МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по производственной практике**

**ПМ01. Разработка модулей программного обеспечения для компьютерных систем**

**Тема: «Разработка программного модуля «Частная стоматологическая клиника»**

Студент

*Кротова Ксения Николаевна*

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*Калинин Арсений Олегович*

Руководитель практики от организации:

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Сергей Михайлович*

подпись

УТВЕРЖДАЮ:

Директор

\_\_ Устюжанинов Вадим Борисович \_\_\_\_\_\_

Наименование организации

ООО «Электрон-Софт»

\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка

М. П.

2024-2025 уч. год

**Содержание**

1. СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ (БАЗЕ ПРАКТИКИ)3
2. РАЗРАБОТКА И АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ 4
3. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПО АЛГОРИТМУ В СООТВЕТСТВИИ ТЕХНИЧЕСКОМУ ЗАДАНИЮ 6
4. ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ МОДУЛЬ 12
5. ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНОГО МОДУЛЯ 21
6. ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ 23
7. ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ 26
8. РЕФАКТОРИНГ И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА 27
9. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА 29
10. ЗАКЛЮЧЕНИЕ34
11. ПРИЛОЖЕНИЯ К ОТЧЕТУ: ССЫЛКА НА РЕПОЗИТОРИЙ ОТЧЕТ В ЭЛЕКТРОННОМ ВИДЕ, ПРЕЗЕНТАЦИЯ ДЛЯ ВЫСТУПЛЕНИЯ, ПРОЕКТ ПРОГРАММНОГО МОДУЛЯ, ПРОЕКТ МОБИЛЬНОГО ПРИЛОЖЕНИЯ И ДР. МАТЕРИАЛЫ. 35
12. **СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ (БАЗЕ ПРАКТИКИ)**

Производственную практику по модулю ПМ.01. «Разработка модулей программного обеспечения для компьютерных систем» я проходила в компании ООО «Электрон-Софт»:

**Юридический адрес:**

* 610004, Кировская область, г. Киров, ул. Ленина, д. 68

**Специализация:**

* ООО «Электрон-Софт» является компанией, специализирующейся на выполнение работ и оказание услуг в сфере 1С.

**Структура (управление):**

* Директор
* Генеральный директор
* Заместитель директора
* Заместитель генерального директора

В первый день прохождения данного модуля нам провели инструктаж по технике безопасности и провели онлайн-тестирование.

1. **РАЗРАБОТКА И АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

На данном этапе было проведено детальное изучение предоставленного технического задания (ТЗ). В рамках анализа ТЗ были уточнены ключевые требования к базе данных, определены основные сущности, их атрибуты и взаимосвязи. Для визуализации процессов и структуры данных были разработаны следующие диаграммы:

Диаграмма вариантов использования — для отображения основных сценариев взаимодействия пользователей с системой (рисунок 2.1).

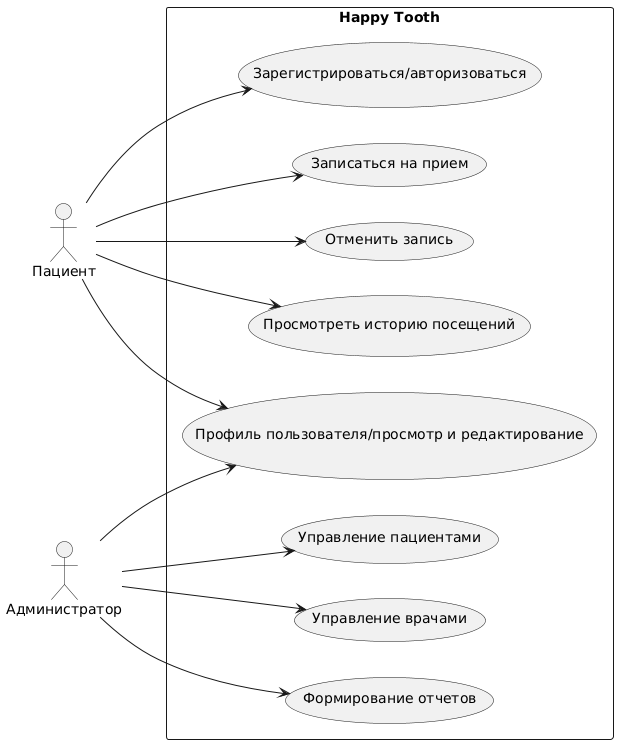


Рисунок 2.1 - Диаграмма вариантов использования

На основе анализа ТЗ и разработанных диаграмм были сформулированы уточненные требования к реализации базы данных. Также был произведен выбор СУБД (Microsoft SQL Server) и подготовлена основа для следующего этапа — непосредственного создания базы данных.

