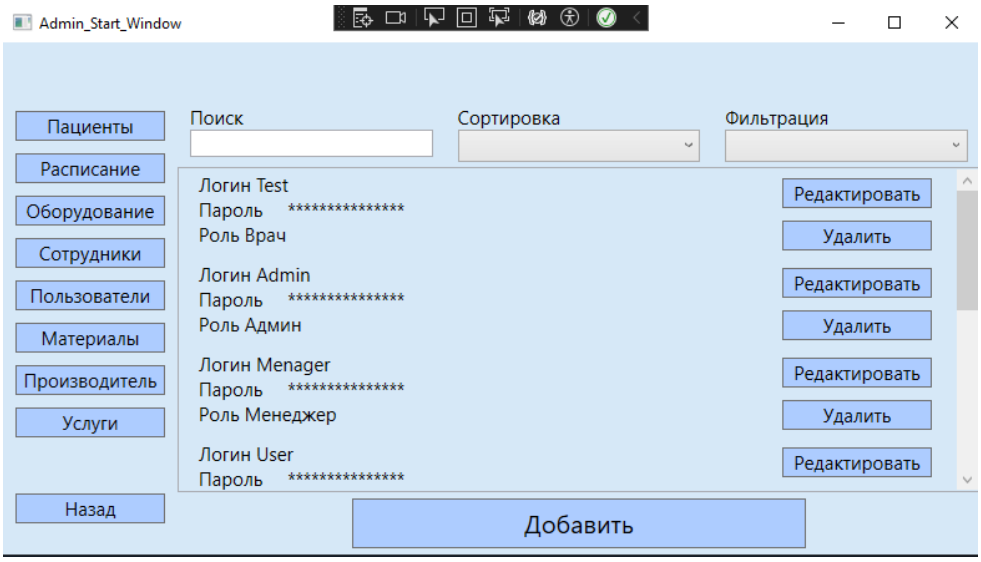


Рисунок 27 – Интерфейс вывода данных из таблицы пользователи

Код вывода данных из базы данных:

```
public void UpdateUser()
{
    List<User> user = CoreBDModel.init().Users.Include(r =>
r.RoleNavigation).Where(s => s.NameUser.Contains(TB_Search.Text)).ToList();
    LvUser.ItemsSource = user;
}
```

У данного окна имеется определённый шаблон, который дублируется для каждой записи в соответствующей таблице (если в таблице 5 записей, то и в программе их будет 5).

3.3 Тестирование приложения

Первое что стоит проверить, это вывод, добавление, редактирование и удаление, для проверки вывода, выберем таблицу пользователей.

Тест 1. Вывод данных.

Задача проверки убедиться, что совпадает количество выведенных записей и их значения с записями в таблице, кроме этого, нужно проверить корректность данных из поля “Role”, они должны выводиться не числом, а словом из таблицы роль.

Данные в таблице представлены на рисунке 28.

idUser	NameUser	Pass	Role
1	Test	123	4
2	Admin	Admin	3
3	Menager	Menager	2
4	User	User	4
5	1	1	2
6	2	2	3
7	3	3	4
NULL	NULL	NULL	NULL

Рисунок 28 – Данные из таблицы пользователи в базе данных

Данные в программе показаны на рисунке 29.

