### Частное учреждение образования

### «Колледж бизнеса и права»

### СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО АВТОМАТИЗАЦИИ РАБОТЫ ПРИЕМНОГО ОТДЕЛЕНИЯ БОЛЬНИЦЫ СКОРОЙ ПОМОЩИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### к курсовому проектированию по дисциплине

### «Конструирование программ и языки программирования»

КП Т.196010.401 ПЗ

Руководитель проекта (А.В.Кривошеина)

Учащийся (С.В.Коваленко)

Минск, 2024

**СОДЕРЖАНИЕ**

Изм.

Изм.

Лист

Лист

№ докум.

№ докум.

Подпись

Подпись

Дата

Дата

Лист

Лист

3

3

КП Т.196010.401 ПЗ

ПЗ

ДП Т.517074.401 ПЗ

Разраб.

Разраб.

Коваленко С.В.

Мовчан П.И.

Провер.

Провер.

Кривошеина А.В.

Багласова Е.В.

Т. контр.

Т. контр.

*Якимович К.О.*

Н. контр.

Н. контр.

Багласова Е.В.

Утверд.

Утверд.

*Багласова Т.Г.*

### *СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО* *АВТОМАТИЗАЦИИ РАБОТЫ ПРИЕМНОГО ОТДЕЛЕНИЯ БОЛЬНИЦЫ СКОРОЙ ПОМОЩИ*

*Программное средство для автоматизации учета заявок жителей Фрунзенского района г.Минска*

Лит.

Лит.

Листов

Листов

51

54

КБП

КБП

[Введение 1](#_Toc169941869)

[1. Описание задачи 2](#_Toc169941870)

[1.1. Анализ предметной области 2](#_Toc169941871)

[1.2. Постановка задачи 3](#_Toc169941872)

[2. Проектирование системы 5](#_Toc169941873)

[2.1. Требования к приложению 5](#_Toc169941874)

[2.2. Проектирование модели 5](#_Toc169941875)

[2.3. Организация данных 8](#_Toc169941876)

[2.4. Концептуальный прототип 10](#_Toc169941877)

[3. Описание реализации программного средства 14](#_Toc169941878)

[3.1. Инструменты разработки и применения технологии 14](#_Toc169941879)

[3.2. Порядок авторизации пользователей 15](#_Toc169941880)

[3.3. Организация данных 16](#_Toc169941881)

[3.4. Функции: логическая и физическая организация 21](#_Toc169941882)

[3.5. Входные и выходные данные 24](#_Toc169941883)

[3.6. Функциональное тестирование 25](#_Toc169941884)

[3.7. Описание справочной системы 31](#_Toc169941885)

[4. Применение 32](#_Toc169941886)

[4.1. Назначение программного средства 32](#_Toc169941887)

[4.2. Условия применения 32](#_Toc169941888)

[Заключение 34](#_Toc169941889)

[Список использованных источников 35](#_Toc169941890)

[Приложение А 37](#_Toc169941891)

[Приложение Б 48](#_Toc169941892)

[Приложение В 49](#_Toc169941893)

# **Введение**

В современном мире, где технологии проникают во все сферы жизни, здравоохранение не является исключением. Автоматизация процессов становится все более актуальной и важной задачей, позволяющей минимизировать потери рабочего времени персонала, экономить на его численности и повышать уровень медицинских сервиса.

Целью курсового проекта «Автоматизация работы приемного отделения больницы скорой помощи» является создание программного средства, которое позволит автоматизировать работу медицинского персонала в поликлинике.

Пояснительная записка к курсовому проекту содержит пять разделов:

Описание задачи: В этом разделе раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть решены. Описываются существующие аналоги.

Проектирование системы: В этом разделе перечисляются требования к аппаратному обеспечению и конфигурации компьютера, проводится характеристика операционной системы, обоснование выбранной среды для разработки приложения. Описываются требования к приложению, строится концептуальный прототип, описывается логическая и физическая организация данных, проектируется справочная система.

Описание реализации программного средства: В этом разделе представлены общие сведения о программном средстве и его функциональном назначении, входные и выходные данные.

Применение предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы.

Заключение будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А будет содержать текст программы.

Графическая часть содержит в себе все необходимые диаграммы для проектирования данного программного средства, такие как:

* диаграмма вариантов использования;
* диаграмма деятельности;
* диаграмма классов;
* диаграмма последовательности.

# **Описание задачи**

# **Анализ предметной области**

Автоматизация работы приемного отделения больницы скорой помощи — это наука об организации совместной деятельности медицинских работников различных подразделений поликлиники с целью эффективного предоставления медицинских услуг. Это включает планирование, координацию и оптимизацию всех процессов, связанных с приемом пациентов, ведением медицинской документации и управлением расписанием, что позволяет минимизировать затраты и улучшить качество обслуживания пациентов.

Автоматизация работы приемного отделения больницы скорой помощи является важнейшей частью современной системы здравоохранения, играя ключевую роль в обеспечении эффективного обслуживания пациентов от первого обращения до завершения лечения. Она охватывает все аспекты управления медицинскими услугами, начиная от планирования приема пациентов и заканчивая организацией обследований, процедур и других медицинских мероприятий.

Важными элементами автоматизации работы приемного отделения больницы скорой помощи:

* **управление медицинской документацией**: это включает в себя управление процессами приема, хранения и обработки медицинской документации в электронном виде;
* **управление информацией о пациентах**: это включает в себя сбор, обработку и анализ данных для поддержки принятия решений в области медицинского обслуживания;
* **управление медицинскими операциями**: это включает в себя планирование и контроль над процессами диагностики и лечения пациентов. Это может включать в себя выбор методов лечения, планирование процедур и управление процессами медицинского обслуживания;
* **управление** информацией: это включает в себя сбор, обработку и распространение информации, необходимой для эффективного управления медицинскими операциями. Это может включать в себя управление данными о пациентах, врачах, расписании, медицинских материалах и другой информацией;

Автоматизация работы приемного отделения больницы скорой помощи занимается решением множества задач, которые контролируют медицинские и административные процессы. Особенности медицинских услуг, такие как срочность и специфичность лечения, напрямую влияют на выбор подходящих методов и инструментов автоматизации.

Администратор поликлиники выполняет следующие задачи:

* привлечение и координация работы медицинского персонала;
* сортировка и распределение пациентов по специалистам;

Автоматизация работы приемного отделения больницы скорой помощи решает множество задач, которые контролируют медицинские и административные процессы. Особенности медицинских услуг, такие как срочность и специфичность лечения, напрямую влияют на выбор подходящих методов и инструментов автоматизации.

Автоматизация процессов с помощью компьютерных систем позволяет значительно увеличить эффективность работы, минимизировать вероятность ошибок, связанных с человеческим фактором, и сократить время на выполнение рутинных задач.

Компьютерная обработка информации обеспечивает быстрый доступ к актуальным данным, возможность их анализа и принятия обоснованных решений на основе полученной информации. Это особенно важно в области здравоохранения, где требуется оперативное реагирование на изменяющиеся условия и быстрое принятие решений.

Потенциальной аудиторией данного приложения являются медицинские учреждения и их сотрудники: врачи, медсестры, администраторы. Приложение также может быть полезно для администраторов и управленцев поликлиник, которым необходимо координировать действия всего медицинского персонала. Кроме того, оно может быть полезно для поставщиков медицинских материалов и оборудования, которые хотят оптимизировать свои операции и улучшить эффективность доставки своих товаров до медицинских учреждений.

# **Постановка задачи**

Исходя из анализа предметной области, можно выделить следующие задачи, подлежащие автоматизации:

* автоматизировать управление медицинскими картами пациентов;
* автоматизировать создание и хранение медицинских записей и заключений;
* автоматизировать генерацию отчетов и аналитики по посещаемости и эффективности работы врачей;

В сравнении с другими приложениями для автоматизации работы поликлиник, такими как «МедИнфо», «Инфоклиника», «EMIAS», «1С:Медицина», «Клиника Онлайн» и другими, ваше приложение Med-Assist имеет следующие преимущества:

* автоматическое управление медицинскими картами: приложение автоматически обновляет информацию о пациентах на основе данных о визитах и назначениях, что помогает медицинскому персоналу оперативно получить доступ к актуальной информации и избежать ошибок при лечении;
* гибкость и масштабируемость: приложение спроектировано так, чтобы быть гибким и масштабируемым, позволяя легко добавлять новые функции и поддерживать большое количество пользователей, включая врачей, медсестер и административный персонал;
* пользовательский интерфейс: приложение имеет интуитивно понятный и простой в использовании пользовательский интерфейс, что облегчит работу с ним для всех пользователей, независимо от их уровня технической грамотности.

Таким образом, предлагает уникальный набор функций, которые помогут улучшить эффективность и производительность в области управления поликлиникой, делая его важным инструментом для любого медицинского учреждения, которое хочет оставаться конкурентоспособным в современном быстро меняющемся бизнес-окружении.

# **Проектирование системы**

# **Требования к приложению**

В приложении будет реализована система аутентификации для обеспечения безопасности данных пациентов и сотрудников. Это означает, что доступ к функционалу приложения будет возможен только после ввода уникальных учетных данных (имя пользователя и пароль). Это помогает предотвратить несанкционированный доступ и защитить конфиденциальность медицинской информации.

Макет приложения будет разработан таким образом, чтобы обеспечить простоту и удобство использования. Однако некоторые функции будут доступны только авторизованным пользователям в зависимости от их роли (например, врач или администратор). Кроме того, некоторые функции могут быть ограничены в зависимости от прав доступа конкретного пользователя в системе.

Макет приложения "Автоматизация работы приемного отделения больницы скорой помощи" будет иметь четкую и последовательную структуру. Все элементы управления будут иметь стандартный размер, форму и расположение, чтобы облегчить навигацию пользователям. Шрифт будет выбран таким образом, чтобы обеспечить хорошую читаемость на всех устройствах. Кроме того, цветовая схема и стиль дизайна будут выбраны так, чтобы обеспечить приятное визуальное восприятие и удобство использования.

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор 2400 МГц и выше;
* оперативная память 4 Гбайт и более;
* свободное место на диске 300 Мбайт;
* интегрированная видеокарта на 2 Гбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

# **Проектирование модели**

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенные средства моделирования включают диаграммы вариантов использования, диаграммы классов, диаграммы последовательности и диаграммы деятельности.

Наиболее распространенные средства моделирования являются: диаграмма вариантов использования, диаграмма классов, диаграмма последовательности, диаграмма деятельности.

Диаграмма вариантов использования – это тип диаграммы, используемой в области разработки программного обеспечения для визуализации системы с точки зрения её акторов и вариантов использования.

Основные элементы диаграммы вариантов использования:

* актеры: ­ актеры представляют собой людей или системы, которые взаимодействуют с системой. Они находятся вне системы и могут быть как людьми (например, пользователи или администраторы), так и другими системами или устройствами;
* варианты использования: варианты использования – это функции или действия, которые система может выполнять в ответ на запросы от актеров. Они представляют собой цели, которые ­ актеры хотят достичь с помощью системы;
* отношения: отношения между актёрами и вариантами использования показывают, как ­ актеры и варианты использования взаимодействуют друг с другом. Они могут быть обозначены стрелками или линиями.

Диаграмма вариантов использования помогает разработчикам и стейкхолдерам лучше понять, как система будет работать и какие функции она будет выполнять. Она также помогает определить требования к системе и спланировать её разработку. Это важный инструмент в процессе проектирования системы.

