МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

Направление специальности 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема Программное средство «Запись к терапевту»

Исполнитель

студент (ка) 2 курса группы 9 Кульнис Александр Юрьевич

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В. (подпись)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

|  |  |
| --- | --- |
| Специальность: 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем  Студент: Кульнис А.Ю. | Группа:\_\_9\_\_ |
| **Тема: Программное средство «Запись к терапевту»** | |

**по дисциплине** "Объектно-ориентированное программирование"

**1. Срок сдачи студентом законченной работы**: "20 мая 2021 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Выполнять функции администратора:
  + Выполнять авторизацию;
  + Просматривать список пользователей;
  + Добавление новых пользователей;
  + Удаление старых пользователей.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Заполнение формы с личной информацией;
  + Возможность записаться на приём к терапевту;
  + Возможность просматривать личную карточку.

**2.2.** При выполнении курсового проекта необходимо использовать принципы и приемы ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2021 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2021 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2021 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2021 |  |
| 5 | Кодирование программного средства | 23.04.2021 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2021 |  |
| 7 | Оформление пояснительной записки | 7.05.2021 |  |
| 9 | Сдача проекта | 20.05.2021 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2021**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О.Л. Панченко*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Содержание**

[ВВЕДЕНИЕ………………………………………………………………………5](#_ВВЕДЕНИЕ)

[1. Аналитический обзор литературы……………………………………….......6](#_Аналитический_обзор_прототипов)

[2. Анализ требований к программному средству и разработка функциональных требований …………………………………………………...9](#_Анализ_требований_к)

[3. Проектирование программного средства…………………………………..12](#_Проектирование_программного_средств)

[4. Создание (реализация) программного средства…………………………...15](#_Toc9543694)

[5. Тестирование, проверка работоспособности и анализ полученных результатов……………………………………………………………………...20](#_Toc9543695)

[6. Руководство по использованию…………………………………………......28](#_Toc9543696)

[ЗАКЛЮЧЕНИЕ…………………………………………………………………33](#_ЗАКЛЮЧЕНИЕ)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ………………………………34](#_Toc9543698)

[Приложение А…………………………………………………………………..35](#_Toc9543699)

[Приложение Б…………………………………………………………………...35](#_Toc9543700)

[Приложение В…………………………………………………………………...36](#_Toc9543701)

[Приложение Г…………………………………………………………………...37](#_Toc9543702)

[Приложение Д…………………………………………………………………...39](#_Toc9543703)

[Приложение Е…………………………………………………………………...39](#_Toc9543704)

[Приложение Ж…………………………………………………………………..43](#_Приложение_Ж)

[Приложение З…………………………………………………………………....44](#_Toc9543706)

# **ВВЕДЕНИЕ**

Данный курсовой проект посвящён разработке программного средства «Запись к терапевту», основной целью которого является возможность пользователей записаться на приём к врачу.

Приложение «Запись к терапевту» предназначено для простого использования в случае необходимости посетить врача-терапевта. Данная программа проста в использовании и не требует никаких усилий, чтобы успешно забронировать время на приём к врачу.

В качестве интерфейса приклaдного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в Microsoft SQL Server.

Цель курсового проекта – создать бесплатный простой и быстрый инструмент для записи к врачу, который может быть использован в случае необходимости посетить врача-терапевта.

# **Аналитический обзор прототипов и литературных источников**

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

Здесь я буду рассматривать бесплатные приложения, или приложения низкой стоимости. Я взял несколько программ для записи к врачу, но их нельзя назвать аналогами моего ПС, так как они разрабатывались целой командой под началом известных фирм, однако ориентировался я на их функционал. Все программы являются desktop-приложениями.

Desktop-приложения для записи к врачу:

* «Запись к врачу | записаться на прием онлайн»;
* «Aibolit онлайн-запись к врачам Беларуси».

«Запись к врачу | записаться на прием онлайн» - приложение, где есть все врачи и все больницы. Вы можете записаться онлайн на прием к врачу в любую государственную больницу.

Интерфейс «Запись к врачу | записаться на прием онлайн» представлен на рисунке 1.1.

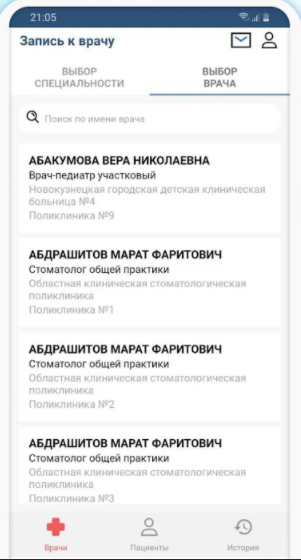


Рисунок 1.1 − Интерфейс «Запись к врачу | записаться на прием онлайн»

Проанализировав «Запись к врачу | записаться на прием онлайн», можно выделить основные минусы данного программного сред.

Основные минусы:

* Загруженный интерфейс;
* Нет записи для того что записаться на приём, врачей;

Основные плюсы:

* Поддерживается 1С;
* Простой интерфейс;
* Надежность;

Продукт «Aibolit онлайн-запись к врачам Беларуси» - приложение для онлайн записи к врачам по Беларуси для сайтов и соцсетей ресторанов и кафе. Онлайн приём броней, депозиты, статистика, база гостей, банкеты. Настраивается за 15 минут, работает в облаке. Стоит от 0 руб. в месяц.

Интерфейс «Aibolit» представлен на рисунке 1.2.

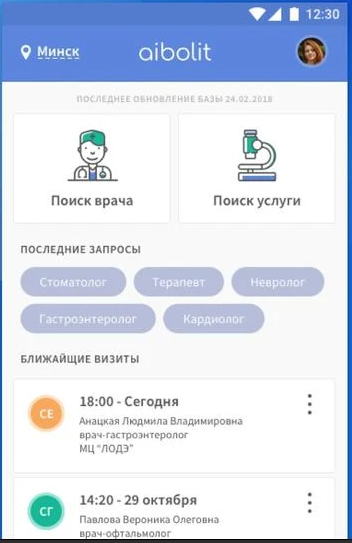


Рисунок 1.2 − Интерфейс «Aibolit»

Проанализировав ПО «Aibolit», можно выделить её основные минусы и плюсы.

Основные минусы:

* Примитивный интерфейс и плохие цвета;
* Отсутствие Фильтрация

Основные плюсы:

* Большой функционал;
* Простота в использовании;
* Наличие поисковой строки;

«К врачу» - Облачное решение для автоматизации ресторана, кафе, магазина или кальянной. Рабочее место кассира и официанта. Управление меню, складом, лояльностью, отчеты. Интеграция с принтерами чеков и фискальными регистраторами.

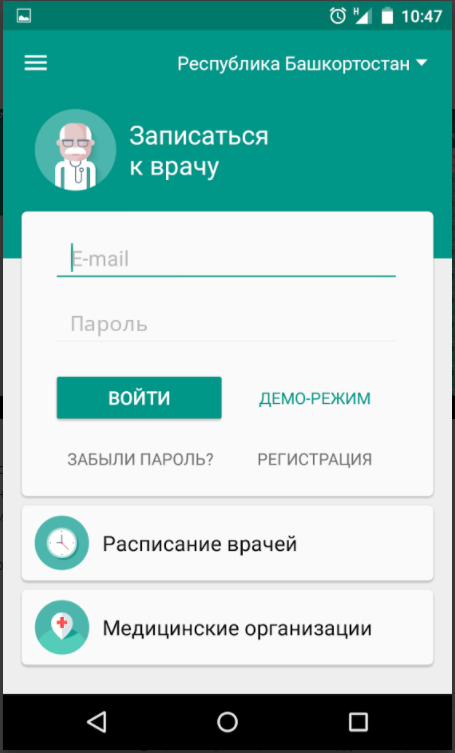
Интерфейс «К-врачу» представлен на рисунке 1.3.

Рисунок 1.3 – Интерфейс «К-врачу»

Данное ПС грамотно спроектировано и выполняет большое количество функций.

Основные минусы:

* Неудобное расписание;
* Много лишних категорий.

Основные плюсы:

* Может интегрироваться с кассовой системой;
* Возможность регистрации;



# **Анализ требований к программному средству и разработка функциональных требований**

Задачей курсового проекта является разработка приложения, предназначенного для записи к терапевту, а также ряда дополнительных операций: добавления юзеров, редактирования, удаления; возможность заполнить личную карту пациента, забронировать талон, отменить бронирование. Информацию о талонах и услугах вводится непосредственно с самого приложения и проверяется администратором.

Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

База данных была разработана в Microsoft SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов —Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* Code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран первый подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

Программное средство предоставляет пользователю следующие функциональные возможности:

* Регистрация и идентификация пользователей
* Заполнение личной карточки
* Возможность забронировать талон к врачу
* Возможность выбрать врача и дату и получить сведения о свободных талонах на этот день
* Возможность отменить забронированный талон
* Просмотр забронированных талонов
* Просмотр истории посещения

Администратор может:

* Добавление пользователей
* Редактирование пользователей
* Удаление пользователей
* Осуществлять поиск по некоторым критериям
* Просмотр всех пользователей
* Подтверждение посещения пользователя врача
* Очистка всех талонов, которые просрочены

Наглядно это видно на UML-схеме, (Приложение А).

# **Проектирование программного средства**

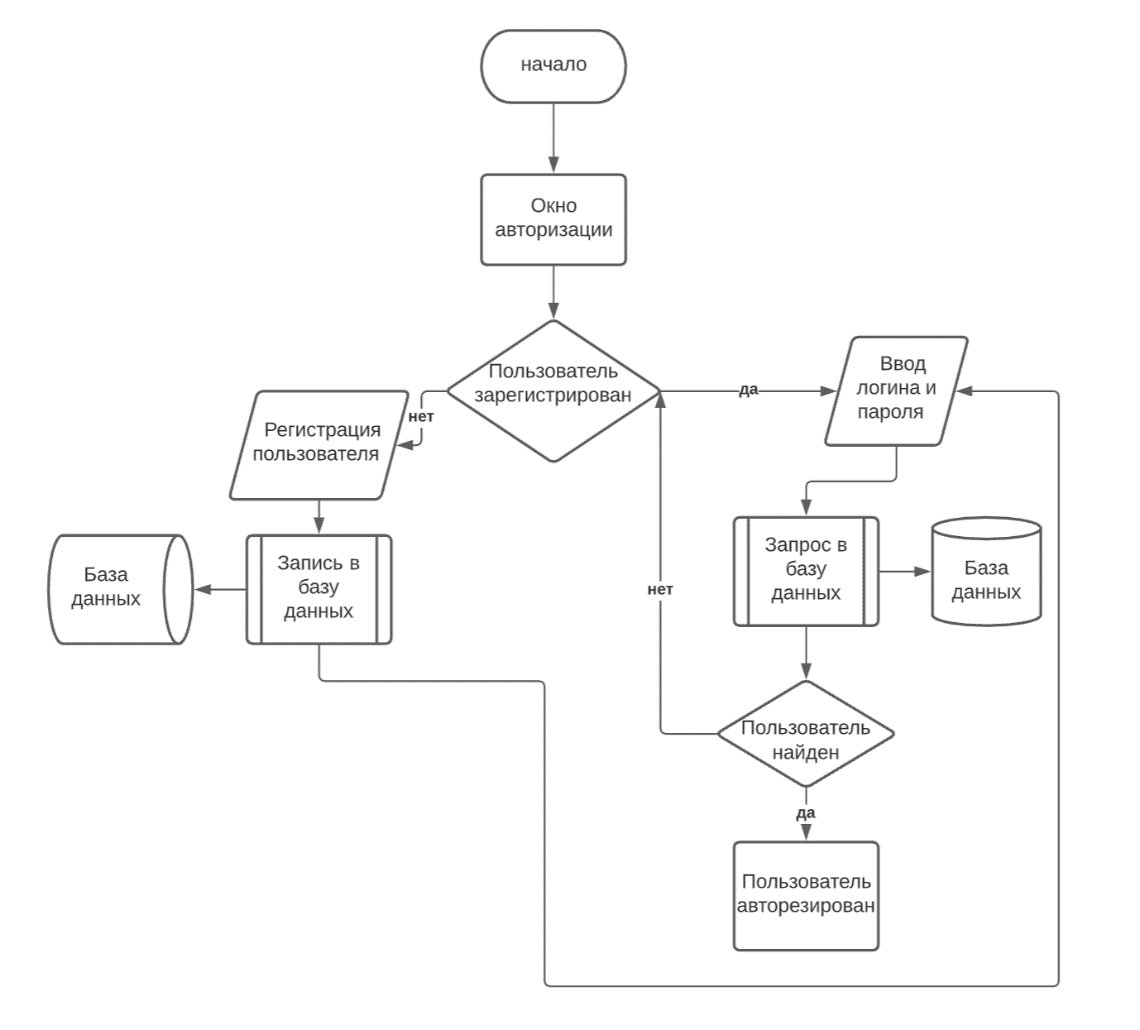
При запуске приложения открывается главное окно. В нем можно выбрать авторизацию либо регистрацию, принцип работы авторизации и регистрации показан в блок-схеме (рисунок 3.0)

Рисунок 3.0 — Блок-схема работы авторизации и регистрации

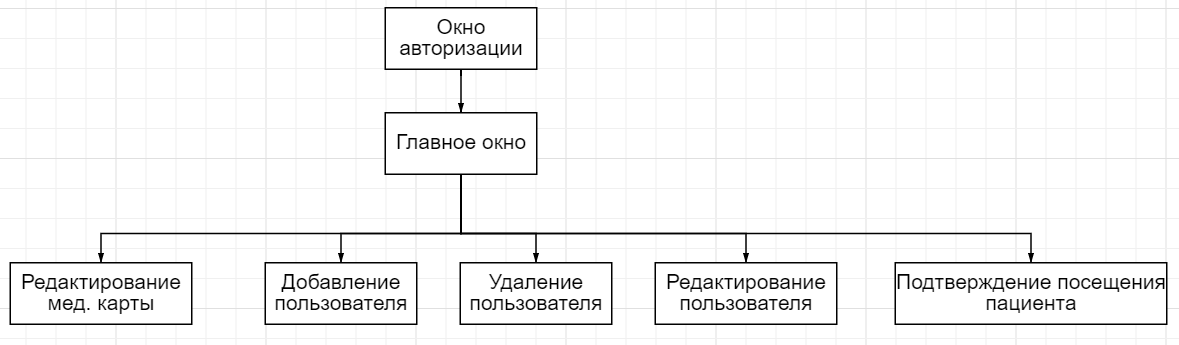
Исходя из схемы взаимодействия окон, можно увидеть, что для пользователей с разными ролями предоставляются различные полномочия. Если войти в приложение под именем администратора, откроется окно, в котором администратор может редактировать личную карточку, подтверждать посещение пациента. Также есть возможность просмотреть, какие пользователи зарегистрировались в приложении и при желании удалить какого-нибудь пользователя, добавить или редактировать. (рисунок 3.1).

Рисунок 3.1 ‒ «Авторизация и главное окно администратора»



Рисунок 3.2 – «Авторизация и главное окно клиента»

На рисунке 3.2 представлена общая схема работы приложения клиента. При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

После ввода логина и пароля в соответствующих полях, проверяется правильность ввода данных.

В случае неверного ввода данных, выведется соответствующее сообщение. В ином случае окно авторизации закроется и откроется главное окно приложения.

В главном окне работа идёт в страничном и оконных режимах.

Структура проекта представлена в приложении В.

В таблице 3.3 описана структура проекта по пакетам классов.

В этой таблице приведены основные логические составляющие пакеты, которые используются в работе приложения.

Таблица 3.3 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Model | Здесь описаны модели, с которыми происходит вся работа в приложении:   * Пользователи * Записи * Мед. карта * История посещений * Доктора |
| Repositories | Содержит классы и интерфейсы для работы с данными БД. Они являются связующим элементом между данными в БД и теми данными, с которыми работает приложение |
| Resources | Файлы иконок |
| View | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, использую при этом Model, после чего передать в базу данных. |
| App.config | Файл конфигурации приложения |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения |

Структуры данных, а также их зависимости друг от друга представлены в приложении Б.

Как показано в приложении Б, в вершине иерархии стоит класс User, который характеризует актуальное и непроверенное объявление соответственно. Класс Region характеризует регион, в котором может быть размещено объявление. User, характеризует пользователя с любым уровнем доступа. Announcement и TmpAnnouncement связаны с User по полю seller и id соответственно. Announcement и TmpAnnouncement связаны с Region по полю idRegion и id соответственно. TmpAnnouncement и Announcement между собой никак не связаны. Они по разным таблицам для того, чтобы не сливать воедино объявления всех типов.

Диаграмма классов показана в приложении Г.

# **Создание (реализация) программного средства**

При создании приложения использовался паттерн проектирование MVVM. Он заключается разделении представления от бизнес логики. Это достигается за счёт ввода новой логической конструкции ViewModel. ViewModel связывает представление и бизнес логику приложения.

Пакетная структура проекта показана на рисунке 4.1

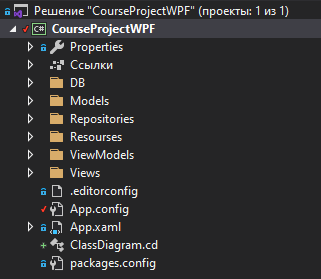


Рисунок 4.1 – «Структура пакетов проекта»

В Model находится структура модели базы данных приложения. При проектировании базы данных данного курсового проекта для взаимодействия с базой данных используется Entity Framework.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Нередко все сущности (модель) наследует класс ViewModelBase, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать класс ViewModelBase, нужно его наследовать в определенном классе.

