МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Новосибирский национальный исследовательский государственный университет»

(Новосибирский государственный университет, НГУ)

Структурное подразделение Новосибирского государственного университета – Высший колледж информатики Университета (ВКИ НГУ)

КАФЕДРА ИНФОРМАТИКИ

Отчет по курсовому проекту

ПМ 01. Разработка программных модулей

**«Документооборот федеральной сети больниц»**

|  |  |
| --- | --- |
| Руководитель | Дувалин А.Е.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2024 г. |
| Студенты 3 курса  гр. 2207 в1 | Злобин Д.С.  Зубенко А.Г.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2024 г. |

Новосибирск 2024

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ 4](#_gjdgxs)

[ВВЕДЕНИЕ 5](#_30j0zll)

[1 ПОСТАНОВКА ЗАДАЧИ ВКР 6](#_1fob9te)

[1.1 Бизнес-требования 7](#_3znysh7)

[1.2 Пользовательские требования 7](#_2et92p0)

[1.3 Системные требования 8](#_tyjcwt)

[1.4 Требования к графическому пользовательскому интерфейсу 8](#_3dy6vkm)

[1.5 План-график выполнения ВКР 9](#_1t3h5sf)

[2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ 10](#_4d34og8)

[2.1 Описание предметной области задачи ВКР 10](#_2s8eyo1)

[2.1.1 Информационные объекты предметной области и взаимосвязи между ними 10](#_17dp8vu)

[2.1.2 Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ) 11](#_3rdcrjn)

[2.1.3 Методы работы с информационными объектами предметной области 12](#_26in1rg)

[2.1.3.1 Используемые математические модели 12](#_lnxbz9)

[2.1.3.2 Применяемые программные технологии, основанные на математических моделях 12](#_35nkun2)

[2.1.4 Обзор существующих программных реализаций решения задачи 13](#_1ksv4uv)

[2.1.5 Концептуальное обоснование разработки 13](#_44sinio)

[2.2 Классы и характеристики пользователей 13](#_2jxsxqh)

[2.3 Функциональные требования 14](#_z337ya)

[2.3.1 Определение функциональных возможностей ПС (ПМ) 14](#_3j2qqm3)

[2.3.2 Описание прецедентов 14](#_1y810tw)

[2.4 Нефункциональные требования 15](#_4i7ojhp)

[3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ 17](#_2xcytpi)

[3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки 17](#_1ci93xb)

[3.2 Характеристика выбранных программных сред и средств 17](#_3whwml4)

[4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 18](#_2bn6wsx)

[4.1 Этапы реализации ПС (ПМ) 18](#_qsh70q)

[4.2 Пользовательский интерфейс ПС (ПМ) 18](#_3as4poj)

[4.2.1 Взаимодействие пользователей с ПС (ПМ) 18](#_1pxezwc)

[4.2.2 Проектирование пользовательских сценариев 18](#_49x2ik5)

[4.2.3 Определение операций пользователей 19](#_2p2csry)

[4.2.4 Составление функциональных блоков 19](#_147n2zr)

[4.2.5 Проектирование структуры экранов ПС (ПМ) и схемы навигации 20](#_3o7alnk)

[4.2.6 Низкоуровневое проектирование 20](#_23ckvvd)

[4.3 Входные, выходные и промежуточные данные 20](#_ihv636)

[4.4 Разработка базы данных, реализуемой в рамках ПС (ПМ) 21](#_32hioqz)

[4.5 Алгоритмы реализации используемых математических моделей 22](#_1hmsyys)

[4.6 Алгоритмы использования применяемых программных технологий 23](#_41mghml)

[4.7 Архитектура и схема функционирования ПС (ПМ) 23](#_2grqrue)

[5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ 25](#_vx1227)

[5.1 План тестирования 25](#_3fwokq0)

[5.2 Результаты тестирования 25](#_1v1yuxt)

[5.3 Оптимизация ПС (ПМ) 25](#_4f1mdlm)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_2u6wntf)

[ЗАКЛЮЧЕНИЕ 27](#_19c6y18)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 29](#_3tbugp1)

[ПРИЛОЖЕНИЯ 30](#_28h4qwu)

[Приложение А 30](#_nmf14n)

[Приложение Б 30](#_37m2jsg)

[Приложение В 30](#_1mrcu09)

[Приложение Г 30](#_46r0co2)

[Приложение Д 30](#_2lwamvv)

[Приложение Е 31](#_111kx3o)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

Специальные термины:

1. ПЗ – проектная документация
2. ЭДО – электронный документооборот
3. КУ – контрольная установка
4. МД – медицинская документация
5. АИС – автоматизированная информационная система
6. СЭД – система электронного документооборота
7. ФГИС – Федеральная государственная информационная система
8. ЛПУ – лечебно-профилактическое учреждение
9. КС – клиническая ситуация
10. ИПП – индивидуальный план пациента
11. Документооборот – процесс создания, обработки, хранения и передачи документов.
12. Медицинская карта – основной документ, содержащий информацию о состоянии здоровья пациента.
13. Приказ – официальный документ, регулирующий внутренние процессы в учреждении.
14. Протокол – документ, фиксирующий результаты обследований, операций и других медицинских манипуляций.
15. Аудит документации – проверка полноты и правильности ведения медицинских документов.
16. Конфиденциальность данных – соблюдение тайны личной информации пациента.
17. Архивирование – процесс хранения документов для дальнейшего использования или проверки.

ВВЕДЕНИЕ

Документооборот в больницах представляет собой критически важный процесс, обеспечивающий эффективное взаимодействие между различными подразделениями медицинского учреждения. В условиях современного здравоохранения, где объем информации постоянно растет, а требования к качеству обслуживания пациентов становятся все более высокими, проблемы, связанные с управлением медицинской документацией, становятся особенно актуальными. Неправильное или неэффективное ведение документооборота может привести к задержкам в предоставлении медицинских услуг, ошибкам в лечении и даже угрозе безопасности пациентов.

Актуальность решения данных проблем обусловлена необходимостью повышения качества медицинского обслуживания и оптимизации процессов внутри больницы. Современные технологии, такие как электронный документооборот (ЭДО), предлагают эффективные решения для автоматизации и упрощения процессов обработки медицинских документов. Однако многие учреждения сталкиваются с трудностями при внедрении таких систем из-за недостатка интеграции с существующими процессами и недостаточной подготовки персонала.

Разработка программных решений для улучшения документооборота в больницах имеет практическую ценность. Она может привести к снижению времени на обработку документов, уменьшению числа ошибок и повышению общей эффективности работы медицинского учреждения. Важно отметить, что существующие наработки в области ЭДО не всегда учитывают специфику работы лечебно-профилактических учреждений. Например, многие системы не обеспечивают необходимую гибкость для адаптации под индивидуальные процессы конкретной больницы или не учитывают требования законодательства в области защиты персональных данных.

Целью данного исследования является разработка комплексного решения для автоматизации документооборота в больнице, которое будет включать как программную, так и техническую части. Задачи исследования включают анализ существующих систем ЭДО, выявление их недостатков, а также разработку и тестирование нового программного обеспечения, способного эффективно интегрироваться в текущие процессы больницы.

Объектом исследования является система документооборота в лечебно-профилактических учреждениях, а выбор материала для исследования обоснован необходимостью создания решения, отвечающего требованиям современного здравоохранения и способствующего повышению качества медицинских услуг. В ходе работы будут использованы методы системного анализа, проектирования и тестирования программного обеспечения.

1 ПОСТАНОВКА ЗАДАЧИ ВКР

Разработка программного обеспечения для автоматизации документооборота в больнице. Это решение направлено на оптимизацию процессов обработки медицинских документов, повышение качества обслуживания пациентов и снижение вероятности ошибок. Задача является частью более сложной системы управления медицинским учреждением, интегрирующей различные компоненты, такие как электронные медицинские карты, системы учета пациентов и финансовые модули.

* 1. **1.1 Бизнес-требования**

1. Сократить время обработки медицинских документов.
2. Уменьшить количество ошибок при работе с документами.
3. Повысить качество обслуживания пациентов за счет быстрого доступа к информации.
4. Обеспечить прозрачность процессов документооборота для руководства учреждения.
5. Упростить взаимодействие между различными подразделениями больницы.

**1.2 Пользовательские требования**

1. Создавать, редактировать и хранить медицинские документы в удобном формате.

2. Получать доступ к информации о пациентах и их истории болезни через интуитивно понятный интерфейс.

3. Генерировать отчеты по различным параметрам (количество обработанных документов, время обработки и т.д.).

4. Использовать систему в рамках более крупной интегрированной платформы управления больницей, что обеспечит совместимость с существующими системами учета и отчетности.

**1.3 Системные требования**

1. Модуль создания и редактирования документов:

• Обеспечивает возможность работы с различными типами медицинских документов (например, рецепты, направления, выписки).

• Поддерживает функции авто форматирования и проверки на ошибки.

1. Модуль хранения данных:

• Отвечает за безопасное хранение и доступ к медицинской информации.

• Реализует механизмы резервного копирования и восстановления данных.

1. Модуль отчетности:

• Генерирует отчеты по запросу пользователей.

• Позволяет настраивать параметры отчетов (по времени, типам документов и т.д.).

1. Модуль интеграции:

• Обеспечивает взаимодействие с другими системами больницы (например, ЭДО, системы учета пациентов).

• Поддерживает стандартные протоколы обмена данными для упрощения интеграции.

**1.4 Требования к графическому пользовательскому интерфейсу**

Характеристики интерфейса:

1. Интуитивно понятный интерфейс: Простая навигация и доступ к основным функциям.
2. Поиск и фильтрация: Возможность быстрого поиска документов по различным критериям.
3. Обучающие материалы: Доступ к справочной информации и инструкциям по использованию системы.
4. Современный дизайн: Соответствие стандартам UX/UI для обеспечения удобства взаимодействия пользователей с системой.
   1. **План-график выполнения ВКР**
5. Анализ требований:

• Сбор информации о текущих процессах документооборота.

• Интервью с пользователями (врачами, медсестрами, администрацией).

• Формирование списка функциональных и нефункциональных требований.

1. Проектирование системы:

• Разработка архитектуры системы.

• Создание прототипов интерфейса.

• Определение баз данных и их структуры.

1. Разработка программного обеспечения:

• Кодирование основных модулей системы.

• Интеграция с существующими системами (если требуется).

• Проведение тестирования (модульное, интеграционное).

1. Тестирование:

• Проведение функционального тестирования.

• Проведение пользовательского тестирования.

• Исправление найденных ошибок.

1. Документация:

• Разработка пользовательских инструкций.

• Подготовка технической документации (архитектура, описание модулей).

• Создание отчетов о тестировании.

1. Внедрение системы:

• Установка программного обеспечения на серверах больницы.

• Обучение пользователей работе с новой системой.

• Сбор обратной связи от пользователей после внедрения.

1. Поддержка и сопровождение:

• Решение возникающих проблем и ошибок.

• Обновление системы по мере необходимости.

• Поддержка пользователей.

Контрольные точки проекта:

1. Завершение анализа требований.
2. Завершение проектирования системы.
3. Завершение разработки программного обеспечения.
4. Завершение тестирования.

5. Завершение документации.

6. Успешное внедрение системы.

7. Оценка работы системы и сбор обратной связи.

2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

* 1. 2.1 Описание предметной области задачи ВКР

Документооборот в больнице представляет собой комплекс процессов, связанных с созданием, обработкой, хранением и передачей медицинских и административных документов. Эти процессы охватывают различные аспекты работы учреждения, включая взаимодействие между медицинским персоналом, пациентами и административными структурами. Эффективный документооборот является критически важным для обеспечения качества медицинского обслуживания, снижения ошибок и повышения уровня удовлетворенности пациентов.

***2.1.1 Информационные объекты предметной области и взаимосвязи между ними***

Информационные объекты

**Пациенты**:

Информационные объекты, содержащие данные о пациентах, такие как имя, дата рождения, история болезни, результаты анализов и т.д.

Взаимосвязи: пациенты связаны с медицинскими записями, назначениями, счетами и т.д.

**Медицинские записи**:

Документы, содержащие информацию о диагнозах, лечении, назначениях и результатах обследований.

Взаимосвязи: медицинские записи связаны с пациентами, врачами и медицинскими процедурами.

**Врачи и медицинский персонал**:

Информационные объекты, представляющие врачей, медсестер и другой медицинский персонал, включая их квалификацию, специализацию и контактные данные.

Взаимосвязи: врачи связаны с пациентами, медицинскими записями и назначениями.

**Медицинские процедуры**:

Описание различных медицинских процедур, включая операции, обследования и лечение.

Взаимосвязи: процедуры связаны с медицинскими записями и пациентами.

**Лабораторные результаты**:

Документы, содержащие результаты анализов и исследований, проведенных в лаборатории.

Взаимосвязи: лабораторные результаты связаны с медицинскими записями и пациентами.

**Счета и финансовые документы**:

Информационные объекты, связанные с финансовыми аспектами лечения, включая счета, страховые документы и платежи.