Диаграмма вариантов использования представлена на листе 1 графической части.

На этой диаграмме можно выделить двух пользователей программным продуктом: Врач, который может просматривать и изменять записи пациентов, добавлять медицинские данные и назначать лечение; Администратор, который может просматривать информацию описанную врачом.

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их атрибуты (полей), методы, Макеты и взаимосвязи (отношений) между ними. Она используется для визуализации структуры классов в системе и их взаимосвязей.

Вот основные элементы диаграммы классов:

* Классы: Классы представляют собой ключевые элементы в объектно-ориентированном моделировании. На диаграмме классы представлены в рамках, содержащих три компонента: имя класса, поля (атрибуты) класса и методы класса.
* Отношения: Отношения между классами на диаграмме классов могут представлять отношения, такие как ассоциация, наследование и реализация.

Диаграмма классов соответствует принципам объектно-ориентированного программирования (ООП) и является одним из базовых инструментов проектирования ООП-систем. Она помогает лучше понимать структуру системы и ее компоненты.

В данном приложении присутствуют следующие классы:

* пациент;
* врач;
* анализ крови;
* анализ мочи;
* анализ крови;
* первичный осмотр;
* жалобы пациента;
* эпикриз;
* операции;
* манипуляции;
* медикаментозное лечение;
* физиотерапевтическое лечение.

Диаграмма классов представлена на листе 2 графической части.

Диаграмма деятельности – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности. Эта диаграмма используется для анализа и оптимизации бизнес-процессов, планирования работы, принятия решений и управления проектами.

Основные элементы диаграммы деятельности:

* начальный узел: Начальный узел активности является первым элементом диаграммы деятельности и представляет начальную точку процесса;
* конечный узел активности: Конечный узел активности является последним элементом диаграммы деятельности и обозначает завершение процесса;
* действие: Действия представляют собой отдельные шаги или операции, которые выполняются в процессе;
* поток управления: Эти потоки показывают последовательность действий и передачу данных между действиями;
* узел принятия решений: Эти узлы используются для представления ветвлений и слияний в потоке управления.

Диаграмма деятельности позволяет визуализировать последовательность задач и операций, которые нужно выполнить, чтобы достичь желаемого результата. Она является мощным инструментом для анализа, планирования и управления проектами и бизнес-процессами.

В рамках проекта рассмотрим функцию добавления операции у пациента.

Диаграмма деятельности для добавления операции представлена на листе 3 графической части.

Диаграмма последовательности – это UML-диаграмма, которая показывает взаимодействие между объектами в системе в виде последовательности сообщений, действий и операций. Она отображает порядок выполнения действий и обмена информацией между объектами во времени.

Вот основные элементы диаграммы последовательности:

* объекты: представлены горизонтальной осью диаграммы. Каждый объект имеет свою колонку;
* сообщения: передаются горизонтально между объектами и располагаются вертикально в хронологическом порядке. Они представлены стрелками;
* время: представлено вертикальной осью диаграммы. Жизненный цикл объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) информационной системы в рамках прецедента показаны на единой временной оси.

Диаграмма последовательности помогает анализировать и понимать взаимодействие между объектами в системе, а также порядок, в котором происходит взаимодействие. Она является важным инструментом для анализа и проектирования систем.

Диаграмма последовательности для создания посылки представлена на листе 4 графической части.

# **Организация данных**

Наиболее распространенным средством моделирования таких типов документации являются схемы-сети (ERD), которые применяются для графического представления многих элементов разработанных программ систем и представляют некоторые габаритные размеры обозначений для определения данных и отношений между ними. С помощью такой диаграммы можно описать отдельные компоненты концептуальной модели или выразить свою точку зрения взаимосвязей между ними, иногда какое значение или реализацию системы. Основными потенциалом данной нотации является понятная сущность в каждом случае каждый рассматриваемый объект может явиться экземпляром одного и только одного сущности, должен иметь уникальное имя или идентификатор, а также отличается от других экземпляров данной сущности. Следовательно: как фотонные линии короткого ассортимента между отделами сущности.

Графическая модель данных строится таким образом, чтобы связь между отделениями сущности органа не только естественный характер соответствующего отношения, но и дополнительные аспекты обязательности связей.

Информационная модель представлена на диаграмме «Сущность-связь» на рисунки 2.2

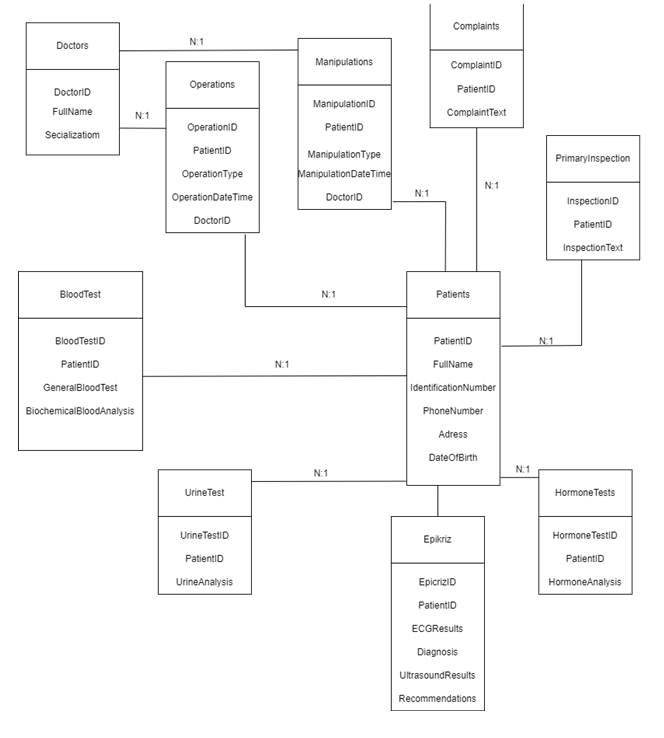


Рисунок 2.2 – Диаграмма «Сущность-связь»

Исследовав предметную область, можно выделить следующие сущности: «Пациент», «Врач», «Анализ крови», «Эпикриз», «Анализ мочи», «Анализ на гормоны» «Первичный осмотр», «Жалобы пациента», «Операции», «Манипуляции», «Медикаментозное лечение», «Физиотерапевтическое лечение».

Для сущности «Пациент» атрибутами будут являться:

* ФИО
* идентификационный номер паспорта;
* номер телефона;
* адрес;
* дата рождения;

Для сущности «Жалобы» атрибутом будет жалоба пациента.

Для сущности «Врач» атрибутами будут являться:

* ФИО;
* специализация врача;

Для сущности «Первичный осмотр» атрибутом будет описание осмотра.

Для сущности «Медикаментозное лечение» атрибутами будут являться:

* тип медикамента;
* дата и время назначения медикамента;
* название медикамента;

Для сущности «Операции» атрибутами будут являться:

* тип операции;
* дата и время операции;

Для сущности «Манипуляции» атрибутами будут являться:

* тип манипуляции;
* дата и время манипуляции;

Для сущности «Анализ крови» атрибутом будут результаты общего и биохимического анализов крови.

Для сущности «Анализ мочи» атрибутом будут результаты анализа мочи.

Для сущности «Анализ на гормоны» будут результаты анализа на гормоны.

Типы связей между сущностями:

* **врач-операции**: один ко многим;
* **врач-манипуляции**: один ко многим
* **пациент-операции**: один ко многим
* **пациент-манипуляции**: один ко многим
* **пациент-жалобы**: один ко многим
* **пациент-первичный осмотр**: один ко многим
* **пациент-анализ крови**: один ко многим
* **пациент-анализ на гормоны**: один ко многим;
* **пациент-анализ мочи**: один ко многим;
* **пациент-эпикриз**: один ко многим.

# **Концептуальный прототип**

Концептуальный прототип состоит из описания внешнего вида пользовательского Макета, а именно, элементов управления.

Разрабатываемое приложение будет содержать две основные формы: «Главная» и «Вход».

В Макете формы «Входа» есть два поля для ввода текста, подписанные как «Логин» и «Пароль». Под полями для ввода находится кнопка с надписью «Войти», которую пользователи могут нажать для входа после ввода своих учетных данных. Внизу Макета есть еще одна кнопка с надписью «Зарегистрироваться» для новых пользователей, которые хотят создать учетную запись.

В Макете формы «Авторизации» есть два поля для ввода текста и поля для выбора деятельности, подписанные как «Логин» и «Пароль», и «Тип работы». Под полями для ввода находится кнопка с надписью «Зарегистрироваться», которую пользователи могут нажать для регистрации после ввода своих учетных данных

Макет формы «Авторизации» представлен в приложении В на рисунке В.1.

Макет главной формы «Главной» для врачей состоит из:

* **кнопка** «Анализы» – перемещает пользователя в окно с различными анализами пациента;
* **кнопка** «Эпикриз» – перемещает пользователя в окно с данными позволяющими поставить пациенту диагноз;
* **кнопка** «Направление» – перемещает пользователя в окно в котором указана основная информация, полученная о пациенте при поступлении;
* **кнопка** «Лечение» – перемещает пользователя в окно в котором указана вся информация о ходе лечения пациента;

Макет формы «Главной» представлен в приложении В на рисунке В.2.

Макет формы «Анализы» состоит из:

* **поле «Общий анализ крови»**: Врач вводит результаты общего анализа крови;
* **поле «Биохимический анализ крови»**: Врач вводит результаты биохимического анализа крови;
* **поле «Анализы на гормоны»**: Врач вводит результаты анализа на гормоны;
* **поле «Анализ мочи»**: Врач вводит результаты анализа мочи;

Макет формы «Анализы» представлен в приложении В на рисунке В.3.

Макет формы «Лечение» состоит из:

* **кнопка** «Операции» – перемещает пользователя в окно с данными о всех операциях проведённых у пациента за время лечения;
* **кнопка** «Манипуляции» – перемещает пользователя в окно с данными о всех манипуляциях проведённых над пациентом за время лечения;
* **кнопка** «Медикаментозное лечение» – перемещает пользователя в окно в котором отображены все препараты, назначенные пациенту для лечения;
* **кнопка** «Физиотерапевтическое лечение» – перемещает пользователя в окно в котором отображены все процедуры, назначенные пациенту для лечения;
* **кнопка «Назад»**: если пользователь решит, что пользователь хочет вернуться на предыдущую формy, он может нажать эту кнопку.

Макет формы «Лечение» представлен в приложении В на рисунке В.4.

Макет формы «Операции» состоит из:

* **таблица «Операции»**: выводит все операции проведённые y пациента;
* **кнопка «**Добавить**»**: позволяет пользователю добавить операцию в БД;
* **кнопка «**Изменить**»**: позволяет пользователю изменить уже существующую операцию в БД;
* **кнопка «**Удалить**»**: – позволяет пользователю удалить операцию, если она например была добавлена по ошибке;
* **кнопка «**Назад**»**: Пользователь вводит полное юридическое название компании;
* выпадающий список **«Пациенты»**: позволяет пользователю выбрать пациента, с которым он будет взаимодействовать;

Макет формы «Операции» представлен в приложении В на рисунке В.5.

Макет формы «Добавить операцию» состоит из:

* выпадающий список **«**Тип операции**»**: используется для выбора типа операции;
* **выбор даты «Дата»**: позволяет выбрать дату проведения операции;
* **поле «Время»**: позволяет указать время проведения операции;
* выпадающий список **«**Врач**»**: позволяет выбрать врача проводящего операцию;
* **кнопка «Сохранить»**: после заполнения всех полей пользователь нажимает эту кнопку для сохранения информации;

Макет формы «Добавить операцию» представлен в приложении В на рисунке В.6.

