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 КУРСОВОГО ПРОЕКТА

Разработка программных модулей

**Разработка desktop-приложения «Дневник самочувствия» для отслеживания ключевых показателей здоровья**

|  |  |
| --- | --- |
| Руководитель  Преподаватель ВКИ НГУ | Пауль С.А.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2025г. |
| Студент 3 курса  гр. 2307б1 | Курс Д.С.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2025 г. |

Новосибирск

2025

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ 3](#_heading=h.yhbumixgqv6u)

[ВВЕДЕНИЕ 5](#_heading=h.71k5pumwh1mf)

[1 ПОСТАНОВКА ЗАДАЧИ КП 6](#_heading=h.604nemo6ua57)

[1.1 Пользовательские требования 6](#_heading=h.ph1hxhde7rk2)

[1.2 Системные требования 6](#_heading=h.fmubgujutanu)

[1.3 Требования к графическому пользовательскому интерфейсу 7](#_heading=h.l69ss1nhrlqp)

[1.4 План-график выполнения КП 7](#_heading=h.runaj3bzdclf)

[2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ 9](#_heading=h.m6xxx8lsztu)

[2.1 Описание предметной области задачи КП 9](#_heading=h.wr0vhpp6agq5)

[2.1.1 Информационные объекты предметной области и взаимосвязи между ними 9](#_heading=h.xv4z3l8btagu)

[2.1.2 Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ) 10](#_heading=h.fb436xq1rwjt)

[2.1.3 Обзор существующих программных реализаций решения задачи 11](#_heading=h.y31rqyujlxnp)

[2.1.4 Концептуальное обоснование разработки 12](#_heading=h.exeeld4gmaov)

[2.2 Классы и характеристики пользователей 12](#_heading=h.olmmpl8043sb)

[2.3 Функциональные требования 12](#_heading=h.2e5mxw9hrlc3)

[2.4 Нефункциональные требования 13](#_heading=h.qmh0k2nnl6cr)

[3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ 14](#_heading=h.mov08z6yau9g)

[4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 16](#_heading=h.1wv3xej4aza3)

[4.1 Этапы реализации ПС 16](#_heading=h.2q7i6kur3oq8)

[4.2 Пользовательский интерфейс ПС 16](#_heading=h.1alfy53bx2pa)

[4.2.1 Взаимодействие пользователя с ПС 16](#_heading=h.lw6p74awpgqg)

[4.2.2 Проектирование структуры экранов ПС и схемы навигации 17](#_heading=h.my71zemrxg6l)

[4.3 Входные, выходные и промежуточные данные 19](#_heading=h.q50c9zqhseh)

[4.4 Разработка базы данных, реализуемой в рамках ПС 19](#_heading=h.75jb0dcm31sl)

[4.5 Архитектура и схема функционирования ПС (ПМ) 24](#_heading=h.r80gcvdkpxpa)

[5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ 27](#_heading=h.51uv80hnrp5n)

[5.1 План тестирования 27](#_heading=h.z4rep83mhglc)

[5.2 Результаты тестирования 27](#_heading=h.xfvmcthd1g6k)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 29](#_heading=h.72ta57dxg3kt)

[ЗАКЛЮЧЕНИЕ 39](#_heading=h.inynkdbnl90l)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 40](#_heading=h.6hezungqi4et)

[ПРИЛОЖЕНИЯ 41](#_heading=h.r114lv3l1mwn)

[Приложение А 41](#_heading=h.75axkni24wxz)

[Приложение Б 44](#_heading=h.knllfuxiyhqs)

[Приложение В 54](#_heading=h.49th6d2ve6j8)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

Десктопное приложение – программа, которая устанавливается на компьютер пользователя и работает под управлением операционной системы.

XAML – язык разметки, используемый для инициализации объектов в технологиях на платформе .NET.

WPF – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML

Скрипт – последовательность действий, описанных с помощью скриптового языка программирования для автоматического выполнения определенных задач.

Пользовательский интерфейс (UI) – интерфейс, обеспечивающий передачу информации между пользователем и программно-аппаратным компонентами компьютерной системы.

.NET Framework – программная платформа от Microsoft, предоставляющая среду выполнения для приложений и библиотеку классов для разработки.

АД (Артериальное давление) – один из ключевых показателей состояния сердечно-сосудистой системы, отслеживаемый в приложении.

БД (База данных) – упорядоченное электронное хранилище информации, которое систематизирует большие объемы данных, позволяя эффективно их хранить, извлекать.

СУБД (Система управления базами данных) – программное обеспечение для управления базами данных и работы с ними.

Visual Studio – среда разработки от Microsoft, предназначенная для разработки программного обеспечения: от написания и редактирования кода до отладки, сборки и развертывания приложений.

LINQ (Language Integrated Query, интегрированный язык запросов) — это компонент платформы .NET, который добавляет возможности выполнения запросов к языку программирования C#. Он позволяет запрашивать и изменять данные из различных источников (например, коллекции, базы данных) с использованием стандартизированного синтаксиса.

# ВВЕДЕНИЕ

В мире, характеризующемся высоким темпом жизни и нагрузками на организм человека, вопрос сохранения здоровья и контроля самочувствия приобретают особую актуальность. Регулярный мониторинг ключевых показателей здоровья становится частью профилактики заболеваний и поддержания качества жизни. Традиционные методы ведения дневников самочувствия в бумажном виде часто оказываются неудобными и недостаточно эффективными для анализа динамики состояния здоровья.

В этом контексте особую значимость приобретает разработка специализированного программного обеспечения, позволяющего автоматизировать процесс контроля самочувствия. «Дневник самочувствия» представляет собой современное решение для систематического отслеживания физиологического состояния. Приложение позволяет регистрировать и анализировать такие важные параметры, как артериальное давление, вес, физическую активность, качество сна, прием лекарственных препаратов и субъективные оценки самочувствия.

Использование цифровых технологий для ведения дневника здоровья предоставляет пользователям ряд существенных преимуществ: возможность быстрого ввода данных, автоматическое формирование отчетов за различные периоды времени, визуализацию динамики показателей, установление напоминаний о приеме лекарств и проведении измерений. Кроме того, электронный формат хранения информации обеспечивает ее сохранность и удобство поиска.

Разрабатываемое десктопное приложение на платформе WPF предлагает интуитивно понятный пользовательский интерфейс, надежное хранение данных и комплексный подход к мониторингу состояния здоровья. Это решение направлено на повышение эффективности контроля самочувствия и способствует формированию ответственного отношения к собственному здоровью.

# 1 ПОСТАНОВКА ЗАДАЧИ КП

Целью курсового проекта является разработка и реализация desktop-приложения «Дневник самочувствия» для операционной системы Windows, предназначенного для персонального учета и анализа ключевых показателей здоровья пользователя.

В качестве основной технологии разработки выбрана платформа Windows Presentation Foundation (WPF) с использованием языка программирования C# и языка разметки XAML для создания графического интерфейса. Для хранения данных используется встраиваемая СУБД SQLite.

## Пользовательские требования

* Возможность регистрации нового пользователя и авторизации в системе;
* Ежедневный ввод и редактирование данных по категориям: артериальное давление, самочувствие, физическая активность, сон и вес;
* Управление списком лекарственных средств: добавление, редактирование и удаление. Настройка расписания приема;
* Формирование наглядных отчетов за выбранный период: графическое представление динамики АД, веса, настроения, физической активности, качества сна и статистических сводок (средних значений, минимума, максимума);
* Просмотр истории всех внесенных записей.

## Системные требования

* Операционная система: Windows 10 и новее;
* Процессор: 1 ГГц или выше, совместимый с архитектурой x64;
* Оперативная память: 2 ГБ (рекомендуется 4 ГБ);

## 1.3 Требования к графическому пользовательскому интерфейсу

Требования к внешнему виду пользовательского интерфейса:

Понимание и удобство в использовании: интерфейс должен быть таковым, чтобы большинство пользователей могли интуитивно понять его функционал. Простые и понятные элементы управления способствуют более быстрому освоению приложения.

Гибкость по отношению к различным разрешениям экрана: необходима поддержка разнообразных разрешений для обеспечения удобства использования приложения на экранах с различными параметрами. Это позволяет адаптировать интерфейс под разные условия эксплуатации.

Графический пользовательский интерфейс должен обеспечивать полный доступ ко всем функциям приложения. Это включает в себя не только управление основными функциональными возможностями, но и предоставление пользователю возможности максимального взаимодействия с приложением через интуитивно понятный интерфейс.

## 1.4 План-график выполнения КП

Чтобы разработать приложение был выбран следующий план-график:

Таблица 1 – План график выполнения КП

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ожидаемое время | Реальное время | Задача |
| 1. | 1 неделя | 1 неделя | Анализ предметной области и формулировка требований к разработке |
| 2. | 1 неделя | 1 неделя | Выбор и обоснование технологий разработки |
| 3. | 1 неделя | 2 недели | Проектирование архитектуры приложения |
| 4. | 2 недели | 3 недели | Разработка пользовательского интерфейса |
| 5. | 2 недели | 2 недели | Реализация логики приложения и работы с базой данных |
| 6. | 1 неделя | 2 недели | Написание пояснительной записки и оформление документации |

# 2 АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

## 2.1 Описание предметной области задачи КП

Предметная область проекта охватывает сферу персонального контроля здоровья и самочувствия. Ключевая проблема заключается в отсутствии у пользователя систематизированного и удобного для анализа представления данных о своем состоянии. Разрозненные записи и бумажные дневники, а также данные в различных приложениях не позволяют выявлять данные корректно.

Разрабатываемое приложение призвано стать единым элементом для учета всех значимых аспектов здоровья. Оно позволяет не только производить запись данных, но и превращает их в информацию за счет визуализации. Это дает пользователю возможность объективно оценить эффективность лечения или изменений в образе жизни, своевременно замечать негативные тенденции, соблюдать режим приема лекарств и начать лучше понимать собственный организм.

## 2.1.1 Информационные объекты предметной области и взаимосвязи между ними

В приложении «Дневник самочувствия» выделяются следующие основные информационные объекты:

Пользователь – физическое лицо, зарегистрировавшееся в системе для ведения личного дневника здоровья. Объект содержит персональные и аутентификационные данные.

Дневная запись (DailyData) – ключевой объект, представляющий совокупность показателей здоровья, введенных пользователем за конкретный день. Связывается с пользователем и может включать данные о давлении, сне, активности и т.д.

Лекарство (Medicine) – объект, описывающий лекарственный препарат, который принимает пользователь. Содержит информацию о названии, дозировке, графике приема и связи с пользователем.

Симптом (Symptom) – справочный объект, описывающий возможный симптом плохого самочувствия (например, головная боль, слабость). Может быть связан с дневными записями.

Отчет (Report) – объект, представляющий собой сформированный по запросу пользователя аналитический отчет за выбранный период. Содержит агрегированные данные, графики и связь с пользователем.

## 2.1.2 Информационные и функциональные потребности пользователей разрабатываемой ПС

Приложение «Дневник самочувствия» призвано удовлетворить следующие потребности пользователя:

* Систематизация данных о здоровье: регулярное внесение показателей о здоровье (АД, вес, сон, активность, настроение, прием лекарств).
* Анализ и отслеживание динамики: получение наглядного представления об изменениях ключевых показателей во времени через графики и отчеты для выявления закономерностей и тревожных тенденций.
* Дисциплина: управление графиком приема лекарств для формирования привычки.
* Сохранение истории: хранение и быстрый доступ ко всей истории записей для предоставления информации врачу или самостоятельного анализа.
* Удобство и доступность: простой, интуитивный интерфейс, позволяющий быстро вносить и просматривать данные.

## 2.1.3 Обзор существующих программных реализаций решения задачи

На рынке представлено множество приложений для отслеживания здоровья, которые можно разделить на несколько категорий:

* Узкоспециализированные приложения: HeartWatch для мониторинга сердечного ритма и ЭКГ; Sleep Cycle для анализа сна. Эти приложения предлагают углубленный анализ в своей области, но не покрывают все аспекты здоровья комплексно.
* Универсальные трекеры здоровья и фитнеса: Google Fit, Apple Health, Samsung Health. Они агрегируют данные с различных устройств и приложений, предоставляя общую картину. Однако их возможности по ручному структурированному вводу данных (особенно медицинских показателей и субъективных ощущений) и кастомизации под индивидуальные нужды часто ограничены.
* Медицинские дневники и журналы: MyTherapy, Medisafe (акцент на приеме лекарств), Bearable. Данные приложения ближе всего по концепции к разрабатываемому проекту, позволяя вносить широкий спектр симптомов, показателей и лекарств. Их недостатком для части пользователей может быть излишняя сложность, перегруженность функциями или отсутствие полноценной desktop-версии.

Конкурентное преимущество разрабатываемого desktop-приложения «Дневник самочувствия» заключается в его простоте, фокусе на основных показателях, работе в среде Windows без привязки к мобильной ОС и наличии локального хранения данных, что может быть важно для пользователей.

## 2.1.4 Концептуальное обоснование разработки

Разработка данного приложения обоснована растущей потребностью людей в контроле своего здоровья в условиях высокой повседневной нагрузки. Бумажные дневники неудобны для анализа, а разрозненные цифровые записи (в заметках, календарях, разных приложениях) не позволяют получить целостную картину. Существующие аналоги часто являются либо частью экосистем конкретных производителей, либо перегружены функциями.

Цель проекта – создать доступный, понятный и эффективный инструмент для ежедневного мониторинга, который поможет пользователю сформировать привычку регулярно отслеживать ключевые показатели, иметь структурированные данные для консультации с врачом.

Таким образом, приложение заполняет нишу персонального desktop-трекера здоровья с акцентом на удобство, конфиденциальность локальных данных и достаточный для рядового пользователя функционал.

## 2.2 Классы и характеристики пользователей

Приложение содержит только одну роль – Пользователь. Это человек, который использует приложение для ведения собственного дневника здоровья. Он может обладать разным уровнем компьютерной грамотности, поэтому интерфейс должен быть интуитивно понятным. В процессе взаимодействия с приложением работает с персональными данными.

## 2.3 Функциональные требования

Пользователь может войти/выйти из приложения, свернуть окно, зарегистрироваться, ввести данные, отредактировать данные, управлять лекарствами, формировать отчеты и просматривать истории записей. Диаграмма взаимодействия отображена на Рисунке 1.

Изображение выглядит как диаграмма, линия, зарисовка, рисунок

Содержимое, созданное искусственным интеллектом, может быть неверным.Рисунок 1 – UML-диаграмма для пользователя «Пользователь»

## 2.4 Нефункциональные требования

* Отклик интерфейса на действия пользователя без видимых задержек;
* Устойчивая работа приложения, недопустимы критические сбои;
* Соответствие интерфейса принципам интуитивной понятности, освоение базовых функций новым пользователем должно занимать не более 15 минут;
* Валидация входных данных, проверка временных интервалов.

# 3 ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ

На основе поддержки от Microsoft графических интерфейсов на 2025 г., был выбран WPF и соответствующий ему язык программирования C# с языком разметки XAML. Для работы с БД выбран Entity Framework Core – модуль для работы с SQLite. В проекте используется LiveCharts.WPF – библиотека для построения графиков и диаграмм в отчетах.

В качестве версии платформы для приложения был выбран .NET 8.0, так как он обладает всеми необходимыми функциями и при этом является версией с долгосрочной поддержкой.