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод RaisePropertiesChanged (“Имя свойства”) как показано в приложении Д.

Для удобной работы с данными, используется паттерн Repository. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. На рисунке 4.2 показана структура.



Рисунок 4.2 – «Паттерн Repository»

Интерфейс IUserRepository имплицируются в соответствующих классах и указывают: какие методы (функционал) должен иметь этот репозиторий. Репозитории показаны в приложении Е.

Интерфейс репозитория также показан в приложении Е.

На рисунке 4.3 показана структура пакета Resources, в котором находятся иконка для приложения и иконка для кнопки “Назад”



Рисунок 4.3 – «Структура пакета Resources»

В проекте используется паттерн MVVM, а это значит, что должны присутствовать ViewModels, которые связывают View и Model. Во ViewModel информация обрабатываться и выводится/записываться в нужном виде. Структура пакета ViewModel представлена в приложении К.

В приложении И показана структура пакета View, в котором хранятся файлы представлений на языке разметки xaml, а также соответствующие им файлы на языке C#. В этом пакете находятся все «окна» и «страницы», которые мы видим, используя приложение. Назначение каждого файла описано в таблице ниже (таблица 4.4)

Таблица 4.4 – «Описание файлов Views»

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| AdminAddClient.xaml (окно) | Окно добавления пользователя, в котором указывается все поля для успешной регистрации |
| AdminChangeClient.xaml (окно) | Окно редактирования пользователя, в котором меняются необходимые поля |
| AuthView.xaml (окно) | Окно авторизации. Требует ввода логина и пароля |
| CardWindow.xaml (страница) | Страница, отображающая личную карточку пациента и даёт возможность редактировать её |
| AuthWindow.xaml (окно) | Окно авторизации. Требует ввода логина и пароля или выбор способа авторизации |
| ChooseTimeAndConfirm.xaml (окно) | Окно, в котором нужно выбрать время посещения врача и подтвердить его |
| ConfirmUsersVisiting (окно) | Окно для подтверждения посещения и просмотра истории посещения пациента |
| HistotyVisitingWindow (страница) | Страница для просмотра истории посещения |
| Loader.xaml (окно) | Окно анимированной загрузки |
| MainAdminWindow.xaml (окно) | Главное окно администратора |
| MainClientWindow.xaml (окно) | Главное окно пользователя |
| RecordingWindow.xaml (страница) | Страница пользователя, в которой выбираешь дату посещения и врача для визита к терапетву. Также просмотр активных талонов и возможность их отменить |
| RegistrView.xaml (окно) | Окно регистрации нового пользователя |

На рисунке 4.5 изображена диаграмма таблиц со связями в базе данных.

Рисунок 4.5 – «Диаграмма таблиц»

Описание таблицы Doctors (таблица 4.6)

Таблица 4.6 – «описание Doctors»

|  |  |
| --- | --- |
| Id | Идентификатор |
| FIO | ФИО врача |
| Post | Специальность врача |
| Cabinet | Кабинет |
| EvenDay\_start | Начало работы врача в чётный день |
| EvenDay\_end | Конец работы врача в чётный день |
| OddDay\_start | Начало работы врача в нечётный день |
| OddDay\_end | Конец работы врача в нечётный день |

Описание таблицы UserDoctors (таблица 4.7)

Таблица 4.7 – «описание UserDoctors»

|  |  |
| --- | --- |
| User\_id | Идентификатор |
| Doctor\_id | Идентификатор |

Описание таблицы Medcards (таблица 4.8)

Таблица 4.8 – «описание Medcards»

|  |  |
| --- | --- |
| Id | Идентификатор |
| Name | Имя |
| Surname | Фамилия |
| Patronomic | Отчество |
| Gender | Пол |
| BDay | День рождения |
| City | Город |
| Street | Улица |
| House | Номер дома |
| Housing | Корпус дома |
| Flat | Номер квартиры |
| Image | Изображение |

Описание таблицы HistoryVisitings (таблица 4.9)

Таблица 4.9 – «описание HistoryVisitings»

|  |  |
| --- | --- |
| id | Идентификатор |
| FIO | ФИО доктора |
| Doctor | Специальность |
| Cabinet | Номер кабинета |
| VisitDay | Дата посещения |
| VisitTime | Время посещения |
| Info | Информация в карточке |
| UserId | Идентификатор |

Описание таблицы Recordings (таблица 4.9)

Таблица 4.9 – «описание Recordings»

|  |  |
| --- | --- |
| id | Идентификатор |
| FIO | ФИО доктора |
| Doctor | Специальность |
| Cabinet | Номер кабинета |
| VisitDay | Дата посещения |
| VisitTime | Время посещения |
| UserId | Идентификатор |

# **Тестирование, проверка работоспособности и анализ полученных результатов**

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент регистрации, возможна такая ситуация, в которой пользователь ничего не ввёл или ввёл неверные данные. Обработка данного сценария приведена на рисунке 5.1

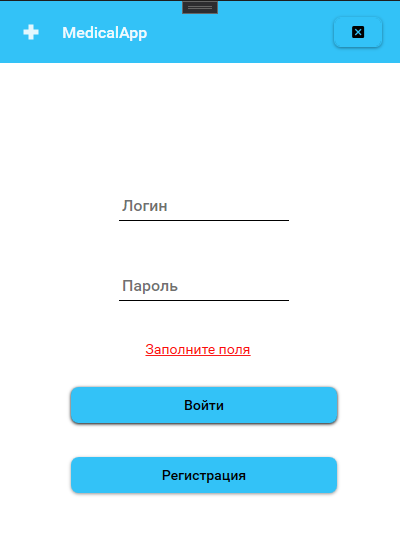


Рисунок 5.1 – «Ошибка корректности ввода»

При неверных данных – рисунок 5.2

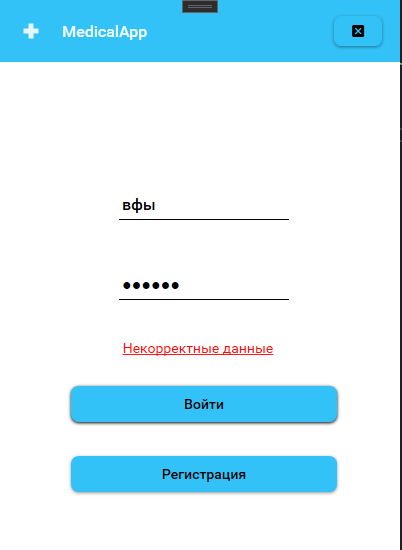


Рисунок 5.2 – «Сценарий, с неверными данными»

Также возможен такие случаи:

* Пользователь хочет зарегистрироваться, при этом не указав никаких данных (рисунок 5.3)
* Пользователь хочет зарегистрироваться, при этом указав неверные данные (рисунок 5.4)

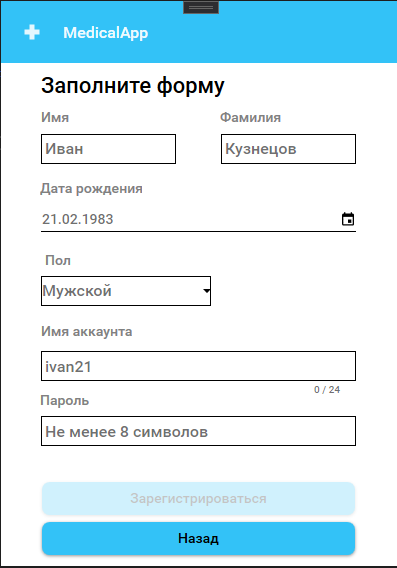


Рисунок 5.3 – «Отсутствие данных»

Как видно на рисунке 5.3, кнопка регистрации просто не будет активна, если не введены данные, необходимые для регистрации.

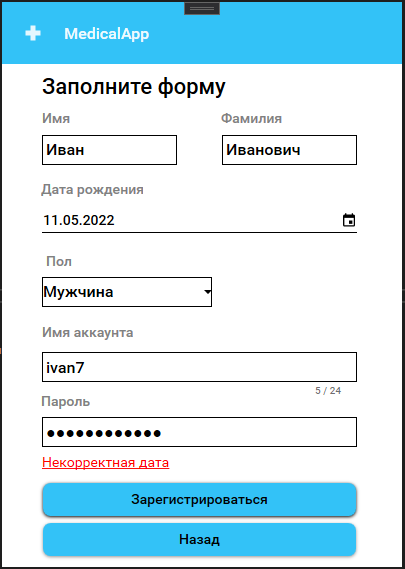


Рисунок 5.4 – «сценарий, при несовпадении паролей»

Также возможен такой исход, при котором пользователь хочет зарегистрироваться при помощи такого логина, который уже есть. Обработка такого рода сценариев показана на рисунке 5.5.