Макет формы «Изменить операцию» состоит из:

* выпадающий список **«**Тип операции**»**: используется для изменения типа операции;
* **выбор даты «Дата»**: позволяет изменить дату проведения операции;
* **поле «Время»**: позволяет изменить время проведения операции;
* выпадающий список **«**Врач**»**: позволяет изменить врача проводящего операцию;
* **кнопка «Сохранить»**: после заполнения и изменения нужных полей пользователь нажимает эту кнопку для сохранения информации;

Макет формы «Изменить операцию» представлен в приложении В на рисунке В.7.

Макет главной формы для сотрудника регистратуры состоит из:

* **кнопка «Карточки»**: пользователь вводит полное имя водителя;
* **кнопка «Направления»**: пользователь вводит номер телефона водителя;

**Кнопка «Назад»**: если пользователь решит вернуться на предыдущее окно, он может нажать эту кнопку.

Макет формы «Главной» для сотрудника регистратуры представлен в приложении В на рисунке В.8.

Макет формы «Направления» состоит из:

* **таблица «Направления»**: здесь отображается информация о пациентах. Сотрудники могут просматривать и управлять этой информацией;
* **кнопка «Экспорт в Excel»**: пользователь может экспортировать дынные о пациентах в таблицу Excel;
* **кнопка «Назад»**: если пользователь решит вернуться на предыдущее окно, он может нажать эту кнопку.

Макет формы «Направления» представлен в приложении В на рисунке В.9.

Макет формы «Карточки» состоит из:

* **таблица «**Карточки**»**: здесь отображается информация о пациентах. Сотрудники могут просматривать и управлять этой информацией;
* **кнопка «Экспорт в Excel»**: пользователь может экспортировать дынные о пациентах в таблицу Excel;
* **поле ««Назад»**: если пользователь решит вернуться на предыдущее окно, он может нажать эту кнопку;

Макет формы «Карточки» представлен в приложении В на рисунке В.10.

# **Описание реализации программного средства**

# **Инструменты разработки и применения технологии**

Инструментами разработки для будущего программного приложения будут являться:

* операционная система MS Widows 11;
* программная среда разработки Microsoft Visual Studio 2022;
* язык программирования C#;
* система управления реляционными базами данных (СУБД) Microsoft SQL Server 2024;
* программа для построения блок-схем Microsoft Visio
* язык определения данных DDL;
* офисный пакет приложений Microsoft Office.

MS Windows 11 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства MS Windows. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами [19].

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского Макета, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию [14].

MS Visual Studio 2022 включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня [14].

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [15].

DDL состоит из набора SQL-команд, используемых для создания, изменения и удаления структур баз данных, но не данных. Он просто имеет дело с описаниями схемы базы данных и используется для создания и изменения структуры объектов, присутствующих в базе данных. Все DDL-команды автоматически фиксируются, что означает, что все изменения навсегда сохраняются в базе данных. DDL подразделяется на пять команд, которые широко используются в SQL-запросах[15].

Microsoft Visio – векторный графический редактор, редактор диаграмм и блок-схем для Windows. Выпускается в трёх редакциях: Standard, Professional и Pro for Office 365.Аналогично с Adobe Reader, в стандартный набор программ Microsoft Office входит только средство для просмотра и печати диаграмм Microsoft Visio Viewer. Полнофункциональная версия Microsoft Visio Professional для создания и редактирования монограмм и диаграмм. Первоначально Visio разрабатывался и выпускался компанией Shapeware, затем переименованной в Visio Corporation. Microsoft приобрела компанию в 2000 году, тогда продукт назывался Visio 2000 [18].

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[6] как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [16].

Microsoft Office – это офисный пакет, разработанный американской корпорацией Microsoft. В его состав входят приложения для работы с текстовыми документами, электронными таблицами, базами данных и другими типами файлов. Он предоставляет инструменты, такие как Word, Excel, PowerPoint, Outlook, Teams и многое другое [17].

# **Порядок авторизации пользователей**

В приложении можно выделить следующие роли и функции:

Сотрудник учреждения здравоохранения:

* Аутентификация: Сотрудник учреждения здравоохранения регистрируется в системе, предоставляя необходимые учетные данные (например, имя пользователя и пароль);
* Авторизация: после успешной аутентификации сотрудник транспортного отдела получает доступ к функциям, связанным с его ролью;
* Функции: Изменение данных о пациенте, назначение диагноза, сбор анализов и назначение лечения;

Сотрудник регистратуры:

* Аутентификация: Сотрудник регистратуры регистрируется в системе, предоставляя необходимые учетные данные (например, имя пользователя и пароль);
* Авторизация: после успешной аутентификации сотрудник транспортного отдела получает доступ к функциям, связанным с его ролью;
* Функции: Просмотр введённых данных о пациенте. Вывод результатов в виде отчёта в Excel.

# **Организация данных**

Организация данных подразумевает создание модели данных, главными элементами, которые являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает шесть таблиц.

Таблица «Patients» содержит информацию о пацинтах, структура таблицы представлена в таблице 3.1

Таблица 3.1 – Patients

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байта | Описание |
| id | int | 4 | Номер пациента |
| FullName | varchar | 250 | ФИО пациента |
| IdentificationNumber | int | 4 | Идентификационный номер паспорта |
| PhoneNumber | int | 4 | Номер телефона |
| Adress | int | 4 | Адрес пациента |
| DateOfBirth | Datetime | 3 | Дата рождения |

Таблица «BloodTest» содержит информацию о всех анализах крови, структура таблицы представлена в таблице 3.2

Таблица 3.2 – BloodTest

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер анализа |
| PatitentID | int | 4 | Номер пациента |
| GenetalBloodTest | varchar(250) | 250 | Общий анализ крови |
| BiochemicalBloodTest | varchar(250) | 250 | Биохимический анализ крови |

Таблица «Complaints» содержит информацию обо всех жалобах на самочувствие, структура таблицы представлена в таблице 3.3

Таблица 3.3 – Complaints

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| ComplaintID | int | 4 | Номер жалобы пациента |
| PatientID | int | 4 | Номер пациента |
| ComplaintText | varchar(250) | 250 | Жалобы пациента |

Таблица «Doctors» содержит информацию о заказчиках, структура таблицы представлена в таблице 3.4

Таблица 3.4 – Doctors

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер врача |
| FullName | varchar | 250 | ФИО врача |
| Specialization | varchar | 250 | Специализация врача |

Таблица «Epikriz» содержит информацию о всех эпикризах всех пациентов, структура таблицы представлена в таблице 3.5

Таблица 3.5 – Epikriz

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| EpikrizID | int | 4 | Номер эпикриза |
| PatientID | int | 4 | Номер пациента |
| ECGResults | varchar | 250 | Результаты ЭКГ |
| Diagnosis | varchar | 250 | Диагноз |
| UltrasoundResult | varchar | 3 | Результаты УЗИ |
| Recomendation |  |  | Рекомендации |

Таблица «HormoneTest» содержит информацию о анализах на гормоны у всех пациентов, структура таблицы представлена в таблице 3.6

Таблица 3.6 – HormoneTest

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| HormoneTestId | int | 4 | Номер анализа на гормоны |
| PatientID | int | 4 | Номер пациента |
| PatientAnalysis | varchar | 250 | Результаты анализа на гормоны |

Таблица «Manupulations» содержит информацию о всех манипуляциях всех пациентов, структура таблицы представлена в таблице 3.7

Таблица 3.7 – Manupulations

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер манипуляции |
| PatientID | int | 4 | логин |
| ManipulationType | varchar | 250 | Тип манипуляции |
| ManipulationDate | Datetime | 3 | Дата проведения манипуляции |
| DoctorID | int | 4 | Номер врача |

Таблица «Operations» содержит информацию обо всех операциях всех пациентов, структура таблицы представлена в таблице 3.8

Таблица 3.8 – Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер пользователя |
| PatientID | int | 4 | логин |
| OperationType | varchar | 250 | Тип операции |
| OperationDate | Datetime | 3 | Дата проведения операции |
| DoctorID | int | 4 | Номер врача |

Таблица «UrineTest» содержит информацию о результатах анализов мочи всех пациентов, структура таблицы представлена в таблице 3.9

Таблица 3.9 – UrineTest

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер анализа |
| PatientID | int | 4 | Номер пациента |
| UrineAnalysis | varchar | 250 | Анализ мочи |

Таблица «PrimaryInspections» содержит информацию о первичных осмотрах всех пациентов, структура таблицы представлена в таблице 4.0

Таблица 4.0 – PrimaryInspections

|  |  |  |  |
| --- | --- | --- | --- |
| Название строки | Тип данных | Размер в байтах | Описание |
| id | int | 4 | Номер осмотра |
| PatientID | int | 4 | Номер пациента |
| InspectionText | varchar | 250 | Первичный осмотр |

Все таблицы представлены на рисунке 4.1

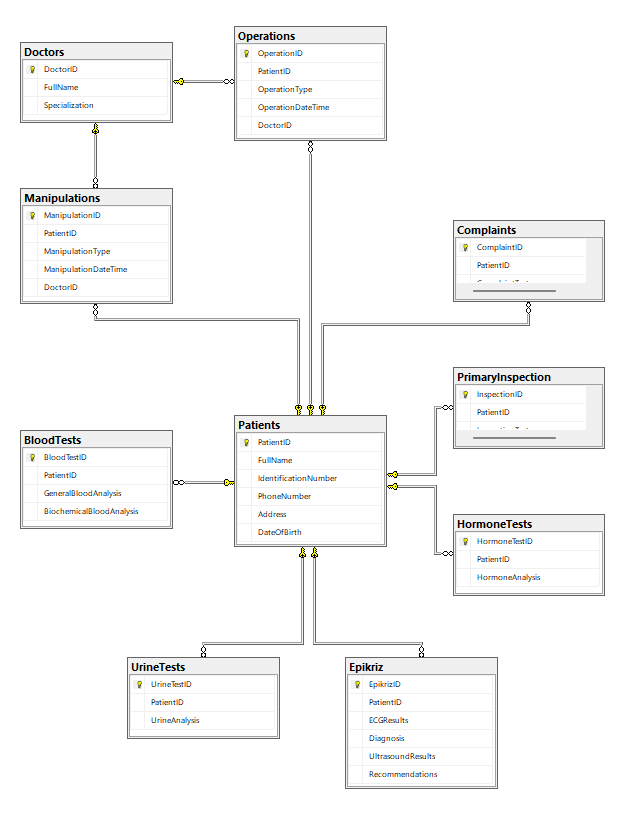


Рисунок 4.1 – Схема данных

# **Функции: логическая и физическая организация**

В данном проекте реализованы функции добавления, редактирования и удаления информации о пациенте, о операциях, о манипуляциях, о медикаментозном и физиотерапевтическом лечении; формирование отчёта обо всех карточках всех пациентов и их направлениях.