В результате работы была успешно создана база данных, соответствующая всем требованиям. Её структура отражена в ER-диаграмме (рисунок 2.2), которая включает:

* Сущности и их атрибуты,
* Связи между сущностями,
* Ограничения и ключи.

Диаграмма наглядно демонстрирует организацию данных и обеспечивает понимание логики работы БД.

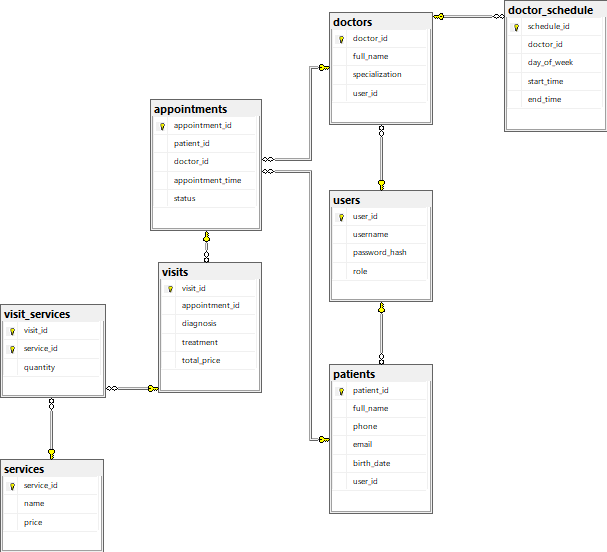


Рисунок 2.2 - ER-диаграмма базы данных

1. **РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПО АЛГОРИТМУ В СООТВЕТСТВИИ ТЕХНИЧЕСКОМУ ЗАДАНИЮ**

На данном этапе была выполнена разработка программного модуля, включающего в себя библиотеку для работы с данными и настольное приложение, в строгом соответствии с техническим заданием (ТЗ). Основной задачей было реализовать функционал, который обеспечивает корректную работу системы.

**Разработка библиотеки классов**

Библиотека HappyToothDLL разработана на языке программирования C# с использованием Entity Framework (ADO.NET Entity Data Model) для работы с базой данных стоматологической клиники. Библиотека предоставляет удобный интерфейс для управления основными бизнес-процессами клиники.

**Основные функциональные возможности:**

**Управление записями на прием:**

* Получение всех записей для конкретного пациента (GetAppointmentsForPatient)
* Получение всех записей для конкретного врача (GetAppointmentsForDoctor)
* Получение предстоящих записей (GetUpcomingAppointments)
* Поиск доступных временных слотов для записи (GetAvailableTimeSlots)

**Работа с визитами пациентов:**

* Получение завершенных визитов пациента (GetCompletedVisitsForPatient)
* Расчет общего дохода от всех визитов (CalculateTotalRevenue)
* Анализ доходов по типам услуг (CalculateRevenueByService)

**Управление расписанием врачей:**

* Получение расписания конкретного врача (GetDoctorSchedule)
* Проверка доступности врача в конкретный день

**Аналитика и отчетность:**

* Получение статистики по пациентам по году рождения (GetPatientsByBirthYear)
* Анализ популярности услуг (GetPopularServices)