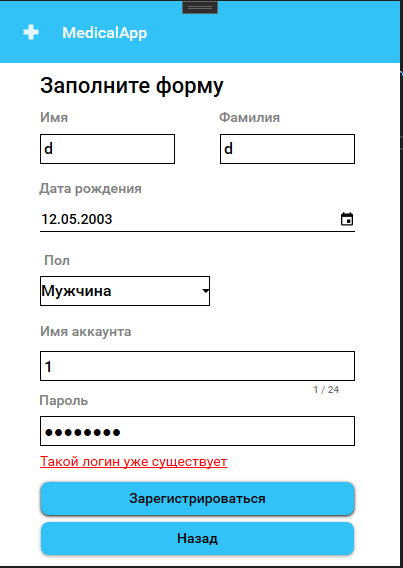


Рисунок 5.5 – «Регистрация пользователя с занятым логином»

Также может возникать такая ситуация, когда пользователь хочет просмотреть забронировать талон или просмотреть историю посещений не заполнив личную карточку пациента, но он сможет этого сделать и т.к. заполнение карточки это обязательное условие, чтобы сделать эти действия. Пример показан на рисунке 5.6

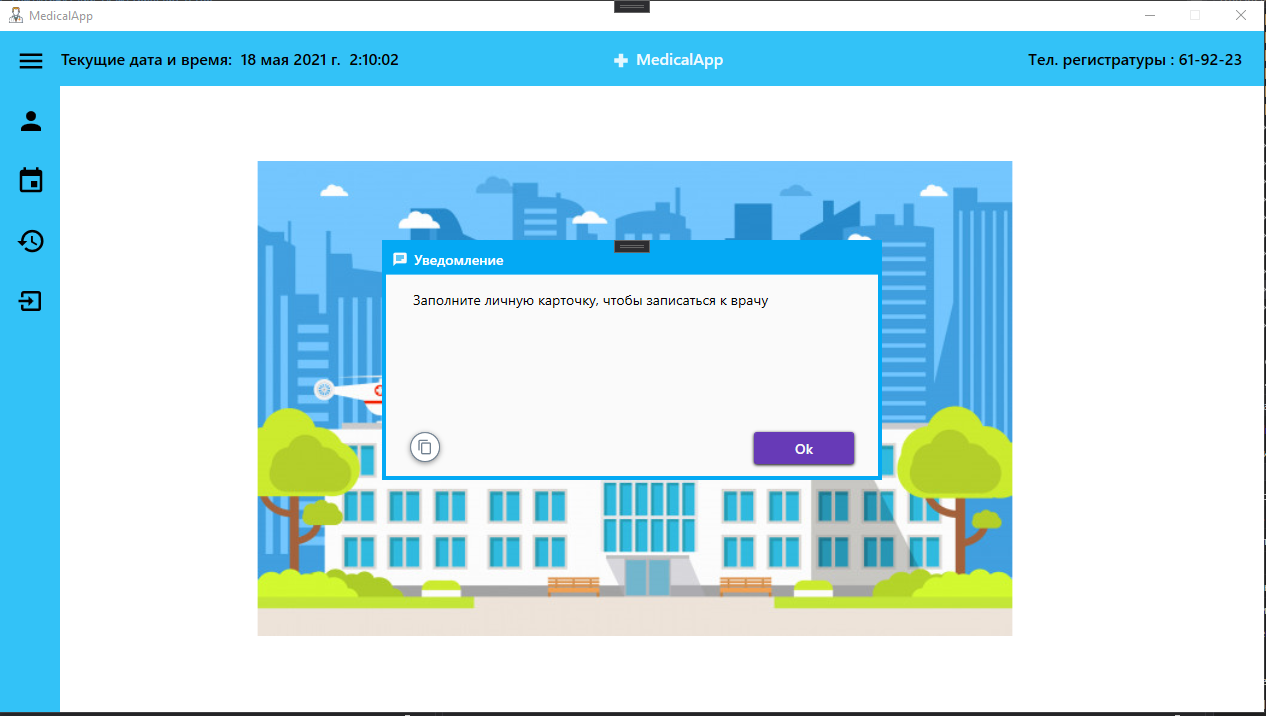


Рисунок 5.6 – «сценарий, записи к врачу, не имея личной карточки»

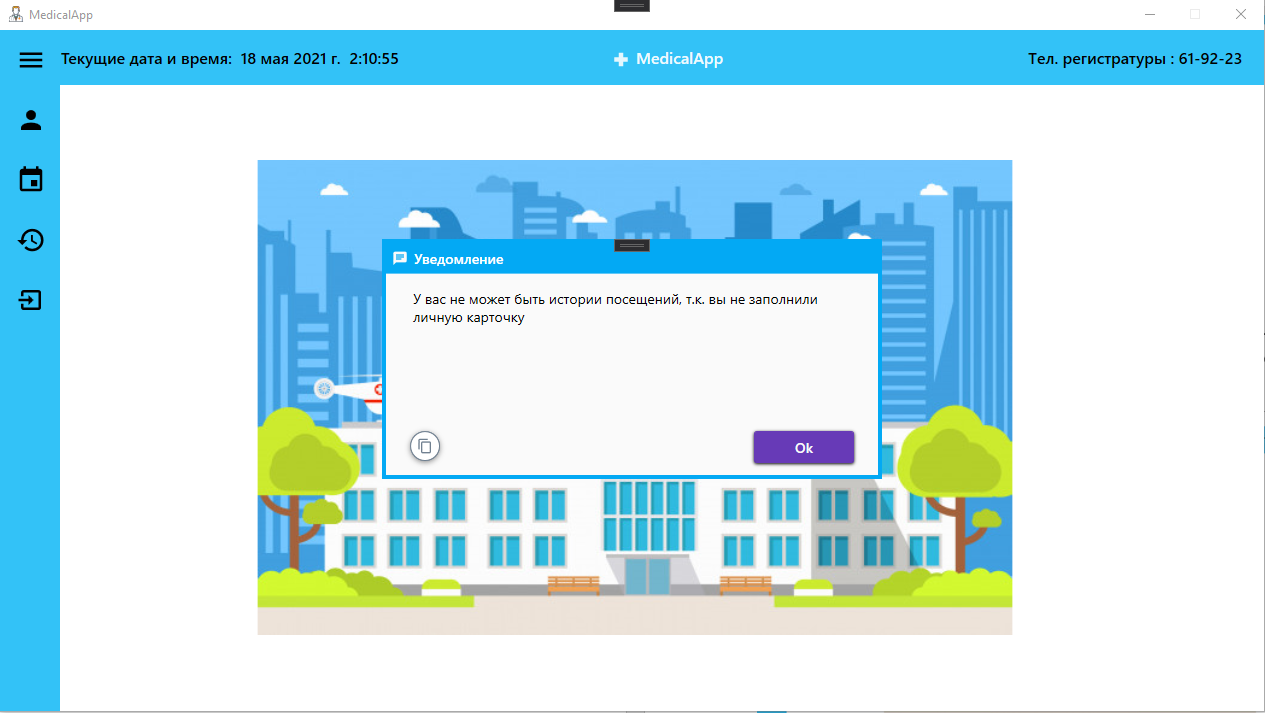


Рисунок 5.6 – «сценарий, просмотра посещений к врачу, не имея личной карточки»

# **Руководство по использованию**

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вас перекинет в окно авторизации и вам нужно ввести ваши данные, указанные при регистрации.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

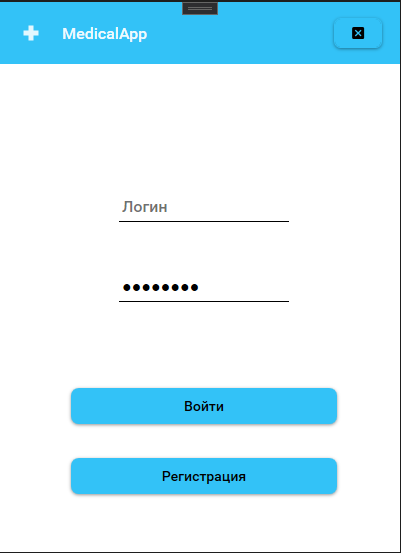
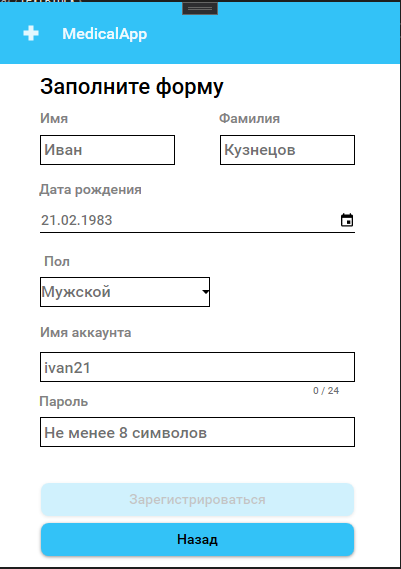


Рисунок 6.1 – «Окно авторизации»

После успешного входа, мы попадаем в главное окно, которое изображено на рисунке 6.2.