Функция добавления информации о пациенте находиться на форме «Направение». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка события Button\_Save\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Napravlеniе». Код функции добавления информации о пациенте:

private void Button\_Save\_Click(object sender, RoutedEventArgs e)

{

string fullName = txtFullName.Text;

string identificationNumber = txtIdentificationNumber.Text;

string phoneNumber = txtPhoneNumber.Text;

string address = txtAddress.Text;

DateTime dateOfBirth = DateTime.Parse(txtDateOfBirth.Text);

string complaints = txtComplaints.Text;

string primaryInspection = txtPrimaryInspection.Text;

ComboBoxItem selectedPatient = cmbPatients.SelectedItem as ComboBoxItem;

if (selectedPatient != null)

{

int patientID = (int)selectedPatient.Tag;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "UPDATE Patients SET FullName = @FullName, IdentificationNumber = @IdentificationNumber, PhoneNumber = @PhoneNumber, Address = @Address, DateOfBirth = @DateOfBirth WHERE PatientID = @PatientID";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@FullName", fullName);

command.Parameters.AddWithValue("@IdentificationNumber", identificationNumber);

command.Parameters.AddWithValue("@PhoneNumber", phoneNumber);

command.Parameters.AddWithValue("@Address", address);

command.Parameters.AddWithValue("@DateOfBirth", dateOfBirth);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

query = "IF EXISTS (SELECT 1 FROM Complaints WHERE PatientID = @PatientID) " +

"UPDATE Complaints SET ComplaintText = @ComplaintText WHERE PatientID = @PatientID " +

"ELSE INSERT INTO Complaints (PatientID, ComplaintText) VALUES (@PatientID, @ComplaintText)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ComplaintText", complaints);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

query = "IF EXISTS (SELECT 1 FROM PrimaryInspection WHERE PatientID = @PatientID) " +

"UPDATE PrimaryInspection SET InspectionText = @InspectionText WHERE PatientID = @PatientID " +

"ELSE INSERT INTO PrimaryInspection (PatientID, InspectionText) VALUES (@PatientID, @InspectionText)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@InspectionText", primaryInspection);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

}

}

}

Функция редактирования информации о пациенте находится на форме «Направление». После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий Button\_Save\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Box».

Функция удаления информации о пациенте находится на форме «Направление». Удаление происходит посредством стирания необходимой информации из поля и нажатия на кнопку «Сохранить».

Функция добавления операции находиться на форме «Операции». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Opеrations».

Функция редактирования информации об операции находится на форме «Операции» После выбора необходимой операции нужно нажать на соответствующую кнопку.. После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Opеrations».

Функция удаления информации об операции находится на форме «Операции». После выбора необходимой операции нужно нажать на соответствующую кнопку. После нажатия на кнопку «Удалить» на данной форме происходит обработка событий DeleteButton\_Click. Выбранные данные будут удалены из таблицы «Opеrations».

Функция добавления манипуляции находиться на форме «Манипуляции». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Manipulations».

Функция редактирования информации об манипуляции находится на форме «Манипуляции». После выбора необходимой манипуляции нужно нажать на соответствующую кнопку. После изменения нужных полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Manipulations».

Функция удаления информации об манипуляции находится на форме «Манипуляции». После выбора необходимой манипуляции нужно нажать на соответствующую кнопку. После нажатия на кнопку «Удалить» на данной форме происходит обработка событий DeleteButton\_Click. Выбранные данные будут удалены из таблицы «Manipulations».

Функция добавления медикаментозного лечения находиться на форме «Медикаментозное лечение». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Mеdications».

Функция редактирования информации о медикаментозном лечении находится на форме «Медикаментозное лечение». После выбора необходимого лечения нужно нажать на соответствующую кнопку. После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Mеdications».

Функция удаления информации о медикаментозном лечении находится на форме «Медикаментозное лечение». После выбора необходимого лечения нужно нажать на соответствующую кнопку. После нажатия на кнопку «Удалить» на данной форме происходит обработка событий DeleteButton\_Click. Выбранные данные будут удалены из таблицы «Mеdications».

Функция добавления физиотерапевтического лечения находиться на форме «Физиотерапевтическое лечение». После заполнения всех полей и нажатия на кнопку «Добавить» на данной форме происходит обработка события SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Phisipthеropy».

Функция редактирования информации о физиотерапевтическом лечении находится на форме «Физиотерапевтическое лечение». После выбора необходимого лечения нужно нажать на соответствующую кнопку. После заполнения всех полей и нажатия на кнопку «Сохранить» на данной форме происходит обработка событий SaveButton\_Click. Все данные, вводимые в поля формы, будут сохранены в таблице «Phisipthеropy».

Функция удаления информации о физиотерапевтическом лечении находится на форме «Физиотерапевтическое лечение». После выбора необходимого лечения нужно нажать на соответствующую кнопку. После нажатия на кнопку «Удалить» на данной форме происходит обработка событий DeleteButton\_Click. Выбранные данные будут удалены из таблицы «Phisipthеropy».

Функция формирования отчёта обо всех пациентах происходит при нажатии на кнопку «**Экспорт в Excel**» на форме «Карточкиии». Происходит обработка события ExportButton\_Click, в процессе которого будет создан документ **Excel** со всеми данными. Код функции создания документа **Excel**:

private void ExportButton\_Click(object sender, RoutedEventArgs e)

{

DataTable dataTable = ((DataView)PatientsDataGrid.ItemsSource).ToTable();

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Excel Files|\*.xlsx",

Title = "Сохранить как Excel файл"

};

if (saveFileDialog.ShowDialog() == true)

{

try

{

using (XLWorkbook workbook = new XLWorkbook())

{

workbook.Worksheets.Add(dataTable, "Patients");

workbook.SaveAs(saveFileDialog.FileName);

}

MessageBox.Show("Данные успешно экспортированы в Excel.", "Экспорт завершен", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка экспорта в Excel: " + ex.Message);

}

}

}

Функция формирования отчёта обо всех направлениях происходит при нажатии на кнопку «**Экспорт в Excel**» на форме «Направления». Происходит обработка события ExportToExcel, в процессе которого будет создан документ Excel со всеми данными о пациентах.

Полный код программных модулей представлен в приложении А.

# **Входные и выходные данные**

Входными данными являются данные, заносимые программой в таблицы: «Operations», «Manipulations», «Patients», «Epikriz». Также входными данными являются данные, вводимые в поля для поиска.

Для добавления данных в таблицу «Operations» необходимо ввести следующие данные:

* пациент;
* тип операции;
* дата операции;
* время операции;
* врач.

Для добавления данных в таблицу «Manipulations» необходимо ввести следующие данные:

* пациент;
* тип манипуляции;
* дата манипуляции;
* время манипуляции;
* врач.

Для добавления данных в таблицу «Patients» необходимо ввести следующие данные:

* ФИО;
* номер телефона;
* идентификационный номер паспорта;
* адрес;
* дата рождения;

Для добавления данных в таблицу «Epikriz» необходимо ввести следующие данные:

* пациент;
* результаты ЭКГ;
* результаты УЗИ;
* диагноз;
* рекомендации.

Выходные данными являются данные, которые экспортируются в файлы Microsoft Office, карточки пациентов и их направления.

Отчёт генерируется по нажатию на кнопку «Создать». Выходные документы представлен в приложении В.11

# **Функциональное тестирование**

Функциональное тестирование – это тестирование функций приложения на соответствия требованиям. Оценка производится в соответствии с ожидаемым и фактическим результатом.

Тестирование программы будет производится последовательно, переходя из одной программы в другую. Во время теста будут проверятся все действия с программой, навигация пунктов меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

В таблицах 4.2-4.9 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Пользователь выбрал вход от имени врача.

Тест-кейс для тестирования функции авторизации от имени врача представлен в таблице 4.2

Таблица 4.2 – Тест-кейс функции авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 1 | Авторизация в приложении | 1. Заполнить поля данными:   — логин: «admin».  — пароль: «password».   1. Нажать на кнопку «Вход». | Ожидаемый  результат: переход на форму «Главная» |
| Фактический  результат: успешный переход на форму «Главная». Результат представлен в приложении В на рисунке В.12. |

Пользователь авторизован и находится на форме «Направление».

Тест-кейс для сохранения изменения данных на форме «Направление» будет представлен в таблицы 4.3

Таблица 4.3 – Тест-кейс сохранения изменения данных на форме «Направление»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 2 | Сохранение изменений на форме «Направление» | 1. Выбрать нужного пациента из выпадающего списка. 2. Изменить данные в полях: — Ф.И.О. пациента. — Идентификационный номера паспорта. — Номер телефона пациента. — Адрес пациента. — Дата рождения пациента. — Жалобы пациента. — Первичный осмотр. 3. Нажать на кнопку «Сохранить». | Ожидаемый  результат: сохранения данных и занесение изменений в БД. |
| Фактический  результат: успешное сохранение изменений и занесение их в БД. Результат представлен в приложении В на рисунке В.13. |

Пользователь авторизован и находится на форме «Анализы».

Тест-кейс сохранения изменения данных на форме «Анализы» будет представлен в таблицы 4.4

Таблица 4.4 – Тест-кейс сохранения изменения данных на форме «Анализы»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 3 | Сохранение изменений на форме «Анализы» | 1. Выбрать нужного пациента из выпадающего списка. 2. Изменить данные в полях: — Общий анализ крови. — Биохимический анализ крови. — Анализ на гормоны. — Анализ мочи. 3. Нажать на кнопку «Сохранить». | Ожидаемый  результат: сохранения данных и занесение изменений в БД. |
| Фактический  результат: успешное сохранение изменений и занесение их в БД. Результат представлен в приложении В на рисунке В.14. |

Пользователь авторизован и находится на форме «Эпикриз».

Тест-кейс сохранения изменения данных на форме «Эпикриз» будет представлен в таблицы 4.5

Таблица 4.5 – Тест-кейс сохранения изменения данных на форме «Эпикриз»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 4 | Сохранение изменений на форме «Эпикриз» | 1. Выбрать нужного пациента из выпадающего списка. 2. Изменить данные в полях: — Результаты ЭКГ. — Результаты УЗИ. — Рекомендации. — Диагноз. 3. Нажать на кнопку «Сохранить». | Ожидаемый  результат: сохранения данных и занесение изменений в БД. |
| Фактический  результат: успешное сохранение изменений и занесение их в БД. Результат представлен в приложении В на рисунке В.15. |

Пользователь авторизован и находится на форме «Операции».

Тест-кейс добавления операции на форме «Операции» будет представлен в таблицы 4.6

Таблица 4.6– Тест-кейс добавления операции на форме «Операции»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 5 | Добавление операции на форме «Операции» | 1. Выбрать нужного пациента из выпадающего списка. 2. Нажать на кнопку «Добавить». 3. Заполнить поля данными: — Тип операции: «Хирургическая». — Дата операции: «05/03/2024». — Время операции: «9:30» — Врач с подходящей специализацией «Селиверстов Иннокентий Даниилович». 4. Нажать на кнопку «Добавить». | Ожидаемый  результат: операция добавлена в таблицу на форме и в БД. |
| Фактический  результат: успешное добавление операции в таблицу на форме и в БД. Результат представлен в приложении В на рисунке В.16. |

Пользователь авторизован и находится на форме «Операции».

Тест-кейс изменения операции на форме «Операции» будет представлен в таблицы 4.7

Таблица 4.7 – Тест-кейс изменения операции на форме «Операции»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 6 | Изменение операции на форме «Операции» | 1. Выбрать нужного пациента из выпадающего списка. 2. Нажать на кнопку «Изменить». 3. Изменить поля данными: — Тип операции: «Нейрохирургическая». — Врач с подходящей специализацией «Смирнов Павел Антонович». 4. Нажать на кнопку «Сохранить». | Ожидаемый  результат: операция сохранена и изменения занесены в БД. |
| Фактический  результат: успешное сохранение и занесение изменений данных в БД. Результат представлен в приложении В на рисунке В.17. |

Пользователь авторизован и находится на форме «Операции».