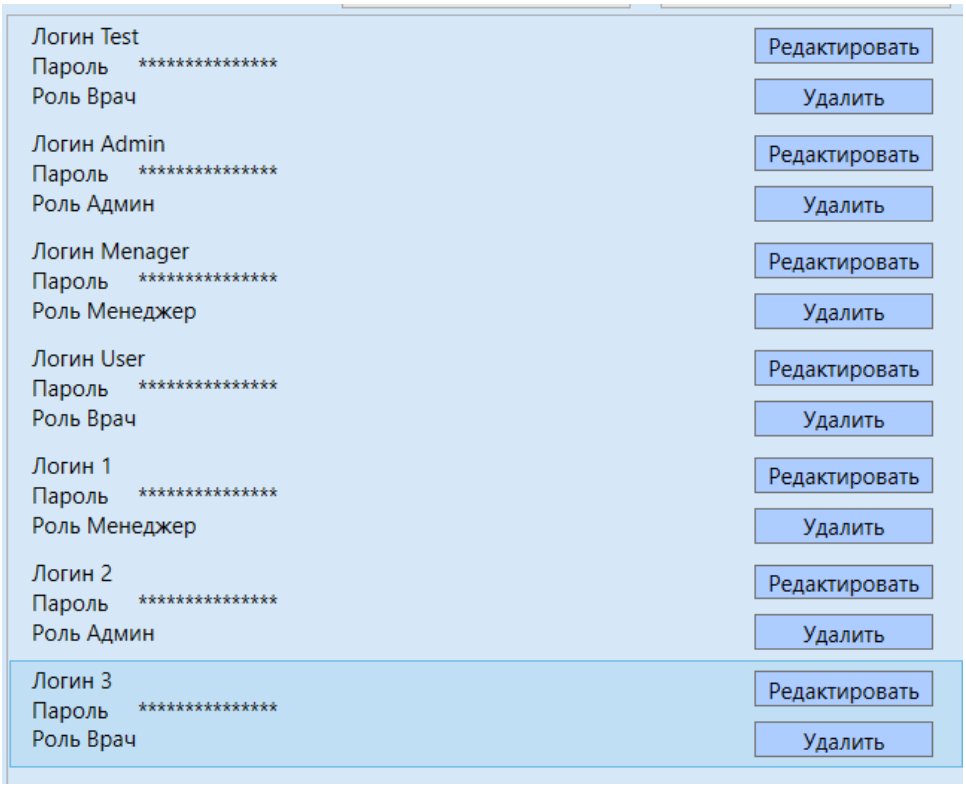


Рисунок 29 – Данные из таблицы пользователи в программе

Итог: тест пройден успешно ведь вывелось 7 записей из 7, при этом все значения совпадают (пароль скрыт для обеспечения безопасности личных данных)

Тест 2. Добавление записей.

Цель проверки, убедиться, что данные из приложения успешно добавляются в базу данных.

Для этого используем таблицу сотрудников, для успешной проверки воспользуемся окном добавления записи, которое представлено на рисунке 30.

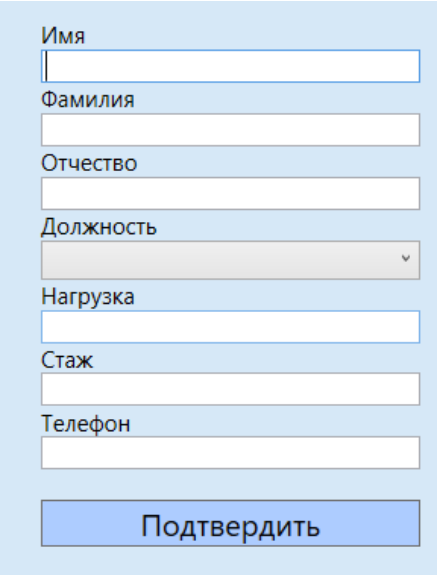


Рисунок 30 – Окно добавления данных

На рисунке 31 показаны поля с заполненной информацией.

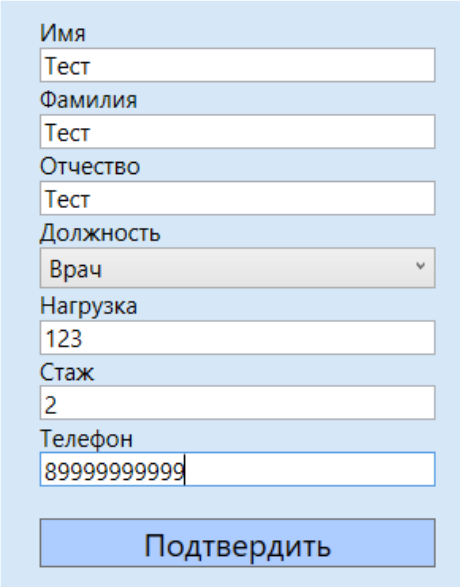
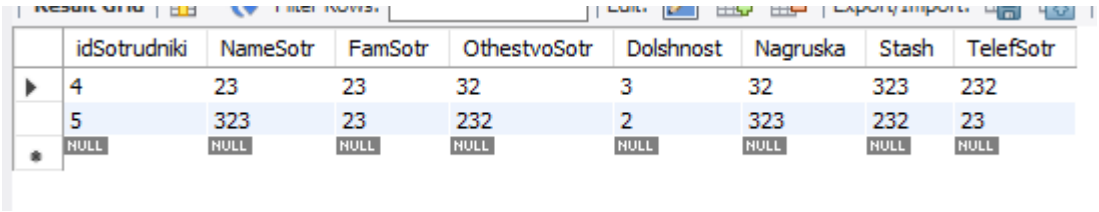