В этом окне мы просто видимо логотип поликлиники. Чтобы далее работать с программой нужно сперва заполнить личную карточку.

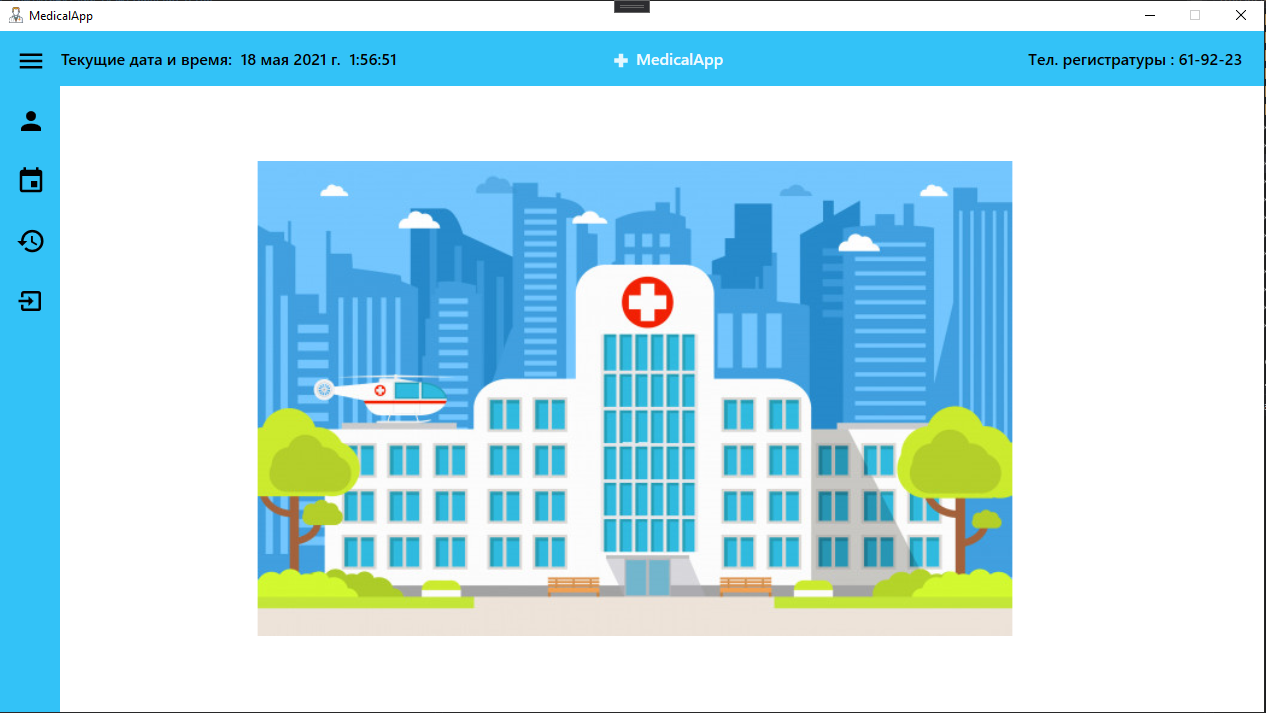


Рисунок 6.2 – «Главное окно»

Для заполнения своей личной карточки, мы переходим на страницу “Личная карта” и заполняем необходимые поля. Есть кнопка, которая позволяет сменить аккаунт.

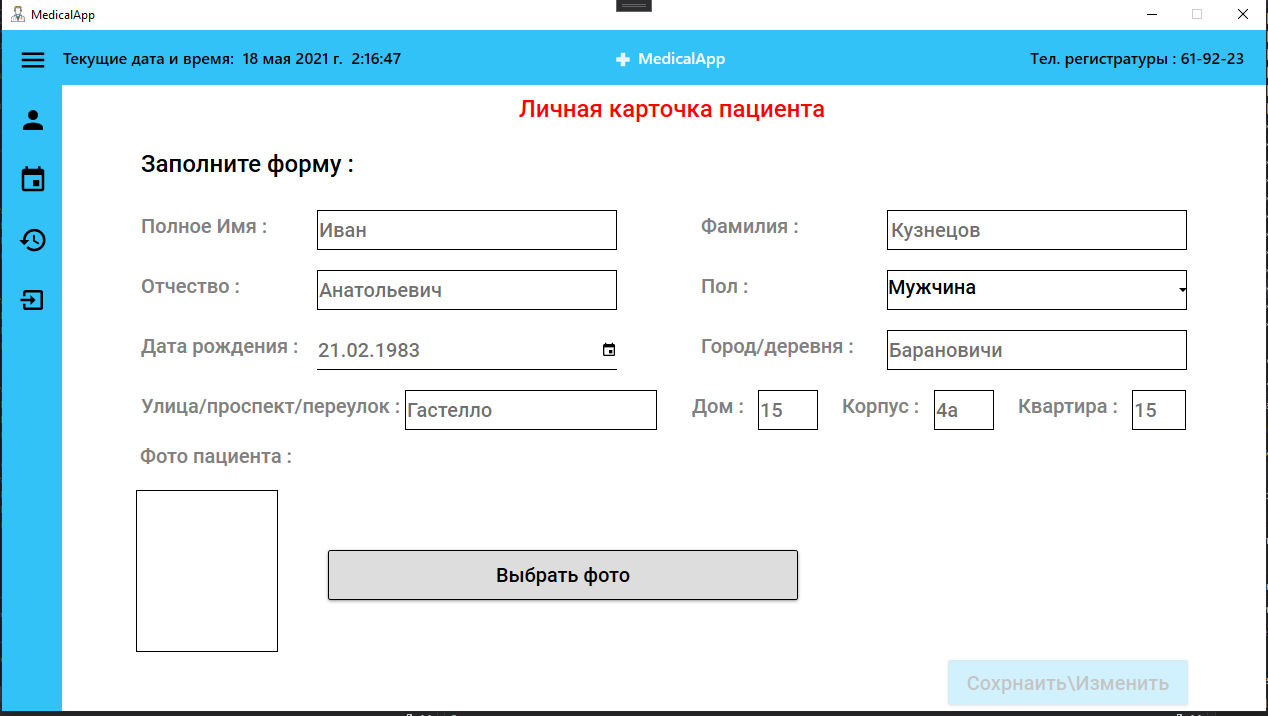
Ниже показан рисунок 6.3, на котором показана страница «Личная карточка».

Рисунок 6.3 – «Страница Личная карточка»

После заполнения личной карты, мы можем переходить на страницу Записи (рисунок 6.4)