Тест-кейс удаления операции на форме «Операции» будет представлен в таблицы 4.8

Таблица 4.8 – Тест-кейс удаления операции на форме «Операции»

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 7 | Удаление операции на форме «Операции» | 1. Выбрать нужного пациента из выпадающего списка. 2. Выбрать нужную операцию, посредством нажатия на неё в таблице на форме. 3. Нажать на кнопку «Удалить» | Ожидаемый  результат: операция удалена и изменения занесены в БД. |
| Фактический  результат: успешное удаление и занесение изменений в БД. Результат представлен в приложении В на рисунке В.18. |

Пользователь авторизован от лица регистратора и находится на форме «Направления».

Тест-кейс экспорта данных из таблицы на форме в документа Excel будет представлен в таблицы 4.9

Таблица 4.9 – Тест-кейс экспорта данных из таблицы на форме в документа Excel

|  |  |  |  |
| --- | --- | --- | --- |
| № Тест-кейса | Функция | Шаги выполнения | Результат |
| 8 | Экспорт данных из таблицы на форме в документ Excel | 1. Нажать на кнопку «Экспорт в Excel». 2. Выбрать место для сохранения. 3. Нажать на кнопку «Сохранить» | Ожидаемый  результат: данные экспортированы в нужное место. |
| Фактический  результат: успешный экспорт данных в указанное место. Результат представлен в приложении В на рисунке В.19. |

# **Описание справочной системы**

Справочная система включает в себя несколько ключевых разделов, которые были определены в предоставленном изображении:

* анализы;
* эпикриз;
* направление;
* лечение;
* операции;
* манипуляции;
* медикаментозное лечение;
* физиотерапевтическое лечение;
* карточки;
* направления.

Справочная система может быть вызвана из главной формы, из формы «Лечение» и главной формы сотрудника регистратуры при нажатии на кнопку «?».

# **Применение**

# **Назначение программного средства**

Приложение предназначено для автоматизации работы приемного отделения больницы скорой помощи. Программное средство предоставляет пользователям возможности для поиска, добавления, отображения и редактирования имеющихся в базе объектов: пациентов, врачей, медсестёр, медицинских услуг, уменьшает затраты времени, используемого при составлении медицинских карт и расписаний приёмов, ведёт учёт посещений пациентов. Для данной роли был создан специальный макет, который позволит медицинскому персоналу быстро и удобно управлять расписанием приёмов и вести электронные медицинские карты.

Основные разработанные функции:

* авторизация и регистрация работников в системе;
* возможность добавлять и редактировать информацию о пациентах;
* возможность создавать и управлять расписанием приёмов;
* возможность назначения диагноза пациенту;
* возможность назначения лечения пациенту;
* возможность добавлять и редактировать операции;
* возможность добавлять и редактировать манипуляции;
* возможность назначать и редактировать медикаментозное лечение;
* возможность назначать и редактировать физиотерапевтическое лечение;
* возможность экспорта данных в документы Microsoft Office.

Приложение «Med-Assist.exe» разработано специально для медицинских операций в поликлиниках и не предназначено для использования в других отраслях, таких как производство или розничная торговля. Оно оптимизировано для работы с медицинскими данными и может не подходить для управления данными, не относящимися к здравоохранению. Кроме того, функциональность экспорта ограничена поддержкой документов Microsoft Office, что может не соответствовать потребностям организаций, использующих другие форматы для документооборота.

# **Условия применения**

Для применения данного программного средства необходимы следующие технические требования:

* Процессор intel Pentium 2.2 и выше;
* минимальный объём оперативной памяти 2000 Мбайт;
* оперативная система Windows 7 и выше;
* платформа .Net Framework v4.8;
* видеокарта с объёмом памяти не менее 256 Мбайт;
* клавиатура, мышь, монитор, принтер.

Процесс установки программного средства «Sub-Zero.exe» включает следующие шаги:

* загрузка программы: скачайте установочный файл «Sub-Zero.exe» с github или носителя, предоставленного разработчиком;
* установка: запустите скачанный установочный файл и следуйте инструкциям мастера установки.

# **Заключение**

Проект представляет собой программное средство, разработанное для автоматизации работы приемного отделения больницы скорой помощи. Организационная сущность задачи заключается в создании инструмента, который облегчит работу врачей, медсестер и административного персонала в области управления медицинскими данными и пациентами. Предметная область проекта охватывает управление записями пациентов, назначение приемов, обработку медицинских карт, выписку рецептов и генерацию отчетов.

Технические требования для программы включают процессор Intel Core i5 или выше, минимальный объем оперативной памяти 500 Мбайт, операционную систему Windows 11, платформу .Net Framework v4.8 и видеокарту с объемом памяти не менее 256 Мбайт. Логическая организация данных включает описание структуры данных и связей между объектами (пациенты, врачи, медсестры, приемы). Физическая организация данных включает выбор СУБД и оптимизацию запросов. Справочная система описывает функциональность и предоставляет помощь пользователям.

Программное средство предоставляет возможности поиска, добавления, отображения и редактирования объектов в базе данных. Входные данные включают информацию о пациентах, врачах, медсестрах и назначенных приемах. Выходные данные могут быть отчетами, статусами приемов и другой информацией.

Среди преимуществ разработанного ПС можно выделить автоматическое управление записями пациентов, гибкость и масштабируемость, а также интуитивный пользовательский интерфейс. Однако следует учитывать ограниченность функциональности экспорта данных в документы Microsoft Office и то, что программа не подходит для управления данными, не связанными с медицинской практикой.

Проект представляет собой важный инструмент для поликлиник, помогая им оставаться конкурентоспособными в современном бизнес-окружении и обеспечивать высокий уровень обслуживания пациентов.

# **Список использованных источников**

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение технологий» / С.В. Банцевич, Т.Г. Багласова. – Минск: КБП, 2022. – 49 c.
2. Михалевич, В.Ю. Методические указания к курсовому проектированию для обучающихся специальности 2-40 01 01 «Программное обеспечение информационных технологий – Минск: КБП, 2023. – 3 c.
3. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
4. 9 лучших программ для автоматизации рабочих процессов [2024 г.] [Электронный ресурс] – Режим доступа: https://nauchniestati.ru/spravka;
5. Автоматизация рабочего места – RPA ROBIN [Электронный ресурс] – Режим доступа: https://www.atlassian.com/ru/agile;
6. Автоматизированное рабочее место (АРМ): что это? [Электронный ресурс] – Режим доступа: https://www.liga-commersantov.ru/articles;
7. Обязанности менеджера по работе с клиентами – чем занимается и как [Электронный ресурс] – Режим доступа: https://sales-generator.ru/blog;
8. Разработка автоматизированного рабочего места менеджера [Электронный ресурс] – Режим доступа: https://spravochnick.ru/informatika;
9. Топ лучших приложений для управления персоналом для автоматизации [Электронный ресурс] – Режим доступа: https://www.guru99.com/ru/best;
10. ТОП-26 программ мониторинга и контроля работы сотрудников за [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles;
11. Удобство и эффективность: разработка автоматизированного рабочего места [Электронный ресурс] – Режим доступа: <https://appmaster.io/ru/blog>;
12. Что значит АРМ (Автоматизированное рабочее место)? [Электронный ресурс] – Режим доступа: https://alexrovich.ru/info;
13. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
14. [Visual Studio 2022 IDE - Programming Tool for Software Developers (microsoft.com)](https://visualstudio.microsoft.com/vs/) [Электронный ресурс] – Режим доступа: https://visualstudio.microsoft.com/vs/.
15. C Sharp – Википедия [2024 г.] [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp;
16. [Microsoft Office – Wikipedia](https://en.wikipedia.org/wiki/Microsoft_Office) [2024 г.] [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Microsoft\_Office;
17. [Microsoft Visio – Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/Microsoft_Visio) 2024 г.] [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visio>;
18. Windows 11 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа: https://support.microsoft.com/ru-ru/meetwindows11. – Дата доступа: 16.03.2024;

# **Приложение А**

**(обязательное)**

**Текст программных модулей**

public partial class AddManipulationWindow : Window

{

private int patientID;

private string connectionString = "Data Source=ssSdkfz251;Initial Catalog=MedicalRecords;Integrated Security=True";

public AddManipulationWindow(int patientID)

{

InitializeComponent();

this.patientID = patientID;

}

private void ManipulationTypeComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ManipulationTypeComboBox.SelectedItem != null)

{

LoadDoctors((ManipulationTypeComboBox.SelectedItem as ComboBoxItem).Content.ToString());

}

}

private void LoadDoctors(string specialization)

{

DoctorComboBox.Items.Clear();

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

SqlCommand command = new SqlCommand("SELECT DoctorID, FullName FROM Doctors WHERE Specialization = @Specialization", connection);

command.Parameters.AddWithValue("@Specialization", specialization);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

DoctorComboBox.Items.Add(new ComboBoxItem { Content = reader["FullName"].ToString(), Tag = reader["DoctorID"] });

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка загрузки врачей: " + ex.Message);

}

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (DoctorComboBox.SelectedItem == null || ManipulationTypeComboBox.SelectedItem == null || !ManipulationDatePicker.SelectedDate.HasValue || string.IsNullOrEmpty(ManipulationTimeTextBox.Text))

{

MessageBox.Show("Пожалуйста, заполните все поля.");

return;

}

string manipulationType = (ManipulationTypeComboBox.SelectedItem as ComboBoxItem).Content.ToString();

DateTime manipulationDateTime = ManipulationDatePicker.SelectedDate.Value.Date + DateTime.Parse(ManipulationTimeTextBox.Text).TimeOfDay;

int doctorID = (int)((ComboBoxItem)DoctorComboBox.SelectedItem).Tag;

using (SqlConnection connection = new SqlConnection(connectionString))

{

try

{

connection.Open();

SqlCommand command = new SqlCommand("INSERT INTO Manipulations (PatientID, ManipulationType, ManipulationDateTime, DoctorID) VALUES (@PatientID, @ManipulationType, @ManipulationDateTime, @DoctorID)", connection);

command.Parameters.AddWithValue("@PatientID", patientID);

command.Parameters.AddWithValue("@ManipulationType", manipulationType);

command.Parameters.AddWithValue("@ManipulationDateTime", manipulationDateTime);

command.Parameters.AddWithValue("@DoctorID", doctorID);

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка сохранения манипуляции: " + ex.Message);

}

}

this.Close();

}

}

public partial class AddMedicationWindow : Window

{

private readonly int patientId;

private readonly string connectionString = "Data Source=ssSdkfz251;Initial Catalog=MedicalRecords;Integrated Security=True";

private Dictionary<string, List<string>> medicationCategories = new Dictionary<string, List<string>>

{

{ "Антибиотики", new List<string> { "Амоксициллин", "Цефтриаксон", "Азитромицин" } },

{ "Обезболивающие", new List<string> { "Парацетамол", "Ибупрофен", "Диклофенак" } },

{ "Противовоспалительные", new List<string> { "Диклофенак", "Ибупрофен", "Напроксен" } }

};

public AddMedicationWindow(int patientId)

{

InitializeComponent();

this.patientId = patientId;

MedicationCategoryComboBox.ItemsSource = medicationCategories.Keys;

LoadDoctors();

}

private void LoadDoctors()

{

try

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT DoctorID, FullName FROM Doctors", conn);

SqlDataReader reader = cmd.ExecuteReader();

DataTable doctors = new DataTable();

doctors.Load(reader);