Рисунок 31 – Поля заполнены верными данными

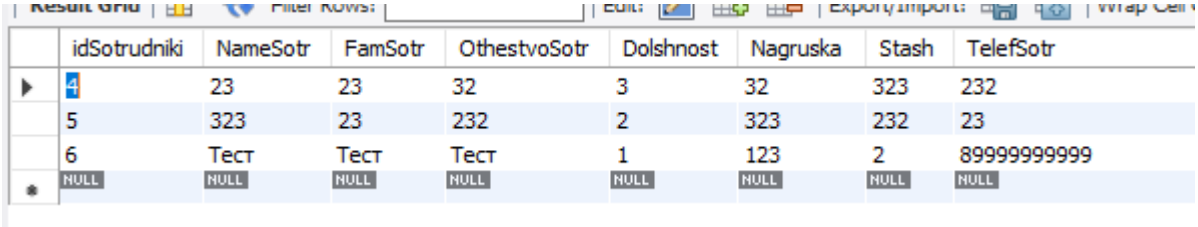
Таблица с сотрудниками до добавления записи представлена на рисунке 32.



idSotrudniki	NameSotr	FamSotr	OthestvoSotr	Dolshnost	Nagruska	Stash	TelefSotr
4	23	23	32	3	32	323	232
5	323	23	232	2	323	232	23
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 32 – Таблица с сотрудниками до добавления записи

На рисунке 33 представлена таблица из базы данных после добавления.



idSotrudniki	NameSotr	FamSotr	OthestvoSotr	Dolshnost	Nagruska	Stash	TelefSotr
4	23	23	32	3	32	323	232
5	323	23	232	2	323	232	23
6	Тест	Тест	Тест	1	123	2	89999999999
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 33 – Таблица с сотрудниками после добавления записи

Итог: Тест успешно пройден, запись добавлена в базу.

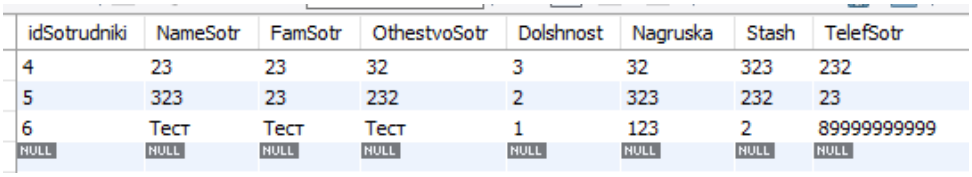
Тест 3. Удаление записей.

Цель проверки, убедиться, что данные из приложения при удалении остаются в базе, если нажать на кнопку нет.

Данные в приложении представлены на рисунке 34.

Рисунок 34 – Данные из таблицы пользователи в программе

Данные в базе показаны на рисунке 35.



idSotrudniki	NameSotr	FamSotr	OthestvoSotr	Dolshnost	Nagruska	Stash	TelefSotr
4	23	23	32	3	32	323	232
5	323	23	232	2	323	232	23
6	Тест	Тест	Тест	1	123	2	89999999999
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 35 – Данные из таблицы пользователи в базе

На рисунке 36 показано сообщение при удалении данных. Для теста нажимаем на кнопку удалить у 1 записи.