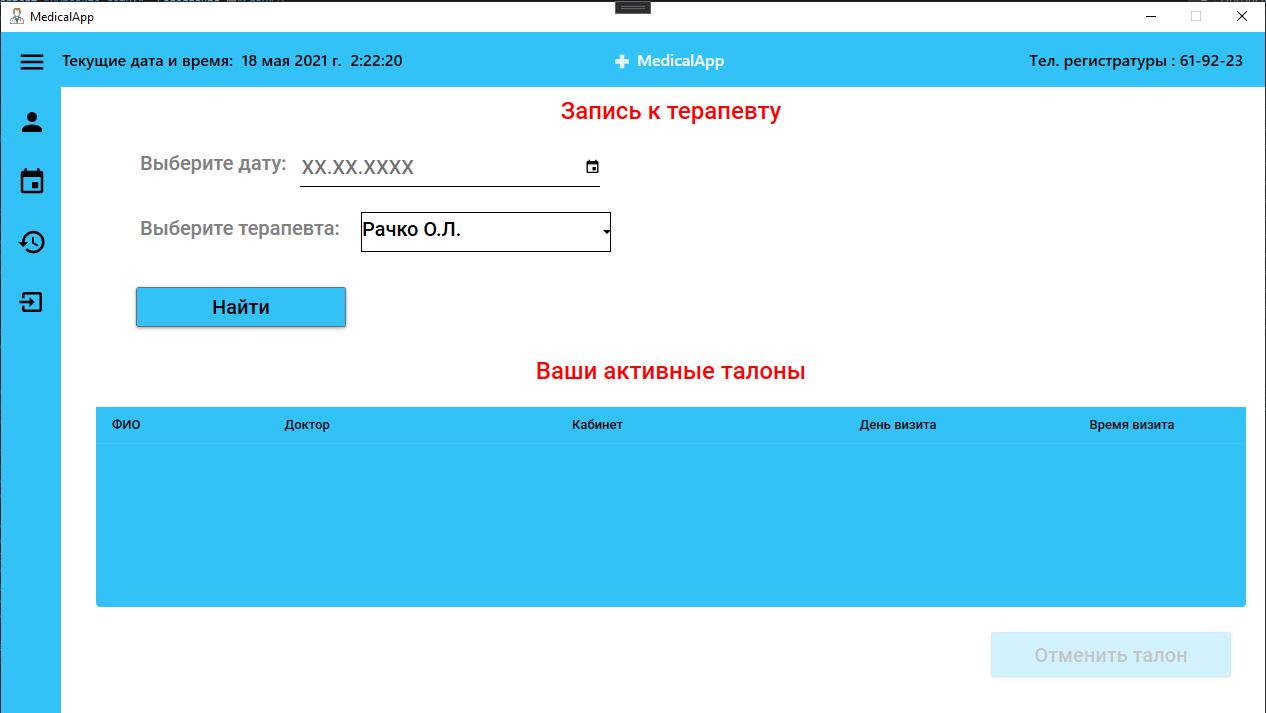


Рисунок 6.4 – «Страница записи к терапевту»

Также на странице «Запись» есть кнопка, нажав на которую будет отменён выделенный талон. Есть поля для ввода даты посещения и врача и кнопка “Найти”, которая открывает окно “Выбора времени и подтверждения”.

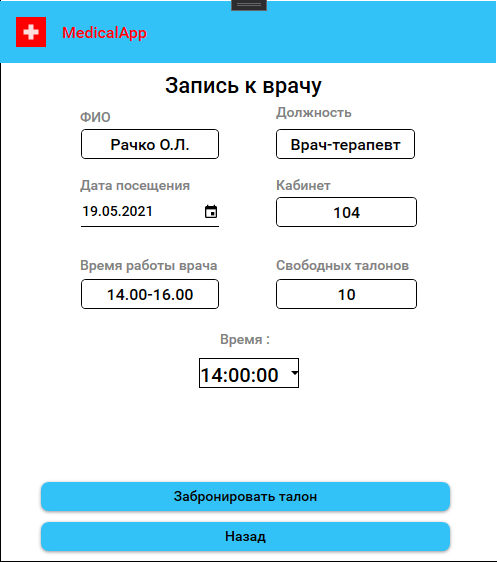
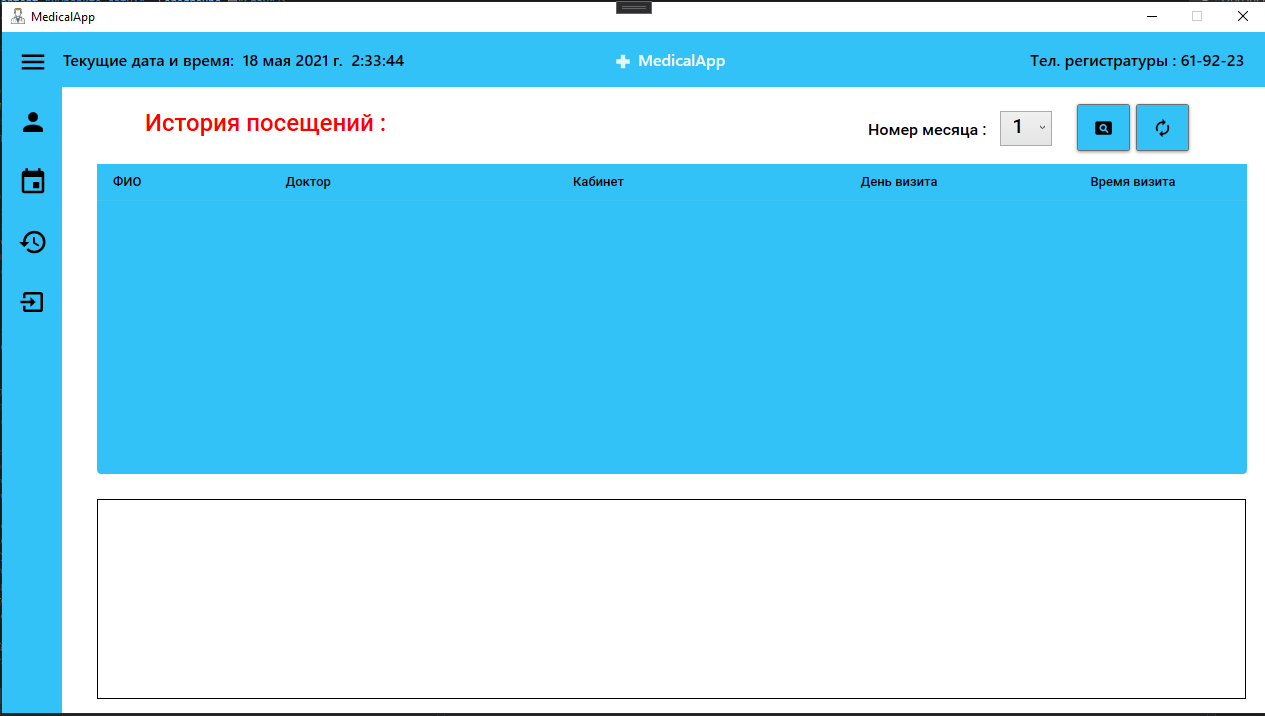


Рисунок 6.5 – «Окно выбора времени и подтверждения»

На рисунке 6.5 показано окно, где пользователь выбирает время посещения врача и подтверждает бронирование талона на это выбранное им время. Также здесь есть поля для чтения, которые показывают ФИО врача, Должность, Дату посещения пациента, Кабинет, Время работы врача, Кол-во свободных таланов и Время визита.

На рисунке 6.6 представлен скриншот страницы истории посещений пациента.

Здесь есть графа для поиска с номером месяца и 2 кнопки, одна из ищет визиты по выбранному номеру месяца, а другая обновляет данные. Есть сортировка по ФИО врача, Должности врача, Кабинет, День визита, Время визита. Также есть блок с текстом, где можно по двойному клику выделенной строки, увидеть в блоке записи, которые оставил врач во время визита пациента.