Взаимосвязи: счета связаны с пациентами и медицинскими записями.

**Электронные документы**:

Все документы, которые хранятся в электронном виде, включая электронные медицинские записи (EMR), электронные рецепты и т.д.

Взаимосвязи: электронные документы связаны с другими информационными объектами, такими как пациенты и медицинские записи.

Взаимосвязи между информационными объектами

**Создание и обновление**:

При каждом новом обращении пациента в больницу создаются или обновляются медицинские записи, что приводит к изменению состояния информационных объектов.

**Передача информации**:

Врачи и медицинский персонал обмениваются информацией о пациентах, что требует передачи медицинских записей и лабораторных результатов.

**Хранение данных**:

Все информационные объекты должны храниться в безопасных и доступных системах, что требует наличия эффективных информационных систем и технологий.

**Анализ и отчетность**:

Информационные объекты могут быть использованы для анализа данных о пациентах, эффективности лечения и финансовых показателях, что требует интеграции различных информационных систем.

2.1.2 Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ)

1. Пользователи системы

**Врачи и медицинский персонал**: Основные пользователи, которые будут взаимодействовать с медицинскими записями, назначениями и результатами анализов.

**Администраторы**: Ответственные за управление данными пациентов, запись на прием и организацию документооборота.

**Бухгалтерия**: обрабатывает финансовые документы, счета и страховые выплаты.

**Пациенты**: могут получать доступ к своим медицинским записям и результатам анализов.

2. Информационные потребности

**Доступ к медицинским записям**: Врачи должны иметь возможность быстро находить и просматривать медицинские записи пациентов, включая историю болезни, результаты анализов и назначения.

**Управление записями на прием**: Администраторы должны иметь возможность управлять расписанием врачей и записывать пациентов на прием.

**Финансовая информация**: Бухгалтерия должна иметь доступ к счетам, страховым документам и платежам для обработки финансовых операций.

**Информация для пациентов**: Пациенты должны иметь возможность получать доступ к своим медицинским записям и результатам анализов через портал или мобильное приложение.

3. Функциональные потребности

**Создание и редактирование медицинских записей**: Врачи должны иметь возможность создавать новые записи и редактировать существующие, добавляя информацию о диагнозах, лечении и результатах обследований.

**Поиск и фильтрация данных**: Система должна предоставлять функции поиска и фильтрации для быстрого доступа к необходимой информации (например, по имени пациента, дате обращения или типу процедуры).

**Генерация отчетов**: Система должна позволять пользователям генерировать отчеты о состоянии пациентов, финансовых показателях и эффективности лечения.

**Уведомления и напоминания**: Система должна отправлять уведомления врачам и пациентам о предстоящих приемах, результатах анализов и других важных событиях.

4. Примеры исходной и получаемой информации

**Исходная информация**:

Данные о пациенте: имя, дата рождения, контактная информация, история болезни.

Результаты анализов: данные лабораторных исследований, рентгеновские снимки, заключения врачей.

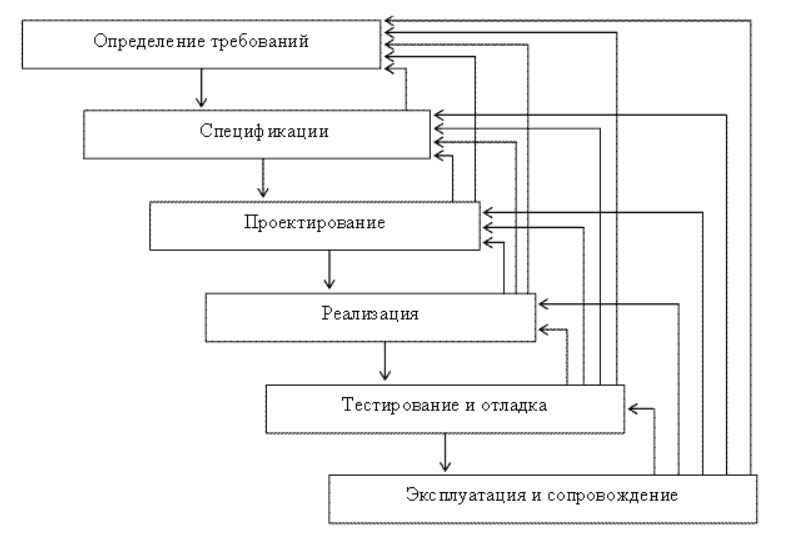
Финансовые документы: счета, страховые полисы, платежные квитанции.

**Получаемая информация**:

Обновленные медицинские записи: полная история болезни пациента, включая все назначения и результаты анализов.

Отчеты о состоянии здоровья: сводная информация о лечении, включая эффективность назначенных процедур и анализов.

Финансовые отчеты: информация о платежах, задолженностях и страховых выплатах.

* + 1. ***2.1.3 Методы работы с информационными объектами предметной области***
       1. ***2.1.3.1 Этапы разработки ПМ***
       2. Мы взяли классическую модель разработки ПМ представленную выше
       3. ***2.1.3.2 Применяемые программные технологии, основанные на математических моделях***
    2. ***2.1.4 Обзор существующих программных реализаций решения задачи***

#### 1. ****Системы электронного документооборота (СЭД)****

**Примеры:**

**1С: Документооборот**

**Электронный архив документов (ЭАД)**

**Достоинства:**

Интуитивно понятный интерфейс.

Возможность интеграции с другими системами (например, с медицинскими информационными системами).

Поддержка различных форматов документов.

**Недостатки:**

Высокая стоимость лицензий и внедрения.

Необходимость обучения персонала.

Ограниченная функциональность в части специфических медицинских процессов.

#### 2. ****Медицинские информационные системы (МИС)****

**Примеры:**

* **МедСистема**
* **Гемотест**

**Достоинства:**

* Комплексный подход к управлению медицинскими данными.
* Автоматизация процессов от записи пациентов до ведения медицинской документации.
* Возможность анализа данных и формирования отчетов.

**Недостатки:**

* Сложность в настройке и интеграции с существующими системами.
* Часто требует значительных затрат на техническую поддержку.
* Ограниченная гибкость в настройке под специфические нужды учреждения.

#### 3. ****Облачные решения для документооборота****

**Примеры:**

* **DocuWare**
* **Google Workspace для бизнеса**

**Достоинства:**

* Доступность из любой точки с интернетом.
* Низкие первоначальные затраты и гибкая модель оплаты.
* Обновления и поддержка со стороны провайдера.

**Недостатки:**

* Зависимость от интернет-соединения.
* Вопросы безопасности и конфиденциальности данных.
* Ограниченные возможности кастомизации.

### Сравнительный анализ аналогов

| **Параметр** | **1С: ДОКУМЕНТООБОРОТ** | **МедСистема** | **DocuWare** |
| --- | --- | --- | --- |
| Стоимость | Высокая | Средняя | Низкая |
| Интуитивность интерфейса | Высокая | Средняя | Высокая |
| Интеграция с другими системами | Хорошая | Отличная | Ограниченная |
| Гибкость настройки | Ограниченная | Высокая | Средняя |
| Безопасность данных | Средняя | Высокая | Средняя |

### Вывод о степени новизны разрабатываемого программного средства

На основе проведенного сравнительного анализа можно сделать вывод, что разрабатываемое в рамках ВКР программное средство должно сочетать в себе лучшие практики существующих решений, а также учитывать их недостатки. Основное внимание следует уделить:

* Упрощению интеграции с существующими системами.
* Повышению гибкости настройки под специфические нужды медицинских учреждений.
* Обеспечению высокого уровня безопасности и конфиденциальности данных.

Таким образом, новизна разрабатываемого решения будет заключаться в создании более адаптивного и безопасного инструмента для автоматизации документооборота в больницах, что позволит значительно повысить эффективность работы медицинского персонала и улучшить качество обслуживания пациентов.

* + 1. ***2.1.5 Концептуальное обоснование разработки***

В современном здравоохранении эффективный документооборот является ключевым фактором, влияющим на качество медицинского обслуживания и организацию работы медицинских учреждений. Существующие системы документооборота в больницах часто сталкиваются с рядом проблем, которые требуют решения.

#### Проблемы существующих систем

1. **Низкая интеграция**: Многие системы не обеспечивают должной интеграции с другими медицинскими информационными системами, что приводит к дублированию данных и увеличению времени на обработку информации.
2. **Сложность использования**: Существующие решения часто имеют сложные интерфейсы, что требует значительных затрат времени на обучение персонала и может приводить к ошибкам в работе.
3. **Ограниченная функциональность**: Многие системы не учитывают специфические процессы, характерные для медицинских учреждений, что ограничивает их применение и эффективность.
4. **Проблемы с безопасностью**: В условиях постоянного роста угроз кибербезопасности, многие системы не обеспечивают достаточный уровень защиты данных пациентов, что может привести к утечкам конфиденциальной информации.

#### Концепция оригинального решения

На основе анализа существующих проблем, предлагается разработка инновационного программного средства для автоматизации документооборота в больницах, которое будет включать в себя следующие ключевые элементы:

1. **Интеграция с существующими системами**: Разработка модульной архитектуры, позволяющей легко интегрировать новое решение с уже используемыми медицинскими информационными системами. Это позволит избежать дублирования данных и упростит процесс обмена информацией.
2. **Интуитивно понятный интерфейс**: Создание пользовательского интерфейса, основанного на принципах UX/UI дизайна, что обеспечит легкость в использовании и минимизирует время на обучение персонала.
3. **Гибкость и адаптивность**: Внедрение системы, которая будет легко настраиваться под специфические нужды каждого медицинского учреждения, включая возможность добавления новых функций и модулей по мере необходимости.
4. **Улучшенная безопасность данных**: Реализация многоуровневой системы защиты данных, включая шифрование, аутентификацию и регулярные аудиты безопасности, что обеспечит высокий уровень защиты конфиденциальной информации пациентов.
5. **Аналитические инструменты**: Внедрение встроенных аналитических инструментов для мониторинга и анализа документооборота, что позволит выявлять узкие места и оптимизировать процессы.
   1. **2.2 Классы и характеристики пользователей**
   2. При анализе информации формируют профили пользователей. Профилями называют описание главных категорий пользователя. Потенциальными пользователями в этой работе является Врач.

Таблица 2.2.1 – Описание пользовательских ролей

|  |  |
| --- | --- |
| Пользователь | Врач |
| Социальные  характеристики | Русскоязычные  Низкий уровень владения компьютером  Знание лечебного дела |
| Мотивационная среда | Производственная необходимость  Низкая мотивация к обучению |
| Навыки и умения | Прошли предварительный тренинг с программой |
| Требования к ПО | Возможность использования в локальной сети  Возможность просмотра и изменения информации о пациентах, врачах, медицинских карт, записей на приём, наличии препаратов в аптеке |
| Задачи пользователя | Назначение рецептов  Запись пациентов на приём |
| Рабочая среда | Стандартизированные ПК, локальная сеть |

## 2.3 Функциональные требования

### 2.3.1 Определение функциональных возможностей

На основе анализа предметной области можно выделить функциональность приложения, исходя из целей и задач пользователя.

Перечень функций, необходимых в приложении для пользовательского профиля Врач:

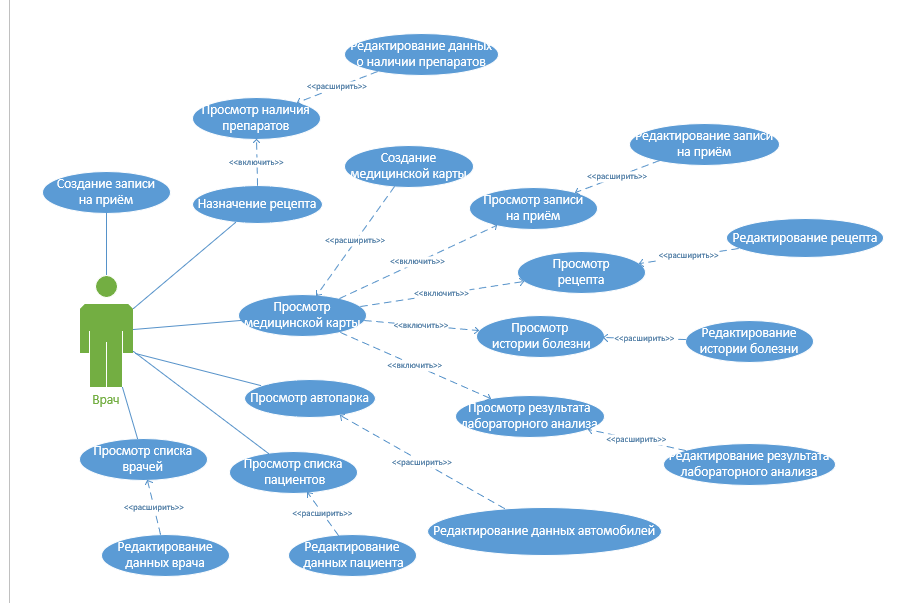
* Просмотр списка пациентов
* Редактирование данных пациента;
* Просмотр списка врачей;
* Редактирование данных врача;
* Создание записи на приём;
* Назначение рецепта;
* Просмотр наличия препаратов;
* Редактирование данных о наличии препаратов;
* Просмотр медицинской карты;
* Создание медицинской карты;
* Просмотр записи на приём;
* Редактирование записи на приём;
* Просмотр рецепта;
* Редактирование рецепта;
* Просмотр истории болезни;
* Редактирование истории болезни;
* Просмотр результата лабораторного анализа;
* Редактирование результата лабораторного анализа;
* Просмотр списка автомобилей врачей;
* Редактирование данных автомобилей.