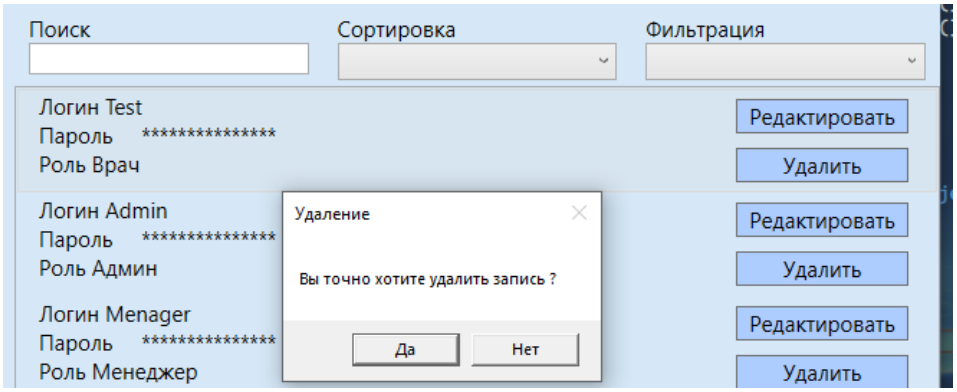


Рисунок 36 – Сообщение об удалении данных

Для успешного удаления нужно нажать нет.

Данные в приложении после этого показаны на рисунке 37.

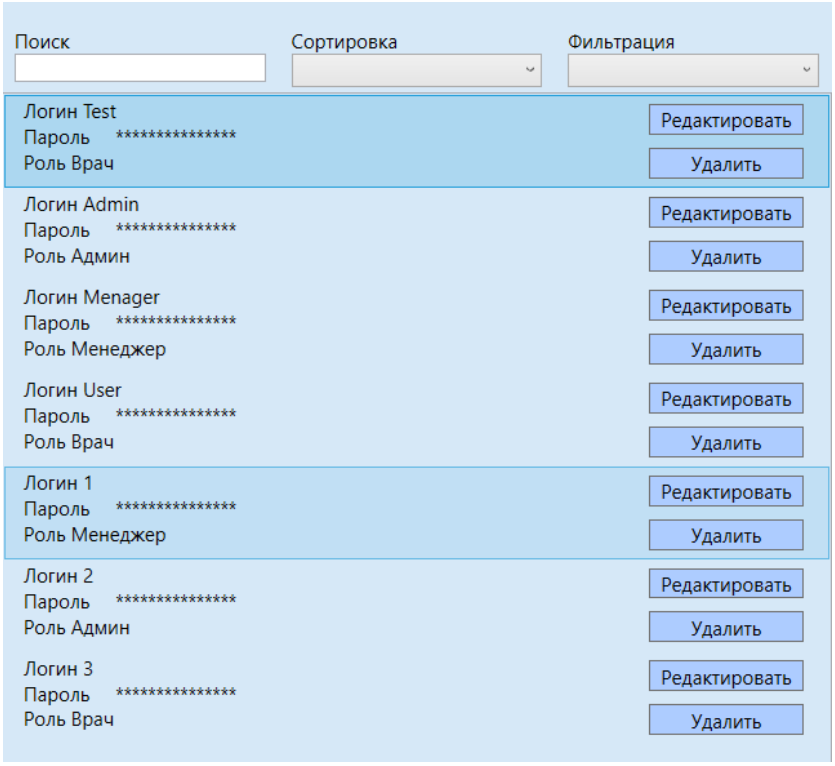


Рисунок 37 – Данные после того как нажали нет

Итог: Тест успешно пройден данные не удалились.

Тест 3.1 Удаление записей.

Цель проверки, убедиться, что данные из приложения при удалении

успешно удаляются из базы данных. Работать будем с той же таблицей пользователи. На рисунке 38 также показано сообщение, которое появляется при нажатии кнопку удалить.

Выбираем 1 запись и в этот раз нажимаем да.