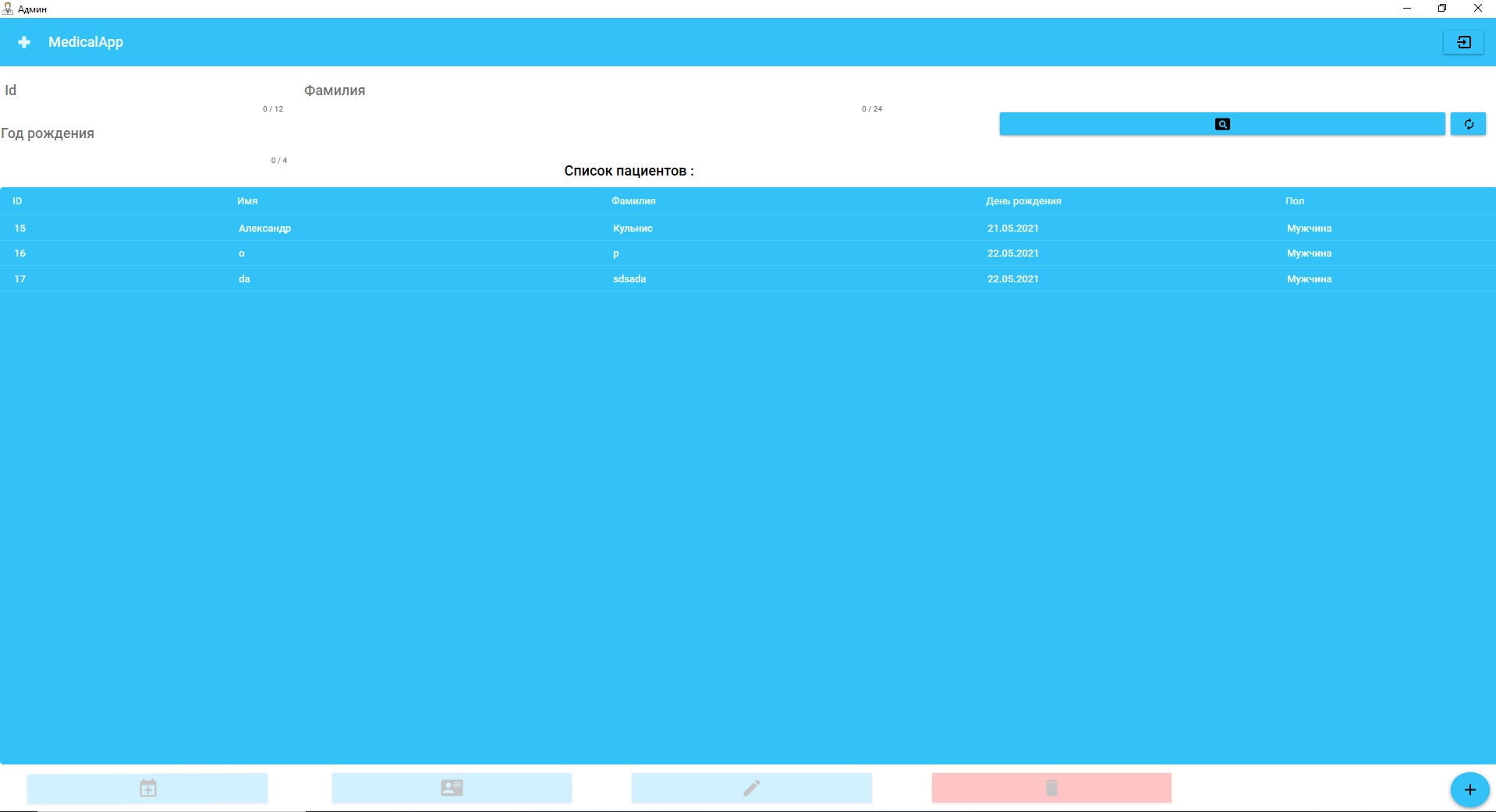


Рисунок 6.7 – «Меню администратора»

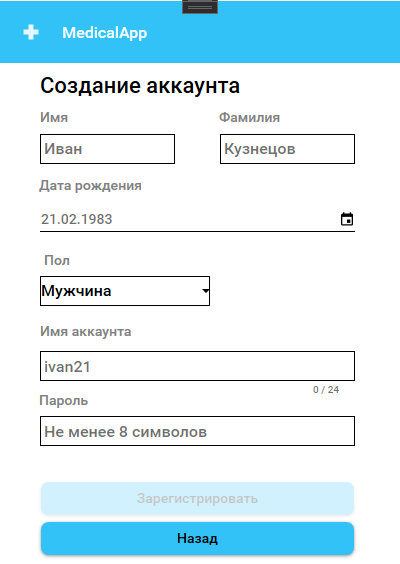
В окно администратора попадаем через окно авторизации. Здесь есть поиск по id пользователей, Фамилии и Год Рождения. Кнопка поиска и обновления данных в таблице. Также есть кнопки добавления, удаления и редактирование пользователей. И последняя кнопка, которая нужна для подтверждения посещения пациента.

Рисунок 6.8 – «Добавления пользователей»

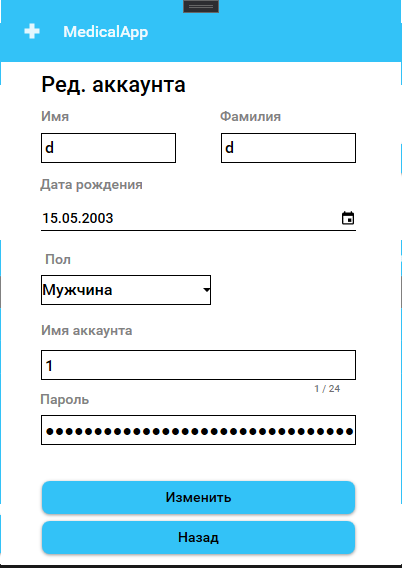
В окне “Добавление пользователей” такой же интерфейс и условия как и при регистрации.

Рисунок 6.9 - «Редактирование аккаунта пользователей»

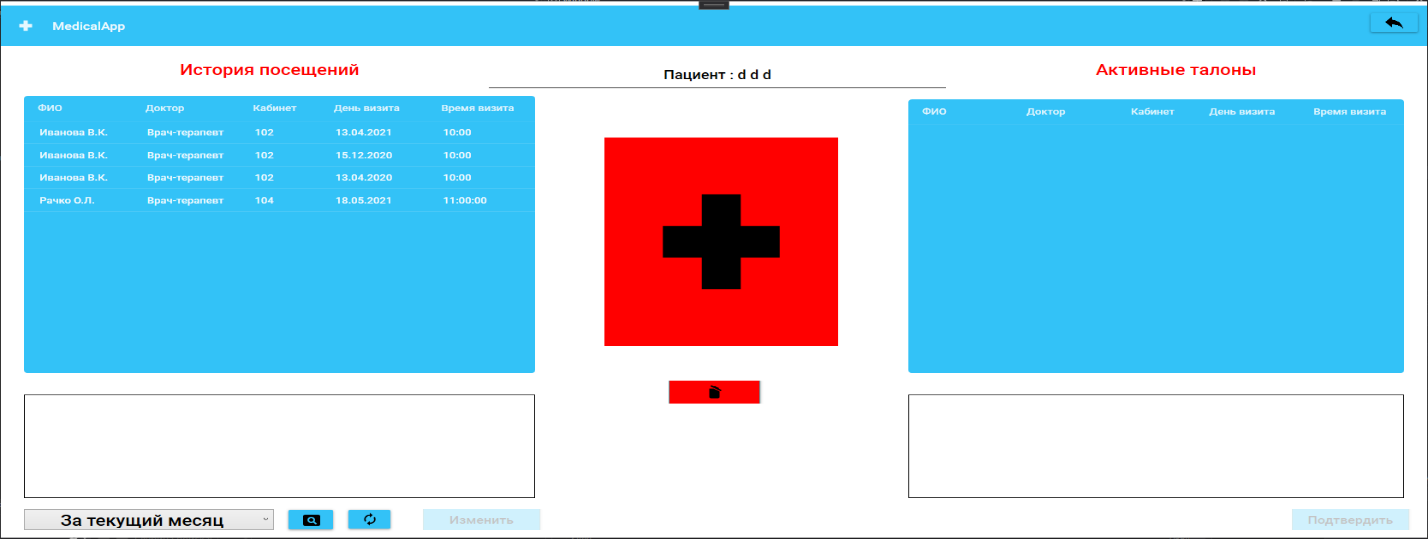
В окне “Редактирование аккаунта пользователя” администратор может менять данные аккаунта. Здесь есть поля с данными аккаунта. Также есть кнопка, которая меняет текущие данные и кнопка, которая возвращает на главную страницу

Рисунок 7.0 - «Подтвердить посещение пациента»

В окне “ Подтвердить посещение пациента” администратор может подтверждать посещение пациента и просматривать его историю посещений. Здесь есть 2 таблицы которые отображают историю посещений и активные талоны. Есть кнопка подтверждения посещения и к ней идёт текстовое поле куда врач записывает свои рекомендации. Есть поиск по различным критериям, к ним добавляются кнопки поиска и обновление данных. Также есть кнопка изменения рекомендаций врача в определенный визит.

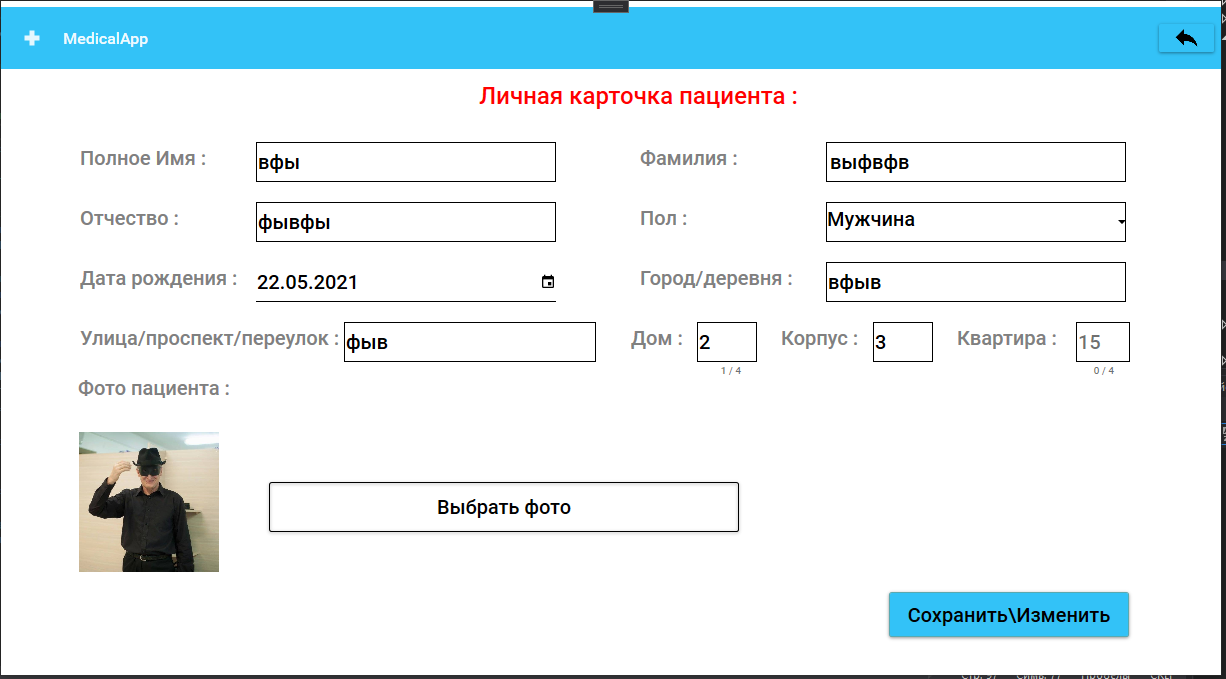


Рисунок 7.1 - «Карточка пациента»

В окне “Карточка пациента” администратор может изменить личную карточку пациента, если изменились его данные.