На основе анализа предметной области и пользовательских требований к разрабатываемой ИС составлен реестр юскейсов для выделенных пользовательских ролей (таблица 2.3.1.1).

Таблица 2.3.1.1 – Реестр юскейсов

|  |  |  |  |
| --- | --- | --- | --- |
| Пользовательская роль | Потребность пользователя | Use Case | Сценарий  использования |
| Врач | Записать пациента на приём | Создание записи на  приём | * Создание записи на приём |
| Назначить рецепт пациенту | Назначение рецепта | * Назначение рецепта * Просмотр наличия препаратов |
| Просмотр списка пациентов / редактирование данных пациента | Просмотр списка пациентов | * Просмотр списка пациентов * Редактирование данных пациента |
| Просмотр списка врачей / редактирование данных врача | Просмотр списка врачей | * Просмотр списка врачей * Редактирование данных врача |
|  | Просмотр списка автомобилей врачей/редактирование данных автомобилей | Просмотр списка автомобилей | * Просмотр списка автомобилей * Редактирование списка автомобилей |

На рисунке 2.3.2.1 представлена UML-диаграмма вариантов использования ИС



1. Рисунок 2.3.2.1 – Диаграмма вариантов использования ИС

### 2.3.2 Описание прецедентов

Описания прецедентов представлены в таблицах (2.3.2.1 - 2.3.2.5).

Таблица 2.3.2.1 – Описание прецедента «Создание записи на приём»

|  |  |
| --- | --- |
| Название прецедента | Создание записи на приём |
| Исполнитель | Врач |
| Цель | Записать пациента на приём |
| Основной успешный сценарий | Врач нажимает на кнопку «Добавить запись на приём». Поля формы становятся пустыми для заполнения. Вносит сведения о записи (номер записи, номер пациента, номер врача, номер медицинской карты, дата и время приёма, причина). |

Таблица 2.3.2.2 – Описание прецедента «Назначение рецепта»

|  |  |
| --- | --- |
| Название прецедента | Назначение рецепта |
| Исполнитель | Врач |
| Цель | Назначить пациенту рецепт |
| Основной успешный сценарий | Врач нажимает на кнопку «Назначить рецепт». Поля формы становятся пустыми для заполнения. Вносит сведения о рецепте (номер рецепта, номер пациента, номер врача, номер препарата, номер медицинской карты, дата выписки рецепта, продолжительность приёма). |

Таблица 2.3.2.3 – Описание прецедента «Просмотр медицинской карты»

|  |  |
| --- | --- |
| Название прецедента | Просмотр медицинской карты |
| Исполнитель | Врач |
| Цель | Просмотр медицинской карты пациента |
| Основной успешный сценарий | Врач нажимает на кнопку «Медицинские карты». Открывается форма с данными медицинской карты. |

Таблица 2.3.2.4 – Описание прецедента «Просмотр списка пациентов»

|  |  |
| --- | --- |
| Название прецедента | Просмотр списка пациентов |
| Исполнитель | Врач |
| Цель | Просмотр списка пациентов |
| Основной успешный сценарий | Врач нажимает на кнопку «Список пациентов». Открывается форма со списком пациентов и их данными. |

Таблица 2.3.2.5 – Описание прецедента «Просмотр списка врачей»

|  |  |
| --- | --- |
| Название прецедента | Просмотр списка врачей |
| Исполнитель | Врач |
| Цель | Просмотр списка врачей |
| Основной успешный сценарий | Врач нажимает на кнопку «Список врачей». Открывается форма со списком врачей и их данными. |

## 2.4 **Нефункциональные требования**

***2.4.1 Требования к программному обеспечению***

Для корректного функционирования ИС необходим компьютер с:

* ОС Windows 7 или более поздней версией;

***2.4.2 Требования к аппаратному обеспечению***

В плане аппаратного обеспечения необходимо соблюдать следующие требования:

* количество свободного места на жестком диске не менее 4 ГБ;
* количество оперативной памяти не менее 4 ГБ;
* количество ядер процессора не менее двух;
* тактовая частота процессора не менее 2.5 ГГц.

**3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ**

В данном разделе проводится анализ программных сред и средств разработки программной системы, а также обоснование выбора программного инструментария. Рассматриваются такие средства, как Visual Studio, C#, XAML, WPF, Git и базы данных.

**3.1 Сравнительный анализ имеющихся возможностей по выбору средств разработки**

При выборе средств разработки важно учитывать несколько факторов, таких как требования к производительности, простота использования, доступность ресурсов и поддержка сообщества. Основные средства и технологии, которые будут рассмотрены:

- Visual Studio: интегрированная среда разработки (IDE) от Microsoft, поддерживающая множество языков программирования. Подходит для создания приложений на языках C#, VB.NET, C++ и других.

- C#: современный объектно-ориентированный язык программирования, разработанный Microsoft. Он обладает простым синтаксисом, широкими возможностями и мощной поддержкой для разработки приложений на платформе .NET.

- XAML: язык разметки, используемый для описания пользовательских интерфейсов в WPF и других технологиях .NET. Позволяет разделять логику приложения и его представление.

- WPF: Windows Presentation Foundation — технология для создания десктопных приложений с современными интерфейсами. Предоставляет мощные возможности для создания графически насыщенных интерфейсов.

- Git: система контроля версий, позволяющая отслеживать изменения в коде и совместно работать над проектами. Обеспечивает простое управление версиями и возможность отката к предыдущим состояниям кода.

- База данных (например, SQL Server): реляционная система управления базами данных (СУБД), предоставляющая возможности хранения и управления данными.

**3.2 Характеристика выбранных программных сред и средств**

На основе проведенного анализа, для разработки программной системы было выбрано следующее программное обеспечение:

1. Visual Studio:

Visual Studio является одной из самых популярных интегрированных сред разработки (IDE) для Windows. Она предоставляет разработчикам мощные инструменты для написания, отладки и тестирования приложений. Поддержка различных языков программирования делает ее универсальным инструментом, а наличие большого количества расширений позволяет адаптировать среду под конкретные нужды.

2. C#:

C# — это современный язык программирования, который идеально подходит для разработки приложений под платформу .NET. Его объектно-ориентированная природа позволяет создавать масштабируемые и поддерживаемые решения. Он также активно используется для создания приложений с графическим интерфейсом благодаря интеграции с WPF.

3. XAML:

XAML используется для определения пользовательских интерфейсов в WPF-приложениях. Этот язык разметки позволяет разработчикам легко создавать сложные интерфейсы, используя декларативный подход. Он также поддерживает привязку данных, что упрощает взаимодействие между логикой приложения и его представлением.

4. WPF:

WPF предоставляет мощные средства для создания графически насыщенных приложений. Он поддерживает такие функции, как анимация, стилизация и шаблоны, что позволяет разработчикам создавать привлекательные интерфейсы. WPF также обеспечивает поддержку MVVM (Model-View-ViewModel), что упрощает организацию кода.

5. Git:

Git является стандартом де-факто для контроля версий в современных проектах. Его возможности по управлению версиями позволяют командам эффективно работать над проектами, минимизируя риски потери данных и конфликтов при слиянии изменений

6. SQL:

Для работы с базами данных было выбрано использование SQL, так как он предоставляет мощные инструменты для управления данными, выполнения запросов и обеспечения целостности данных. SQL является стандартом для большинства реляционных баз данных, что обеспечивает высокую совместимость и производительность.

**4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

**4.1 Этапы реализации программной системы (ПС) или программного продукта (ПП)**

Реализация программной системы включает несколько последовательных этапов, на каждом из которых выполняются специфические задачи. Ниже представлены эти этапы в словесной форме.

Словесное описание этапов реализации

1. Определение требований:

- Сбор и анализ требований от заинтересованных сторон (заказчиков и пользователей), чтобы понять их потребности и ожидания от системы.

2. Проектирование системы:

- Разработка архитектуры программной системы, включая структурное и детальное проектирование. Определение компонентов и их взаимодействия.

3. Выбор инструментов разработки:

- Определение технологий, языков программирования, средств разработки и библиотек, которые будут использоваться для реализации системы.

4. Разработка:

- Написание исходного кода для всех компонент системы в соответствии с проектной документацией. Также включает написание тестов и документации.

5. Тестирование:

- Проведение тестирования программного обеспечения для выявления ошибок и проверки его соответствия заявленным требованиям. Включает различные виды тестирования: модульное, интеграционное, системное.

6. Внедрение:

- Установка и настройка системы в рабочей среде; сеанс обучения пользователей и подготовка их к использованию системы.

7. Эксплуатация и поддержка:

- Запуск системы в эксплуатацию, выполнение мониторинга работы системы и устранение возникающих ошибок и проблем.

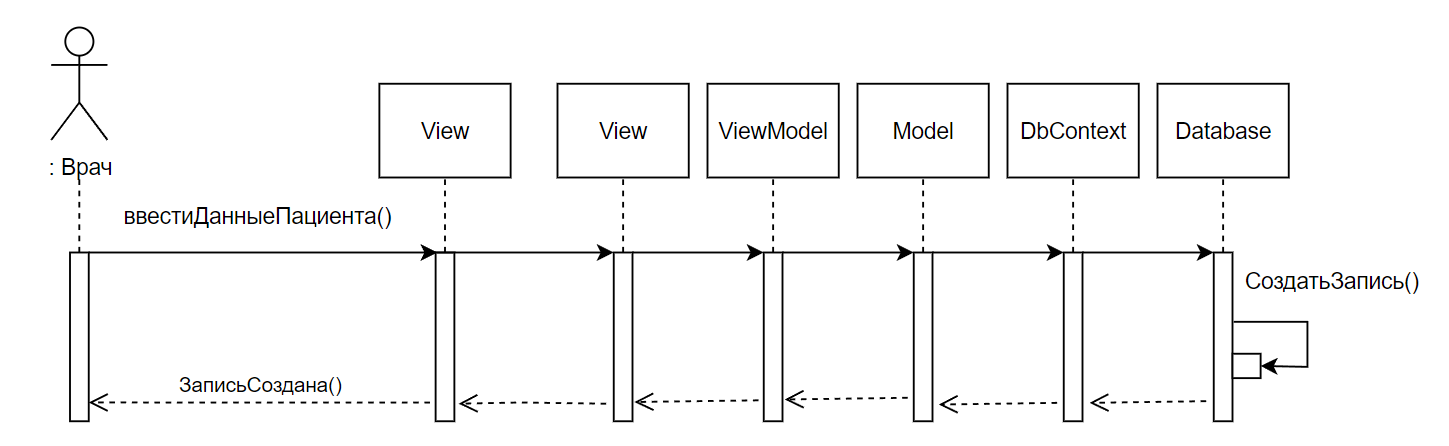
8. Обновление и улучшение:

- Регулярное обновление системы в соответствии с новыми требованиями, добавление нового функционала и улучшение производительности.