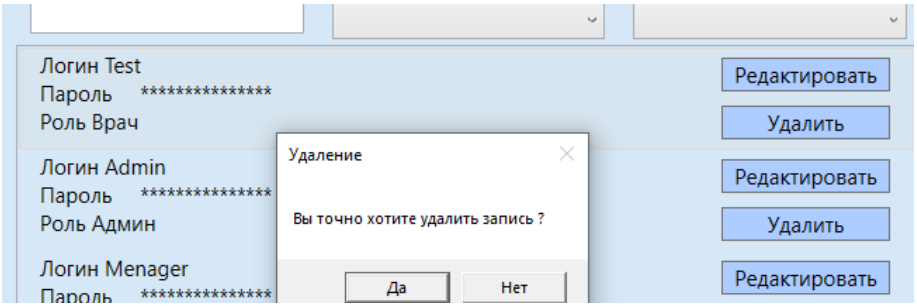


Рисунок 38 – Сообщение об удалении данных

Данные в приложении после того как нажали да, показаны на рисунке 39.

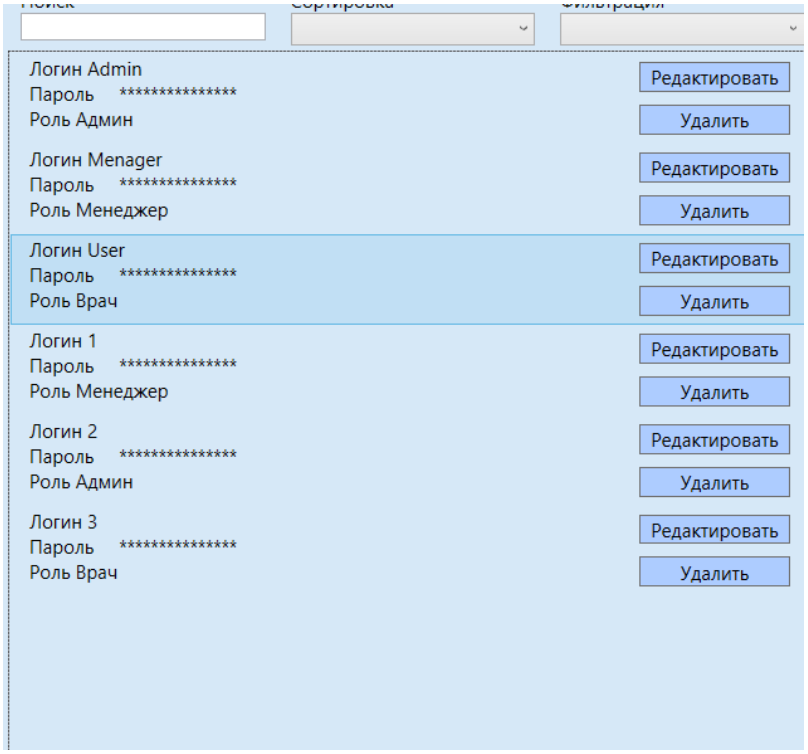


Рисунок 39 – Данные после того, как нажали да

Данные в таблице после того, как нажали да, показаны на рисунке 40.

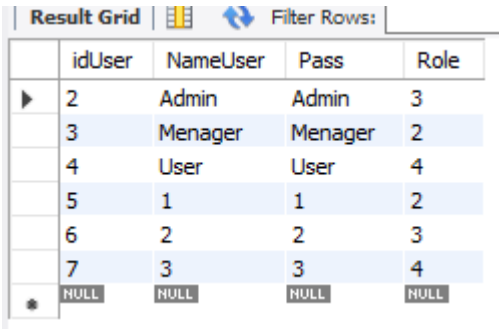


Рисунок 40 – Данные в базе после того, как нажали да

Итог: Тест пройден, данные успешно удалены из базы.

После проведения тестирования стоит провести оценку эффективности данной программы.

Среди плюсов стоит отметить, что скорость работы сотрудников заметно ускорилась, ведь при совершении ошибки на бумажном носителе, все приходилось переписывать заново, а в электронном варианте достаточно сразу исправить ошибку, не меняя всю информацию.

Кроме этого заметно возросла скорость записи пациентов, ведь если документацию заполнял человек с плохим подчерком, приходилось тратить время на расшифровку.

Выводы по главе 3

По результатам работы, имеется готовый и протестированный программный продукт, который успешно выполняет все заданные функции.

Заключение

В процессе выполнения выпускной работы была выполнена главная цель, а именно разработано программное обеспечение для оптимального распределения ресурсов в задачах управления проектами, для достижения данной цели, были выполнены все поставленные задачи.