Рассмотрим подробнее выбранные средства:

* Выбор языка программирования C# обоснован его безупречной интеграцией с платформой .NET и мощными возможностями объектно-ориентированного программирования. C# является статически типизированным языком, что способствует высокой производительности и предотвращает множество потенциальных ошибок на этапе компиляции. Благодаря поддержке от Microsoft и активному сообществу разработчиков, C# постоянно обновляется и совершенствуется, предоставляя средства для создания эффективных и надежных приложений.
* Выбор WPF обусловлен его мощными средствами для создания современных графических интерфейсов. WPF предоставляет разработчикам широкий набор инструментов для создания интерактивных и стильных пользовательских интерфейсов, что является важным фактором для обеспечения привлекательности и удобства использования приложения. Технология WPF также обладает гибкостью и возможностью масштабирования, что важно для разработки приложений с разнообразными требованиями.
* Библиотека LiveCharts.WPF выбрана благодаря своей гибкости, производительности и богатым возможностям для создания интерактивных, высококачественных диаграмм, особенно в сценариях с данными реального времени, используя его мощную интеграцию с XAML и .NET, что делает её идеальным инструментом для визуализации сложных данных в современных настольных приложениях.
* SQLite используется для небольших проектов. Это связано с простотой, портативностью и отсутствием необходимости в сервере. Она встраивается прямо в приложение как один файл, не требуя сложной установки или настройки, что идеально для десктопных приложений, где не нужно многопользовательское одновременное соединение.
* LINQ интегрирует запросы в язык программирования C#, позволяя писать выразительный код для работы с данными в среде разработки. Интегрированный язык запросов позволяет легко фильтровать, сортировать, делать выборки данных.

# 4 АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

## 4.1 Этапы реализации ПС

Этапы реализации ПС:

1. Аналитический этап – изучение технического задания, анализ подобных программ (приложений для здоровья), постановка требований.
2. Этап проектирования – разработка диаграммы базы данных, проектирование классов-моделей, создание интерфейса и карты навигации.
3. Этап реализации инфраструктуры – настройка проекта в Visual Studio, подключение NuGet-пакетов для работы с SQLite и LiveCharts.WPF. Создание структуры БД.
4. Этап разработки – написание кода обработчиков событий в классах окон, которые реализуют валидацию данных, формирование отчетов и взаимодействие с базой данных через Entity Framework Core.
5. Этап реализации пользовательского интерфейса – верстка окон на XAML.
6. Этап тестирования – проверка функций, исправление ошибок и оптимизация производительности
7. Заключительный этап – завершение работы с проектом, оформление документации.

## 4.2 Пользовательский интерфейс ПС

### 4.2.1 Взаимодействие пользователя с ПС

Пользователь взаимодействует с программой способом, показанным на рисунке 2.

Изображение выглядит как текст, диаграмма, линия, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – Взаимодействие пользователя с программным средством

### 4.2.2 Проектирование структуры экранов ПС и схемы навигации

В ПС представлено шесть экранов: окно авторизации/регистрации – точка входа в приложение; главное окно, которое содержит сводку за день, кнопки навигации; окно детального ввода данных – форма с вкладками для ввода данных каждой категории; окно управления лекарствами – список препаратов, форма для добавления и редактирования; окно отчетов, позволяющее выбирать тип отчета, период и отображает результат в виде графиков и таблиц; окно просмотра данных пользователя. Когда пользователь открывает программу, появляется окно авторизации. Схема навигации между экранами представлена на рисунке 3.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – Схема навигации в ПС

0 – вход пользователя на окно авторизации на главную страницу

1 – авторизация пользователя

2 – переход на страницу просмотра самочувствия

3 – возврат в окно авторизации

4 – просмотр дневных данных

5 – нажатие на кнопку «Контроль лекарств»

6 – переход на страницу создания списка лекарств  
7 – создание списка лекарств

8 – удаление списка лекарств  
9 – редактирование списка лекарств

10 – нажатие на кнопку «Главная»

11 – возврат на главную страницу

12 – нажатие на кнопку «Отчеты»

13 – переход на страницу формирования отчетов  
14 – выбор типа отчета

15 – выбор интервала формирования отчета

16 – нажатие на кнопку «Сформировать отчет»

17 – просмотр отчета

18 – нажатие на кнопку «Внести данные»

19 – переход на страницу внесения данных

20 – заполнение дневных данных

21 – сохранение показателей

22 – нажатие на кнопку «Профиль»

23 – переход на страницу просмотра профиля

24 – просмотр данных пользователя

25 – редактирование данных пользователя

26 – сохранение изменений

## 4.3 Входные, выходные и промежуточные данные

* Входные данные:
  + Вводимые пользователем значения через формы (числа, текстовые данные, дата/время)
  + Конфигурационные данные (настройки приложения)
* Промежуточные данные:
  + Объекты моделей-классов в оперативной памяти (User, Medicine, DailyData)
* Выходные данные:
  + Визуальные: графики, диаграммы, отображаемые на экране в разделе «Отчеты»
  + Структурированные: таблицы (плитки) с историей записей

## 4.4 Разработка базы данных, реализуемой в рамках ПС

Для хранения данных пользователей спроектирована база данных SQLite. Она создается при первом запуске приложения с помощью технологии Entity Framework Core и, если уже существует, то к ней происходит подключение. В базе данных реализовано 5 таблиц: Users, DailyData, Medicines, Reports, Symptoms. Следует учитывать, что SQLite имеет только четыре примитивных типа данных: INTEGER, REAL, TEXT и BLOB.

В таблице User хранятся все пользователи сервиса. В таблице 2 подробно представлены поля этой таблицы.

Таблица 2 – Поля таблицы User

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | INTEGER | Идентификатор пользователя |
| Name | TEXT | Имя пользователя |
| Gender | TEXT | Пол пользователя (мужской/женский) |
| Height | INTEGER | Рост пользователя |
| Weight | REAL | Вес пользователя |
| DateOfBirth | TEXT | Дата рождения пользователя |
| Login | TEXT | Логин пользователя |
| Password | TEXT | Пароль пользователя |
| PhotoPath | TEXT | Путь к картинке пользователя, допускает значение NULL |

В таблице Symptoms хранятся основные симптомы, характерные для человека. В таблице 3 подробно представлены поля этой таблицы.

Таблица 3 – Поля таблицы Symptoms

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | INTEGER | Идентификатор симптома |
| Name | TEXT | Название симптома |

В таблице DailyData хранятся все данные, внесенные за какой-либо день. В таблице 4 подробно представлены поля этой таблицы.

Таблица 4 – Поля таблицы DailyData

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | INTEGER | Идентификатор дневной записи |
| UserId | INTEGER | Внешний ключ к таблице User, поле хранит идентификатор пользователя |
| Date | TEXT | Дата создания дневной записи |
| SystolicPressure | INTEGER | Значение систолического АД в дневной записи, допускает значение NULL |
| DiastolicPressure | INTEGER | Значение диастолического АД в дневной записи, допускает значение NULL |
| Pulse | INTEGER | Значение пульса в дневной записи, допускает значение NULL |
| WellBeingRating | INTEGER | Значение самочувствия в дневной записи, допускает значение NULL |
| Symptoms | TEXT | Симптомы в дневной записи, допускает значение NULL |
| Notes | TEXT | Заметки в дневной записи, допускает значение NULL |
| StepCount | INTEGER | Шаги в дневной записи, допускает значение NULL |
| SleepStart | TEXT | Начало сна в дневной записи, допускает значение NULL |
| SleepEnd | TEXT | Конец сна в дневной записи, допускает значение NULL |
| SleepQuality | INTEGER | Качество сна в дневной записи, допускает значение NULL |
| NightAwake | INTEGER | Количество ночных пробуждений в дневной записи, допускает значение NULL |
| Weight | REAL | Значение веса в дневной записи, допускает значение NULL |
| CreateAt | TEXT | Время создания дневной записи |
| UpdateAt | TEXT | Время обновления дневной записи |

В таблице Medicines хранятся все лекарственные средства. В таблице 5 подробно представлены поля этой таблицы.

Таблица 5 – Поля таблицы Medicines

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | INTEGER | Идентификатор лекарства |
| UserId | INTEGER | Внешний ключ к таблице User, поле хранит идентификатор пользователя |
| MedicineName | TEXT | Название лекарства |
| Dose | TEXT | Информация о дозировке лекарства, допускает значение NULL |
| Instructions | TEXT | Инструкции к применению лекарства, допускает значение NULL |
| StartDate | TEXT | Дата начала приема лекарства |
| EndDate | TEXT | Дата окончания приема лекарства, допускает значение NULL |
| IsActive | INTEGER | Показывает, принимается лекарство сейчас или нет |
| IntakeTime | TEXT | Время (чч:мм) приема лекарства |
| Monday | INTEGER | Показывает, принимается ли лекарство в понедельник |
| Tuesday | INTEGER | Показывает, принимается ли лекарство во вторник |
| Wednesday | INTEGER | Показывает, принимается ли лекарство в среду |
| Thursday | INTEGER | Показывает, принимается ли лекарство в четверг |
| Friday | INTEGER | Показывает, принимается ли лекарство в пятницу |
| Saturday | INTEGER | Показывает, принимается ли лекарство в субботу |
| Sunday | INTEGER | Показывает, принимается ли лекарство в воскресенье |
| CreateAt | TEXT | Время создания лекарства |
| UpdateAt | TEXT | Время обновления лекарства |

В таблице Reports хранятся все лекарственные средства. В таблице 6 подробно представлены поля этой таблицы.

Таблица 6 – Поля таблицы Reports

|  |  |  |
| --- | --- | --- |
| Название поля | Тип поля | Описание |
| Id | INTEGER | Идентификатор отчета |
| UserId | INTEGER | Внешний ключ к таблице User, поле хранит идентификатор пользователя |
| ReportType | TEXT | Тип отчета |
| StartDate | TEXT | Дата начала временного интервала формирования отчета |
| EndDate | TEXT | Дата окончания временного интервала формирования отчета |
| ReportData | TEXT | Текстовое содержимое отчета |
| Summary | TEXT | Вывод по отчету, допускает значение NULL |
| Recommends | INTEGER | Рекомендации к отчету, допускает значение NULL |
| CreateAt | TEXT | Время сохранения отчета |

На рисунке 4 представлена ER-диаграмма – схема базы данных.

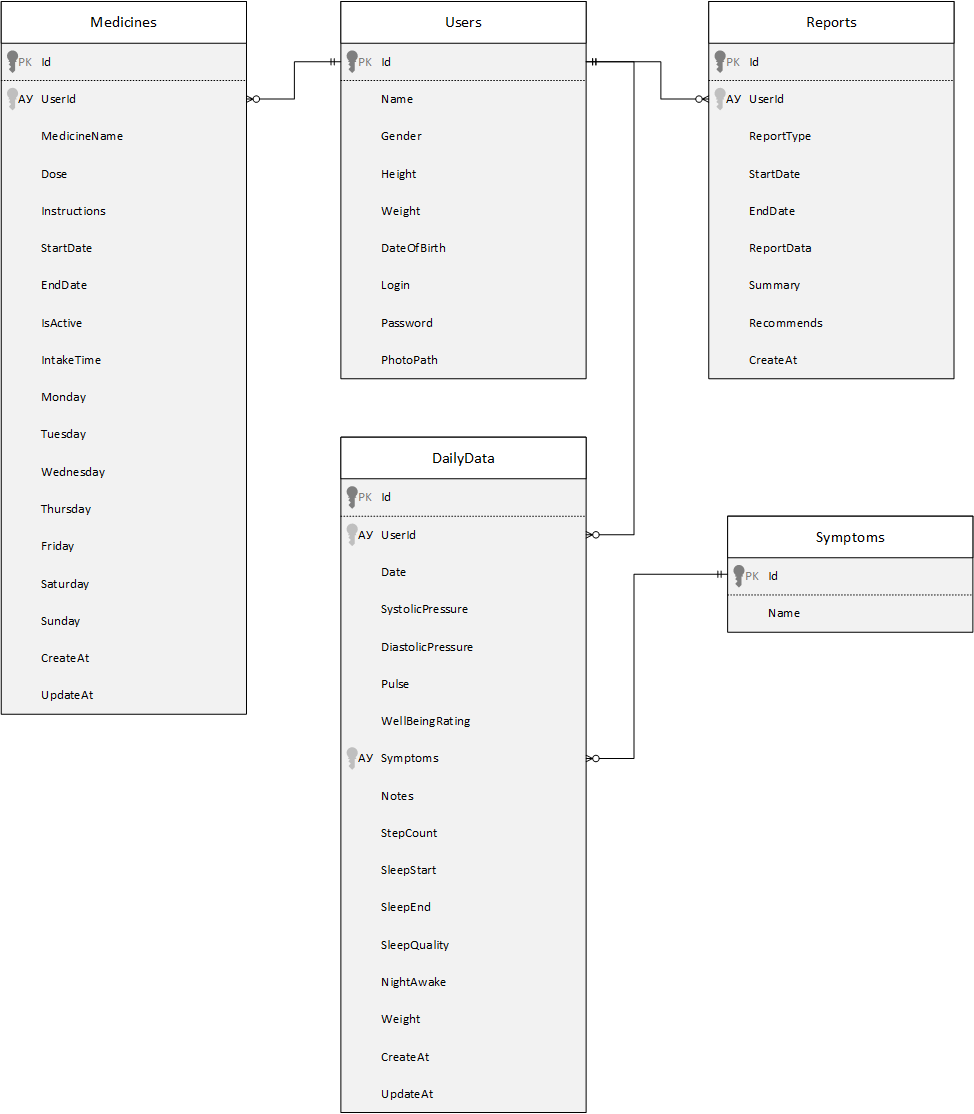


Рисунок 4 – ER-диаграмма – схема базы данных

## 4.5 Архитектура и схема функционирования ПС (ПМ)

Приложение построено по стандартной архитектуре WPF приложения с прямым доступом к данным из слоя представления (окна). Ключевой особенностью является прямой доступ к БД из UI – контекст базы данных передается между окнами и используется напрямую для выполнения запросов. Архитектура включает в себя следующие компоненты:

1. Слой представления:
   * Отвечает за отображение информации пользователю и обработку пользовательского ввода.
   * Окна реализованы с использованием WPF/XAML.
   * Каждое окно имеет связанный файл .xaml.cs, который содержит обработчики событий и логику взаимодействия с пользователем
2. Слой доступа к данным:
   * Содержит модели данных и обеспечивает взаимодействие с базой данных.
   * Реализован через классы моделей (User, DailyData, Medicine, Symptom, Report), которые определяют структуру данных и соответствуют таблицам в БД; контекст базы (AppContext) – класс, наследуемый от DbContext, определяет подключение к SQLite и набор сущностей (DbSet); прямые LINQ-запросы, с помощью которых производится валидация и выборка данных в обработчиках событий окон.
3. Слой данных:
   * Предназначен для физического хранения данных и реализован в файле БД SQLite (WellBeingDiary.db). Этот файл создается и управляется через Entity Framework Core.

Алгоритм функционирования (на примере добавления данных):

1. Пользователь нажимает кнопку «Внести данные» на главном окне;
2. Вызывается обработчик события, который открывает окно внесения данных;
3. В конструкторе создается экземпляр класса AppContext и загружаются соответствующие данные;
4. При нажатии кнопки «Сохранить» вызывается метод валидации данных, который проверяет корректность введенных значений;
5. Если валидация прошла успешно, то данные упаковываются в объект класса DailyData и сохраняются в базе данных.
6. Пользователь получает сообщение о сохранении.

Для реализации архитектуры проекта использовалась следующая файловая структура:

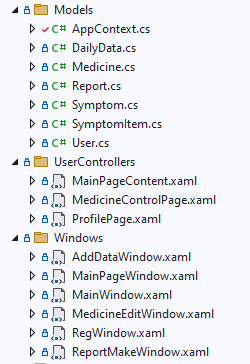


Рисунок 5 – Файловая структура проекта

# 5 ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

## 5.1 План тестирования

Для обеспечения корректной работы приложения «Дневник самочувствия» был разработан план тестирования, включающий следующие направления:

* Функциональное тестирование: проверка регистрации и авторизации пользователей; проверка ввода, редактирования, удаления данных (дневные записи, лекарства); проверка формирования отчетов за различные периоды.
* Тестирование пользовательского интерфейса: проверка отзывчивости интерфейса; проверка навигации между окнами; проверка валидации вводимых данных.
* Тестирование работы с базой данных: проверка целостности данных при операциях добавления, обновления, удаления; проверка корректности связей между таблицами.
* Производительность и стабильность: проверка времени отклика при загрузке данных.