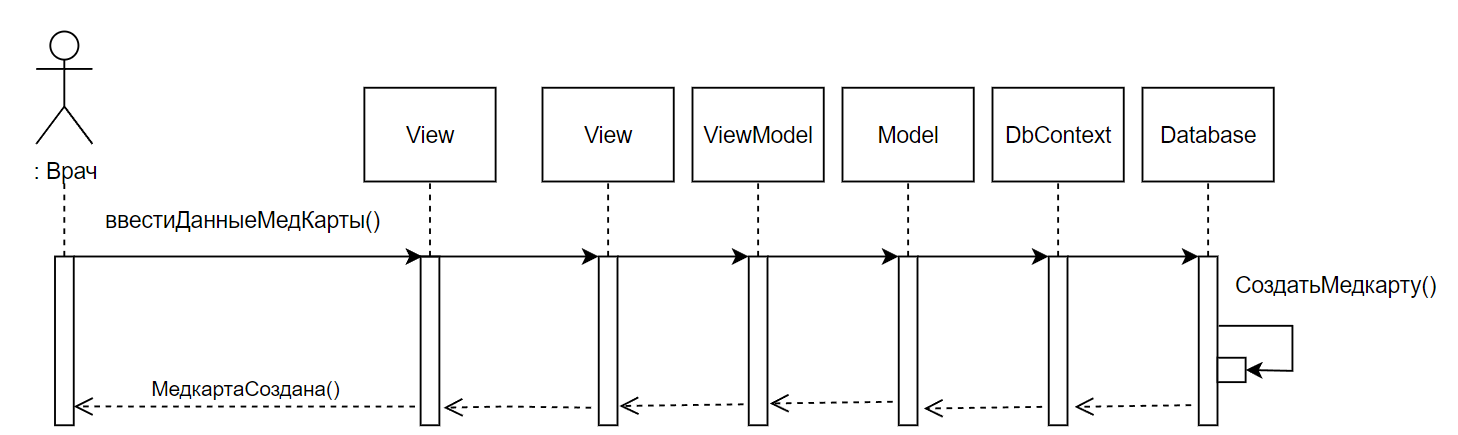
**4.2 Пользовательский интерфейс ПС (ПМ)**

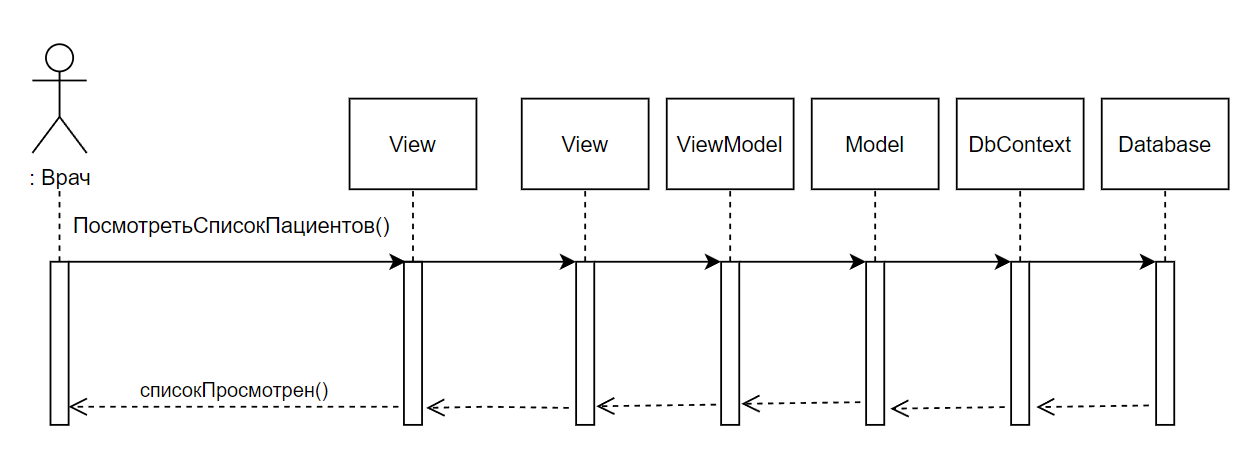
***4.2.1 Взаимодействие пользователей с ПС (ПМ)***

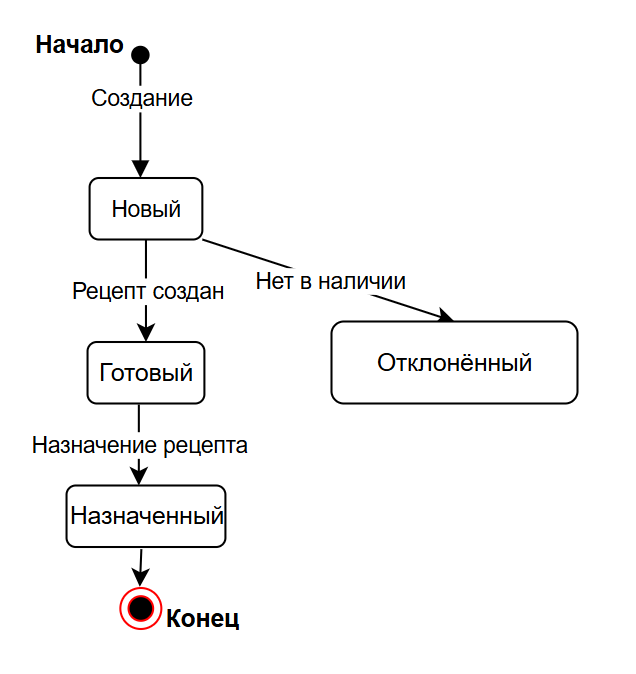
Взаимодействие пользователей с программной системой (ПС) можно описать с помощью различных моделей и диаграмм, которые дают представление о том, как пользователи выполняют действия в системе, какие прецеденты они инициируют, а также как система реагирует на эти действия. Разработаны три диаграммы последовательности представлены и одна диаграмма состояний, которые представлены на рисунках (4.2.1.1 – 4.2.1.4).



1. Рисунок 4.2.1.1 – Диаграмма последовательности «Создание записи на приём»



1. Рисунок 4.2.1.2 – Диаграмма последовательности «Создание медицинской карты»
2. 
3. Рисунок 4.2.1.3 – Диаграмма последовательности «Просмотр списка пациентов»



1. Рисунок 4.2.1.4 – Диаграмма состояний «Назначение рецепта»

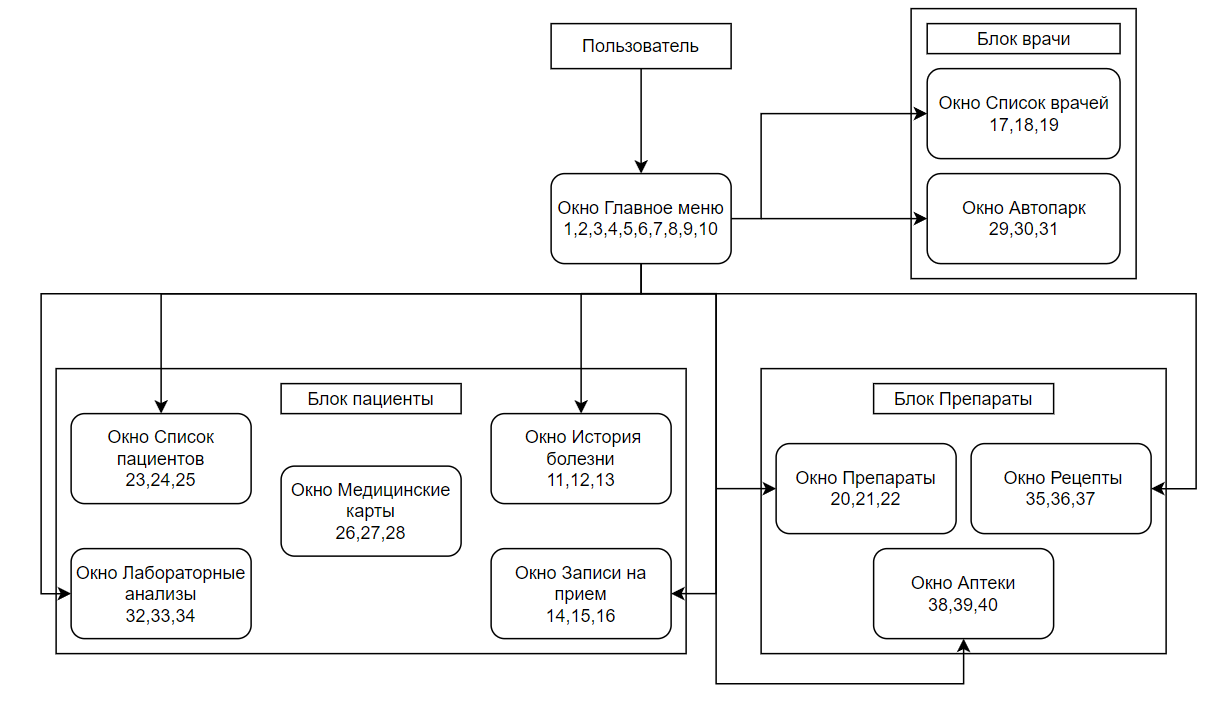
***4.2.2 Проектирование пользовательских сценариев***

Логика взаимодействия пользователей с системой реализована через форму главный экран, который содержит 10 кнопок. Проектирование сценариев предполагает стандартное поведение интерфейса: при успешном добавлении, изменении, удалении выводится сообщение о успехе. Иначе выводится сообщение об ошибке.

* + 1. ***4.2.3 Определение операций пользователей***
    2. Проанализировав функциональные возможности потенциальных пользователей, можно определить операции, которые они могут выполнять в рамках предложенного им приложения:
    - Просмотр данных истории болезни, врачей, пациентов, автомобилей, препаратов, рецептов, лабораторных анализов, записей о приёме, аптек, медицинских карт.
    - Добавление/изменение данных истории болезни, врачей, пациентов, автомобилей, препаратов, рецептов, лабораторных анализов, записей о приёме, аптек, медицинских карт.
    - Удаление данных истории болезни, врачей, пациентов, автомобилей, препаратов, рецептов, лабораторных анализов, записей о приёме, аптек, медицинских карт.

1. ***4.2.4 Составление функциональных блоков***
2. Функциональные блоки обеспечивают структурирование системы и помогают организовать её работу более эффективно. На этом можно выделить следующие функциональные блоки:

* Врачи: данный блок отвечает за список врачей и их автопарк.
* Пациенты: данный блок отвечает за список пациентов, их истории болезни, лабораторные анализы, записи на приём, медицинские карты.
* Препараты: данный блок отвечает за рецепты, препараты и аптеки.

1. ***4.2.5 Проектирование структуры экранов ПС (ПМ) и схемы навигации***
2. Проделав анализ предметной области, сформировав профили и сценарии действий пользователей и определив функциональность приложения, можно составить схему навигационной системы. Разработанная схема навигации представлена на рисунке 4.2.5.1.
   * 1. ******
3. Рисунок 4.2.5.1 – Схема навигации

В данной схеме три функциональных блока и десять основных экранных форм:

* Главное меню;

Пациенты:

* Список пациентов;
* Лабораторные анализы;
* Медицинские карты;
* История болезни;
* Записи на прием;

Врачи:

* Список врачей;
* Автопарк;

Препараты:

* Препараты;
* Рецепты;
* Аптеки.

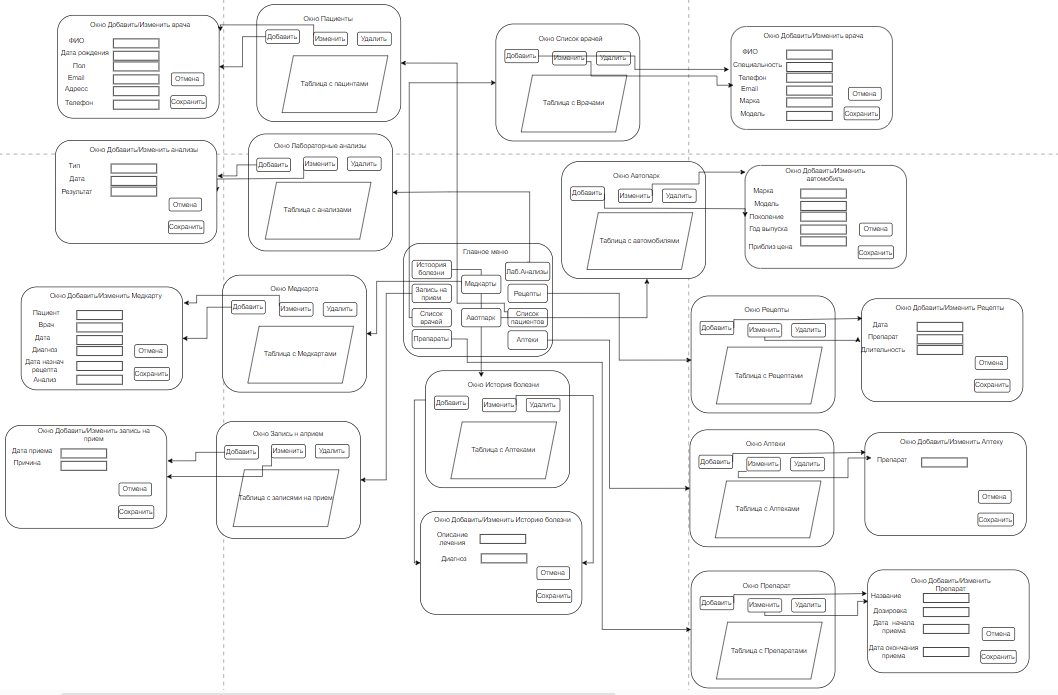
Данные функциональные блоки соответствуют работе пользователей с информацией:

* Открытие окна Список пациентов – 1;
* Открытие окна Запись на прием – 2;
* Открытие окна История болезни – 3;
* Открытие окна Лабораторные анализы – 4;
* Открытие окна Медицинская карта – 5;
* Открытие окна Список врачей – 6;
* Открытие окна Автопарк– 7;
* Открытие окна Препараты – 8;
* Открытие окна Рецепты– 9;
* Открытие окна Аптеки – 10;
* Добавление записи в историю болезни – 11;
* Изменение записи истории болезни – 12;
* Удаление записи истории болезни – 13;
* Добавление записи на прием– 14;
* Изменение записи на прием – 15;
* Удаление записи на прием – 16;
* Добавление записи в список врачей – 17;
* Изменение записи в списке врачей – 18;
* Удаление записи в списке врачей – 19;
* Добавление записи в список препаратов – 20;
* Изменение записи в списке препаратов – 21;
* Удаление записи в списке препаратов – 22;
* Добавление записи в список пациентов – 23;
* Изменение записи в списке пациентов – 24;
* Удаление записи в списке пациентов – 25;
* Добавление записи в медицинской карте – 26;
* Изменение записи в медицинской карте – 27;
* Удаление записи в медицинской карте – 28;
* Добавление записи в автопарк – 29;
* Изменение записи в автопарке – 30;
* Удаление записи в автопарке – 31;
* Добавление записи в список лабораторных анализов – 32;

Цифрами на рисунке указаны отдельные операции, выполняемые пользователями.