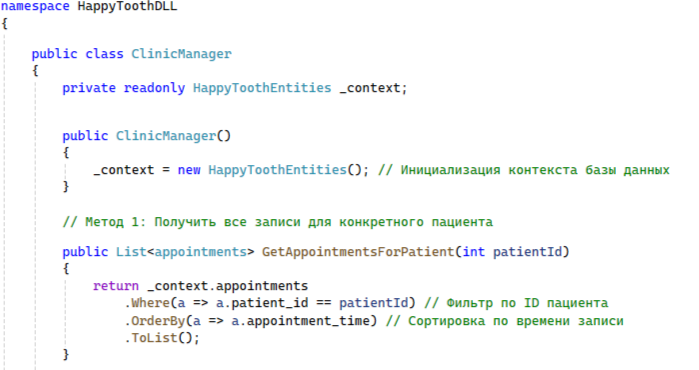


Рисунок 3.1 – Методы библиотеки классов

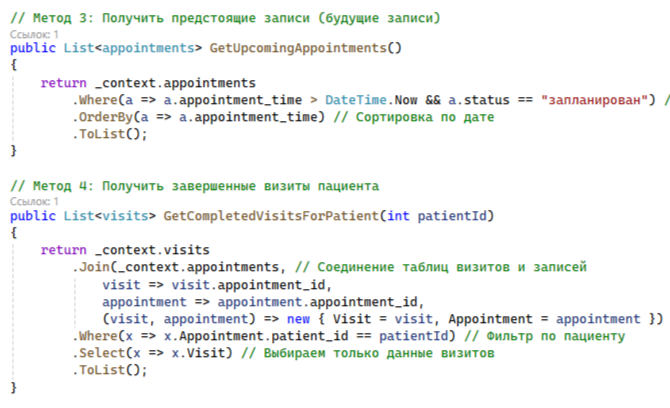


Рисунок 3.2 – Методы библиотеки классов

**Разработка настольного приложения**

Настольное приложение было разработано для администратора клиники с использованием WPF для C#. Основные функции приложения:

* учет пациентов,
* управление врачами,
* просмотр записей на прием и истории посещений пациентов,
* формирование отчетности.
* Профиль пользователя: редактирование данных и смена пароля.
* Подключение к базе данных — приложение работает через подключение к БД по технологии ADO NET.EDM.

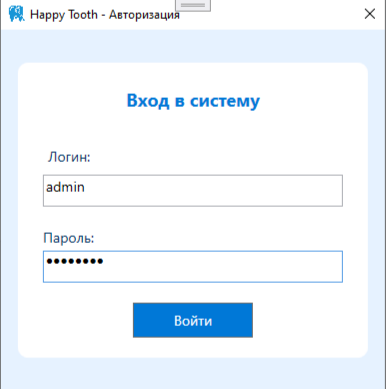


Рисунок 3.3 – Настольное приложение

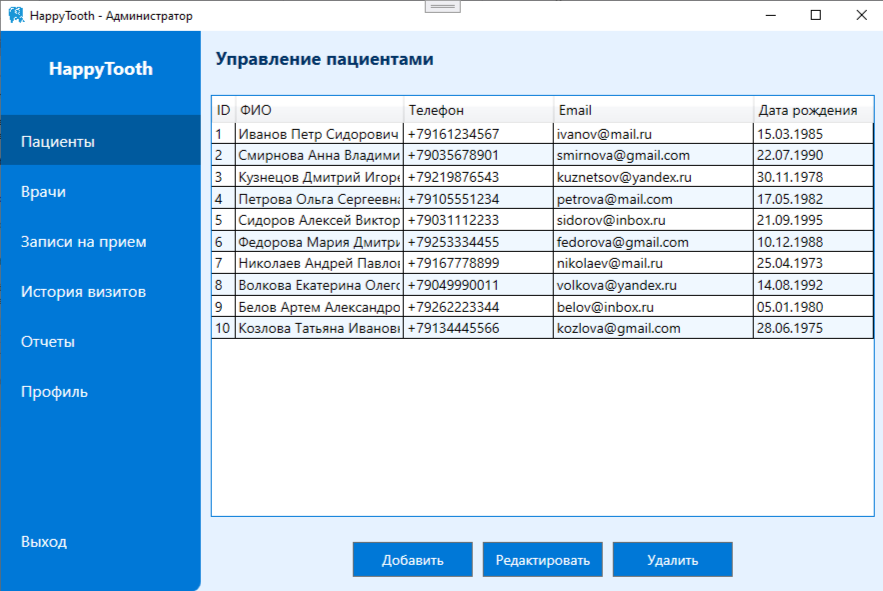


Рисунок 3.4 – Настольное приложение

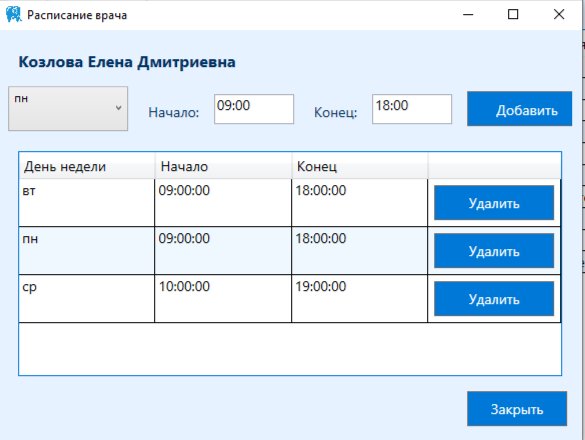


Рисунок 3.5 – Настольное приложение

**Разработка API**

На основе библиотеки HappyToothDLL разработан RESTful API с использованием ASP.NET Core.