## 5.2 Результаты тестирования

В результате тестирования было обнаружено, что при регистрации программой ошибочно выводится ошибка отсутствующего пути для изображения. Ошибка представлена на рисунке 6. Для исправления ошибки необходимо исправить путь к папке с изображениями.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – Первая ошибка, обнаруженная при тестировании

Вторая ошибка возникла при попытке ввода в момент регистрации чисел, превышающих допустимые значения. Для решения данной проблемы добавлена конструкция catch (OverflowException), которая позволяет «поймать» исключение и вывести сообщение об ошибке.

**Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Рисунок 7 – Вторая ошибка, обнаруженная при тестировании

В процессе тестирования пользовательского интерфейса, работы с базой данных, проверки производительности и стабильности ошибок не обнаружено.

# 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

После запуска программы пользователю откроется окно авторизации, представленное на рисунке 7. Для авторизованного пользователя открывается окно главной страницы, которое показано на рисунке 9.

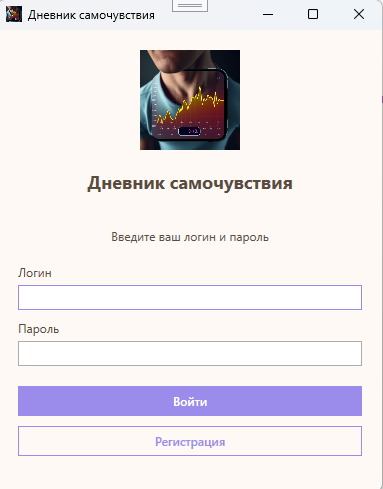


Рисунок 7 – Окно авторизации

Если пользователь ранее не проходил процедуру регистрации, то ему необходимо нажать кнопку «Регистрация» и пройти ее. Окно регистрации представлено на рисунке 8. После регистрации пользователь вернется к окну авторизации, где ему необходимо будет ввести логин и пароль. После этого он попадет на главную страницу (рисунок 9).

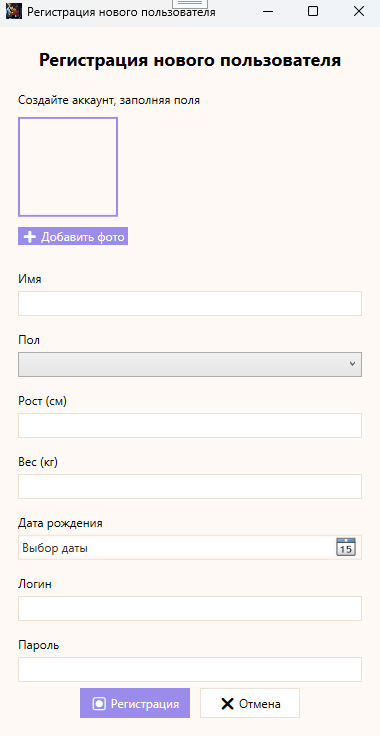


Рисунок 8 – Окно регистрации

На рисунке 9 представлена главная страница пользователя. Из нее осуществляется управление всем приложением.

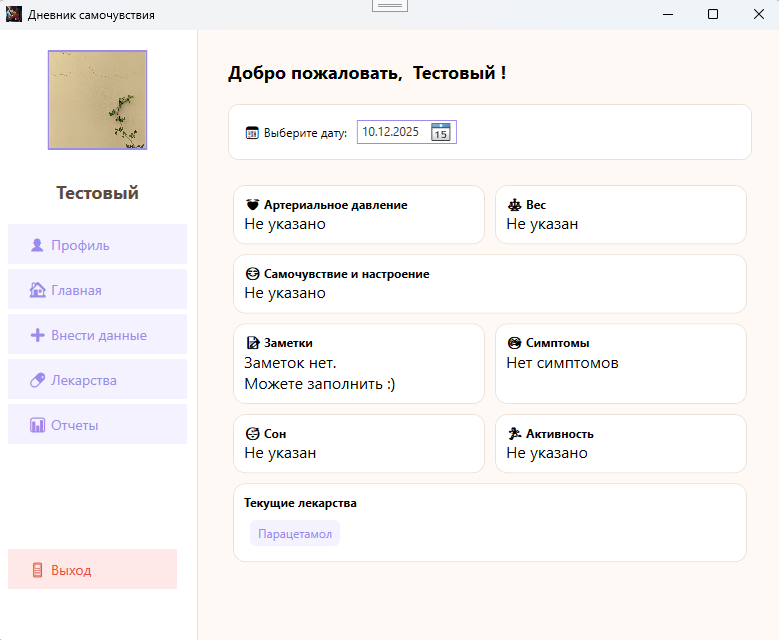


Рисунок 9 – Главная страница ПС

Просмотрим данные пользователя. Для этого нажмем на кнопку «Профиль». Откроется окно просмотра информации пользователя, где он может отредактировать информацию о себе, изменить фотографию. Данное окно отображено на рисунке 10.

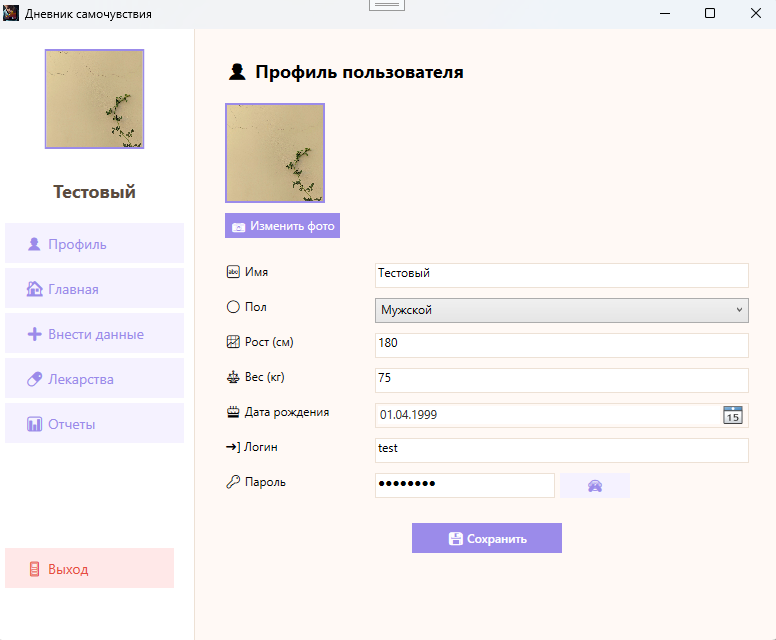


Рисунок 10 – Окно просмотра информации о пользователе

Заполним дневные данные пользователя. Для этого нажмем на кнопку «Внести данные». Откроется окно внесения дневных данных, представленное на рисунках 11.1, 11.2, 11.3. Здесь пользователю предлагается заполнить поля.

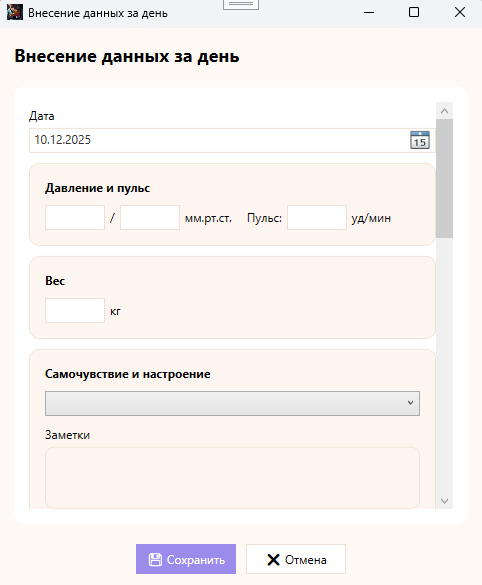


Рисунок 11.1 – Окно ввода дневных данных пользователя (давление, вес, самочувствие, заметки)

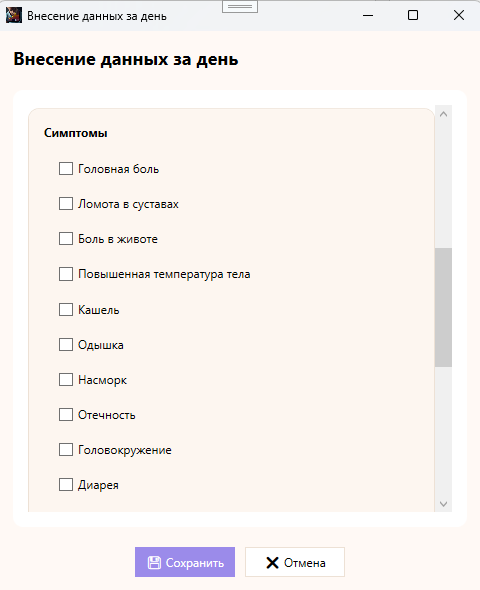


Рисунок 11.2 – Окно ввода дневных данных пользователя (симптомы)

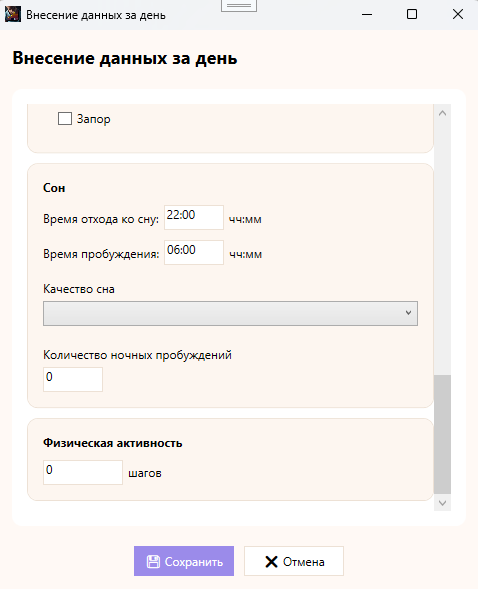


Рисунок 11.3 – Окно ввода дневных данных пользователя (сон, физическая активность)

Добавим лекарство для пользователя. Для этого нажмем на кнопку «Лекарства». Откроется окно управления лекарствами (рисунок 12). У пользователя «Тестовый» уже существует добавленное лекарство «Парацетамол». Рассмотрим функционал взаимодействия.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 – Окно управления лекарствами

Нажмем на кнопку «Добавить лекарство». В поля необходимо ввести данные о лекарстве: название, дозировку, краткие инструкции (если необходимы), дату начала приема, дату окончания (необязательно), расписание приема (рисунок 13). Лекарства так же можно удалять и редактировать. Редактирование лекарства схоже с созданием.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 13 – Окно создания лекарства

Рассмотрим функционал формирования отчетов. На главной странице приложения нажмем на кнопку «Отчеты» (рисунок 14). Откроется окно формирования отчетов, где необходимо выбрать тип отчета, интервал дат для отчета. При нажатии на кнопку «Сгенерировать отчет» на экране мы получим информацию. Также отчет представляется в виде графиков.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 14 – Окно формирования отчетов

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано desktop-приложение «Дневник самочувствия» для операционной системы Windows. Приложение предназначено для систематического учёта и анализа ключевых показателей здоровья пользователя.

В рамках проекта были решены следующие задачи:

1. Проведён анализ предметной области и сформулированы функциональные и нефункциональные требования.
2. Выбраны и обоснованы технологии разработки: WPF, C#, XAML, SQLite, Entity Framework Core, LiveCharts.WPF.
3. Разработана архитектура приложения, спроектирована база данных и пользовательский интерфейс.
4. Реализованы основные функции: регистрация и авторизация, ввод и редактирование данных, управление лекарствами, формирование графических отчётов.
5. Проведено тестирование приложения, выявлены и исправлены ошибки.

Приложение обладает интуитивно понятным интерфейсом, поддерживает русский язык, адаптировано под различные разрешения экрана и обеспечивает надёжное хранение данных. Все поставленные цели достигнуты, функциональные требования выполнены в полном объёме.

Разработанный «Дневник самочувствия» может быть использован как инструмент для персонального контроля здоровья, способствующий формированию осознанного подхода к профилактике заболеваний и улучшению качества жизни пользователя.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. WPF [Электронный ресурс] / Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/>

(Дата обращ. 13.12.2025).

1. XAML [Электронный ресурс] / Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/>

(Дата обращ. 13.12.2025).

1. C# [Электронный ресурс] / Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/tutorials/oop> (Дата обращ. 13.12.2025).
2. Основы LINQ [Электронный ресурс] / Metanit – URL: https://metanit.com/sharp/tutorial/15.1.php (Дата обращ. 13.12.2025).
3. LiveCharts.WPF [Электронный ресурс] / LiveCharts – URL: <https://livecharts.dev/docs/wpf/2.0.0-rc6.1/gallery> (Дата обращ. 13.12.2025).
4. Обучение по C# [Электронный ресурс] / Metanit – URL: <https://metanit.com/sharp/tutorial/> (Дата обращ. 13.12.2025).
5. Обучение по WPF [Электронный ресурс] / Metanit – URL: <https://metanit.com/sharp/wpf/> (Дата обращ. 13.12.2025).
6. Использование базы данных SQLite в приложении Windows

[Электронный ресурс] / Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/windows/apps/develop/data-access/sqlite-data-access> (Дата обращ. 13.12.2025).

1. Исходный код [Электронный ресурс] / GitHub – URL: <https://github.com/danilkadurilka/WellBeingDiary> (Дата обращ. 13.12.2025).