***4.2.6 Низкоуровневое проектирование***

На уровне экранов предусмотрены следующие элементы:

1. Экраны просмотров: Кнопки редактирования (Добавить, удалить, редактировать), таблицу для просмотра данных.
2. Экраны редактирования: Текстовые поля для ввода данных, и выпадающие списки для связанных таблиц.
   1. **4.3 Входные, выходные и промежуточные данные**
   2. **4.4 Разработка базы данных, реализуемой в рамках ПС (ПМ)**
3. При разработке базы данных была использована реляционная модель. На основании даталогического проектирования в SQL были созданы 10 таблиц (таблицы 4.4.1 - 4.4.10).
4. Таблица 4.4.1 – Описание полей таблицы PatientEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Name | nvarchar(100) | Имя |
| DateOfBirth | date | Дата рождения |
| Sex | nvarchar(100) | Пол |
| Address | nvarchar(100) | Адрес |
| PhoneNumber | nvarchar(100) | Номер телефона |
| Email | nvarchar(100) | Электронная почта |

1. Таблица 4.4.2 – Описание полей таблицы DoctorEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Name | nvarchar(100) | Имя |
| Speciality | nvarchar(100) | Специальность |
| PhoneNumber | nvarchar(100) | Номер телефона |
| Email | nvarchar(100) | Электронная почта |
| CarIDFK | int | Внешний ключ – автомобили доктор |

2. Таблица 4.4.3 – Описание полей таблицы AppointmentRecordEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Date | date | Дата записи на приём |
| Reason | nvarchar(255) | Причина |

1. Таблица 4.4.4 – Описание полей таблицы MedicalHistoryEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Diagnosis | nvarchar(100) | Диагноз |
| DescriptionOfTreatment | nvarchar(max) | Описание лечения |

1. Таблица 4.4.5 – Описание полей таблицы LaboratoryTestEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Type | nvarchar(100) | Тип анализа |
| Date | date | Дата анализа |
| Result | nvarchar(max) | Результаты |

1. Таблица 4.4.6 – Описание полей таблицы DrugEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Name | nvarchar(100) | Название |
| Dosage | nvarchar(50) | Дозировка |
| StartDate | date | Дата начала приёма |
| EndDate | date | Дата окончания приёма |

1. Таблица 4.4.7 – Описание полей таблицы RecipeEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| DrugIDFK | int | Внешний ключ - препарат |
| Date | date | Дата выписки рецепта |
| Duration | nvarchar(100) | Продолжительность приёма |

1. Таблица 4.4.8 – Описание полей таблицы DrugstoreEntity

|  |  |  |
| --- | --- | --- |
| 1. Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| DrugIDFK | int | Внешний ключ - препарат |

1. Таблица 4.4.9 – Описание полей таблицы MedicalRecordEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| PatientIDFK | int | Внешний ключ – пациент |
| DoctorIDFK | int | Внешний ключ – доктор |
| AppointmentRecordIDFK | int | Внешний ключ – запись о приёме |
| MedicalHistoryIDFK | int | Внешний ключ – история болезни |
| RecipeIDFK | int | Внешний ключ – рецепт |
| LaboratoryTestIDFK | int | Внешний ключ – лабораторный анализ |

1. Таблица 4.4.10 – Описание полей таблицы CarEntity

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | int | Идентификатор |
| Mark | nvarchar(100) | Марка |
| Model | nvarchar(100) | Модель |
| Generation | nvarchar(100) | Поколение |
| ReleaseYear | int | Год выпуска |
| ApproximatePrice | int | Приблизительная стоимость |

Схема базы данных (от англ. Database schema) – её структура, описанная на формальном языке, поддерживаемом СУБД. В реляционных базах данных схема определяет таблицы, поля в каждой таблице (обычно с указанием их названия, типа, обязательности), и ограничения целостности (первичный, потенциальные и внешние ключи и другие ограничения).

* + - * 1. Основными объектами графического представления схемы являются таблицы и связи, определяемые внешними ключами.

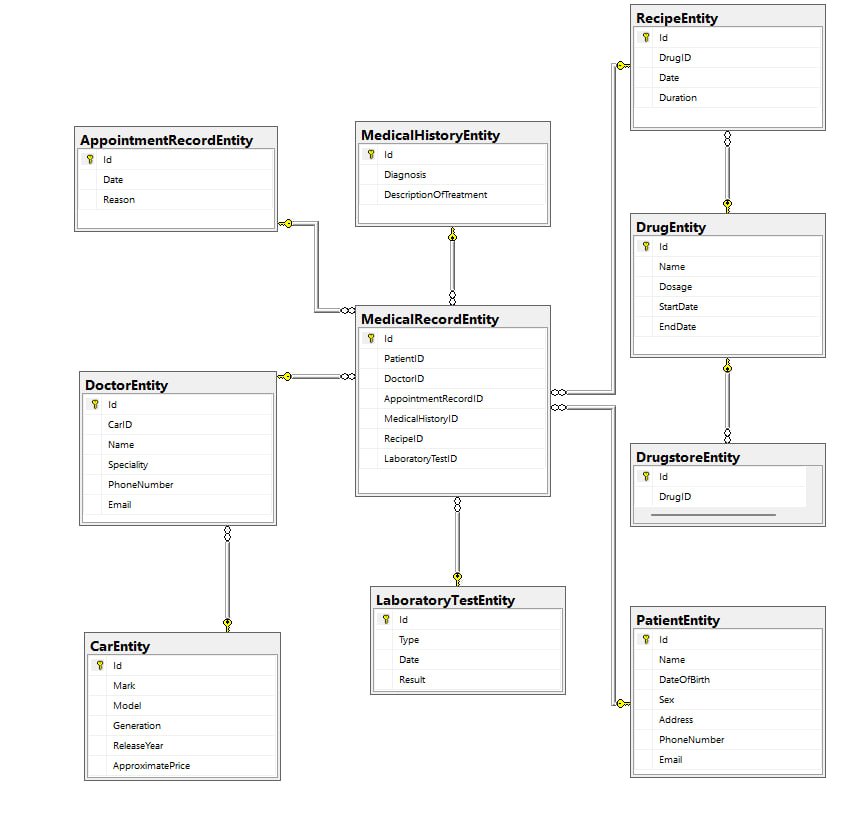
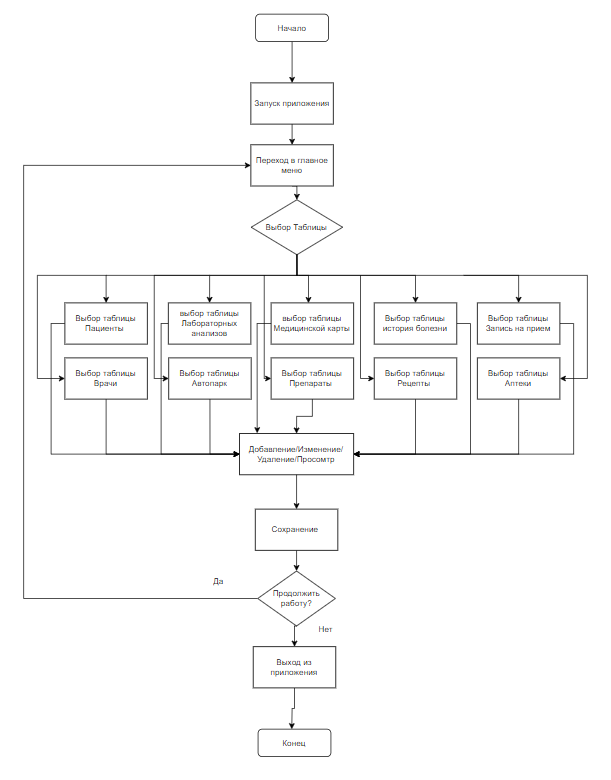
1. На рисунке 4.4 представлена схема проектируемой базы данных.
2. 

Рисунок 4.4.1 – Схема базы данных для больницы

Характеристика связей между таблицами на схеме базы данных.

* «PatientEntity» и «MedicalRecordEntity» имеют связь 1:1, т.к. у одного пациента может быть только одна медицинская карта.
* «DoctorEntity» и «MedicalRecordEntity» имеют связь 1:М, т.к. у одного доктора может быть только несколько медицинских карт.
* «MedicalRecordEntity» и «AppointmentRecordEntity» имеют связь 1:М, т.к. у одной медицинской карты может быть только несколько записей о приёме.
* «MedicalRecordEntity» и «MedicalHistoryEntity» имеют связь 1:М, т.к. у одной медицинской карты может быть только несколько историй болезни.
* «MedicalRecordEntity» и «LaboratoryTestEntity» имеют связь 1:М, т.к. у одной медициснкой карты может быть только несколько лабораторных анализов.
* «MedicalRecordEntity» и «RecipeEntity» имеют связь 1:М, т.к. у одной медицинской карты может быть только несколько рецептов.
* «DoctorEntity» и «CarEntity» имеют связь 1:1, т.к. у одного врача может быть только одна машина.
* «RecipeEntity» и «DrugEntity» имеют связь 1:1, т.к. у одного рецепта может быть только один препарат.
* «DrugstoreEntity» и «DrugEntity» имеют связь 1:М, т.к. у одной аптеки может быть несколько препаратов.
  1. **4.5 Схема работы приложения**
  2. ****
  3. **4.6 Алгоритмы использования применяемых программных технологий**

**Алгоритм взаимодействия с базой данных (MySQL):**

1. Подключение к базе данных через MySQL Connector.
2. Выполнение CRUD-операций:
   * **Создание**: Вставка новых записей.
   * **Чтение**: Выборка данных для отображения.
   * **Обновление**: Изменение существующих записей.
   * **Удаление**: Удаление ненужных данных.

**Алгоритм обработки данных для отчета:**

1. Выполнение SQL-запросов для выборки агрегированных данных.
2. Построение графиков в пользовательском интерфейсе.
   1. **4.7 Архитектура и схема функционирования ПС (ПМ)**

**Архитектура системы с использованием шаблона MVVM**

Для реализации ПМ используется архитектурный шаблон **MVVM** (Model-View-ViewModel), который упрощает разработку и поддержку пользовательского интерфейса. MVVM обеспечивает четкое разделение логики представления, бизнес-логики и данных, что делает систему гибкой и масштабируемой.

**Основные модули MVVM:**

1. **Model (Модель)**:
   * Представляет данные и бизнес-логику приложения.
   * Отвечает за взаимодействие с базой данных (MySQL).
   * Реализует классы данных, такие как Brand, Car, CheckedOffense, Model, Offense, Officer, Proofs, User, Verdict, Database, NotifyProperty, RelayCommand.
   * Содержит методы для выполнения CRUD-операций.

**2. View (Представление)**:

* Отображает данные пользователю и отвечает за пользовательский интерфейс.
* Реализуется с использованием **XAML** для создания графического интерфейса.
* Примеры элементов интерфейса:
  + Экран авторизации/регистрации.
  + Форма заполнения правонарушения.
  + Форма ответа инспектора по правонарушению.
  + Экран просмотра подданых правонарушений пользователем.

**3. ViewModel (Модель представления)**:

* Связывает данные из модели с представлением.
* Содержит команды и свойства, используемые для работы с данными

**Взаимодействие модулей MVVM:**

1. **View**:
   * Связывается с ViewModel через механизмы привязки данных (data binding) в XAML.
   * Реагирует на изменения в данных ViewModel.
2. **ViewModel**:
   * Взаимодействует с Model для получения или обновления данных.
   * Обеспечивает команды для обработки пользовательских событий.
3. **Model**:

* Взаимодействует с базой данных (MySQL) для выполнения операций чтения, записи, обновления и удаления данных.