Основные маршруты API:

**Управление записями**

* GET /api/appointments/patient/{patientId} - Получить все записи пациента
* GET /api/appointments/doctor/{doctorId} - Получить все записи врача
* GET /api/appointments/upcoming - Получить предстоящие записи (с параметрами фильтрации)

**Работа с визитами**

* GET /api/visits/patient/{patientId}/completed - Получить завершенные визиты пациента
* POST /api/visits - Создать новый визит

**Финансовая аналитика**

* GET /api/revenue/total - Получить общий доход (с параметрами периода)
* GET /api/revenue/by-service - Получить доход по услугам (с параметрами периода)

**Управление расписанием**

* GET /api/schedule/doctor/{doctorId} - Получить расписание врача
* GET /api/schedule/doctor/{doctorId}/available-slots - Получить доступные слоты (с параметром даты)

**Аналитика пациентов**

* GET /api/patients/by-birth-year/{year} - Получить пациентов по году рождения

**Аналитика услуг**

* GET /api/services/popular/{topCount} - Получить популярные услуги (топ-N)

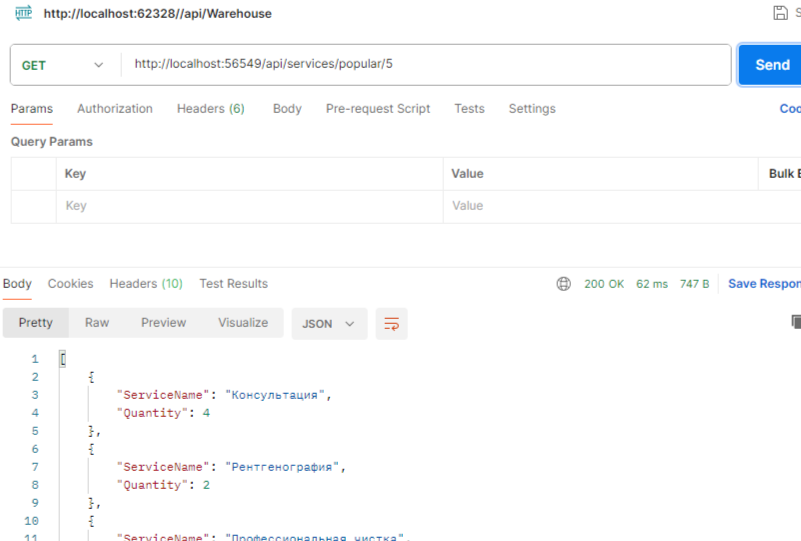
****

Рисунок 3.6 – API

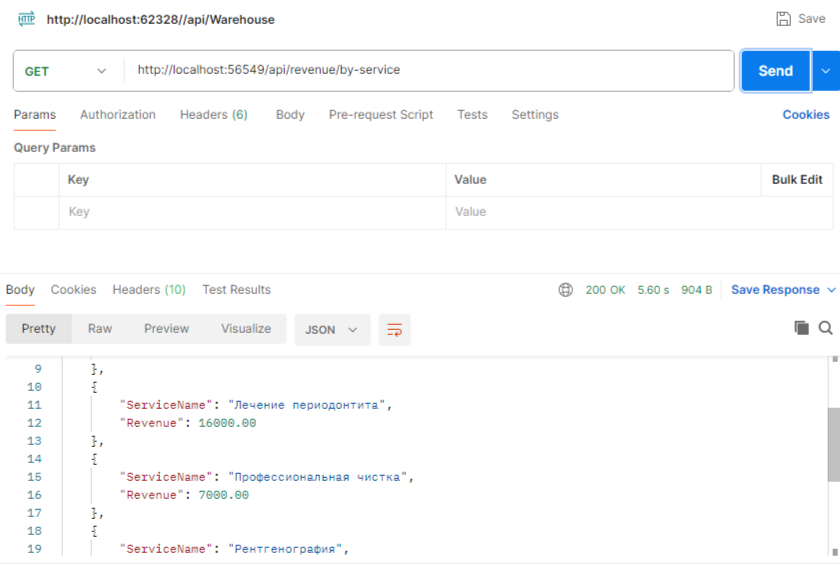
****

Рисунок 3.7 – API

Результатом этапа стал полностью функциональный программный модуль, включающий в себя библиотеку для работы с данными, API и настольное приложение, соответствующие всем требованиям ТЗ. Модуль готов к дальнейшему тестированию и отладке.