# ПРИЛОЖЕНИЯ

## Приложение А

AppContext.cs

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class AppContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<DailyData> DailyData { get; set; }

public DbSet<Medicine> Medicines { get; set; }

public DbSet<Report> Reports { get; set; }

public DbSet<Symptom> Symptoms { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=WellBeingDiary.db");

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<User>().HasIndex(u => u.Login).IsUnique();

modelBuilder.Entity<DailyData>().HasIndex(d => new { d.UserId, d.Date }).IsUnique();

modelBuilder.Entity<Symptom>().HasData(

new Symptom { Id = 1, Name = "Головная боль" },

new Symptom { Id = 2, Name = "Ломота в суставах" },

new Symptom { Id = 3, Name = "Боль в животе" },

new Symptom { Id = 4, Name = "Повышенная температура тела" },

new Symptom { Id = 5, Name = "Кашель" },

new Symptom { Id = 6, Name = "Одышка" },

new Symptom { Id = 7, Name = "Насморк" },

new Symptom { Id = 8, Name = "Отечность" },

new Symptom { Id = 9, Name = "Головокружение" },

new Symptom { Id = 10, Name = "Диарея" },

new Symptom { Id = 11, Name = "Рвота" },

new Symptom { Id = 12, Name = "Запор" });

modelBuilder.Entity<User>().HasData(

new User { Id = 1, Login = "test", Password = "test", Name = "Тестовый", Gender = "Мужской", Height = 180, Weight = 75, DateOfBirth = new DateOnly(1999, 4, 1) });

}

}

}

DailyData.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WellBeingDiary.Models;

namespace WellBeingDiary.Models

{

public class DailyData

{

public int Id { get; set; }

public int UserId { get; set; }

public DateTime Date { get; set; } = DateTime.Today;

public int? SystolicPressure { get; set; }

public int? DiastolicPressure { get; set; }

public int? Pulse { get; set; }

public int? WellBeingRating { get; set; }

public string? Symptoms { get; set; }

public string? Notes { get; set; }

public int? StepCount { get; set; }

public DateTime? SleepStart { get; set; }

public DateTime? SleepEnd { get; set; }

public int? SleepQuality { get; set; }

public int? NightAwake { get; set; }

public double? Weight { get; set; }

public DateTime CreateAt { get; set; } = DateTime.Now;

public DateTime UpdateAt { get; set; } = DateTime.Now;

}

}

Medicine.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class Medicine

{

public int Id { get; set; }

public int UserId { get; set; }

public User User { get; set; }

public string MedicineName { get; set; }

public string? Dose { get; set; }

public string? Instructions { get; set; }

public DateTime StartDate { get; set; }

public DateTime? EndDate { get; set; }

public bool IsActive { get; set; }

public string IsActiveText

{

get

{

if (IsActive) return "Да";

else return "Нет";

}

}

public TimeSpan IntakeTime { get; set; }

public bool Monday { get; set; }

public bool Tuesday { get; set; }

public bool Wednesday { get; set; }

public bool Thursday { get; set; }

public bool Friday { get; set; }

public bool Saturday { get; set; }

public bool Sunday { get; set; }

public DateTime CreateAt { get; set; } = DateTime.Now;

public DateTime UpdateAt { get; set; } = DateTime.Now;

}

}

Report.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class Report

{

public int Id { get; set; }

public int UserId { get; set; }

public User User { get; set; }

public string ReportType { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public string? ReportData { get; set; }

public string? Summary { get; set; }

public string? Recommends { get; set; }

public DateTime CreateAt { get; set; }

}

}

Symptom.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class Symptom

{

public int Id { get; set; }

public string Name { get; set; }

}

}

SymptomItem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class SymptomItem

{

public int Id { get; set; }

public string Name { get; set; }

public bool IsSelected { get; set; }

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WellBeingDiary.Models

{

public class User

{

public int Id { get; set; }

public string Name { get; set; }

public string Gender { get; set; }

public int Height { get; set; }

public double Weight { get; set; }

public DateOnly DateOfBirth { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public string? PhotoPath { get; set; }

public List<DailyData> DailyData { get; set; } = new();

public List<Medicine> Medicines { get; set; } = new();

public List<Report> Reports { get; set; } = new();

}

}

## Приложение Б

Окно MainWindow.xaml

<Window x:Class="WellBeingDiary.Windows.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WellBeingDiary.Windows"

mc:Ignorable="d"

Title="Дневник самочувствия" Height="500" Width="400"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5"

Foreground="#5A4B3F">

<Grid Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Border BorderThickness="0" Margin="0 0 0 20" Height="100" Width="100" HorizontalAlignment="Center">

<Image Grid.Row="0" Source="/Images/ico.png"/>

</Border>

<TextBlock Grid.Row="1" Text="Дневник самочувствия" FontSize="18" FontWeight="Bold" HorizontalAlignment="Center" Margin="0 0 0 20"/>

<StackPanel Grid.Row="2" VerticalAlignment="Center">

<TextBlock Text="Введите ваш логин и пароль" Margin="0 0 0 10" TextAlignment="Center"/>

<TextBlock Text="Логин" Margin="0 10 0 5"/>

<TextBox Name="BoxLogin" Height="25" Foreground="#5A4B3F" BorderBrush="#9B8BEA" BorderThickness="1"/>

<TextBlock Text="Пароль" Margin="0 10 0 5"/>

<PasswordBox Name="BoxPassword" Height="25"/>

<Button Name="ButtonLogin" Content="Войти" Click="ButtonLogin\_Click" Height="30" Background="#9B8BEA" Foreground="White" BorderThickness="0" Margin="0 20 0 10"

FontWeight="SemiBold" IsDefault="True" Cursor=""/>

<Button Name="ButtonRegister" Content="Регистрация" Click="ButtonRegister\_Click" Background="Transparent" Foreground="#9B8BEA"

BorderBrush="#9B8BEA" BorderThickness="1" Height="30" FontWeight="SemiBold"/>

</StackPanel>

</Grid>

</Window>

Окно MainPageWindow.xaml

<Window x:Class="WellBeingDiary.Windows.MainPageWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Дневник самочувствия" Height="650" Width="800"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Border Grid.Column="0" Background="#FFFFFF" BorderThickness="0 0 1 0" BorderBrush="#EDE1D5">

<StackPanel>

<Border Height="100" Width="100" HorizontalAlignment="Center" Margin="20" BorderThickness="2" BorderBrush="#9B8BEA" ClipToBounds="True">

<Image Name="UserImage" Stretch="Fill"/>

</Border>

<StackPanel Margin="10">

<TextBlock Name="BoxUserName" FontSize="18" FontWeight="Bold" Margin="20 0 20 20" TextAlignment="Center" Foreground="#5A4B3F" Text="{Binding currentUser.Name}"/>

<Button Name="ViewProfileButton" Content="👤 Профиль" FontSize="14" Click="ViewProfileButton\_Click" Height="40"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0" HorizontalContentAlignment="Left" Margin="0 0 0 5" Padding="20 0"/>

<Button Name="MainPageButton" Content="🏠 Главная" FontSize="14" Click="MainPageButton\_Click" Height="40"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0" HorizontalContentAlignment="Left" Margin="0 0 0 5" Padding="20 0"/>

<Button Name="AddDataButton" Content="➕ Внести данные" FontSize="14" Click="AddDataButton\_Click" Height="40"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0" HorizontalContentAlignment="Left" Margin="0 0 0 5" Padding="20 0"/>

<Button Name="MedControlButton" Content="💊 Лекарства" FontSize="14" Click="MedControlButton\_Click" Height="40"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0" HorizontalContentAlignment="Left" Margin="0 0 0 5" Padding="20 0"/>

<Button Name="ReportsButton" Content="📊 Отчеты" FontSize="14" Click="ReportsButton\_Click" Height="40"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0" HorizontalContentAlignment="Left" Margin="0 0 0 5" Padding="20 0"/>

</StackPanel>

<Button Name="ExitButton" Content="🚪 Выход" FontSize="14" Click="ExitButton\_Click" Height="40"

Background="#FFE8E8" Foreground="#E74C3C" BorderThickness="0" HorizontalContentAlignment="Left" Margin="10 90 20 20" Padding="20 0"/>

</StackPanel>

</Border>

<Border Grid.Column="1" Background="#FFF9F5" Padding="20">

<ContentControl Name="MainContent" />

</Border>

</Grid>

</Window>

UserControl MainPageContent.xaml

<UserControl x:Class="WellBeingDiary.UserControllers.MainPageContent"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WellBeingDiary.UserControllers"

mc:Ignorable="d"

d:DesignHeight="600" d:DesignWidth="600"

Background="#FFF9F5">

<ScrollViewer VerticalScrollBarVisibility="Auto" Padding="10">

<StackPanel>

<TextBlock FontSize="18" FontWeight="Bold" Margin="0 0 0 20">

<Run Text="Добро пожаловать, "/>

<Run Text="{Binding currentUser.Name}"/>

<Run Text="!"/>

</TextBlock>

<Border Background="#FFFFFF" CornerRadius="10" Padding="15" Margin="0 0 0 20" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel Orientation="Horizontal">

<TextBlock Text="📅 Выберите дату:" VerticalAlignment="Center" Margin="0 0 10 0"/>

<DatePicker Name="DatePickerControl" SelectedDateChanged="DatePickerControl\_SelectedDateChanged" Width="100"

Background="#FFFFFF" BorderBrush="#9B8BEA" BorderThickness="1"/>

</StackPanel>

</Border>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Border Grid.Row="0" Grid.Column="0" CornerRadius="12" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" >

<StackPanel>

<TextBlock Text="💓 Артериальное давление" FontWeight="Bold"/>

<TextBlock Name="BoxBloodPressure" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="0" Grid.Column="1" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="⚖️ Вес" FontWeight="Bold"/>

<TextBlock Name="BoxWeight" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="😊 Самочувствие и настроение" FontWeight="Bold"/>

<TextBlock Name="BoxWellBeing" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="2" Grid.Column="0" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="📝 Заметки" FontWeight="Bold"/>

<TextBlock Name="BoxNotes" FontSize="16" TextWrapping="Wrap"/>

</StackPanel>

</Border>

<Border Grid.Row="2" Grid.Column="1" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="🤒 Симптомы" FontWeight="Bold"/>

<TextBlock Name="BoxSymptoms" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="3" Grid.Column="0" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="😴 Сон" FontWeight="Bold"/>

<TextBlock Name="BoxSleep" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="3" Grid.Column="1" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="🏃 Активность" FontWeight="Bold"/>

<TextBlock Name="BoxSteps" FontSize="16"/>

</StackPanel>

</Border>

<Border Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Margin="5" Padding="10" CornerRadius="12">

<StackPanel>

<TextBlock Text="Текущие лекарства" FontWeight="Bold" Margin="0 0 0 5"/>

<ListView Name="ItemsMedicines" BorderThickness="0" Background="Transparent">

<ListView.ItemTemplate>

<DataTemplate>

<Border Margin="0 2" Padding="8 5" CornerRadius="6" Background="#F5F2FF">

<TextBlock Text="{Binding MedicineName}" Foreground="#9B8BEA"/>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<TextBlock Name="TextNoMedicines" Text="Нет активных лекарств" FontStyle="Italic" Foreground="#7F8C8D" Margin="0 5"/>

</StackPanel>

</Border>

</Grid>

</StackPanel>

</ScrollViewer>

</UserControl>

UserControl ProfilePage.xaml

<UserControl x:Class="WellBeingDiary.UserControllers.ProfilePage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WellBeingDiary.UserControllers"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="500"

Background="#FFF9F5">

<StackPanel Margin="10">

<TextBlock Text="👤 Профиль пользователя" FontSize="18" FontWeight="Bold" Margin="0 0 0 20"/>

<Border Margin="0 0 0 10" Height="100" Width="100" HorizontalAlignment="Left" BorderThickness="2" BorderBrush="#9B8BEA" ClipToBounds="True">

<Image Name="ProfilePhoto" Stretch="Fill"/>

</Border>

<Button Name="ButtonChangePhoto" Content="📷 Изменить фото" Click="ChangePhotoButton\_Click" Width="115" Height="25" HorizontalAlignment="Left" Margin="0 0 0 20"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="🔤 Имя" Margin="0 5"/>

<TextBox Grid.Row="0" Grid.Column="1" Name="BoxName" Height="25" Margin="0 5" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Grid.Row="1" Grid.Column="0" Text="⚪ Пол" Margin="0 5"/>

<ComboBox Grid.Row="1" Grid.Column="1" Name="ComboBoxGender" Height="25" Margin="0 5" BorderBrush="#EDE1D5" BorderThickness="1">

<ComboBoxItem Content="Мужской"/>

<ComboBoxItem Content="Женский"/>

</ComboBox>

<TextBlock Grid.Row="2" Grid.Column="0" Text="📈 Рост (см)" Margin="0 5"/>

<TextBox Grid.Row="2" Grid.Column="1" Name="BoxHeight" Height="25" Margin="0 5" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Grid.Row="3" Grid.Column="0" Text="⚖️ Вес (кг)" Margin="0 5"/>

<TextBox Grid.Row="3" Grid.Column="1" Name="BoxWeight" Height="25" Margin="0 5" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="🎂 Дата рождения" Margin="0 5"/>

<DatePicker Grid.Row="4" Grid.Column="1" Name="DPBirthDate" Height="25" BorderBrush="#EDE1D5" BorderThickness="1" Margin="0 5"/>

<TextBlock Grid.Row="5" Grid.Column="0" Text="➜] Логин" Margin="0 5"/>

<TextBox Grid.Row="5" Grid.Column="1" Name="BoxLogin" Height="25" BorderBrush="#EDE1D5" BorderThickness="1" Margin="0 5"/>

<TextBlock Grid.Row="6" Grid.Column="0" Text="🔑 Пароль" Margin="0 5"/>

<StackPanel Grid.Row="6" Grid.Column="1" Orientation="Horizontal">

<PasswordBox Name="BoxPassword" Height="25" Margin="0 5" Width="180" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBox Name="BoxPasswordVisible" Height="25" Margin="0 5" Width="180" Visibility="Collapsed" BorderBrush="#EDE1D5" BorderThickness="1"/>

<Button Name="ViewPasswordButton" Content="🙈" Width="70" Margin="5 5 0 5" Click="ViewPasswordButton\_Click"

Background="#F5F2FF" Foreground="#9B8BEA" BorderThickness="0"/>

</StackPanel>

</Grid>

<Button Name="SaveProfileButton" Content="💾 Сохранить"

Click="SaveProfileButton\_Click" Width="150" Height="30" Margin="0 20 0 0" Background="#9B8BEA" Foreground="White" BorderThickness="0" FontWeight="SemiBold"/>

</StackPanel>

</UserControl>

Окно AddDataWindow.xaml

<Window x:Class="WellBeingDiary.Windows.AddDataWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WellBeingDiary.Windows"

mc:Ignorable="d"

Title="Внесение данных за день" Height="600" Width="500"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5">

<Grid Margin="15">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Внесение данных за день" FontSize="18" FontWeight="Bold" Margin="0 0 0 20"/>

<Border Grid.Row="1" Background="#FFFFFF" CornerRadius="10" Padding="15">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<StackPanel>

<TextBlock Text="Дата" Margin="0 5 0 5"/>

<DatePicker Name="DPDate" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1" SelectedDateChanged="DPDate\_SelectedDateChanged"/>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Давление и пульс" FontWeight="Bold" Margin="0 0 0 10"/>

<StackPanel Orientation="Horizontal" VerticalAlignment="Center">

<TextBox Name="BoxSystolic" Width="60" Height="25" Margin="0 0 5 0" BorderBrush="#EDE1D5" BorderThickness="1" TextAlignment="Center"/>

<TextBlock Text="/" VerticalAlignment="Center"/>

<TextBox Name="BoxDiastolic" Width="60" Height="25" Margin="5 0 5 0" BorderBrush="#EDE1D5" BorderThickness="1" TextAlignment="Center"/>

<TextBlock Text="мм.рт.ст." VerticalAlignment="Center" Margin="0 0 15 0"/>

<TextBlock Text="Пульс:" VerticalAlignment="Center"/>

<TextBox Name="BoxPulse" Width="60" Height="25" Margin="5 0 0 0" BorderBrush="#EDE1D5" BorderThickness="1" TextAlignment="Center"/>

<TextBlock Text="уд/мин" VerticalAlignment="Center" Margin="5 0 0 0"/>

</StackPanel>

</StackPanel>

</Border>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Вес" FontWeight="Bold" Margin="0 0 0 10"/>

<StackPanel Orientation="Horizontal">

<TextBox Name="BoxWeight" Height="25" Width="60" HorizontalAlignment="Left" BorderBrush="#EDE1D5" BorderThickness="1" TextAlignment="Center"/>

<TextBlock Text="кг" Margin="5 0 0 0" VerticalAlignment="Center"/>

</StackPanel>

</StackPanel>

</Border>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Самочувствие и настроение" FontWeight="Bold" Margin="0 0 0 10"/>

<ComboBox Name="ComboBoxWellBeing" Height="25" Margin="0 0 0 5" BorderBrush="#EDE1D5" BorderThickness="1">

<ComboBoxItem Content="1 - Плохое"/>

<ComboBoxItem Content="2 - Удовлетворительное"/>

<ComboBoxItem Content="3 - Нормальное"/>

<ComboBoxItem Content="4 - Хорошее"/>

<ComboBoxItem Content="5 - Отличное"/>

</ComboBox>

<TextBlock Text="Заметки" Margin="0 5 0 5"/>

<Border Background="#FDF6F0" BorderBrush="#EDE1D5" BorderThickness="1" CornerRadius="8">

<TextBox Name="BoxNotes" Height="60" Background="Transparent" BorderThickness="0" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto"/>

</Border>

</StackPanel>

</Border>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Симптомы" FontWeight="Bold" Margin="0 0 0 10"/>

<ListView Name="SymptomsItems" BorderThickness="0" Background="Transparent">

<ListView.ItemTemplate>

<DataTemplate>

<Border Margin="0 3" Padding="8 5" CornerRadius="8">

<CheckBox Content="{Binding Name}" IsChecked="{Binding IsSelected}" Tag="{Binding Id}"/>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackPanel>