# **ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте было разработано программное средство «Запись к терапевту» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* Идентификация пользователей;
* Заполнение личной карточки;
* Возможность записаться к врачу на определенную дату;
* Просмотр и отмена забронированных талонов;
* Сортировка талонов по различным критериям;
* Просмотр своих посещений к врачу;
* Поиск и сортировка по различным критериям в истории посещений;
* Видеть текущее время(визуальное);

Администратор имеет следующий функционал:

* Добавление пользователей;
* Удаление пользователей;
* Редактирование аккаунтов пользователей;
* Редактирование личной карточки пользователя;
* Просмотр активных талонов у пользователя.
* Просмотр истории посещений у пользователя.
* Возможность писать рекомендации на определенный визит пациента
* Возможность просмотра и редактирования рекомендации на определенный визит пациента;
* Поиск по различным критериям в истории посещений пациента

Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository.

Приложение хорошо работает с базой данных. В ходе тестирования, не было выявлено нарушений в работе приложения с базой данных.

Также осуществлено взаимодействие с социальными сетями VK и Instagram, а именно вход в приложение, использую данные одной из социальных сетей.

Для разработки дизайна использовалась библиотека MaterialDesign, которая имеет огромное количество красивых и удобных элементов управления.

Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Для хранения исходного кода и удобства контроля версий проекта использовался крупнейший веб-сервис для совместной разработки – GitHub. Ссылка на проект: (https://github.com/alex01-kulnis).

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1]. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2018. – 175 с.

[2]. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с. [3]. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: http://msdn.microsoft.com/library/

[4]. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

[5]. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

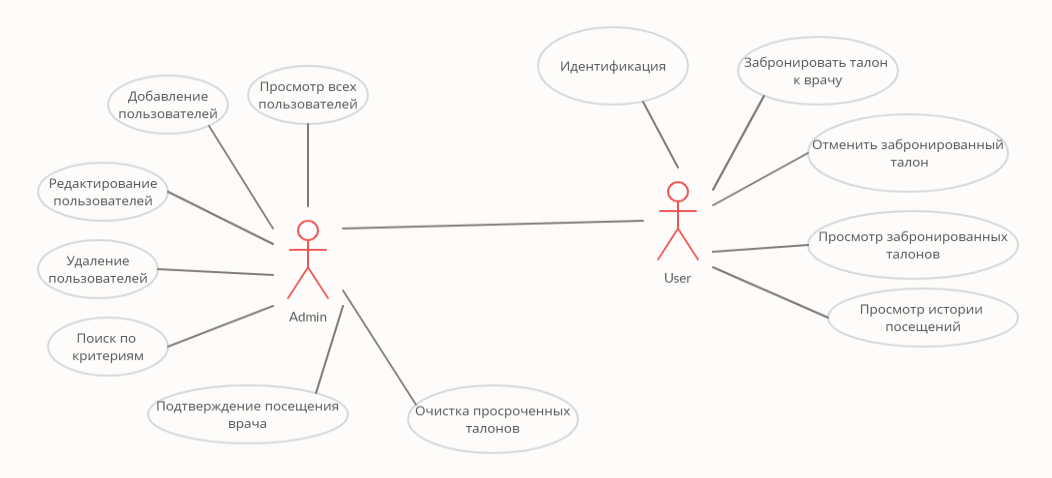
[6]. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

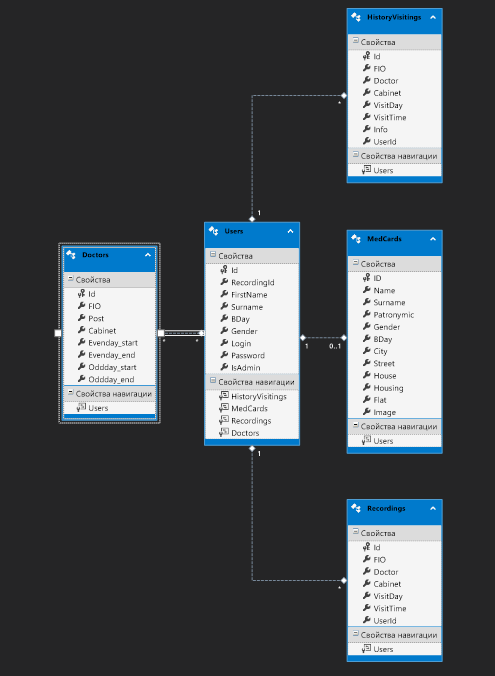
[7]. Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>

[8]. VkNet – Вконтакте API для C# [Электронные данные] / Режим доступа: <https://vknet.github.io/vk/>

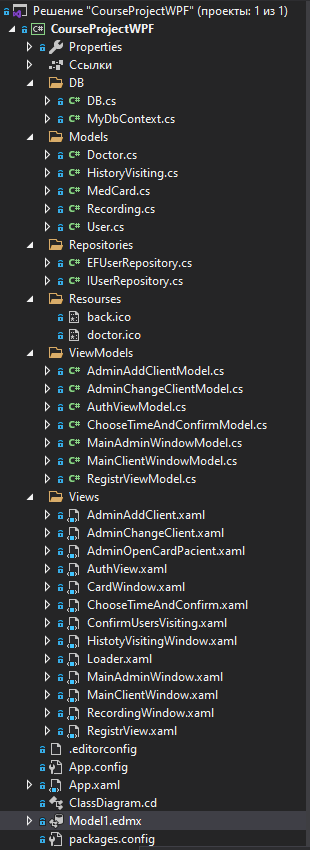
[9]. InstaSharp – API для взаимодействия с Instagram [электронный ресурс] / Режим доступа: <https://github.com/instasharp>

Приложение А



Приложение Б

Приложение В

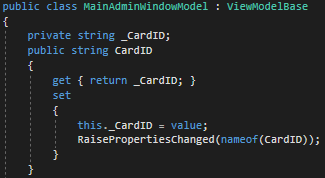


Приложение Г



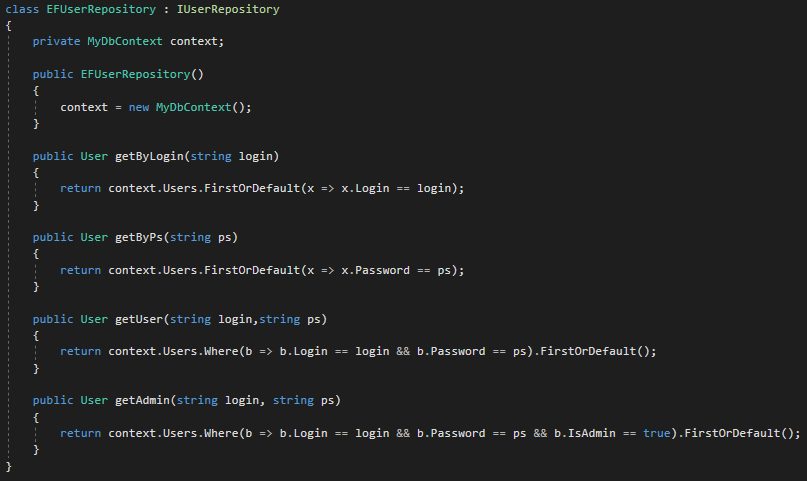
Приложение Д

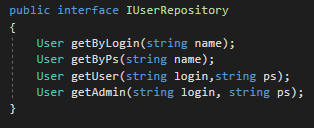
Листинг 1 – «ViewModelBase»



Приложение Е

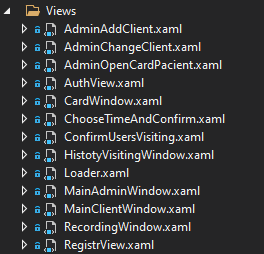
Листинг 2 – «Repositories»





# **Приложение Ж**

Листинг 3 – «Структура пакета View»



# **Приложение З**

Листинг 4 – «Структура пакета ViewModel»