1. **ОФОРМЛЕНИЕ ДОКУМЕНТАЦИИ НА ПРОГРАММНЫЙ МОДУЛЬ**

На данном этапе была выполнена разработка и оформление документации на программный модуль в соответствии с требованиями технического задания (ТЗ) и стандартами разработки.

**Руководство оператора для настольного приложения**

Функциональным назначением программы является автоматизация работы стоматологической клиники администратора.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* учет пациентов,
* управление врачами,
* просмотр записей на прием и истории визитов,
* формирование отчетности.
* Профиль пользователя: редактирование данных и смена пароля.

Условия выполнения программы

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средства в части условий их эксплуатации.

Минимальный состав технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор с тактовой частотой, 1 ГГц, не менее;
* оперативную память объемом, 512 Мб, не менее;
* жесткий диск со свободным местом 500 Мб, не менее;
* монитор, с разрешением экрана 1024\*768, не менее;
* компьютерная мышь;
* клавиатура;
* принтер;
* CD – привод.

Минимальный состав программных средств

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7 и выше.

Требования к персоналу (пользователю)

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – пользователь программы, оператор.

Пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы семейства Windows.

Выполнение программы

Для установки программы необходимо открыть файл setup.exe от имени администратора с CD–диска. При установке выбрать установку для всех пользователей. После установки ярлыки приложений отобразиться на рабочем столе. Для запуска клиентской программы необходимо открыть ярлык либо файл Happy\_Tooth\_WPF.exe в папке установки, после пользователю отобразится окно авторизации программы (Рисунок 4.1).

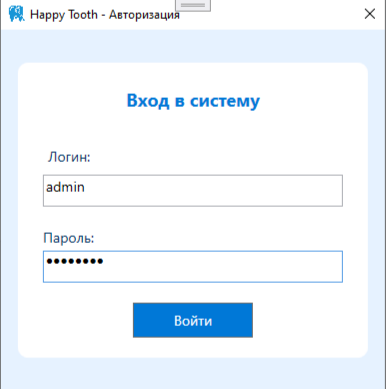
****

Рисунок 4.1 – Окно авторизации

Для того что бы начать использовать приложение вам необходимо авторизоваться. Для этого, необходимо ввести в поля Логин и Пароль соответствующие данные и нажать на кнопку «Войти». После нажатия откроется окно-меню с соответствующим функционалом. (Рисунок 4.2).

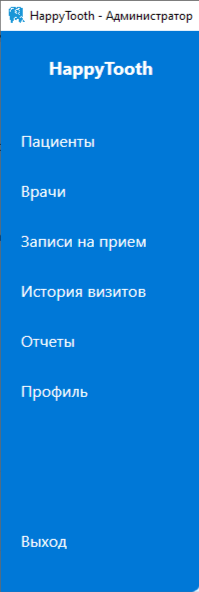


Рисунок 4.2 – Меню приложения

Разберем подробнее весь функционал. Раздел Пациенты позволяет управлять пациентами: просмотр, редактирование, добавление и удаление данных о пациентах клиники (Рисунок 4.3-4.5).