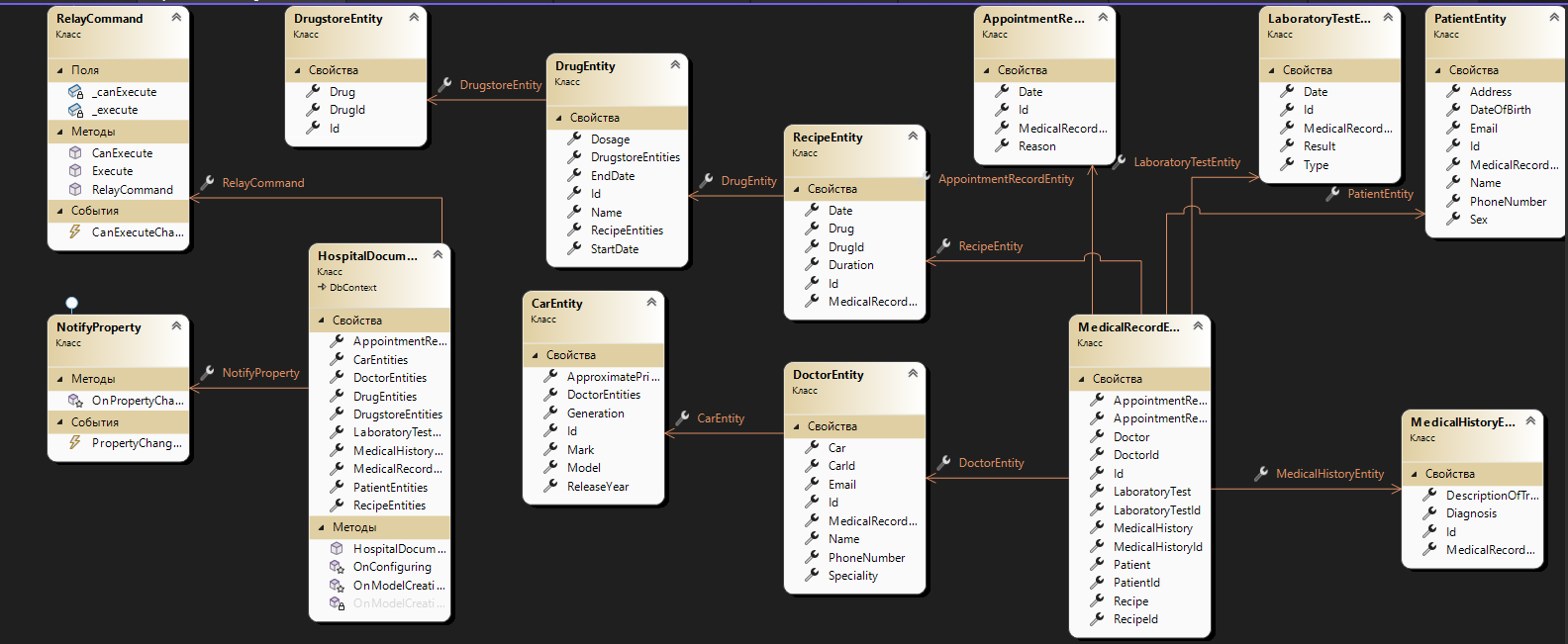
****

Рисунок 4.7.1 – Диаграмма классов

* 1. **ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ**
  2. **План тестирования**

В данном разделе будет представлен план тестирования программного обеспечения (ПС), который включает в себя контрольный список проверок для тестировщика. Этот список поможет провести тщательное тестирование приложения и его функций.

Виды тестирования

1. Функциональное тестирование

- Описание: Это тестирование, которое проверяет соответствие каждой функции приложения заданным требованиям. Будут тестироваться все пользовательские сценарии, чтобы убедиться, что приложение выполняет свои функции корректно.

- Обоснование: Данный вид тестирования является критически важным, так как служит основой для проверки того, что приложение выполняет свои функции на уровне требований и ожиданий пользователей.

2. Регрессионное тестирование

- Описание: после внесения изменений в код (например, исправления ошибок или добавления новых функций) необходимо проверить, что существующий функционал приложения не был нарушен. - Обоснование: Регрессионное тестирование позволит убедиться в стабильности ПС после изменений и снизит риски появления новых ошибок в уже работающих функциях.

3. Тестирование производительности

- Описание: Проверка приложения на устойчивость и быстродействие под разными нагрузками.

- Обоснование: это тестирование необходимо для оценки производительности приложения в реальных условиях использования. Оно поможет выявить узкие места и оптимизировать их.

Виды тестирования, которые не будут проведены и обоснование выбора

1. Тестирование удобства использования (Usability Testing)

- Обоснование: Удобство использования выделяется, как одна из второстепенных задач, так как интерфейс приложения будет простым и наглядным, что ожидается от пользователей. Однако это не отменяет необходимости потратить часть времени на изучение обратной связи от реальных пользователей после внедрения.

2. Тестирование совместимости

- Обоснование: так как приложение разрабатывается для узкого числа платформ и систем, данное тестирование может занять значительное время без принципиальной необходимости, принимая во внимание целевую аудиторию.

3. Модульное тестирование

- Обоснование: В данном проекте принято решение проводить интеграционное тестирование вместо модульного, так как разработка происходит с использованием подхода Agile, где часовая проверка.

**5.2 Результаты тестирования**

1. **Добавление нового пациента**:
   * Действие: Ввод данных через форму.
   * Ожидаемый результат: Ввод данных через форму и сохранение в БД.
   * Фактический результат: Успешно.
2. **Изменение данных пациента**:
   * Действие: Ввод данных через форму.
   * Ожидаемый результат: Ввод данных через форму и сохранение в БД.
   * Фактический результат: Успешно.
3. **Удаление пациента**:
   * Действие: Удаление данных через форму.
   * Ожидаемый результат: Удаление данных через форму и удаление в БД.
   * Фактический результат: Успешно.

**Нагрузочное тестирование**:

* При одновременном добавлении 1 000 заявлений занимает менее 2 секунд.

**Пользовательское тестирование**:

* Участники тестирования отметили удобство интерфейса и интуитивно понятную навигацию.

**5.3 Оптимизация ПС (ПМ)**

**Выявленные проблемы**:

отсутствуют

**Решения**:

Не требуется

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При открытии приложения появляется главное меню.

На рисунке 6.1 представлено «Главное меню».



Рисунок 6.1 – окно «Главное меню»

На экране представлены 10 кнопок для навигации по приложению:

1. Истории болезни: при нажатии открывается новое окно с историями болезней. На рисунке 6.2 представлено окно «Истории болезни».

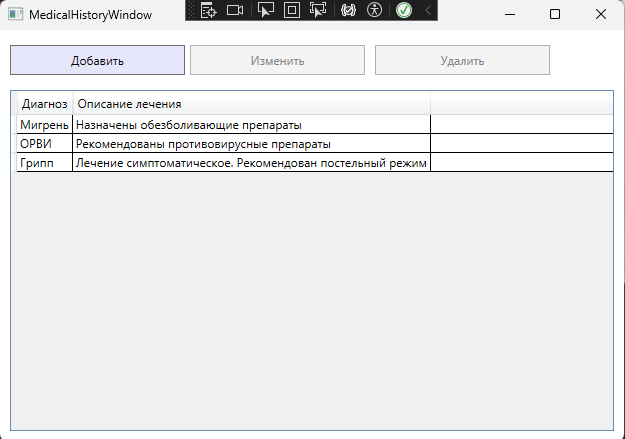


Рисунок 6.2 – окно «Истории болезни»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно создать новую историю болезни, либо отредактировать выбранную. На рисунке 6.3 представлено окно «Добавить/Изменить Историю болезни».

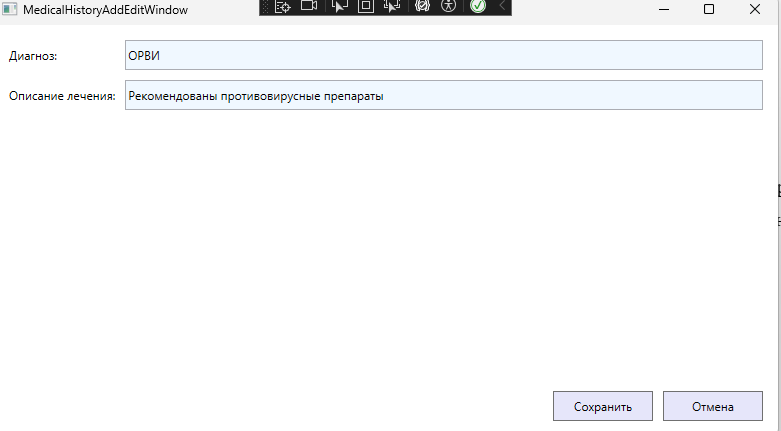


Рисунок 6.3 – окно «Добавить/Изменить Историю болезни»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

2. Записи на приём: при нажатии открывается новое окно с записями на приём. На рисунке 6.4 представлено окно «Записи на приём».

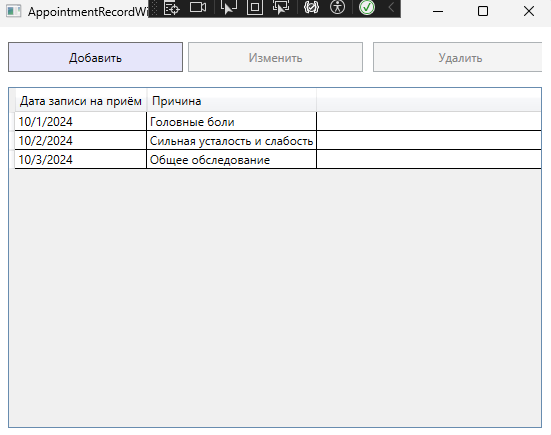


Рисунок 6.4 – окно «Записи на приём»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно создать новую запись на приём, либо отредактировать выбранную. На рисунке 6.5 представлено окно «Добавить/Изменить Запись на приём».

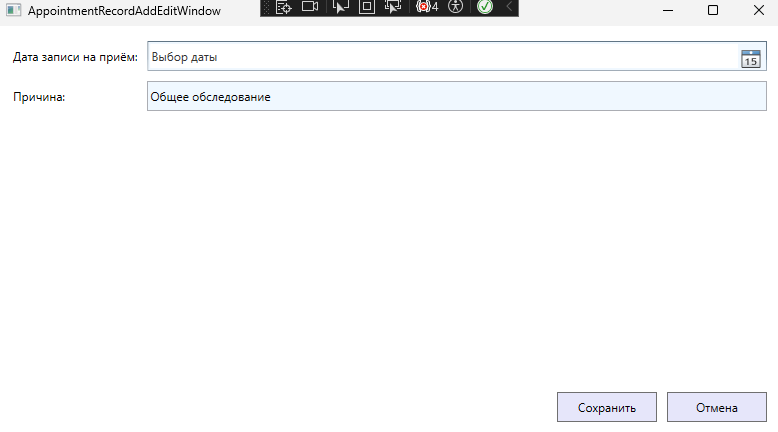


Рисунок 6.5 – окно «Добавить/Изменить Историю болезни»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

3.Список врачей: при нажатии открывается новое окно со списком врачей. На рисунке 6.6 представлено окно «Список врачей».

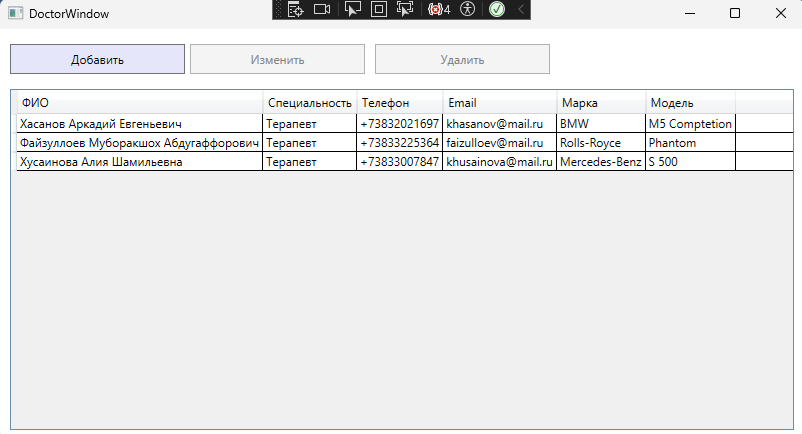


Рисунок 6.6 – окно «Список врачей»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить нового врача, либо отредактировать выбранного. На рисунке 6.7 представлено окно «Добавить/Изменить Врача».

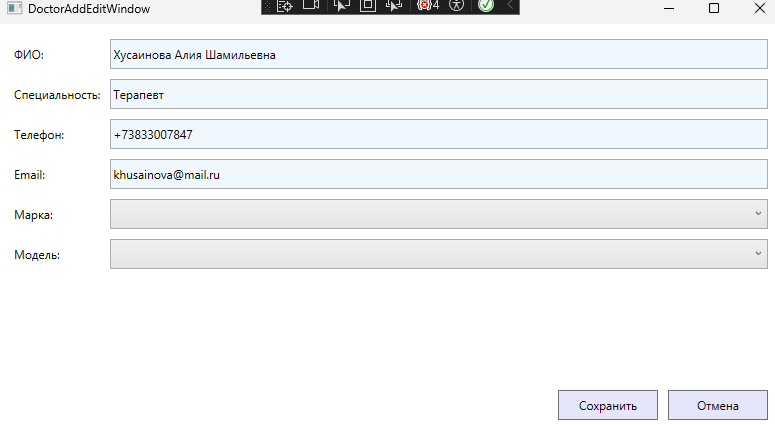


Рисунок 6.7 – окно «Добавить/Изменить Врача»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

4. Препараты: при нажатии открывается новое окно с препаратами. На рисунке 6.8 представлено окно «Препараты».