В первой главе был проведен анализ предметной области и анализ аналогов для автоматизируемого процесса, также были составлены требования к программному продукту. Создана организационная структура предприятия и описаны основные процессы, которые происходят на предприятии, и в организации.

Во второй главе создана архитектура программного продукта, проведен сравнительный анализ и выбраны технологии для реализации приложения и базы данных (Visual Studio 2022, MySQL Workbench, C#). Также составлена диаграмма вариантов использования, диаграмма классов и составлена диаграмма.

В третьей главе описан процесс разработки приложения (интерфейса и логики программы, при помощи WPF и .NET Framework) и создания удаленной базы данных, также было проведено тестирование готового приложения. По результатам проверки, можно сделать вывод, что готовый программный продукт соответствует заданным требованиям и выполняет все заданные функции.

По итогам выполнения выпускной квалификационной работы, все поставленные цели и задачи были успешно выполнены, также была доказана практическая польза программы.

Список используемой литературы и используемых источников

1. Базы данных MySQL // beget URL: https://beget.com/ru/kb/manual/mysql (дата обращения: 01.09.2025).

2. Бизнес-процесс / [Электронный ресурс] // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/Бизнес-процесс (дата обращения: 05.09.2025).

3. Больница им. Т.И. Ерошевского // Самарская областная клиническая офтальмологическая больница имени Т.И. Ерошевского. URL: https://zrenie-samara.ru/ (дата обращения: 10.09.2025).

4. Документация по C# // Microsoft URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 01.09.2025).

5. Ерошевский, Тихон Иванович // Википедия. URL: https://ru.wikipedia.org/wiki/Ерошевский,_Тихон_Иванович (дата обращения: 5.09.2025).

6. Полное руководство по языку программирования C# 14 и платформе .NET 10 // METANIT.COM URL: https://metanit.com/sharp/tutorial/ (дата обращения: 01.10.2025).

7. Порядки и стандарты // vision-kursk URL: https://vision-kursk.ru/порядки-и-стандарты/ (дата обращения: 21.09.2025).

8. Руководство по ASP.NET Core 9 // METANIT.COM URL: https://metanit.com/sharp/aspnet6/ (дата обращения: 04.10.2025).

9. Руководство по WPF // METANIT.COM URL: https://metanit.com/sharp/wpf/ (дата обращения: 14.09.2025). 1

10. Семантическое моделирование. Проектирование БД с помощью ER-модели // Хабр URL: https://habr.com/ru/companies/timeweb/articles/916824/ (дата обращения: 01.10.2025).

11. СЗИ Secret Net Studio // Сиссофт URL:

https://www.syssoft.ru/Kod-Bezopasnosti/SZI-Secret-Net-Studio/ (дата обращения: 10.11.2025).

12. Третья нормальная форма // hexlet URL: https://ru.hexlet.io/courses/rdb-basics/lessons/3nf/theory_unit (дата обращения: 04.09.2025).

13. Центральная Научная Медицинская Библиотека // URL: https://rucml.ru/pages/resources (дата обращения: 21.10.2025). ### Источники на английском языке

14. .NET Core // versionsof URL: https://versionsof.net/core/ (дата обращения: 01.09.2025).

15. C#: описание и особенности языка. [Электронный ресурс]: https://otus.ru/journal/si-sharp-opisanie-i-osobennosti-yazyka/ (дата обращения: 15.09.2025).

16. ER-диаграммы // Miro URL: https://miro.com/ru/diagramming/what-is-an-er-diagram/ (дата обращения: 04.09.2025).

17. UML / [Электронный ресурс] // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/UML (дата обращения: 05.09.2025).

18. Visual Studio / [Электронный ресурс] // Microsoft : [сайт]. — URL: https://visualstudio.microsoft.com/ru/downloads/ (дата обращения: 14.10.2025).

19. Visual Studio / [Электронный ресурс] // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio (дата обращения: 04.09.2025).

20. Windows Presentation Foundation // Введение в C# URL: https://metanit.com/sharp/tutorial/1.1.php (дата обращения: 11.09.2025).