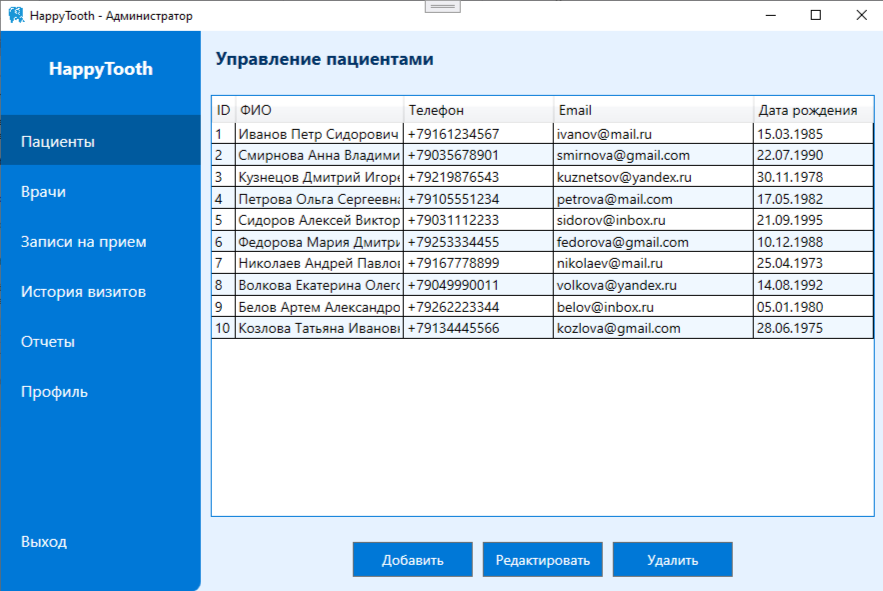


Рисунок 4.3 – Управление пациентами

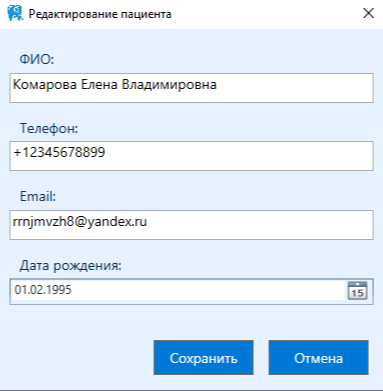


Рисунок 4.4 – Редактирование/добавление пациента

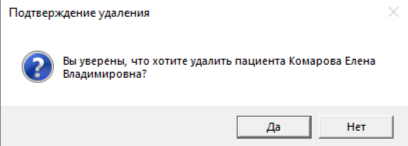


Рисунок 4.5 – Удаление пациента

Раздел Врачи позволяет управлять врачами: просмотр, редактирование, добавление и удаление данных о врачах клиники, настройка их графика работы (Рисунок 4.6-4.9).



Рисунок 4.6 – Управление врачами

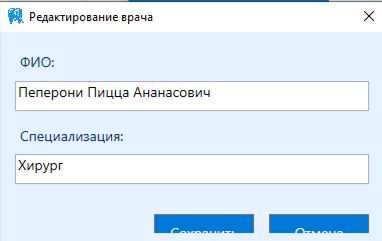


Рисунок 4.7 – Редактирование/добавление врача

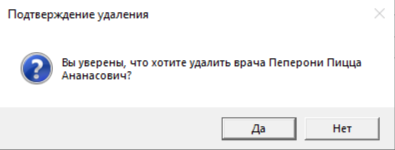


Рисунок 8 – Удаление врача

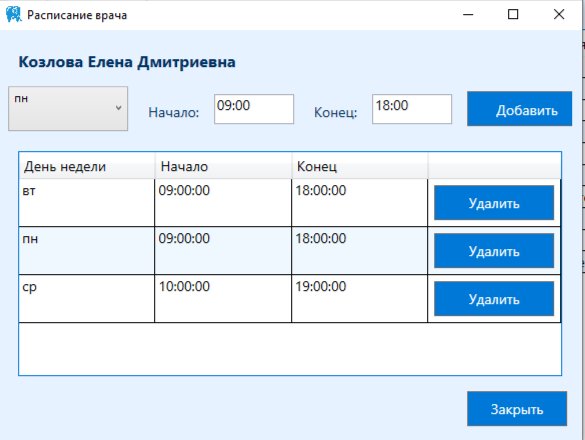


Рисунок 4.9 – Настройка графика работы врача

Раздел Записи на прием позволяет управлять записями пациентов на прием к врачу: просмотр, редактирование и удаление данных о приемах, также присутствует фильтрация данных (Рисунок 4.10-4.11).