</Border>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Сон" FontWeight="Bold" Margin="0 0 0 10"/>

<StackPanel Orientation="Horizontal" Margin="0 0 0 10">

<TextBlock Text="Время отхода ко сну:" VerticalAlignment="Center" Margin="0 0 5 0"/>

<TextBox Name="BoxSleepStart" Width="60" Height="25" Margin="0 0 5 0" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="чч:мм" VerticalAlignment="Center"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 0 0 10">

<TextBlock Text="Время пробуждения:" VerticalAlignment="Center" Margin="0 0 5 0"/>

<TextBox Name="BoxSleepEnd" Width="60" Height="25" Margin="0 0 5 0" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="чч:мм" VerticalAlignment="Center"/>

</StackPanel>

<TextBlock Text="Качество сна" Margin="0 5 0 5" VerticalAlignment="Center"/>

<ComboBox Name="ComboBoxSleepQuality" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1">

<ComboBoxItem Content="1 - Очень плохо"/>

<ComboBoxItem Content="2 - Плохо"/>

<ComboBoxItem Content="3 - Нормально"/>

<ComboBoxItem Content="4 - Хорошо"/>

<ComboBoxItem Content="5 - Отлично"/>

</ComboBox>

<TextBlock Text="Количество ночных пробуждений" Margin="0 10 0 5"/>

<TextBox Name="BoxNightAwakening" Height="25" Width="60" HorizontalAlignment="Left" Text="0" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Border>

<Border Margin="0 0 0 10" Padding="15" Background="#FDF6F0" CornerRadius="10" BorderThickness="1" BorderBrush="#EDE1D5">

<StackPanel>

<TextBlock Text="Физическая активность" FontWeight="Bold" Margin="0 0 0 10"/>

<StackPanel Orientation="Horizontal">

<TextBox Name="BoxSteps" Height="25" Width="80" HorizontalAlignment="Left" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="шагов" VerticalAlignment="Center" Margin="5 0 0 0" />

</StackPanel>

</StackPanel>

</Border>

</StackPanel>

</ScrollViewer>

</Border>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0 20 0 0">

<Button Name="SaveButton" Content="💾 Сохранить" Click="SaveButton\_Click" Width="100" Height="30" Margin="0 0 10 0"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Button Name="CancelButton" Content="❌ Отмена" Click="CancelButton\_Click" Width="100" Height="30"

Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Grid>

</Window>

Окно MedicineEditWindow.xaml

<Window x:Class="WellBeingDiary.Windows.MedicineEditWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WellBeingDiary.Windows"

mc:Ignorable="d"

Title="Редактирование лекарства" Height="500" Width="450"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5">

<Grid Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Редактирование лекарства" FontSize="18" FontWeight="Bold" Margin="0 0 0 20"/>

<Border Grid.Row="1" Background="#FFFFFF" CornerRadius="10" Padding="15">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<StackPanel>

<TextBlock Text="Название лекарства" Margin="0 5 0 5"/>

<TextBox Name="BoxMedicineName" Height="25" Margin="0 0 0 10" Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="Дозировка лекарства" Margin="0 5 0 5"/>

<TextBox Name="BoxDose" Height="25" Margin="0 0 0 10" Background="White" BorderBrush="#EDE1D5" BorderThickness="1" TextWrapping="Wrap"/>

<TextBlock Text="Краткие инструкции к лекарству" Margin="0 5 0 5"/>

<TextBox Name="BoxInstructions" Height="60" Margin="0 0 0 10" Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="Дата начала приема" Margin="0 5 0 5"/>

<DatePicker Name="DPStartDate" Height="25" Margin="0 0 0 10" Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="Дата окончания приема (не заполняйте, если принимаете постоянно)" Margin="0 5 0 5"/>

<DatePicker Name="DPEndDate" Height="25" Margin="0 0 0 10" Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

<Border Margin="0 10" Padding="10" Background="#F5F2FF" CornerRadius="8">

<CheckBox Name="BoxIsActive" Content="Сейчас принимается?" IsChecked="True"/>

</Border>

<TextBlock Text="Расписание приема" FontWeight="Bold" Margin="0 10 0 5"/>

<Border Padding="10" Background="#FDF6F0" CornerRadius="8" BorderThickness="1" BorderBrush="#EDE1D5">

<WrapPanel Margin="0 0 0 10">

<CheckBox Name="BoxMonday" Content="Пн" Margin="0 0 8 5"/>

<CheckBox Name="BoxTuesday" Content="Вт" Margin="0 0 8 5"/>

<CheckBox Name="BoxWednesday" Content="Ср" Margin="0 0 8 5"/>

<CheckBox Name="BoxThursday" Content="Чт" Margin="0 0 8 5"/>

<CheckBox Name="BoxFriday" Content="Пт" Margin="0 0 8 5"/>

<CheckBox Name="BoxSaturday" Content="Сб" Margin="0 0 8 5"/>

<CheckBox Name="BoxSunday" Content="Вс" Margin="0 0 0 5"/>

</WrapPanel>

</Border>

<TextBlock Text="Время приема" Margin="0 5 0 5"/>

<TextBox Name="BoxIntakeTime" Height="25" Background="#FFFFFF" BorderBrush="#EDE1D5" BorderThickness="1" Text="08:00"/>

</StackPanel>

</ScrollViewer>

</Border>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0 20 0 0">

<Button Name="SaveButton" Content="💾 Сохранить" Click="SaveButton\_Click" Width="100" Height="30" Margin="0 0 10 0"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Button Name="CancelButton" Content="❌ Отмена" Click="CancelButton\_Click" Width="100" Height="30"

Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Grid>

</Window>

UserControl MedicineControlPage.xaml

<UserControl x:Class="WellBeingDiary.UserControllers.MedicineControlPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:WellBeingDiary.UserControllers"

mc:Ignorable="d"

d:DesignHeight="600" d:DesignWidth="600"

Background="#FFF9F5">

<Grid Margin="15">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="💊 Контроль лекарств" FontSize="18" FontWeight="Bold" Margin="0 0 0 20"/>

<Border Grid.Row="1" Background="White" CornerRadius="10" Padding="15">

<StackPanel>

<Button Name="AddMedicineButton" Content="➕ Добавить лекарство" Click="AddMedicineButton\_Click" Width="140" Height="30" Margin="0 0 0 10" HorizontalAlignment="Left"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Border BorderBrush="#EDE1D5" CornerRadius="8" Height="350" Background="#FFF9F5" BorderThickness="1">

<ListBox Name="MedicinesListBox" Background="Transparent" BorderThickness="0" ItemsSource="{Binding Medicines}" Height="300" Margin="0 0 0 10">

<ListBox.ItemTemplate>

<DataTemplate>

<Border Background="White" BorderBrush="#EDE1D5" BorderThickness="1" Margin="2" Padding="15" CornerRadius="10">

<StackPanel>

<TextBlock Text="{Binding MedicineName}" FontWeight="Bold" Margin="0 0 0 5"/>

<TextBlock Text="{Binding Dose}" Foreground="#7F8C8D" Margin="0 0 0 3"/>

<TextBlock Text="{Binding Instructions}" Foreground="#7F8C8D" Margin="0 0 0 3" TextWrapping="Wrap"/>

<TextBlock>

<Run Text="с" Foreground="#7F8C8D"/>

<Run Text="{Binding StartDate, StringFormat=\{0:dd.MM.yyyy\}}" Foreground="#7F8C8D"/>

</TextBlock>

<TextBlock>

<Run Text="Сейчас принимается: " Foreground="#7F8C8D"/>

<Run Text="{Binding IsActiveText, Mode=OneWay}" Foreground="#9B8BEA"/>

</TextBlock>

<TextBlock>

<Run Text="Время приема: " Foreground="#7F8C8D"/>

<Run Text="{Binding IntakeTime}" Foreground="#9B8BEA"/>

</TextBlock>

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Border>

</StackPanel>

</Border>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center">

<Button Name="EditButton" Content="✎ Редактировать" Click="EditButton\_Click" Width="100" Height="30" Margin="0 0 10 0"

Background="#FFD166" BorderThickness="0"/>

<Button Name="DeleteButton" Content="🗑️ Удалить" Click="DeleteButton\_Click" Width="100" Height="30"

Background="#FF6B6B" Foreground="#FFFFFF" BorderThickness="0"/>

</StackPanel>

</Grid>

</UserControl>

Окно ReportMakeWindow.xaml

<Window x:Class="WellBeingDiary.Windows.ReportMakeWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WellBeingDiary.Windows"

xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"

mc:Ignorable="d"

Title="Создание отчетов" Height="700" Width="800"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5">

<Grid Margin="15">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Создание отчетов" FontSize="18" FontWeight="Bold" Margin="0 0 0 20"/>

<Border Grid.Row="1" Background="White" CornerRadius="10" Margin="0 0 0 15">

<StackPanel>

<TextBlock Text="Выберите тип отчета:" Margin="0 0 0 8"/>

<ComboBox Name="ComboBoxReportType" Height="25" Margin="5 0 5 10" SelectionChanged="ComboBoxReportType\_SelectionChanged" BorderBrush="#EDE1D5" BorderThickness="1">

<ComboBoxItem Content="Отчет по самочувствию"/>

<ComboBoxItem Content="Отчет по давлению"/>

<ComboBoxItem Content="Отчет по сну"/>

<ComboBoxItem Content="Отчет по двигательной активности"/>

<ComboBoxItem Content="Отчет по весу"/>

</ComboBox>

<StackPanel Orientation="Horizontal" Margin="0 0 0 10">

<StackPanel>

<TextBlock Text="Начальная дата" Margin="0 5 0 5"/>

<DatePicker Name="DPStartDate" Height="25" Width="100" Margin="0 0 10 0" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

<StackPanel>

<TextBlock Text="Конечная дата" Margin="0 5 0 5"/>

<DatePicker Name="DPEndDate" Height="25" Width="100" Margin="0 0 10 0" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</StackPanel>

<Button Name="GenerateReportButton" Content="🔄 Сгенерировать отчет" Click="GenerateReportButton\_Click" Height="30"

Foreground="White" Background="#9B8BEA" BorderThickness="0"/>

</StackPanel>

</Border>

<Border Grid.Row="2" CornerRadius="10" Margin="0 0 0 10">

<TabControl>

<TabItem Header="📄 Текстовый отчет" Background="#F5F2FF" BorderBrush="#EDE1D5" BorderThickness="1">

<Border Background="#FDF6F0" CornerRadius="0 0 10 10">

<TextBox Name="BoxReport" Height="300" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" BorderThickness ="0" IsReadOnly="True" Background="Transparent" Padding="10"/>

</Border>

</TabItem>

<TabItem Header="📈 График" Name="ChartTab" Background="#F5F2FF" BorderBrush="#EDE1D5" BorderThickness="1">

<Border Background="#FDF6F0" CornerRadius="0 0 10 10">

<lvc:CartesianChart Name="ReportChart" Series="{Binding ReportCollection}" LegendLocation="Right" Background="Transparent"/>

</Border>

</TabItem>

</TabControl>

</Border>

<StackPanel Grid.Row="3" Orientation="Horizontal" HorizontalAlignment="Center" Margin="0 20 0 0">

<Button Name="SaveReportButton" Content="💾 Сохранить" Click="SaveReportButton\_Click" Width="100" Height="30" Margin="0 0 10 0"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Button Name="CloseButton" Content="❌ Отмена" Click="CloseButton\_Click" Width="100" Height="30"

Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Grid>

</Window>

Окно RegWindow.xaml

<Window x:Class="WellBeingDiary.Windows.RegWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Регистрация нового пользователя" Height="750" Width="400"

WindowStartupLocation="CenterScreen"

Icon="/Images/ico.png"

Background="#FFF9F5">

<Grid Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Text="Регистрация нового пользователя" FontSize="18" FontWeight="Bold" HorizontalAlignment="Center" Margin="0 0 0 20"/>

<Border Grid.Row="1" CornerRadius="10">

<StackPanel>

<TextBlock Text="Создайте аккаунт, заполняя поля" Margin="0 0 0 10"/>

<Border Margin="0 0 0 10" Height="100" Width="100" HorizontalAlignment="Left" BorderThickness="2" BorderBrush="#9B8BEA" ClipToBounds="True">

<Image Name="ImgUserPhoto" Source="{Binding PhotoPath}" Stretch="Fill"/>

</Border>

<Button Name="AddPhotoButton" Content="➕ Добавить фото" Click="AddPhotoButton\_Click" Width="110" HorizontalAlignment="Left" Margin="0 0 0 20"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<TextBlock Text="Имя" Margin="0 5 0 5"/>

<TextBox Name="BoxName" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1" Text="{Binding Name}"/>

<TextBlock Text="Пол" Margin="0 5 0 5"/>

<ComboBox Name="ComboBoxGender" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1" SelectedValue="{Binding Gender}">

<ComboBoxItem Content="Мужской" Tag="Мужской"/>

<ComboBoxItem Content="Женский" Tag="Женский"/>

</ComboBox>

<TextBlock Text="Рост (см)" Margin="0 5 0 5"/>

<TextBox Name="BoxHeight" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1" Text="{Binding Height}"/>

<TextBlock Text="Вес (кг)" Margin="0 5 0 5"/>

<TextBox Name="BoxWeight" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1" Text="{Binding Weight}"/>

<TextBlock Text="Дата рождения" Margin="0 5 0 5"/>

<DatePicker Name="DPBirthDate" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="Логин" Margin="0 5 0 5"/>

<TextBox Name="BoxLogin" Height="25" Margin="0 0 0 10" BorderBrush="#EDE1D5" BorderThickness="1"/>

<TextBlock Text="Пароль" Margin="0 5 0 5"/>

<PasswordBox Name="BoxPassword" Height="25" Margin="0 0 0 20" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Border>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center">

<Button Name="RegButton" Content="⏺️ Регистрация" Click="RegButton\_Click" Width="110" Height="30" Margin="0 0 10 0"

Background="#9B8BEA" Foreground="White" BorderThickness="0"/>

<Button Name="CancelButton" Content="❌ Отмена" Click="CancelButton\_Click" Width="100" Height="30"

Background="White" BorderBrush="#EDE1D5" BorderThickness="1"/>

</StackPanel>

</Grid>

</Window>

## Приложение В

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private Models.AppContext \_context;

public MainWindow()

{

InitializeComponent();

InitializeDB();

}

public void InitializeDB()

{

try

{

\_context = new Models.AppContext();

\_context.Database.EnsureCreated();

if (!\_context.Users.Any(u => u.Login == "test"))

{

\_context.Users.Add(new User { Id = 1, Login = "test", Password = "test", Name = "Тестовый", Gender = "Мужской", Height = 180, Weight = 75, DateOfBirth = new DateOnly(1999, 4, 1) });

\_context.SaveChanges();

}

}

catch (Exception ex)

{

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

}

}

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

{

string login = BoxLogin.Text;

string password = BoxPassword.Password;

try

{

if (string.IsNullOrWhiteSpace(login) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Введите логин и пароль!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

var user = \_context.Users.FirstOrDefault(u => u.Login == login && u.Password == password);

if (user != null)

{

MainPageWindow mainWindow = new MainPageWindow(user, \_context);

mainWindow.Show();

this.Close();

}

else

{

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

}

}

catch (Exception ex)

{

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

}

}

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

{

RegWindow regWindow = new RegWindow(\_context);

regWindow.Show();

this.Close();

}

}

}

MainPageWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WellBeingDiary.Models;

using WellBeingDiary.UserControllers;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для MainPageWindow.xaml

/// </summary>

public partial class MainPageWindow : Window

{

public User currentUser { get; set; }

private Models.AppContext \_context;

public MainPageWindow(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

DataContext = this;

ShowMainPage();