DoctorComboBox.ItemsSource = doctors.DefaultView;

DoctorComboBox.DisplayMemberPath = "FullName";

DoctorComboBox.SelectedValuePath = "DoctorID";

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке списка врачей: " + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void MedicationCategoryComboBox\_SelectionChanged(object sender, RoutedEventArgs e)

{

string selectedCategory = MedicationCategoryComboBox.SelectedItem as string;

if (selectedCategory != null)

{

MedicationNameComboBox.ItemsSource = medicationCategories[selectedCategory];

}

}

private void AddMedication\_Click(object sender, RoutedEventArgs e)

{

string medicationType = MedicationCategoryComboBox.SelectedItem as string;

string medicationName = MedicationNameComboBox.SelectedItem as string;

DateTime medicationDateTime = MedicationDatePicker.SelectedDate ?? DateTime.Now;

string medicationTime = MedicationTimeTextBox.Text;

int doctorId = (int)DoctorComboBox.SelectedValue;

try

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Medications (PatientID, MedicationType, MedicationName, MedicationDateTime, DoctorID) VALUES (@PatientID, @MedicationType, @MedicationName, @MedicationDateTime, @DoctorID)", conn);

cmd.Parameters.AddWithValue("@PatientID", patientId);

cmd.Parameters.AddWithValue("@MedicationType", medicationType);

cmd.Parameters.AddWithValue("@MedicationName", medicationName);

cmd.Parameters.AddWithValue("@MedicationDateTime", medicationDateTime);

cmd.Parameters.AddWithValue("@DoctorID", doctorId);

cmd.ExecuteNonQuery();

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении медикамента: " + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

public partial class AddOperation : Window

{

private string connectionString = "Data Source=ssSdkfz251;Initial Catalog=MedicalRecords;Integrated Security=True";

private int patientId;

public AddOperation(int patientId)

{

InitializeComponent();

this.patientId = patientId;

LoadDoctors("Хирургическая"); // При инициализации загружаем врачей для типа операции "Хирургическая"

}

private void OperationTypeComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

if (OperationTypeComboBox.SelectedItem != null)

{

string specialization = (OperationTypeComboBox.SelectedItem as System.Windows.Controls.ComboBoxItem).Content.ToString();

LoadDoctors(specialization);

}

}

private void LoadDoctors(string specialization)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT DoctorID, FullName FROM Doctors WHERE Specialization = @Specialization", conn);

cmd.Parameters.AddWithValue("@Specialization", specialization);

SqlDataReader reader = cmd.ExecuteReader();

DoctorComboBox.Items.Clear();

while (reader.Read())

{

int doctorId = reader.GetInt32(0);

string fullName = reader.GetString(1);

DoctorComboBox.Items.Add(new { FullName = fullName, DoctorID = doctorId });

}

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

string operationType = (OperationTypeComboBox.SelectedItem as System.Windows.Controls.ComboBoxItem).Content.ToString();

DateTime operationDate = OperationDatePicker.SelectedDate.Value;

string operationTime = OperationTimeTextBox.Text;

int doctorId = (int)DoctorComboBox.SelectedValue;

DateTime operationDateTime;

if (!DateTime.TryParse($"{operationDate.ToShortDateString()} {operationTime}", out operationDateTime))

{

MessageBox.Show("Пожалуйста, введите корректные дату и время.");

return;

}

if (IsOperationExists(patientId, operationDateTime))

{

MessageBox.Show("Операция для выбранного пациента на эту дату уже существует.");

return;

}

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Operations (PatientID, OperationType, OperationDateTime, DoctorID) VALUES (@PatientID, @OperationType, @OperationDateTime, @DoctorID)", conn);

cmd.Parameters.AddWithValue("@PatientID", patientId);

cmd.Parameters.AddWithValue("@OperationType", operationType);

cmd.Parameters.AddWithValue("@OperationDateTime", operationDateTime);

cmd.Parameters.AddWithValue("@DoctorID", doctorId);

cmd.ExecuteNonQuery();

}

this.Close();

}

private bool IsOperationExists(int patientId, DateTime operationDate)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT COUNT(\*) FROM Operations WHERE PatientID = @PatientID AND CONVERT(date, OperationDateTime) = @OperationDate", conn);

cmd.Parameters.AddWithValue("@PatientID", patientId);

cmd.Parameters.AddWithValue("@OperationDate", operationDate.Date);

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

}

public partial class AddPhysiotherapy : Window

{

private string connectionString = "Data Source=ssSdkfz251;Initial Catalog=MedicalRecords;Integrated Security=True";

private int patientId;

public AddPhysiotherapy(int patientId)

{

InitializeComponent();

this.patientId = patientId;

LoadDoctors();

}

private void LoadDoctors()

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT DoctorID, FullName FROM Doctors", conn);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int doctorId = reader.GetInt32(0);

string fullName = reader.GetString(1);

PhysiotherapyDoctorComboBox.Items.Add(fullName); // Добавляем только ФИО в ComboBox

}

}

}

private void AddButton\_Click(object sender, RoutedEventArgs e)

{

string physiotherapyType = (PhysiotherapyTypeComboBox.SelectedItem as ComboBoxItem).Content.ToString();

DateTime physiotherapyDate = PhysiotherapyDatePicker.SelectedDate.Value;

string physiotherapyTime = PhysiotherapyTimeTextBox.Text;

string physiotherapyPurpose = PhysiotherapyPurposeTextBox.Text;

string selectedFullName = PhysiotherapyDoctorComboBox.SelectedItem.ToString(); // Получаем выбранное ФИО

int doctorId = GetDoctorIdByFullName(selectedFullName); // Получаем ID врача по выбранному ФИО

DateTime physiotherapyDateTime;

if (!DateTime.TryParse($"{physiotherapyDate.ToShortDateString()} {physiotherapyTime}", out physiotherapyDateTime))

{

MessageBox.Show("Пожалуйста, введите корректные дату и время.");

return;

}

using (SqlConnection conn = new SqlConnection(connectionString))

{

try

{

conn.Open();

SqlCommand cmd = new SqlCommand("INSERT INTO Physiotherapy (PatientID, ProcedureType, ProcedureDate, ProcedureTime, DoctorID, Recommendations) VALUES (@PatientID, @PhysiotherapyType, @PhysiotherapyDate, @PhysiotherapyTime, @DoctorID, @PhysiotherapyPurpose)", conn);

cmd.Parameters.AddWithValue("@PatientID", patientId);

cmd.Parameters.AddWithValue("@PhysiotherapyType", physiotherapyType);

cmd.Parameters.AddWithValue("@PhysiotherapyDate", physiotherapyDate);

cmd.Parameters.AddWithValue("@PhysiotherapyTime", physiotherapyTime);

cmd.Parameters.AddWithValue("@DoctorID", doctorId);

cmd.Parameters.AddWithValue("@PhysiotherapyPurpose", physiotherapyPurpose);

int rowsAffected = cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении данных в базу данных: {ex.Message}");

}

}

this.Close();

}

private int GetDoctorIdByFullName(string fullName)

{

int doctorId = -1;

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT DoctorID FROM Doctors WHERE FullName = @FullName", conn);

cmd.Parameters.AddWithValue("@FullName", fullName);

object result = cmd.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

doctorId = Convert.ToInt32(result);

}

}

return doctorId;

}

}   
  
private void Button\_Save\_Click(object sender, RoutedEventArgs e)

{

string generalBloodAnalysis = txtGeneralBloodAnalysis.Text;

string biochemicalBloodAnalysis = txtBiochemicalBloodAnalysis.Text;

string hormoneAnalysis = txtHormoneAnalysis.Text;

string urineAnalysis = txtUrineAnalysis.Text;

ComboBoxItem selectedPatient = cmbPatients.SelectedItem as ComboBoxItem;

if (selectedPatient != null)

{

int patientID = (int)selectedPatient.Tag;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "IF EXISTS (SELECT 1 FROM BloodTests WHERE PatientID = @PatientID) " +

"UPDATE BloodTests SET GeneralBloodAnalysis = @GeneralBloodAnalysis, BiochemicalBloodAnalysis = @BiochemicalBloodAnalysis WHERE PatientID = @PatientID " +

"ELSE " +

"INSERT INTO BloodTests (PatientID, GeneralBloodAnalysis, BiochemicalBloodAnalysis) VALUES (@PatientID, @GeneralBloodAnalysis, @BiochemicalBloodAnalysis)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@GeneralBloodAnalysis", generalBloodAnalysis);

command.Parameters.AddWithValue("@BiochemicalBloodAnalysis", biochemicalBloodAnalysis);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

query = "IF EXISTS (SELECT 1 FROM HormoneTests WHERE PatientID = @PatientID) " +

"UPDATE HormoneTests SET HormoneAnalysis = @HormoneAnalysis WHERE PatientID = @PatientID " +

"ELSE " +

"INSERT INTO HormoneTests (PatientID, HormoneAnalysis) VALUES (@PatientID, @HormoneAnalysis)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@HormoneAnalysis", hormoneAnalysis);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

query = "IF EXISTS (SELECT 1 FROM UrineTests WHERE PatientID = @PatientID) " +

"UPDATE UrineTests SET UrineAnalysis = @UrineAnalysis WHERE PatientID = @PatientID " +

"ELSE " +

"INSERT INTO UrineTests (PatientID, UrineAnalysis) VALUES (@PatientID, @UrineAnalysis)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@UrineAnalysis", urineAnalysis);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

}

}

}   
private void Button\_Click(object sender, RoutedEventArgs e)

{

string connectionString = "Data Source=ssSdkfz251;Initial Catalog=MedicalRecords;Integrated Security=True";

string username = login.Text;

string password = passwordBox.Password;

try

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

string query = "SELECT PasswordHash FROM Users WHERE Username = @Username";

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

var result = cmd.ExecuteScalar();

if (result != null && VerifyPassword(password, result.ToString()))

{

Main mainW = new Main();

mainW.Show();

this.Close();

}

else

{

MessageBox.Show("Неверный логин или пароль");

}

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка подключения к базе данных: " + ex.Message);

}

}

private bool VerifyPassword(string password, string storedHash)

{

using (SHA256 sha256 = SHA256.Create())

{

byte[] computedHash = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));

string computedHashString = BitConverter.ToString(computedHash).Replace("-", "").ToLower();

return computedHashString == storedHash;

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (DoctorComboBox.SelectedItem == null || ManipulationTypeComboBox.SelectedItem == null || !ManipulationDatePicker.SelectedDate.HasValue || string.IsNullOrEmpty(ManipulationTimeTextBox.Text))

{

MessageBox.Show("Пожалуйста, заполните все поля.");

return;

}

string manipulationType = (ManipulationTypeComboBox.SelectedItem as ComboBoxItem).Content.ToString();

DateTime manipulationDateTime = ManipulationDatePicker.SelectedDate.Value.Date + DateTime.Parse(ManipulationTimeTextBox.Text).TimeOfDay;

int doctorID = (int)((ComboBoxItem)DoctorComboBox.SelectedItem).Tag;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand("UPDATE Manipulations SET ManipulationType = @ManipulationType, ManipulationDateTime = @ManipulationDateTime, DoctorID = @DoctorID WHERE ManipulationID = @ManipulationID", connection);

command.Parameters.AddWithValue("@ManipulationType", manipulationType);

command.Parameters.AddWithValue("@ManipulationDateTime", manipulationDateTime);

command.Parameters.AddWithValue("@DoctorID", doctorID);

command.Parameters.AddWithValue("@ManipulationID", manipulationID);

command.ExecuteNonQuery();