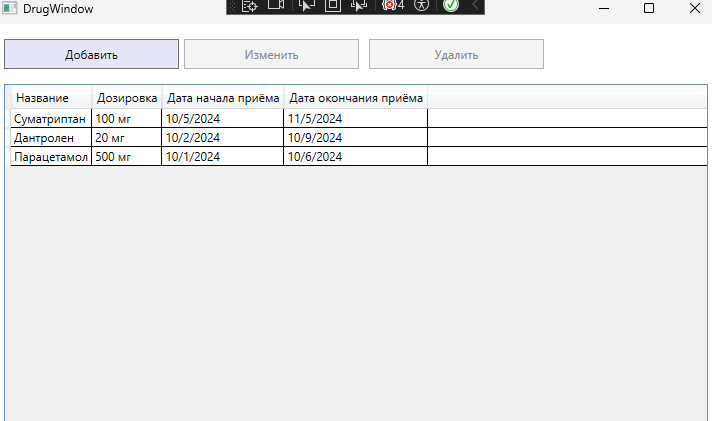


Рисунок 6.8 – окно «Препараты»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новый препарат, либо отредактировать выбранный. На рисунке 6.9 представлено окно «Добавить/Изменить Препарат».

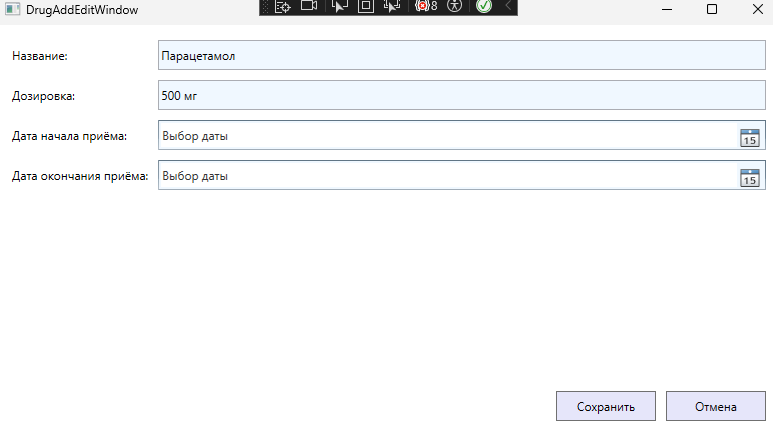


Рисунок 6.9 – окно «Добавить/Изменить Препарат»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

5. Медицинские карты: при нажатии открывается новое окно с медицинскими картами. На рисунке 6.10 представлено окно «Медицинские карты».

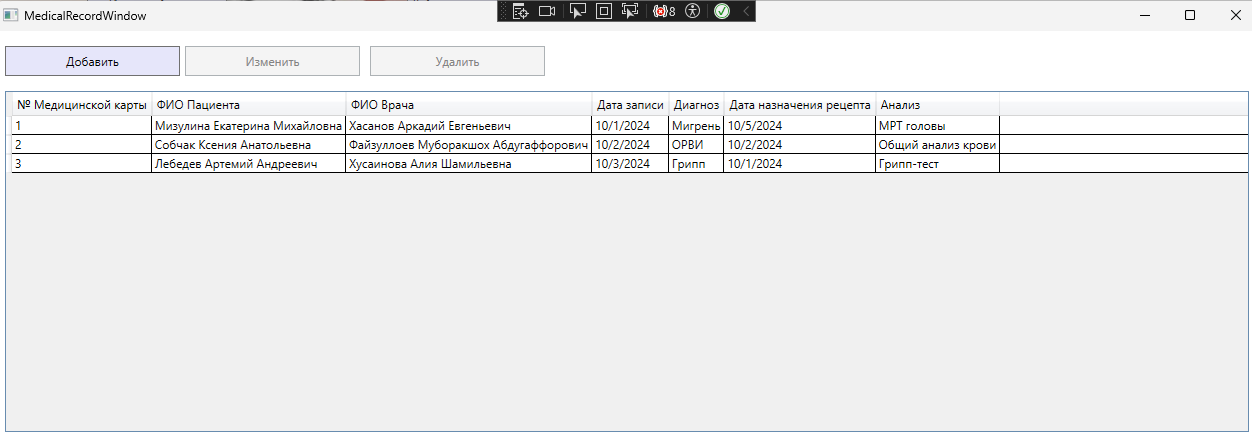


Рисунок 6.8 – окно «Медицинские карты»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новую медицинскую карту, либо отредактировать выбранную. На рисунке 6.11 представлено окно «Добавить/Изменить Медицинскую карту».

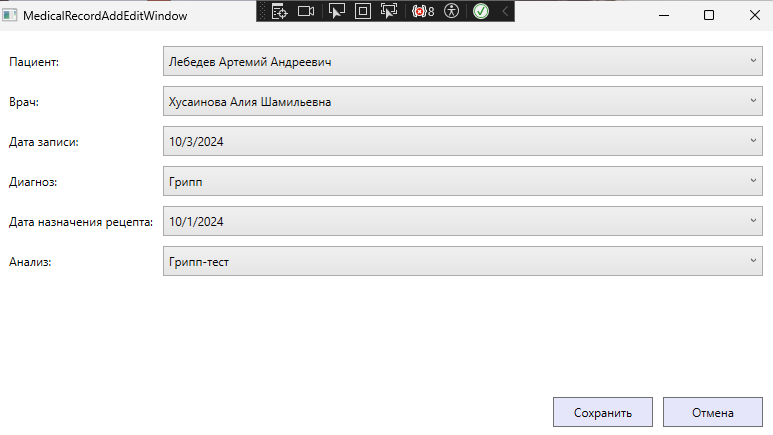


Рисунок 6.11 – окно «Добавить/Изменить Медицинскую карту»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

6.Автомобили: при нажатии открывается новое окно с автомобилями. На рисунке 6.12 представлено окно «Автомобили».

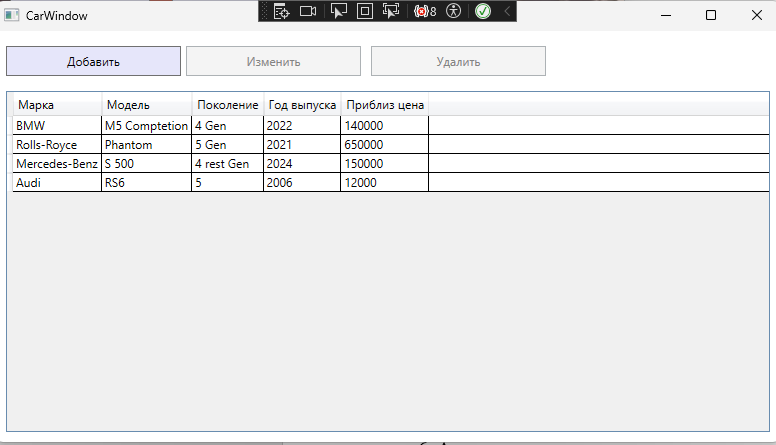


Рисунок 6.12 – окно «Автомобили»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новый автомобиль, либо отредактировать выбранный. На рисунке 6.13 представлено окно «Добавить/Изменить Автомобиль».

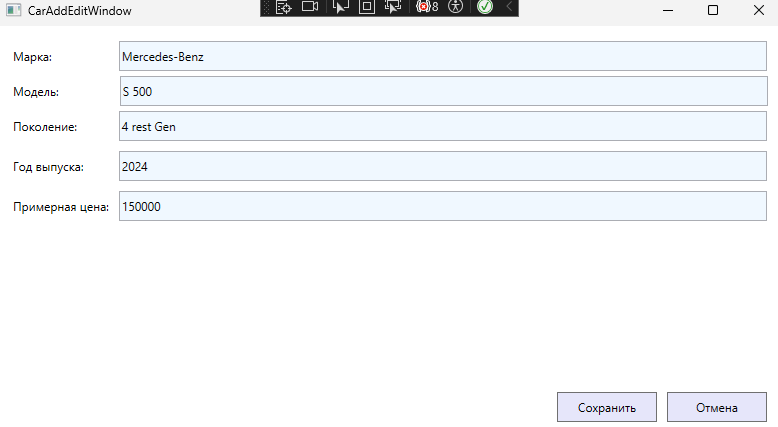


Рисунок 6.13 – окно «Добавить/Изменить Автомобиль»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

7.Лабораторные анализы: при нажатии открывается новое окно с лабораторными анализами. На рисунке 6.14 представлено окно «Лабораторные анализы».

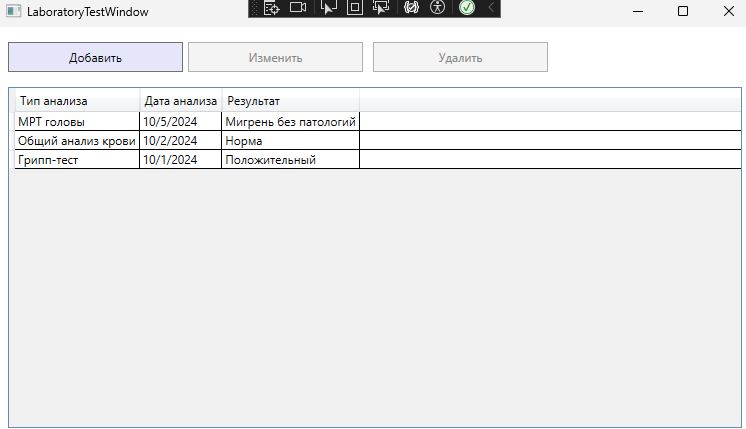


Рисунок 6.14 – окно «Лабораторные анализы»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новый лабораторный анализ, либо отредактировать выбранный. На рисунке 6.15 представлено окно «Добавить/Изменить Лабораторный анализ».

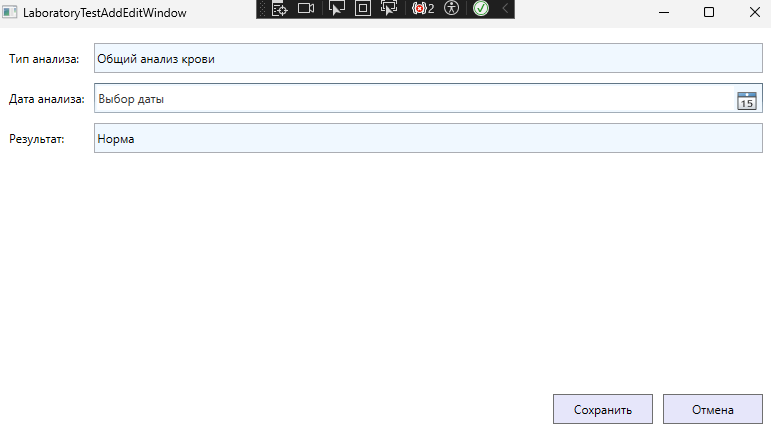


Рисунок 6.15 – окно «Добавить/Изменить Лабораторный анализ»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

8.Рецепты: при нажатии открывается новое окно с рецептами. На рисунке 6.16 представлено окно «Рецепты».

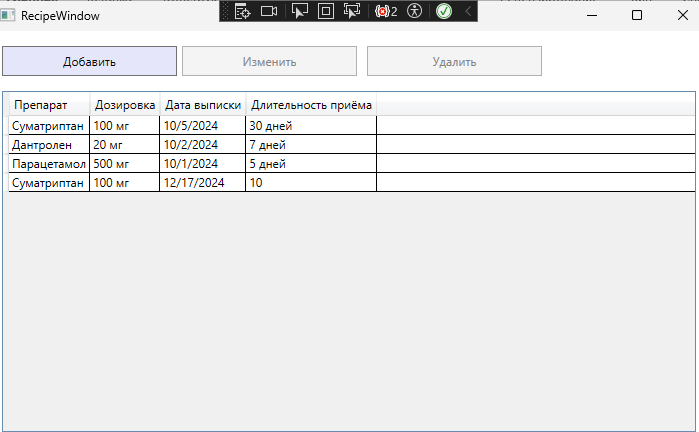


Рисунок 6.16 – окно «Рецепты»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новый рецепт, либо отредактировать выбранный. На рисунке 6.17 представлено окно «Добавить/Изменить Рецепт».

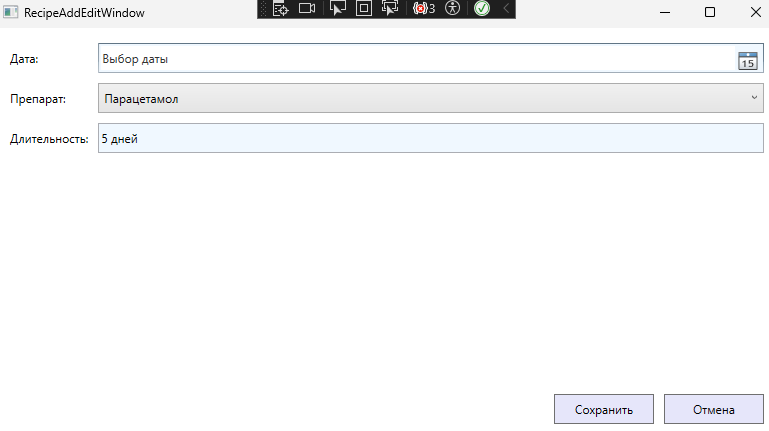


Рисунок 6.17 – окно «Добавить/Изменить Рецепт»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

9.Список пациентов: при нажатии открывается новое окно со списком пациентов. На рисунке 6.18 представлено окно «Список пациентов».