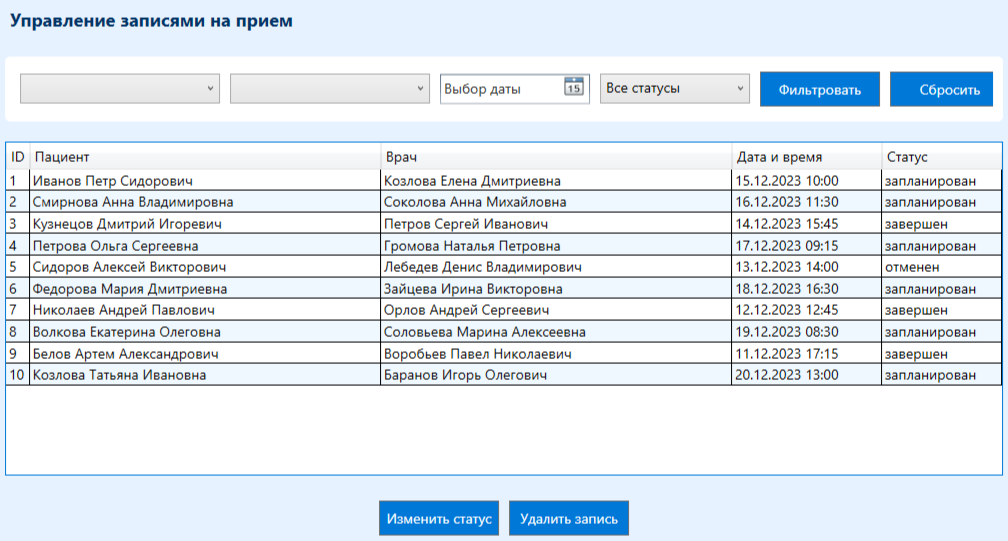


Рисунок 4.10 – Управление записями на прием

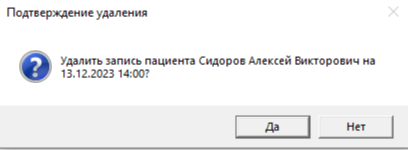


Рисунок 4.11 – Удаление записи

Раздел История визитов позволяет управлять историей визитов пациентов: просмотр и фильтрация данных (Рисунок 4.12).

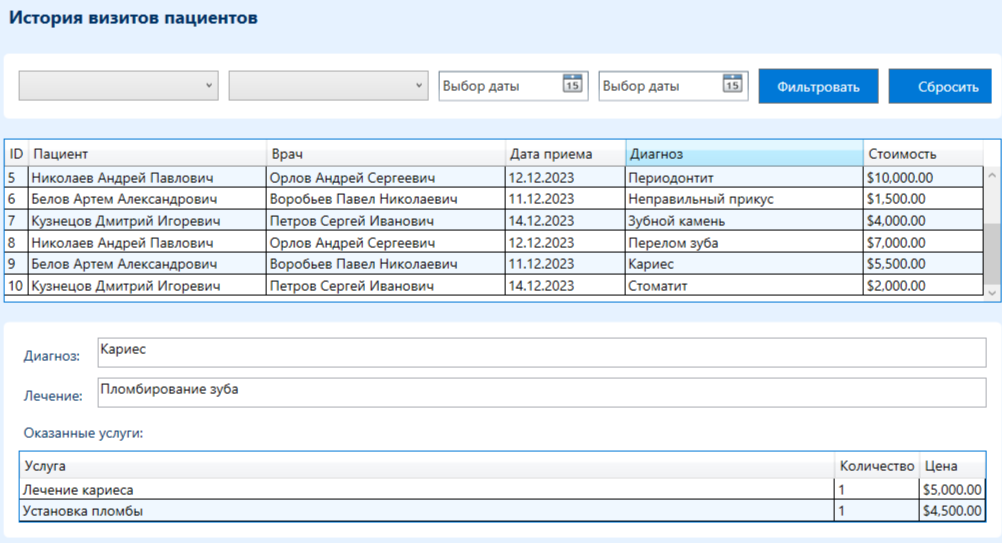


Рисунок 4.12 – Управление историей визитов

Раздел Отчеты позволяет создавать различные отчеты и экспортировать их в Excel (Рисунок 4.13-4.15).