ShowUserImage();

}

public void ShowUserImage()

{

try

{

string path = currentUser.PhotoPath;

if (string.IsNullOrEmpty(path) || !System.IO.File.Exists(path))

{

path = "Images/none.png";

}

if (System.IO.File.Exists(path))

{

BitmapImage bitmap = new();

bitmap.BeginInit();

bitmap.UriSource = new Uri(System.IO.Path.GetFullPath(path), UriKind.Absolute);

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.EndInit();

UserImage.Source = bitmap;

}

}

catch

{

MessageBox.Show("Ошибка инициализации изображения");

}

}

public void ShowMainPage()

{

MainContent.Content = new MainPageContent(currentUser, \_context);

}

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

{

MainContent.Content = new ProfilePage(currentUser, \_context);

}

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

{

ShowMainPage();

}

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

{

Window addDataWindow = new AddDataWindow(currentUser, \_context);

if (addDataWindow.ShowDialog() == true)

ShowMainPage();

}

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

{

MainContent.Content = new MedicineControlPage(currentUser, \_context);

}

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

{

Window reportWindow = new ReportMakeWindow(currentUser, \_context);

reportWindow.ShowDialog();

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

MainPageContent.xaml.cs

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.UserControllers

{

/// <summary>

/// Логика взаимодействия для MainPageContent.xaml

/// </summary>

public partial class MainPageContent : UserControl

{

private User currentUser;

private DailyData selectedData;

private DateTime currentDate = DateTime.Today;

private Models.AppContext \_context;

public MainPageContent(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

DatePickerControl.SelectedDate = currentDate;

LoadData();

}

private void DatePickerControl\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

if (DatePickerControl.SelectedDate.HasValue)

{

currentDate = DatePickerControl.SelectedDate.Value;

LoadData();

}

}

public void LoadData()

{

selectedData = \_context.DailyData?.FirstOrDefault(d => d.UserId == currentUser.Id && d.Date.Date == currentDate.Date);

if (selectedData == null)

selectedData = new DailyData();

UpdateDataDisplay();

LoadMedicines();

}

public void UpdateDataDisplay()

{

if (selectedData.SystolicPressure.HasValue && selectedData.DiastolicPressure.HasValue)

{

string pulse = string.Empty;

if (selectedData.Pulse.HasValue)

{

pulse = $"{selectedData.Pulse}";

}

BoxBloodPressure.Text = $"{selectedData.SystolicPressure}/{selectedData.DiastolicPressure}/{pulse}";

}

else { BoxBloodPressure.Text = "Не указано"; }

if (selectedData.Weight.HasValue)

{

BoxWeight.Text = $"{selectedData.Weight} кг";

}

else { BoxWeight.Text = "Не указан"; }

if (selectedData.WellBeingRating.HasValue)

{

int? rating = selectedData.WellBeingRating.Value;

string ratingText;

if (rating == 1)

{

ratingText = "Плохое";

}

else if (rating == 2)

{

ratingText = "Удовлетворительное";

}

else if (rating == 3)

{

ratingText = "Нормальное";

}

else if (rating == 4)

{

ratingText = "Хорошее";

}

else if (rating == 5)

{

ratingText = "Отличное";

}

else

{

ratingText = "Не указано";

}

BoxWellBeing.Text = $"{ratingText} ({rating}/5)";

}

else { BoxWellBeing.Text = "Не указано"; }

if (!string.IsNullOrEmpty(selectedData.Notes))

BoxNotes.Text = selectedData.Notes;

else

BoxNotes.Text = "Заметок нет.\nМожете заполнить :)";

if (!string.IsNullOrEmpty(selectedData.Symptoms))

{

try

{

List<int> symptomIds = selectedData.Symptoms.Split(',').Where(s => !string.IsNullOrEmpty(s)).Select(int.Parse).ToList();

List<string> symptomNames = \_context.Symptoms.Where(s => symptomIds.Contains(s.Id)).Select(s => s.Name).ToList();

if (symptomNames.Count > 0)

{

BoxSymptoms.Text = string.Join("\n", symptomNames);

}

else

{

BoxSymptoms.Text = "Нет симптомов";

}

}

catch (Exception ex)

{

BoxSymptoms.Text = "Ошибка загрузки симптомов";

Console.WriteLine($"Ошибка при загрузке симптомов: {ex.Message}");

}

}

else

{

BoxSymptoms.Text = "Нет симптомов";

}

if (selectedData.SleepStart.HasValue && selectedData.SleepEnd.HasValue)

{

TimeSpan sleepTime = selectedData.SleepEnd.Value - selectedData.SleepStart.Value;

string quality = string.Empty;

if (selectedData.SleepQuality.HasValue)

{

quality = $", качество сна: {selectedData.SleepQuality}/5";

}

BoxSleep.Text = $"{sleepTime.Hours}ч {sleepTime.Minutes}м{quality}";

}

else { BoxSleep.Text = "Не указан"; }

if (selectedData.StepCount.HasValue)

{

BoxSteps.Text = $"{selectedData.StepCount} шагов";

}

else { BoxSteps.Text = "Не указано"; }

}

public void LoadMedicines()

{

List<Medicine>? medicines = \_context.Medicines?.Where(m => m.UserId == currentUser.Id && m.IsActive && m.StartDate.Date <= currentDate.Date

&& (m.EndDate == null || m.EndDate.Value.Date >= currentDate.Date)).ToList();

if (medicines != null && medicines.Any())

{

ItemsMedicines.ItemsSource = medicines;

TextNoMedicines.Visibility = Visibility.Collapsed;

}

else

{

ItemsMedicines.ItemsSource = null;

TextNoMedicines.Visibility = Visibility.Visible;

}

}

}

}

ProfilePage.xaml.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.UserControllers

{

/// <summary>

/// Логика взаимодействия для ProfilePage.xaml

/// </summary>

public partial class ProfilePage : UserControl

{

private User currentUser;

private bool isVisiblePassword = false;

private string realPassword = string.Empty;

private Models.AppContext \_context;

private string selectedImagePath = string.Empty;

public User CurrentUser

{

get { return currentUser; }

set { currentUser = value; }

}

public ProfilePage(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

LoadUserData();

}

public void LoadUserData()

{

DataContext = this;

BoxName.Text = currentUser.Name;

ComboBoxGender.Text = currentUser.Gender;

BoxHeight.Text = currentUser.Height.ToString();

BoxWeight.Text = currentUser.Weight.ToString();

BoxLogin.Text = currentUser.Login;

realPassword = currentUser.Password;

BoxPassword.Password = "\*\*\*\*\*\*\*\*";

BoxPasswordVisible.Text = realPassword;

isVisiblePassword = false;

ViewPasswordButton.Content = "🙈";

BoxPasswordVisible.Visibility = Visibility.Collapsed;

BoxPassword.Visibility = Visibility.Visible;

LoadUserImage();

if (currentUser.DateOfBirth != default)

{

DPBirthDate.SelectedDate = new DateTime(

currentUser.DateOfBirth.Year,

currentUser.DateOfBirth.Month,

currentUser.DateOfBirth.Day);

}

}

public void LoadUserImage()

{

try

{

string path = currentUser.PhotoPath;

if (string.IsNullOrEmpty(path) || !System.IO.File.Exists(path))

{

path = "Images/none.png";

}

if (System.IO.File.Exists(path))

{

BitmapImage bitmap = new();

bitmap.BeginInit();

bitmap.UriSource = new Uri(System.IO.Path.GetFullPath(path), UriKind.Absolute);

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.EndInit();

ProfilePhoto.Source = bitmap;

}

}

catch

{

MessageBox.Show("Ошибка инициализации изображения");

}

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Image files (\*.jpg, \*.jpeg, \*.png)|\*.jpg;\*.jpeg;\*.png";

if (openFileDialog.ShowDialog() == true)

{

selectedImagePath = openFileDialog.FileName;

currentUser.PhotoPath = selectedImagePath;

}

}

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

{

try

{

if (string.IsNullOrWhiteSpace(BoxName.Text))

{

MessageBox.Show("Имя не может быть пустым");

return;

}

if (string.IsNullOrWhiteSpace(BoxLogin.Text))

{

MessageBox.Show("Логин пользователя не может быть пустым");

return;

}

if (BoxLogin.Text != currentUser.Login && \_context.Users.Any(u => u.Login == BoxLogin.Text))

{

MessageBox.Show("Пользователь с таким логином уже существует");

return;

}

try

{

int height = int.Parse(BoxHeight.Text);

if (height < 60 || height > 250)

{

MessageBox.Show("Введите корректное значение роста (от 60 до 250 см)");

return;

}

currentUser.Height = height;

}

catch(FormatException)

{

MessageBox.Show("Рост должен быть целым числом");

return;

}

try

{

double weight = double.Parse(BoxWeight.Text);

if (weight < 20 || weight > 300)

{

MessageBox.Show("Введите корректное значение веса (от 20 до 300 кг)");

return;

}

currentUser.Weight = weight;

}

catch (FormatException)

{

MessageBox.Show("Вес должен быть числом");

return;

}

currentUser.Name = BoxName.Text;

currentUser.Gender = ComboBoxGender.Text;

currentUser.Login = BoxLogin.Text;

if (isVisiblePassword)

currentUser.Password = BoxPasswordVisible.Text;

else if (BoxPassword.Password != "\*\*\*\*\*\*\*\*")

currentUser.Password = BoxPassword.Password;

if (DPBirthDate.SelectedDate.HasValue)

{

currentUser.DateOfBirth = DateOnly.FromDateTime(DPBirthDate.SelectedDate.Value);

}

if (!string.IsNullOrEmpty(selectedImagePath))

{

string extension = System.IO.Path.GetExtension(selectedImagePath);

string fileName = $"{currentUser.Id}\_{currentUser.Login}{extension}";

string imagePath = "./Images/" + fileName;

System.IO.File.Copy(selectedImagePath, imagePath, true);

currentUser.PhotoPath = imagePath;

selectedImagePath = string.Empty;

}

if (string.IsNullOrEmpty(currentUser.PhotoPath) || !System.IO.File.Exists(currentUser.PhotoPath))

{

currentUser.PhotoPath = "./Images/none.png";

}

\_context.SaveChanges();

LoadUserImage();

MessageBox.Show("Изменения сохранены");

}

catch (Exception ex)

{

MessageBox.Show("Возникла ошибка при сохранении: " + ex.Message);

}

}

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

{

if (!isVisiblePassword)

{

BoxPasswordVisible.Text = realPassword;

BoxPassword.Visibility = Visibility.Collapsed;

BoxPasswordVisible.Visibility = Visibility.Visible;

ViewPasswordButton.Content = "🐵";

}

else

{

if (BoxPasswordVisible.Text != realPassword)

realPassword = BoxPasswordVisible.Text;

BoxPassword.Password = "\*\*\*\*\*\*\*\*";

BoxPasswordVisible.Visibility = Visibility.Collapsed;

BoxPassword.Visibility = Visibility.Visible;

ViewPasswordButton.Content = "🙈";

}

isVisiblePassword = !isVisiblePassword;

}

}

}

AddDataWindow.xaml.cs

using Microsoft.EntityFrameworkCore.Sqlite.Query.Internal;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Printing.IndexedProperties;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для AddDataWindow.xaml

/// </summary>

public partial class AddDataWindow : Window

{

private User currentUser;

private DailyData dailyData;

private Models.AppContext \_context;

private List<SymptomItem> symptomItems;

public AddDataWindow(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

DPDate.SelectedDate = DateTime.Today;

LoadThisDateData();

}

public void LoadThisDateData()

{

if (currentUser == null)

{

MessageBox.Show("Ошибка! Данные пользователя не загрузились");

return;

}

List<Symptom> allSymptoms = \_context.Symptoms.ToList();

symptomItems = new List<SymptomItem>();

foreach (Symptom? symptom in allSymptoms)

{

SymptomItem? item = new SymptomItem();

item.Id = symptom.Id;

item.Name = symptom.Name;

item.IsSelected = false;

symptomItems.Add(item);

}

SymptomsItems.ItemsSource = symptomItems;

DateTime selectedDate = DPDate.SelectedDate ?? DateTime.Today;

dailyData = \_context.DailyData?.FirstOrDefault(d => d.UserId == currentUser.Id && d.Date.Date == selectedDate.Date);

if (dailyData != null)

{

BoxSystolic.Text = dailyData.SystolicPressure?.ToString();

BoxDiastolic.Text = dailyData.DiastolicPressure?.ToString();

BoxPulse.Text = dailyData.Pulse?.ToString();

BoxWeight.Text = dailyData.Weight?.ToString();

if (dailyData.WellBeingRating.HasValue)

ComboBoxWellBeing.SelectedIndex = dailyData.WellBeingRating.Value - 1;

BoxNotes.Text = dailyData.Notes;

if (dailyData.SleepStart.HasValue)

BoxSleepStart.Text = dailyData.SleepStart.Value.ToString("HH:mm");

if (dailyData.SleepEnd.HasValue)

BoxSleepEnd.Text = dailyData.SleepEnd.Value.ToString("HH:mm");

if (dailyData.SleepQuality.HasValue)

ComboBoxSleepQuality.SelectedIndex = dailyData.SleepQuality.Value - 1;

BoxNightAwakening.Text = dailyData.NightAwake?.ToString();

BoxSteps.Text = dailyData.StepCount?.ToString();

if (!string.IsNullOrEmpty(dailyData.Symptoms))

{

var selectedSymptomIds = dailyData.Symptoms.Split(',')

.Where(s => !string.IsNullOrEmpty(s))

.Select(int.Parse)

.ToList();

foreach (var item in symptomItems)

{

item.IsSelected = selectedSymptomIds.Contains(item.Id);

}

SymptomsItems.Items.Refresh();

}

}

else

{

dailyData = new DailyData();

BoxSleepStart.Text = "22:00";

BoxSleepEnd.Text = "06:00";

BoxNightAwakening.Text = "0";

BoxSteps.Text = "0";

}

}

public bool CheckData()

{

if(!string.IsNullOrEmpty(BoxSystolic.Text) && !string.IsNullOrEmpty(BoxDiastolic.Text) && !string.IsNullOrEmpty(BoxPulse.Text))

{

try

{

int systolic = int.Parse(BoxSystolic.Text);

int diastolic = int.Parse(BoxDiastolic.Text);

int pulse = int.Parse(BoxPulse.Text);

if (systolic < 60 || systolic > 250)

{

MessageBox.Show("Систолическое давление должно быть между 60 и 250");

return false;

}

if (diastolic < 40 || diastolic > 150)

{

MessageBox.Show("Диастолическое давление должно быть между 40 и 150");

return false;

}

if (systolic <= diastolic)

{

MessageBox.Show("Систолическое давление должно быть больше диастолического");

return false;

}

if (pulse < 30 || pulse > 200)

{

MessageBox.Show("Пульс должен быть между 30 и 200");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Давление и пульс должны быть целыми числами");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Превышено допустимое значение!");

return false;

}

}

if (!string.IsNullOrEmpty(BoxWeight.Text))

{

try

{

double weight = double.Parse(BoxWeight.Text);

if (weight < 20 || weight > 300)

{

MessageBox.Show("Вес должен быть между 20 и 300 кг");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Вес должен быть числом");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Превышено допустимое значение!");

return false;

}

}

if (!string.IsNullOrEmpty(BoxSleepStart.Text) && !string.IsNullOrEmpty(BoxSleepEnd.Text))

{

try

{

TimeSpan sleepStartTime = TimeSpan.Parse(BoxSleepStart.Text);

TimeSpan sleepEndTime = TimeSpan.Parse(BoxSleepEnd.Text);

}

catch (FormatException)

{

MessageBox.Show("Время сна должно быть указано в формате чч:мм");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Превышено допустимое значение!");

return false;

}

}

if (!string.IsNullOrEmpty(BoxNightAwakening.Text))