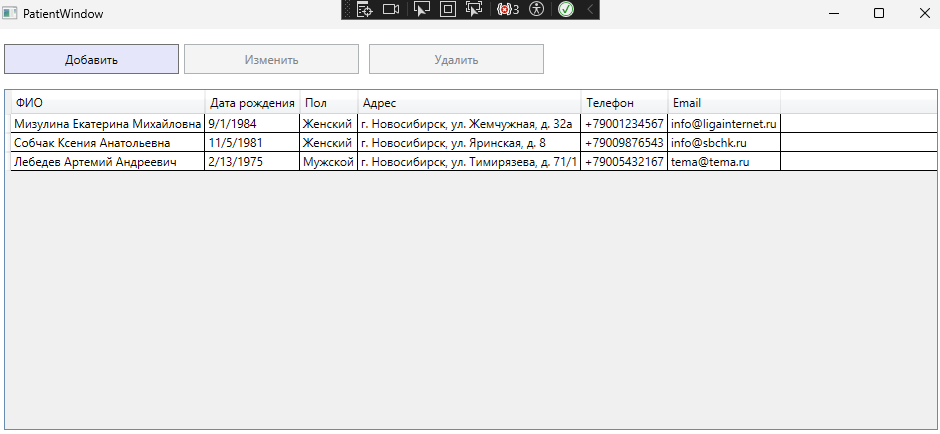


Рисунок 6.18 – окно «Список пациентов»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить нового пациента, либо отредактировать выбранного. На рисунке 6.19 представлено окно «Добавить/Изменить Пациента».

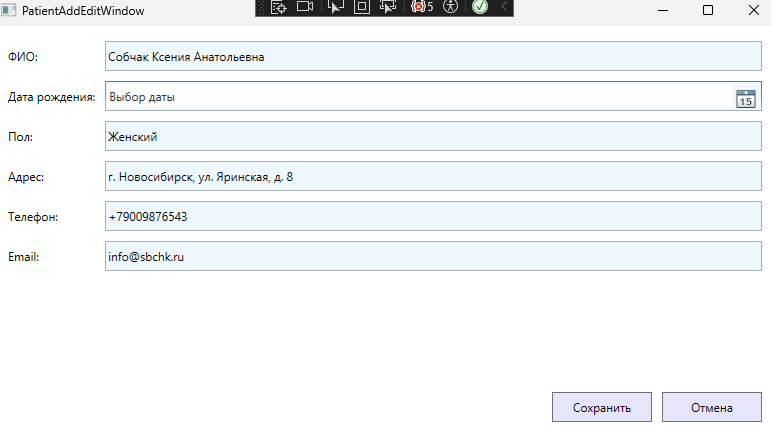


Рисунок 6.19 – окно «Добавить/Изменить Пациента»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

10.Аптеки: при нажатии открывается новое окно с аптеками. На рисунке 6.20 представлено окно «Аптеки».

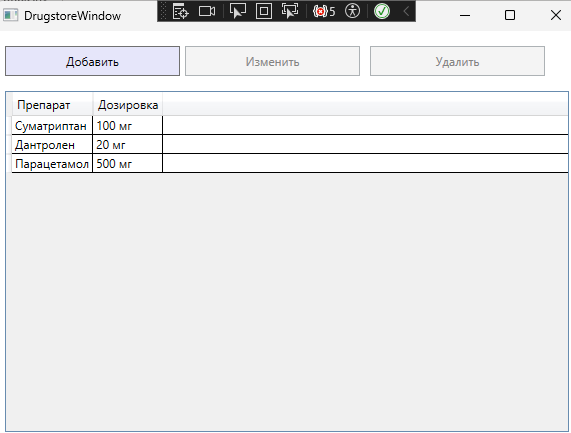


Рисунок 6.20 – окно «Аптеки»

На данном окне располагаются 3 кнопки:

* Добавить
* Изменить
* Удалить

Если выбрать запись и нажать кнопку удалить, то запись удалится. При нажатии на кнопку добавить/изменить открывается окно, где можно добавить новую аптеку, либо отредактировать выбранную. На рисунке 6.21 представлено окно «Добавить/Изменить Аптеку».

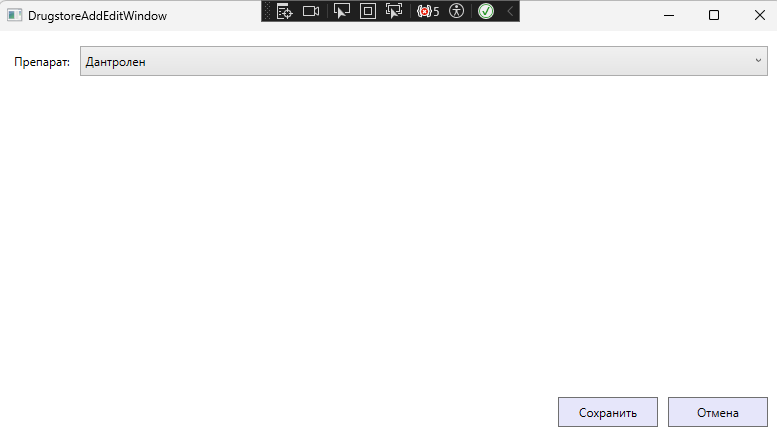


Рисунок 6.21 – окно «Добавить/Изменить Аптеку»

На данном окне располагаются 2 кнопки:

* Сохранить
* Отмена

При нажатии на кнопку сохранить запись сохраняется. При нажатии на кнопку отмена данное окно закрывается и перекидывает на окно с общими записями.

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано WPF приложение, предназначенное для врачей терапевтического отделения больницы.

Проделаны следующие виды работ:

* + разработана база данных;
  + спроектирован и реализован пользовательский интерфейс программы;
  + оформлено руководство пользователя;

Были получены навыки проведения этапа низкоуровнего и высокоуровнего проектирования интерфейса: проанализирована предметная область, выделены профили потенциальных пользователей, составлены схемы навигации и пользовательские сценарии.

Разработанное WPF приложение, предназначенное для автоматизации процессов обработки данных, их корректировки и хранения в БД, связанных с деятельностью в терапевтическом отделении больницы.

В приложении реализованы функции добавления, изменения, удаления. С помощью MS Excel возможен экспорт данных из таблиц БД для дальнейшего их анализа, построения диаграмм, графиков, регрессий и т.д., также это позволяет вести отчетность на бумаге.

В БД хранятся все записи, которые хотя бы единожды были внесены в WPF приложении.

Реализация системы проводилась с использованием Visual Studio и SQL Server Management Studio.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

{Список опубликованных печатных, либо интернет- источников информации, используемых при написании текста ПЗ.

Сведения об источниках можно располагать в порядке появления ссылок на источники в тексте документа.

Источники необходимо нумеровать арабскими цифрами без точки и печатать с абзацного отступа.

Ссылки на использованные источники следует приводить в квадратных скобках в тексте, в конце предложения.

Примеры библиографического описания источников}

1. Численное исследование процессов переноса и трансформации газовых и аэрозольных примесей в шлейфе выбросов Норильского промышленного района / В.Ф. Рапута, В. Симоненков, Б.Д. Белан, Т.В. Ярославцева / «Оптика атмосферы и океана», 31, № 6, 2018 – C. 438-439.
2. Вострикова Л.Г. Финансовое право [Электронный ресурс] : учебник / Л.Г. Вострикова. – Электрон. текстовые дан. – М.: Равновесие: Юстинформ, 2005. – 1 электрон. опт. диск (СD). – Загл. с контейнера.
3. Исследовано в России: многопредмет. науч. журн. / Моск. физ.-техн. ин-т. – Электрон. журн. – Долгопрудный : МФТИ, 1998.– URL : http://zhurnal.mipt.rssi.ru, свободный. –Яз. рус. – (Дата обращ. ……..)
4. Мозолин В. П. Модернизация права собственности в экономическом измере-нии [Электронный ресурс] / В. П. Мозолин. – Электрон. ст. – М., 2011. – URL : http://www.norma-verlag.com/journal/2011/1, свободный. – Яз. рус. – Аналог печат. изд. (Журнал российского права. – 2011. – № 1). – (Дата обращ. 23.06.2011).
5. **ПРИЛОЖЕНИЯ**
   1. **Приложение А**

{Содержит **техническое задание на реализацию программной системы**, разработанное при непосредственном участии представителей заинтересованных организаций.}

* 1. **Приложение Б**

{Содержит примеры входных данных в виде изображений, таблиц, фрагментов текстовых документов и.т.д.}

* 1. **Приложение В**

{Содержит различные схемы, диаграммы, иллюстрирующие алгоритмы работы разработанной программной системы.}

* 1. **Приложение Г**

{Содержит результаты тестовых испытаний на различных тестовых данных.

*Например, при наличии в составе ПС расчетно-графического модуля, полученные в результате работы, иллюстрации и таблицы, по которым можно судить о корректной работе программы*.}

* 1. **Приложение Д**

AppointmentRecordEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class AppointmentRecordEntity

{

public int Id { get; set; }

public DateOnly Date { get; set; }

public string Reason { get; set; } = null!;

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

CarEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class CarEntity

{

public int Id { get; set; }

public string Mark { get; set; } = null!;

public string Model { get; set; } = null!;

public string Generation { get; set; } = null!;

public int ReleaseYear { get; set; }

public int ApproximatePrice { get; set; }

public virtual ICollection<DoctorEntity> DoctorEntities { get; set; } = new List<DoctorEntity>();

}

DoctorEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class DoctorEntity

{

public int Id { get; set; }

public int? CarId { get; set; }

public string Name { get; set; } = null!;

public string Speciality { get; set; } = null!;

public string PhoneNumber { get; set; } = null!;

public string Email { get; set; } = null!;

public virtual CarEntity? Car { get; set; }

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

DrugEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class DrugEntity

{

public int Id { get; set; }

public string Name { get; set; } = null!;

public string Dosage { get; set; } = null!;

public DateOnly StartDate { get; set; }

public DateOnly EndDate { get; set; }

public virtual ICollection<DrugstoreEntity> DrugstoreEntities { get; set; } = new List<DrugstoreEntity>();

public virtual ICollection<RecipeEntity> RecipeEntities { get; set; } = new List<RecipeEntity>();

}

DrugstoreEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class DrugstoreEntity

{

public int Id { get; set; }

public int? DrugId { get; set; }

public virtual DrugEntity? Drug { get; set; }

}

LaboratoryTestEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class LaboratoryTestEntity

{

public int Id { get; set; }

public string Type { get; set; } = null!;

public DateOnly Date { get; set; }

public string Result { get; set; } = null!;

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

MedicalHistoryEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class MedicalHistoryEntity

{

public int Id { get; set; }

public string Diagnosis { get; set; } = null!;

public string DescriptionOfTreatment { get; set; } = null!;

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

MedicalRecordEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class MedicalRecordEntity

{

public int Id { get; set; }

public int? PatientId { get; set; }

public int? DoctorId { get; set; }

public int? AppointmentRecordId { get; set; }

public int? MedicalHistoryId { get; set; }

public int? RecipeId { get; set; }

public int? LaboratoryTestId { get; set; }

public virtual AppointmentRecordEntity? AppointmentRecord { get; set; }

public virtual DoctorEntity? Doctor { get; set; }

public virtual LaboratoryTestEntity? LaboratoryTest { get; set; }

public virtual MedicalHistoryEntity? MedicalHistory { get; set; }

public virtual PatientEntity? Patient { get; set; }

public virtual RecipeEntity? Recipe { get; set; }

}

PatientEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class PatientEntity

{

public int Id { get; set; }

public string Name { get; set; } = null!;

public DateOnly DateOfBirth { get; set; }

public string Sex { get; set; } = null!;

public string Address { get; set; } = null!;

public string PhoneNumber { get; set; } = null!;

public string Email { get; set; } = null!;

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

RecipeEntity:

using System;

using System.Collections.Generic;

namespace HospitalDocumentation;

public partial class RecipeEntity

{

public int Id { get; set; }

public int? DrugId { get; set; }

public DateOnly Date { get; set; }

public string Duration { get; set; } = null!;

public virtual DrugEntity? Drug { get; set; }

public virtual ICollection<MedicalRecordEntity> MedicalRecordEntities { get; set; } = new List<MedicalRecordEntity>();

}

* 1. **Приложение Е**

{При наличии, можно привести Акт о внедрении в эксплуатацию разработанной программной системы.}