}

this.Close();

}

private void SaveMedication\_Click(object sender, RoutedEventArgs e)

{

string medicationType = MedicationCategoryComboBox.SelectedItem as string;

string medicationName = MedicationNameComboBox.SelectedItem as string;

DateTime medicationDateTime = MedicationDatePicker.SelectedDate ?? DateTime.Now;

medicationDateTime = medicationDateTime.Add(TimeSpan.Parse(MedicationTimeTextBox.Text));

int? doctorId = DoctorComboBox.SelectedValue as int?;

try

{

if (doctorId != null)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("UPDATE Medications SET MedicationType = @MedicationType, MedicationName = @MedicationName, MedicationDateTime = @MedicationDateTime, DoctorID = @DoctorID WHERE MedicationID = @MedicationID", conn);

cmd.Parameters.AddWithValue("@MedicationID", medicationId);

cmd.Parameters.AddWithValue("@MedicationType", medicationType);

cmd.Parameters.AddWithValue("@MedicationName", medicationName);

cmd.Parameters.AddWithValue("@MedicationDateTime", medicationDateTime);

cmd.Parameters.AddWithValue("@DoctorID", doctorId);

cmd.ExecuteNonQuery();

this.Close();

}

}

else

{

MessageBox.Show("Пожалуйста, выберите врача.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении изменений медикамента: " + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Button\_Save\_Click(object sender, RoutedEventArgs e)

{

string ecgResults = txtECGResults.Text;

string diagnosis = txtDiagnosis.Text;

string ultrasoundResults = txtUltrasoundResults.Text;

string recommendations = txtRecommendations.Text;

ComboBoxItem selectedPatient = cmbPatients.SelectedItem as ComboBoxItem;

if (selectedPatient != null)

{

int patientID = (int)selectedPatient.Tag;

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "IF EXISTS (SELECT 1 FROM Epikriz WHERE PatientID = @PatientID) " +

"UPDATE Epikriz SET ECGResults = @ECGResults, Diagnosis = @Diagnosis, UltrasoundResults = @UltrasoundResults, Recommendations = @Recommendations WHERE PatientID = @PatientID " +

"ELSE " +

"INSERT INTO Epikriz (PatientID, ECGResults, Diagnosis, UltrasoundResults, Recommendations) VALUES (@PatientID, @ECGResults, @Diagnosis, @UltrasoundResults, @Recommendations)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ECGResults", ecgResults);

command.Parameters.AddWithValue("@Diagnosis", diagnosis);

command.Parameters.AddWithValue("@UltrasoundResults", ultrasoundResults);

command.Parameters.AddWithValue("@Recommendations", recommendations);

command.Parameters.AddWithValue("@PatientID", patientID);

command.ExecuteNonQuery();

}

}

}

}

private void ExportButton\_Click(object sender, RoutedEventArgs e)

{

DataTable dataTable = ((DataView)PatientsDataGrid.ItemsSource).ToTable();

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Excel Files|\*.xlsx",

Title = "Сохранить как Excel файл"

};

if (saveFileDialog.ShowDialog() == true)

{

try

{

using (XLWorkbook workbook = new XLWorkbook())

{

workbook.Worksheets.Add(dataTable, "Patients");

workbook.SaveAs(saveFileDialog.FileName);

}

MessageBox.Show("Данные успешно экспортированы в Excel.", "Экспорт завершен", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка экспорта в Excel: " + ex.Message);

}

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

string operationType = (OperationTypeComboBox.SelectedItem as ComboBoxItem).Content.ToString();

DateTime operationDate = OperationDatePicker.SelectedDate.Value;

string operationTime = OperationTimeTextBox.Text;

string doctor = (DoctorComboBox.SelectedItem as ComboBoxItem)?.Content.ToString();

int doctorId = (int)(DoctorComboBox.SelectedItem as ComboBoxItem)?.Tag;

DateTime operationDateTime;

if (!DateTime.TryParse($"{operationDate.ToShortDateString()} {operationTime}", out operationDateTime))

{

MessageBox.Show("Пожалуйста, введите корректные дату и время.");

return;

}

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

SqlCommand cmd = new SqlCommand("UPDATE Operations SET OperationType = @OperationType, OperationDateTime = @OperationDateTime, DoctorID = @DoctorID WHERE OperationID = @OperationID", conn);

cmd.Parameters.AddWithValue("@OperationID", operationId);

cmd.Parameters.AddWithValue("@OperationType", operationType);

cmd.Parameters.AddWithValue("@OperationDateTime", operationDateTime);

cmd.Parameters.AddWithValue("@DoctorID", doctorId);

cmd.ExecuteNonQuery();

}

this.Close();

}

# **Приложение Б**

**(справочное)**

**Формы выходных документов**

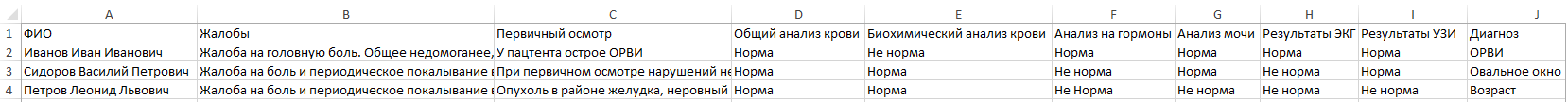


Рисунок Б.1 – Направления

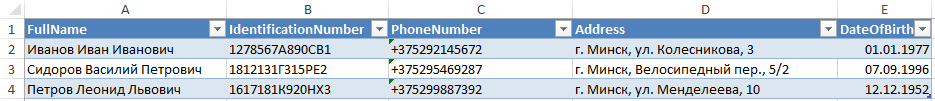


Рисунок Б.2 – Карточки

# **Приложение В**

**(справочное)**

**Результаты работы программы**

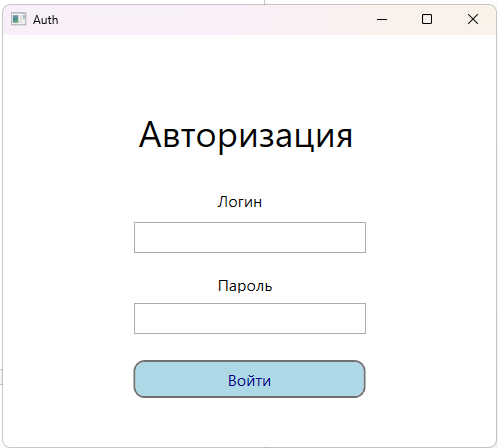


Рисунок В.1 – Форма входа от имени врача



Рисунок В.2 – Форма «Главная» для врача

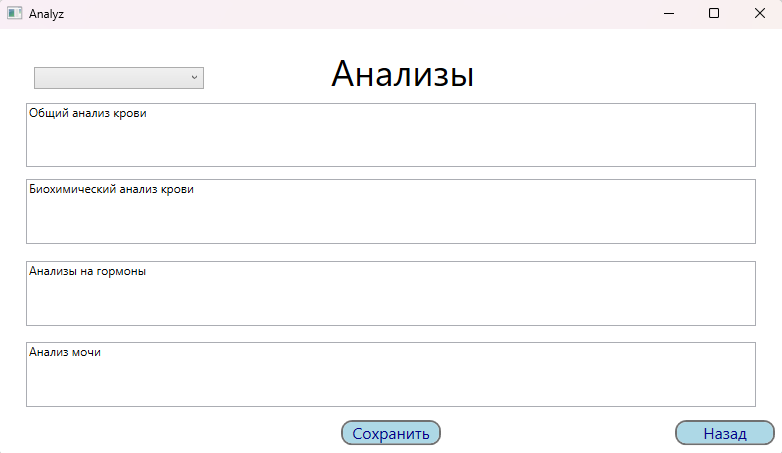


Рисунок В.3 – Форма «Анализы»



Рисунок В.4 – Форма «Лечение»

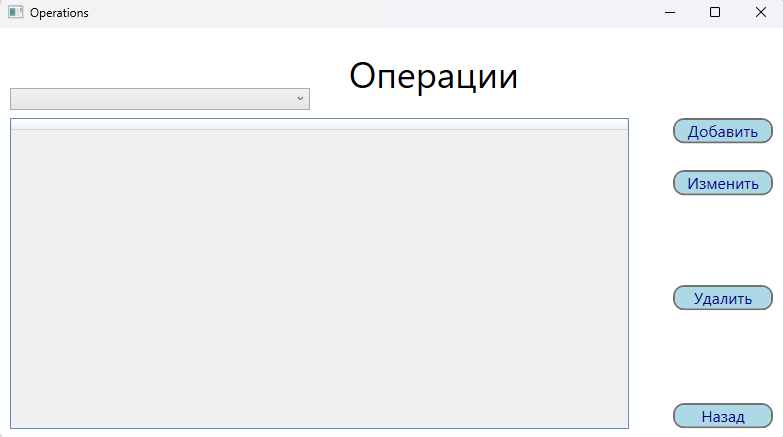


Рисунок В.5– Форма «Операции»

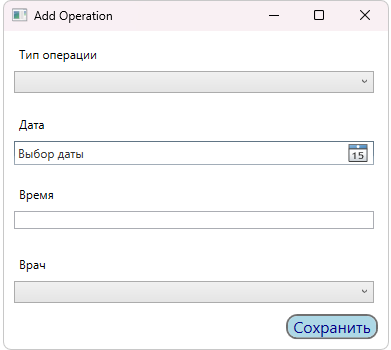


Рисунок В.6– Форма «Добавить операцию»

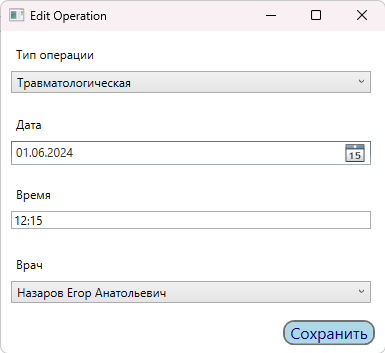


Рисунок В.7– Форма «Изменить операцию»

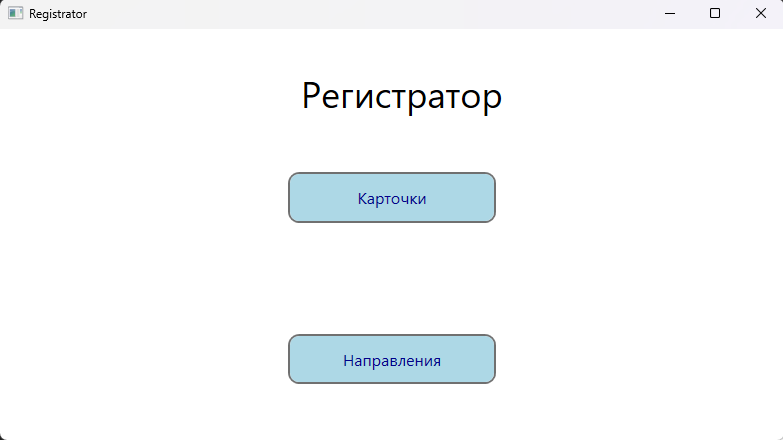


Рисунок В.8 – Форма «Главная» сотрудника регистратуры

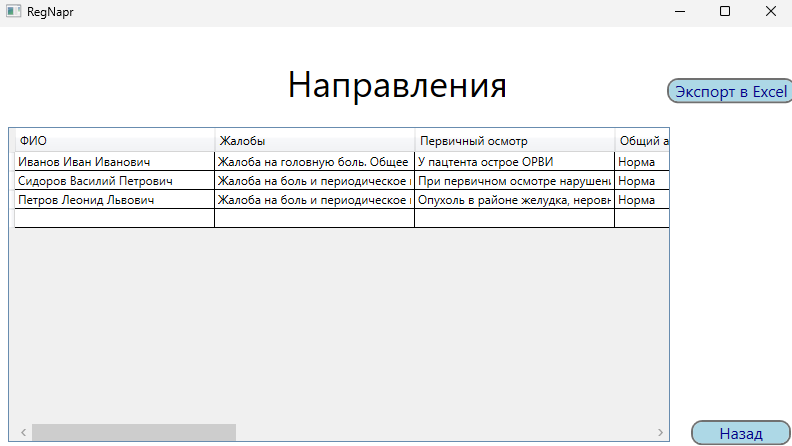


Рисунок В.9 – Форма «Направления»