{

try

{

int awake = int.Parse(BoxNightAwakening.Text);

if (awake < 0 || awake > 20)

{

MessageBox.Show("Количество ночных пробуждений не может быть отрицательным или превышать значение 20");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Количество ночных пробуждений должно быть целым числом");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Превышено допустимое значение!");

return false;

}

}

if (!string.IsNullOrEmpty(BoxSteps.Text))

{

try

{

int steps = int.Parse(BoxSteps.Text);

if (steps < 0 || steps > 200000)

{

MessageBox.Show("Количество шагов не может быть отрицательным или превышать значение 200000");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Количество шагов должно быть целым числом");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Превышено допустимое значение!");

return false;

}

}

return true;

}

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

{

if (CheckData() == false)

return;

try

{

dailyData.UserId = currentUser.Id;

dailyData.Date = DPDate.SelectedDate ?? DateTime.Today;

dailyData.SystolicPressure = Convert.ToInt32(BoxSystolic.Text);

dailyData.DiastolicPressure = Convert.ToInt32(BoxDiastolic.Text);

dailyData.Pulse = Convert.ToInt32(BoxPulse.Text);

dailyData.Weight = Convert.ToDouble(BoxWeight.Text);

dailyData.WellBeingRating = ComboBoxWellBeing.SelectedIndex + 1;

dailyData.Notes = BoxNotes.Text;

List<int> selectedSymptomIds = symptomItems.Where(s => s.IsSelected).Select(s => s.Id).ToList();

if (selectedSymptomIds.Count > 0)

{

dailyData.Symptoms = string.Join(",", selectedSymptomIds);

}

else

dailyData.Symptoms = null;

if (!string.IsNullOrEmpty(BoxSleepStart.Text) && !string.IsNullOrEmpty(BoxSleepEnd.Text))

{

try

{

TimeSpan sleepStartTime = TimeSpan.Parse(BoxSleepStart.Text);

TimeSpan sleepEndTime = TimeSpan.Parse(BoxSleepEnd.Text);

DateTime selectedDate = DPDate.SelectedDate ?? DateTime.Today;

DateTime sleepStart = selectedDate.Date + sleepStartTime;

DateTime sleepEnd = selectedDate.Date + sleepEndTime;

if (sleepEnd <= sleepStart)

sleepEnd = sleepEnd.AddDays(1);

dailyData.SleepStart = sleepStart;

dailyData.SleepEnd = sleepEnd;

dailyData.SleepQuality = ComboBoxSleepQuality.SelectedIndex + 1;

}

catch (FormatException)

{

MessageBox.Show("Неверный формат даты! Введите время сна в формате чч:мм (22:00)");

return;

}

}

dailyData.NightAwake = Convert.ToInt32(BoxNightAwakening.Text);

dailyData.StepCount = Convert.ToInt32(BoxSteps.Text);

dailyData.UpdateAt = DateTime.Now;

if (dailyData.Id == 0)

{

dailyData.CreateAt = DateTime.Now;

\_context.DailyData.Add(dailyData);

}

\_context.SaveChanges();

MessageBox.Show("Данные успешно внесены!");

this.DialogResult = true;

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"При внесении данных возникла ошибка: {ex.Message}");

}

}

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

{

this.DialogResult=false;

this.Close();

}

private void DPDate\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

if (!DPDate.SelectedDate.HasValue)

return;

if (DPDate.SelectedDate > DateTime.Today)

{

MessageBox.Show("Нельзя вносить данные за следующие дни");

DPDate.SelectedDate = DateTime.Today;

return;

}

LoadThisDateData();

if (DPDate.SelectedDate < DateTime.Today.AddDays(-1) && (dailyData == null || dailyData.Id == 0))

{

MessageBox.Show("Запрещено создавать новые данные за предыдущие дни. Можно создать запись за вчерашний день или редактировать только существующие записи");

DPDate.SelectedDate = DateTime.Today;

LoadThisDateData();

return;

}

}

}

}

MedicineEditWindow.xaml.cs

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для MedicineEditWindow.xaml

/// </summary>

public partial class MedicineEditWindow : Window

{

private User currentUser;

private Medicine editMedicine;

private bool isEditing = false;

private Models.AppContext \_context;

public MedicineEditWindow(User user, Models.AppContext appContext, Medicine medicine = null)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

if (medicine != null)

{

editMedicine = medicine;

isEditing = true;

LoadMedicineData();

}

else

{

editMedicine = new Medicine();

DPStartDate.SelectedDate = DateTime.Today;

BoxIntakeTime.Text = "08:00";

BoxMonday.IsChecked = true;

}

}

public bool CheckMedicineData()

{

if (string.IsNullOrWhiteSpace(BoxMedicineName.Text))

{

MessageBox.Show("Название лекарства не может быть пустым");

return false;

}

if (DPStartDate.SelectedDate.HasValue && DPStartDate.SelectedDate.Value > DateTime.Today.AddDays(1))

{

MessageBox.Show("Дата начала приема не может быть в будущем");

return false;

}

if (DPEndDate.SelectedDate.HasValue && DPStartDate.SelectedDate.HasValue)

{

if(DPEndDate.SelectedDate.Value < DPStartDate.SelectedDate.Value)

{

MessageBox.Show("Дата окончания приема не может быть раньше даты начала");

return false;

}

}

if (string.IsNullOrWhiteSpace(BoxIntakeTime.Text))

{

MessageBox.Show("Введите время приема");

return false;

}

try

{

TimeSpan intakeTime = TimeSpan.Parse(BoxIntakeTime.Text);

}

catch (Exception ex)

{

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

return false;

}

if (!(BoxMonday.IsChecked == true || BoxTuesday.IsChecked == true || BoxWednesday.IsChecked == true || BoxThursday.IsChecked == true ||

BoxFriday.IsChecked == true || BoxSaturday.IsChecked == true || BoxSunday.IsChecked == true))

{

MessageBox.Show("Выберите хотя бы один день для приема");

return false;

}

return true;

}

public bool ShouldTakeOnDay(DayOfWeek dayOfWeek)

{

if (dayOfWeek == DayOfWeek.Monday) return BoxMonday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Tuesday) return BoxTuesday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Wednesday) return BoxWednesday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Thursday) return BoxThursday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Friday) return BoxFriday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Saturday) return BoxSaturday.IsChecked == true;

if (dayOfWeek == DayOfWeek.Sunday) return BoxSunday.IsChecked == true;

return false;

}

public void LoadMedicineData()

{

BoxMedicineName.Text = editMedicine.MedicineName;

BoxDose.Text = editMedicine.Dose;

BoxInstructions.Text = editMedicine.Instructions;

DPStartDate.SelectedDate = editMedicine.StartDate;

if (editMedicine.EndDate.HasValue)

DPEndDate.SelectedDate = editMedicine.EndDate.Value;

BoxIsActive.IsChecked = editMedicine.IsActive;

BoxMonday.IsChecked = editMedicine.Monday;

BoxTuesday.IsChecked = editMedicine.Tuesday;

BoxWednesday.IsChecked = editMedicine.Wednesday;

BoxThursday.IsChecked = editMedicine.Thursday;

BoxFriday.IsChecked = editMedicine.Friday;

BoxSaturday.IsChecked = editMedicine.Saturday;

BoxSunday.IsChecked = editMedicine.Sunday;

BoxIntakeTime.Text = editMedicine.IntakeTime.ToString(@"hh\:mm");

}

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

{

if (CheckMedicineData() == false)

return;

try

{

editMedicine.MedicineName = BoxMedicineName.Text;

editMedicine.Dose = BoxDose.Text;

editMedicine.Instructions = BoxInstructions.Text;

editMedicine.StartDate = DPStartDate.SelectedDate ?? DateTime.Today;

editMedicine.EndDate = DPEndDate.SelectedDate;

editMedicine.IsActive = BoxIsActive.IsChecked ?? true;

editMedicine.UserId = currentUser.Id;

editMedicine.IntakeTime = TimeSpan.Parse(BoxIntakeTime.Text);

editMedicine.Monday = BoxMonday.IsChecked ?? false;

editMedicine.Tuesday = BoxTuesday.IsChecked ?? false;

editMedicine.Wednesday = BoxWednesday.IsChecked ?? false;

editMedicine.Thursday = BoxThursday.IsChecked ?? false;

editMedicine.Friday = BoxFriday.IsChecked ?? false;

editMedicine.Saturday = BoxSaturday.IsChecked ?? false;

editMedicine.Sunday = BoxSunday.IsChecked ?? false;

editMedicine.UpdateAt = DateTime.Now;

if (!isEditing)

{

\_context.Medicines.Add(editMedicine);

}

\_context.SaveChanges();

this.DialogResult = true;

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Возникла ошибка в момент сохранения: {ex.Message}");

}

}

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

{

this.DialogResult = false;

this.Close();

}

}

}

MedicineControlPage.xaml.cs

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WellBeingDiary.Models;

using WellBeingDiary.Windows;

namespace WellBeingDiary.UserControllers

{

/// <summary>

/// Логика взаимодействия для MedicineControlPage.xaml

/// </summary>

public partial class MedicineControlPage : UserControl

{

private User currentUser;

private Models.AppContext \_context;

public MedicineControlPage(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

LoadMedicines();

}

public void LoadMedicines()

{

List<Medicine>? medicines = \_context.Medicines?.Where(m => m.UserId == currentUser.Id).ToList();

MedicinesListBox.ItemsSource = medicines;

}

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

{

Window medicineEditWindow = new MedicineEditWindow(currentUser, \_context);

if (medicineEditWindow.ShowDialog() == true)

LoadMedicines();

}

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

{

Medicine selectedMedicine = (Medicine)MedicinesListBox.SelectedItem;

if (selectedMedicine != null)

{

Window medicineEditWindow = new MedicineEditWindow(currentUser, \_context, selectedMedicine);

if (medicineEditWindow.ShowDialog() == true)

LoadMedicines();

}

else

{

MessageBox.Show("Выберите лекарство для редактирования!");

}

}

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

{

Medicine selectedMedicine = (Medicine)MedicinesListBox.SelectedItem;

if (selectedMedicine != null)

{

var result = MessageBox.Show("Удалить это лекарство?", "Подтвердите действие", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result.ToString() == "Yes")

{

try

{

\_context.Medicines.Remove(selectedMedicine);

\_context.SaveChanges();

LoadMedicines();

MessageBox.Show("Лекарство удалено");

}

catch (Exception ex)

{

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

}

}

}

else

{

MessageBox.Show("Выберите лекарство для удаления!");

}

}

}

}

ReportMakeWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using LiveCharts;

using LiveCharts.Wpf;

using WellBeingDiary.Models;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для ReportMakeWindow.xaml

/// </summary>

public partial class ReportMakeWindow : Window

{

private User currentUser;

private Models.AppContext \_context;

public SeriesCollection ReportCollection { get; set; }

public List<string> Labels { get; set; }

public ReportMakeWindow(User user, Models.AppContext appContext)

{

currentUser = user;

\_context = appContext;

InitializeComponent();

DPStartDate.SelectedDate = DateTime.Today.AddDays(-30);

DPEndDate.SelectedDate = DateTime.Today;

ComboBoxReportType.SelectedIndex = 0;

ReportCollection = new SeriesCollection();

Labels = new List<string>();

ConfigureAxes();

DataContext = this;

}

public void ConfigureAxes()

{

ReportChart.AxisX.Clear();

ReportChart.AxisX.Add(new Axis { Title = "Дата", Labels = Labels });

ReportChart.AxisY.Clear();

ReportChart.AxisY.Add(new Axis { Title = "Значения" });

}

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

{

if (!DPStartDate.SelectedDate.HasValue || !DPEndDate.SelectedDate.HasValue)

{

MessageBox.Show("Выберите период для формирования отчета!");

return;

}

DateTime startDate = DPStartDate.SelectedDate.Value;

DateTime endDate = DPEndDate.SelectedDate.Value;

if (startDate > endDate)

{

MessageBox.Show("Начальная дата не может быть больше конечной!");

return;

}

if ((endDate-startDate).TotalDays > 365)

{

MessageBox.Show("Период отчета не может превышать 1 год!");

return;

}

List<DailyData>? reportData = \_context.DailyData?.Where(d => d.UserId == currentUser.Id && d.Date >= startDate && d.Date <= endDate).OrderBy(d => d.Date).ToList();

if (reportData == null || reportData.Count == 0)

{

BoxReport.Text = "Нет данных за выбранный период.";

return;

}

string reportType = (ComboBoxReportType.SelectedItem as ComboBoxItem)?.Content.ToString();

string reportText = GenerateReportText(reportType, reportData, startDate, endDate);

BoxReport.Text = reportText;

GenerateReportChart(reportType, reportData);

}

public string GenerateReportText(string reportType, List<DailyData> data, DateTime startDate, DateTime endDate)

{

StringBuilder sb = new StringBuilder();

sb.AppendLine($"Тип отчета: {reportType}");

sb.AppendLine($"Период: {startDate:dd.MM.yyyy} - {endDate:dd.MM.yyyy}");

sb.AppendLine($"Пользователь: {currentUser.Name}");

sb.AppendLine();

if (reportType == "Отчет по самочувствию")

{

WellBeingReport(sb, data);

}

else if (reportType == "Отчет по давлению")

{

BloodPressureReport(sb, data);

}

else if (reportType == "Отчет по сну")

{

SleepReport(sb, data);

}

else if (reportType == "Отчет по двигательной активности")

{

ActivityReport(sb, data);

}

else if (reportType == "Отчет по весу")

{

WeightReport(sb, data);

}

sb.AppendLine();

sb.AppendLine(ReportRecommendations(reportType, data));

return sb.ToString();

}

public void WellBeingReport(StringBuilder sb, List<DailyData> data)

{

List<DailyData> validData = data.Where(d => d.WellBeingRating.HasValue).ToList();

if (validData.Count == 0)

{

MessageBox.Show("Нет данных о самочувствии за период");

return;

}

double avgRating = validData.Average(d => d.WellBeingRating.Value);

sb.AppendLine($"Средняя оценка самочувствия: {avgRating:f1}/5");

sb.AppendLine();

sb.AppendLine("Динамика изменений:");

foreach (DailyData day in validData)

{

string rateText = WellBeingText(day.WellBeingRating.Value);

sb.AppendLine($"{day.Date:dd.MM.yyyy}: {rateText} ({day.WellBeingRating}/5)");

}

}

public void BloodPressureReport(StringBuilder sb, List<DailyData> data)

{

List<DailyData> validData = data.Where(d => d.SystolicPressure.HasValue && d.DiastolicPressure.HasValue && d.Pulse.HasValue).ToList();

if (validData.Count == 0)

{

MessageBox.Show("Нет данных о давлении за период");

return;

}

double avgSystolic = validData.Average(d => d.SystolicPressure.Value);

double avgDiastolic = validData.Average(d => d.DiastolicPressure.Value);

double avgPulse = validData.Average(d => d.Pulse.Value);

sb.AppendLine($"Среднее давление и пульс: {avgSystolic:f0}/{avgDiastolic:f0}/{avgPulse:f0}");

sb.AppendLine();

List<DailyData> anomalies = validData.Where(d => d.SystolicPressure > 140 || d.DiastolicPressure > 90 || d.Pulse > 80 ||

d.SystolicPressure < 90 || d.DiastolicPressure < 60 || d.Pulse < 60).ToList();

if (anomalies.Count > 0)

{

sb.AppendLine("Выявлены нарушения!");

foreach (DailyData anomaly in anomalies)

{

sb.AppendLine($"{anomaly.Date:dd.MM.yyyy}: {anomaly.SystolicPressure}/{anomaly.DiastolicPressure}/{anomaly.Pulse}");

}

sb.AppendLine();

}

sb.AppendLine("Динамика изменений:");

foreach (DailyData? day in validData)

{

sb.AppendLine($"{day.Date:dd.MM.yyyy}: {day.SystolicPressure}/{day.DiastolicPressure}/{day.Pulse}");

}

}

public void SleepReport(StringBuilder sb, List<DailyData> data)

{

List<DailyData> validData = data.Where(d => d.SleepStart.HasValue && d.SleepEnd.HasValue).ToList();

if (validData.Count == 0)

{

MessageBox.Show("Нет данных о сне за период");

return;

}

double avgSleepHours = validData.Average(d => (d.SleepEnd.Value - d.SleepStart.Value).TotalHours);

double avgSleepQuality = validData.Where(d => d.SleepQuality.HasValue).Average(d => d.SleepQuality.Value);

sb.AppendLine($"Средняя продолжительность сна: {avgSleepHours:f1} ч");

sb.AppendLine($"Среднее качество сна: {avgSleepQuality:f1}/5");

sb.AppendLine();

sb.AppendLine("Динамика изменений:");

foreach (DailyData day in validData)

{

TimeSpan sleepTime = day.SleepEnd.Value - day.SleepStart.Value;

string quality = $", качество сна: {day.SleepQuality}/5";

sb.AppendLine($"{day.Date:dd.MM.yyyy}: {sleepTime.Hours}ч {sleepTime.Minutes}м{quality}");

}

}

public void ActivityReport(StringBuilder sb, List<DailyData> data)

{

List<DailyData> validData = data.Where(d => d.StepCount.HasValue).ToList();

if (validData.Count == 0)

{

MessageBox.Show("Нет данных о двигательной активности за период");

return;

}

double avgSteps = validData.Average(d => d.StepCount.Value);

int totalSteps = validData.Sum(d => d.StepCount.Value);

sb.AppendLine($"Среднее количество шагов: {avgSteps:f0}");

sb.AppendLine($"Общее количество шагов: {totalSteps}");

sb.AppendLine();

sb.AppendLine("Динамика изменений:");

foreach (DailyData day in validData)

{

sb.AppendLine($"{day.Date:dd.MM.yyyy}: {day.StepCount} шагов");

}

}

public void WeightReport(StringBuilder sb, List<DailyData> data)

{

List<DailyData> validData = data.Where(d => d.Weight.HasValue).ToList();

if (validData.Count == 0)

{

MessageBox.Show("Нет данных о весе за период");

return;

}

double firstWeight = validData.OrderBy(d => d.Date).First().Weight.Value;

double lastWeight = validData.OrderBy(d => d.Date).Last().Weight.Value;

double weightChange = lastWeight - firstWeight;

double days = (validData.Max(d => d.Date) - validData.Min(d => d.Date)).TotalDays;

double changeInDay;

if (days > 0)

changeInDay = weightChange / days;

else

changeInDay = 0;

sb.AppendLine($"Начальное значение: {firstWeight} кг");

sb.AppendLine($"Конечное значение: {lastWeight} кг");

sb.AppendLine($"Темп изменения: {changeInDay:f2} кг в день");

sb.AppendLine();

sb.AppendLine("Динамика изменений:");

foreach (DailyData day in validData)

{

sb.AppendLine($"{day.Date:dd.MM.yyyy}: {day.Weight} кг");

}

}

public string ReportRecommendations(string reportType, List<DailyData> data)

{

List<string> recommendations = new();

if (reportType == "Отчет по самочувствию")

{

List<DailyData> wellBeingData = data.Where(d => d.WellBeingRating.HasValue).ToList();

if (wellBeingData.Any())

{

double avgRating = wellBeingData.Average(d => d.WellBeingRating.Value);

if (avgRating < 3)

recommendations.Add("Рекомендуется отдых и снижение уровня стресса!");

if (avgRating < 2)

recommendations.Add("Стоит посетить врача");

}

}

else if (reportType == "Отчет по давлению")

{

List<DailyData> pressureData = data.Where(d => d.SystolicPressure.HasValue && d.DiastolicPressure.HasValue && d.Pulse.HasValue).ToList();

if (pressureData.Any())

{

double avgSystolic = pressureData.Average(d => d.SystolicPressure.Value);

double avgDiastolic = pressureData.Average(d => d.DiastolicPressure.Value);

double avgPulse = pressureData.Average(d => d.Pulse.Value);

if (avgSystolic > 130 || avgDiastolic > 85)

recommendations.Add("Рекомендуется снизить потребление соли");

if (avgSystolic > 140 || avgDiastolic > 90)

recommendations.Add("Проконсультируйтесь с врачом о возможной гипертонии");

if (avgSystolic < 100 || avgDiastolic < 60)

recommendations.Add("Увеличьте потребление жидкости");

if (avgPulse > 80)

recommendations.Add("Уменьшите стресс: чем выше уровень стресса, тем более высокий пульс");

if (avgPulse < 60)

recommendations.Add("Добавьте легкие физические нагрузки");

}

}

else if (reportType == "Отчет по сну")

{

List<DailyData> sleepData = data.Where(d => d.SleepStart.HasValue && d.SleepEnd.HasValue).ToList();

if (sleepData.Any())

{

double avgSleepHours = sleepData.Average(d => (d.SleepEnd.Value - d.SleepStart.Value).TotalHours);

if (avgSleepHours < 7)

recommendations.Add("Старайтесь спать не менее 7-8 часов в сутки");

if (avgSleepHours > 9)

recommendations.Add("Избыток сна может приводить к переутомлению");

}

}

else if (reportType == "Отчет по двигательной активности")

{

List<DailyData> activityData = data.Where(d => d.StepCount.HasValue).ToList();

if (activityData.Any())

{

double avgSteps = activityData.Average(d => d.StepCount.Value);

if (avgSteps < 5000)

recommendations.Add("Старайтесь проходить не менее 5000 шагов в день");

if (avgSteps < 3000)

recommendations.Add("Рекомендуется добавить легкие физические нагрузки");

if (avgSteps > 10000)

recommendations.Add("Вы - молодец! Продолжайте держать активность на этом уровне");

}

}

else if (reportType == "Отчет по весу")

{

List<DailyData> weightData = data.Where(d => d.Weight.HasValue).ToList();

double firstWeight = weightData.OrderBy(d => d.Date).First().Weight.Value;

double lastWeight = weightData.OrderBy(d => d.Date).Last().Weight.Value;

double change = lastWeight - firstWeight;

if (change < 0)

recommendations.Add($"Ваш вес уменьшился на {Math.Abs(change)} кг");

else if (change > 2)

recommendations.Add($"Ваш вес увеличился на {change} кг");

}

if (recommendations.Any())

return string.Join("\n", recommendations);

else

return "Рекомендаций нет. У вас все в порядке!";

}

public string WellBeingText(int rating)

{

if (rating == 1)

{

return "Плохое";

}

else if (rating == 2)

{

return "Удовлетворительное";

}

else if (rating == 3)

{

return "Нормальное";

}

else if (rating == 4)

{

return "Хорошее";

}

else if (rating == 5)

{

return "Отличное";

}

else

{

return "Не указано";

}

}

public void GenerateReportChart(string reportType, List<DailyData> data)

{

ReportCollection.Clear();

Labels.Clear();

ConfigureAxes();

if (reportType == "Отчет по самочувствию")

{

List<DailyData> wellBeingData = data.Where(d => d.WellBeingRating.HasValue).OrderBy(d => d.Date).ToList();

if (wellBeingData.Any())

{

Labels = wellBeingData.Select(d => d.Date.ToString("dd.MM")).ToList();

ReportCollection.Add(new LineSeries

{

Title = "Самочувствие",

Values = new ChartValues<int>(wellBeingData.Select(d => d.WellBeingRating.Value)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

}

}

else if (reportType == "Отчет по давлению")

{

List<DailyData> pressureData = data.Where(d => d.SystolicPressure.HasValue && d.DiastolicPressure.HasValue && d.Pulse.HasValue).OrderBy(d => d.Date).ToList();

if (pressureData.Any())

{

Labels = pressureData.Select(d => d.Date.ToString("dd.MM")).ToList();

ReportCollection.Add(new LineSeries

{

Title = "Систолическое",

Values = new ChartValues<int>(pressureData.Select(d => d.SystolicPressure.Value)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

ReportCollection.Add(new LineSeries

{

Title = "Диастолическое",

Values = new ChartValues<int>(pressureData.Select(d => d.DiastolicPressure.Value)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

ReportCollection.Add(new LineSeries

{

Title = "Пульс",

Values = new ChartValues<int>(pressureData.Select(d => d.Pulse.Value)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

}

}

else if (reportType == "Отчет по сну")

{

List<DailyData> sleepData = data.Where(d => d.SleepStart.HasValue && d.SleepEnd.HasValue).OrderBy(d => d.Date).ToList();

if (sleepData.Any())

{

Labels = sleepData.Select(d => d.Date.ToString("dd.MM")).ToList();

ReportCollection.Add(new LineSeries

{

Title = "Часы сна",

Values = new ChartValues<double>(sleepData.Select(d => (d.SleepEnd.Value - d.SleepStart.Value).TotalHours)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

}

}

else if (reportType == "Отчет по двигательной активности")

{

List<DailyData> activityData = data.Where(d => d.StepCount.HasValue).OrderBy(d => d.Date).ToList();

if (activityData.Any())

{

Labels = activityData.Select(d => d.Date.ToString("dd.MM")).ToList();

ReportCollection.Add(new ColumnSeries

{

Title = "Шаги",

Values = new ChartValues<int>(activityData.Select(d => d.StepCount.Value))

});

}

}

else if (reportType == "Отчет по весу")

{

List<DailyData> weightData = data.Where(d => d.Weight.HasValue).OrderBy(d => d.Date).ToList();

if (weightData.Any())

{

Labels = weightData.Select(d => d.Date.ToString("dd.MM")).ToList();

ReportCollection.Add(new LineSeries

{

Title = "Вес",

Values = new ChartValues<double>(weightData.Select(d => d.Weight.Value)),

PointGeometry = DefaultGeometries.Circle,

PointGeometrySize = 8

});

}

}

ReportChart.AxisX.Clear();

ReportChart.AxisX.Add(new Axis

{

Title = "Дата",

Labels = Labels

});

ReportChart.Update(true);

}

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

{

if (string.IsNullOrEmpty(BoxReport.Text) || BoxReport.Text.StartsWith("Нет данных"))

{

MessageBox.Show("Сначала сгенерируйте отчет!");

return;

}

int newId;

if (\_context.Reports.Any())

{

newId = \_context.Reports.Max(r => r.Id) + 1;

}

else

{

newId = 1;

}

try

{

Report? report = new Report

{

Id = newId,

UserId = currentUser.Id,

ReportType = (ComboBoxReportType.SelectedItem as ComboBoxItem)?.Content.ToString() ?? "Общий отчет",

StartDate = DPStartDate.SelectedDate ?? DateTime.Today.AddDays(-7),

EndDate = DPEndDate.SelectedDate ?? DateTime.Today,

ReportData = BoxReport.Text,

CreateAt = DateTime.Now

};

\_context.Reports.Add(report);

\_context.SaveChanges();

MessageBox.Show("Отчет сохранен!");

}

catch (Exception ex)

{

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

}

}

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

{

this.Close();

}

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

{

ReportCollection?.Clear();

Labels?.Clear();

ConfigureAxes();

}

}

}

RegWindow.xaml.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WellBeingDiary.Models;

namespace WellBeingDiary.Windows

{

/// <summary>

/// Логика взаимодействия для RegWindow.xaml

/// </summary>

public partial class RegWindow : Window

{

private User newUser = new User();

private Models.AppContext \_context;

private string selectedImagePath = string.Empty;

public RegWindow(Models.AppContext appContext)

{

\_context = appContext;

InitializeComponent();

}

public bool CheckData()

{

if (string.IsNullOrWhiteSpace(BoxName.Text))

{

MessageBox.Show("Имя не может быть пустым");

return false;

}

if (string.IsNullOrWhiteSpace(BoxLogin.Text))

{

MessageBox.Show("Логин пользователя не может быть пустым");

return false;

}

if (\_context.Users.Any(u => u.Login == BoxLogin.Text))

{

MessageBox.Show("Пользователь с таким логином уже существует");

return false;

}

if (string.IsNullOrWhiteSpace(BoxPassword.Password))

{

MessageBox.Show("Пароль пользователя не может быть пустым");

return false;

}

if (!string.IsNullOrWhiteSpace(BoxHeight.Text))

{

try

{

int height = int.Parse(BoxHeight.Text);

if (height < 60 || height > 250)

{

MessageBox.Show("Введите корректное значение роста (от 60 до 250 см)");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Рост должен быть целым числом");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Введите корректное значение!");

return false;

}

}

else if (string.IsNullOrWhiteSpace(BoxHeight.Text))

{

MessageBox.Show("Заполните ваш рост!");

return false;

}

if (!string.IsNullOrWhiteSpace(BoxWeight.Text))

{

try

{

double weight = double.Parse(BoxWeight.Text);

if (weight < 20 || weight > 300)

{

MessageBox.Show("Введите корректное значение веса (от 20 до 300 кг)");

return false;

}

}

catch (FormatException)

{

MessageBox.Show("Вес должен быть числом");

return false;

}

catch (OverflowException)

{

MessageBox.Show("Введите корректное значение!");

return false;

}

}

else if (string.IsNullOrWhiteSpace(BoxWeight.Text))

{

MessageBox.Show("Заполните ваш вес!");

return false;

}

if (DPBirthDate.SelectedDate.HasValue)

{

if (DPBirthDate.SelectedDate.Value > DateTime.Today)

{

MessageBox.Show($"Вы что, из будущего? Дата рождения не может быть позже {DateTime.Today}");

return false;

}

if (DateTime.Today.Year - DPBirthDate.SelectedDate.Value.Year > 125)

{

MessageBox.Show($"Проверьте введенные данные. Дата рождения пользователя не может быть больше {DateTime.Now.AddYears(-125)}");

return false;

}

}

return true;

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Image files (\*.jpg, \*.jpeg, \*.png)|\*.jpg;\*.jpeg;\*.png";

openFileDialog.Title = "Выберите файл:";

if (openFileDialog.ShowDialog() == true)

{

selectedImagePath = openFileDialog.FileName;

newUser.PhotoPath = selectedImagePath;

BitmapImage bitmap = new();

bitmap.BeginInit();

bitmap.UriSource = new Uri(System.IO.Path.GetFullPath(selectedImagePath), UriKind.Absolute);

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.EndInit();

ImgUserPhoto.Source = bitmap;

}

}

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

{

if (CheckData() == false)

return;

try

{

newUser.Name = BoxName.Text;

newUser.Login = BoxLogin.Text;

newUser.Password = BoxPassword.Password;

if (ComboBoxGender.SelectedItem == null)

{

MessageBox.Show("Выберите пол");

return;

}

newUser.Gender = (ComboBoxGender.SelectedItem as ComboBoxItem)?.Tag?.ToString();

if (!string.IsNullOrWhiteSpace(BoxHeight.Text))

newUser.Height = int.Parse(BoxHeight.Text);

if (!string.IsNullOrWhiteSpace(BoxWeight.Text))

newUser.Weight = double.Parse(BoxWeight.Text);

if (DPBirthDate.SelectedDate.HasValue)

newUser.DateOfBirth = DateOnly.FromDateTime(DPBirthDate.SelectedDate.Value);

if (!string.IsNullOrEmpty(selectedImagePath))

{

string extension = System.IO.Path.GetExtension(selectedImagePath);

string fileName = $"{newUser.Id}\_{newUser.Login}{extension}";

string imagePath = "./Images/" + fileName;

System.IO.File.Copy(selectedImagePath, imagePath, true);

newUser.PhotoPath = imagePath;

}

else

newUser.PhotoPath = "./Images/none.png";

\_context.Users.Add(newUser);

\_context.SaveChanges();

MessageBox.Show("Регистрация завершена!");

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Возникла ошибка при регистрации: {ex.Message}");

}

}

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

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}