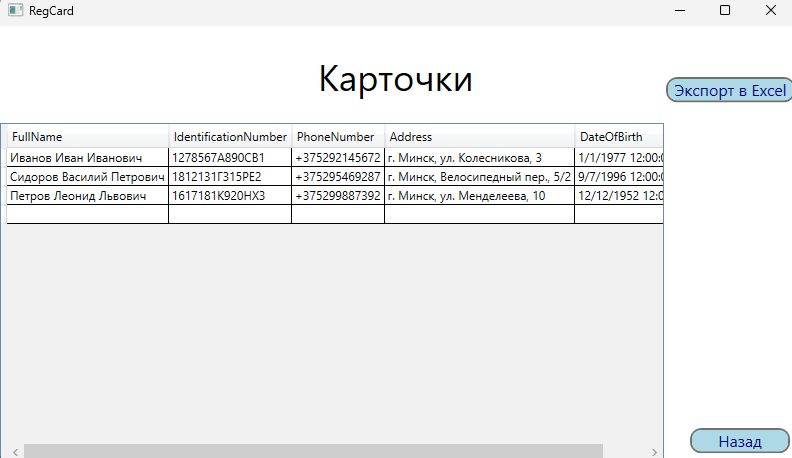


Рисунок В.10 – Форма «Карточки»

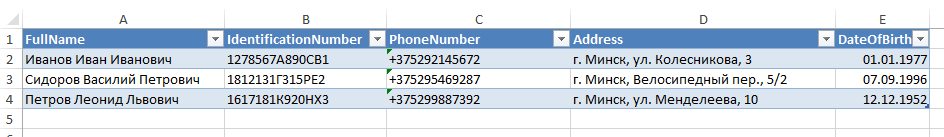


Рисунок В.11– Отчёт

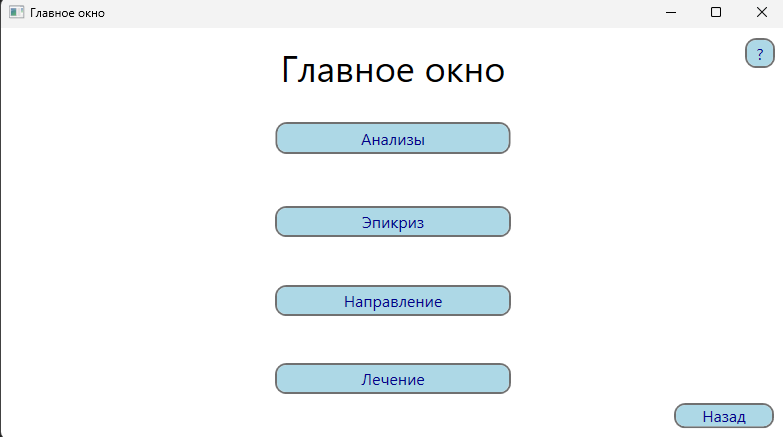


Рисунок В.12 – Результат входа врача

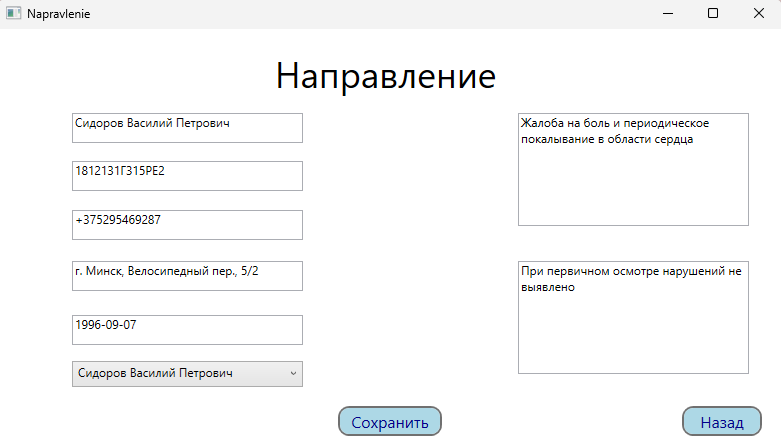


Рисунок В.13 – Результат сохранения направления

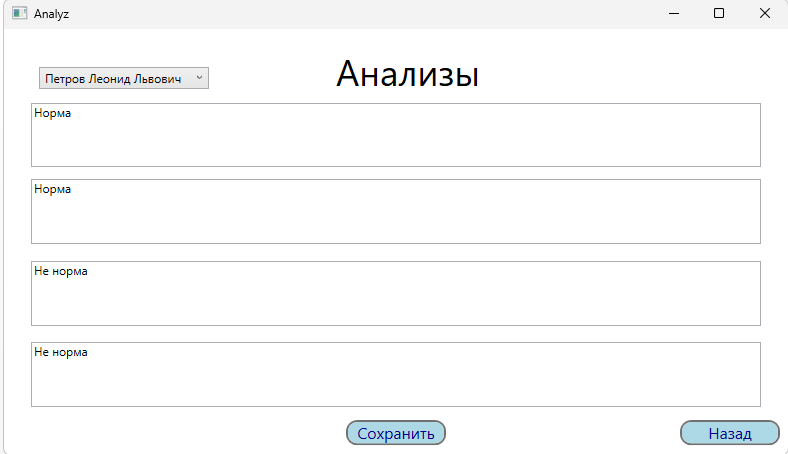


Рисунок В.14 – Результат сохранения анализов

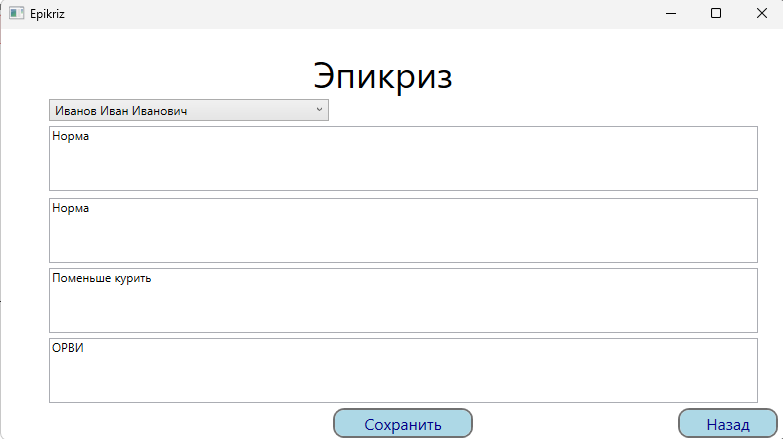


Рисунок В.15 – Результат сохранения эпикриза

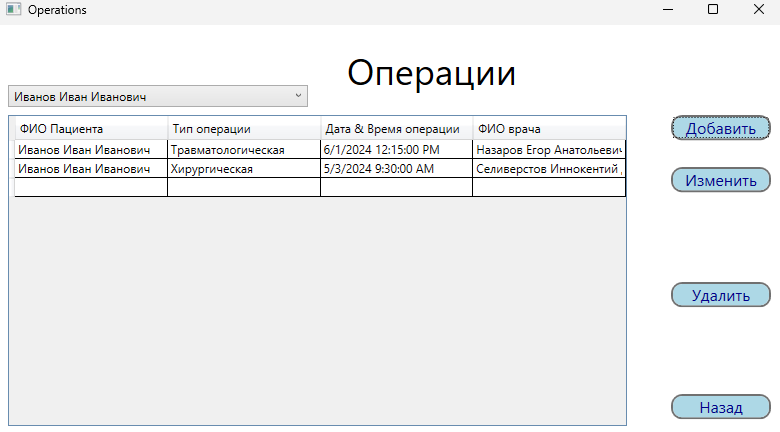


Рисунок В.16 – Результат добавления операции

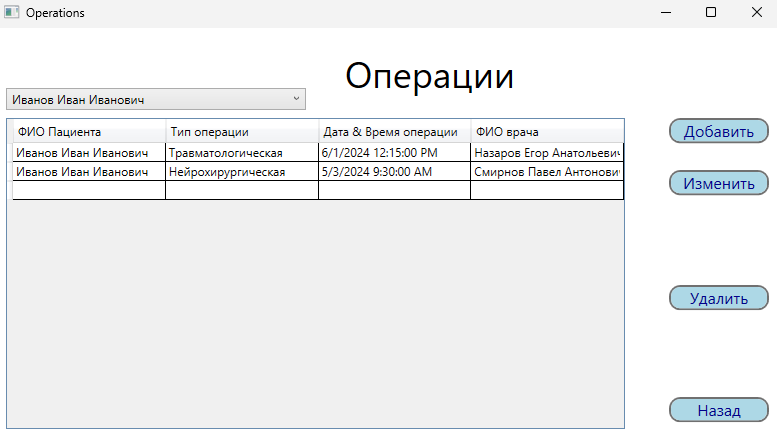


Рисунок В.17 – Результат изменения операции

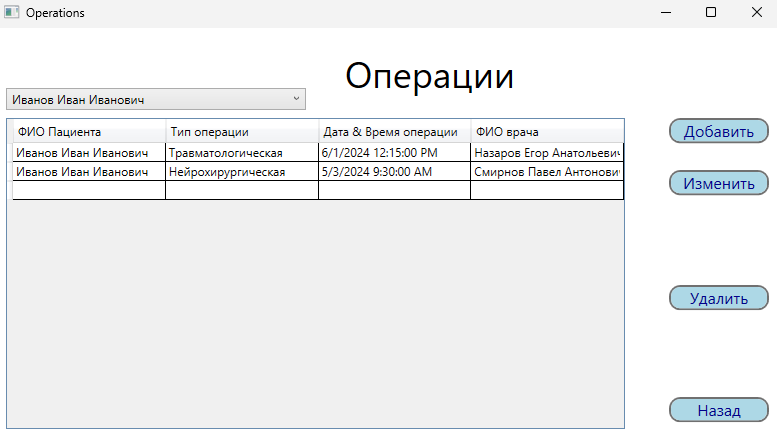


Рисунок В.18 – Результат удаление операции

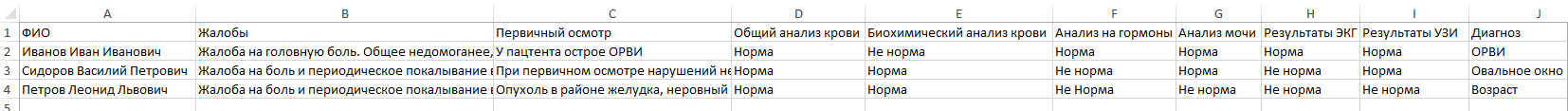


Рисунок В.19 – Результат создания отчета с формы «Карточки»