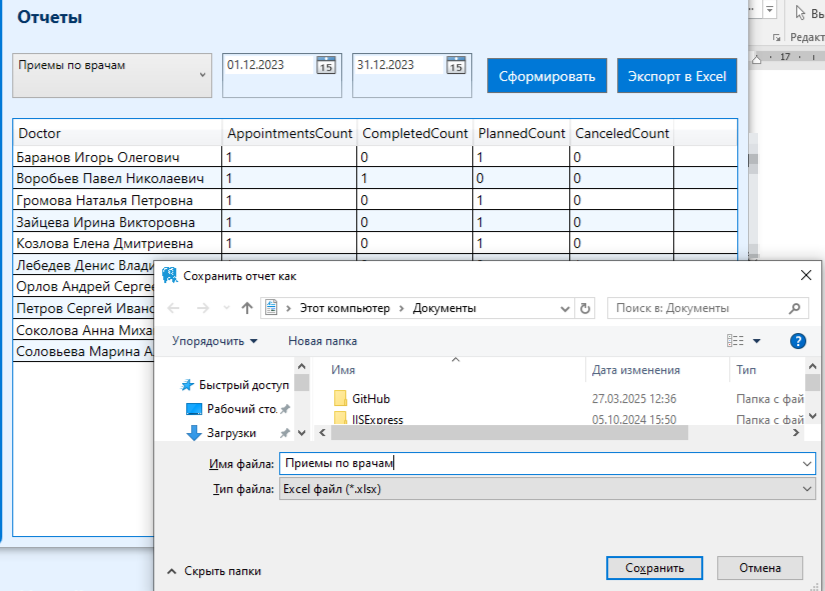


Рисунок 4.13 – Создание отчета Приемы по врачам и его экспорт

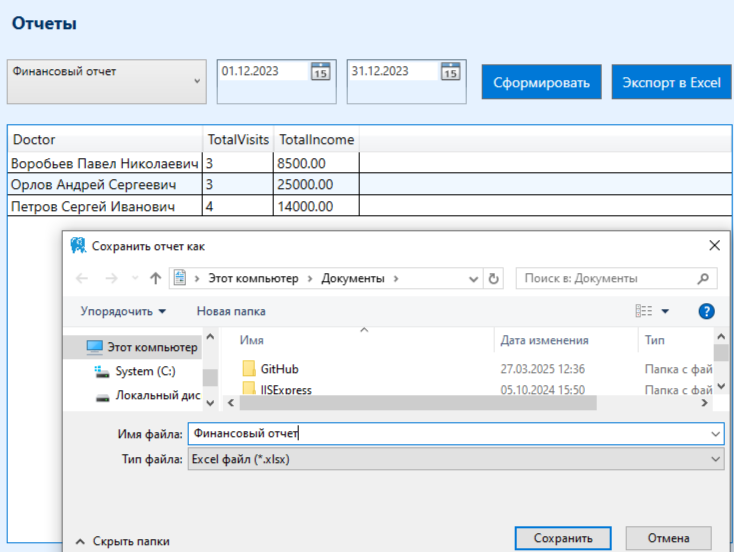


Рисунок 4.14 – Создание финансового отчета и его экспорт

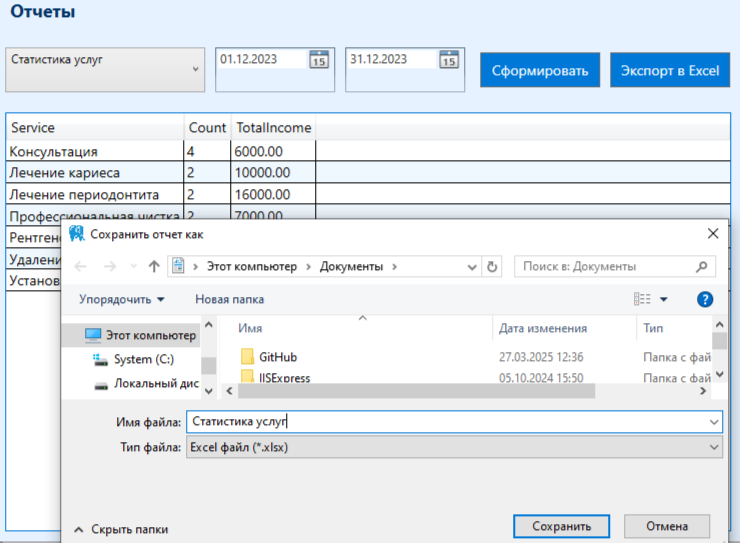


Рисунок 4.15 – Создание отчета Статистика услуг и его экспорт

Раздел Профиль позволяет администратору редактировать свои данные, такие как логин и пароль (Рисунок 16).

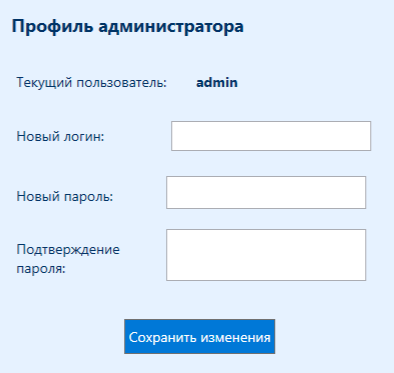


Рисунок 4.16 – Профиль пользователя

Кнопка «Выход» на панели меню позволяет вернуться к окну авторизации.

1. **ВЫПОЛНЕНИЕ ОТЛАДКИ ПРОГРАММНОГО МОДУЛЯ**

На данном этапе была проведена отладка программного модуля, в ходе которой были исправлены ошибки в коде, оптимизирована работа библиотеки, api и проверена корректность функционирования настольного приложения. Основной задачей было обеспечить стабильную работу модуля в соответствии с требованиями технического задания (ТЗ).

Отладка библиотеки классов (Рисунок 5.1):

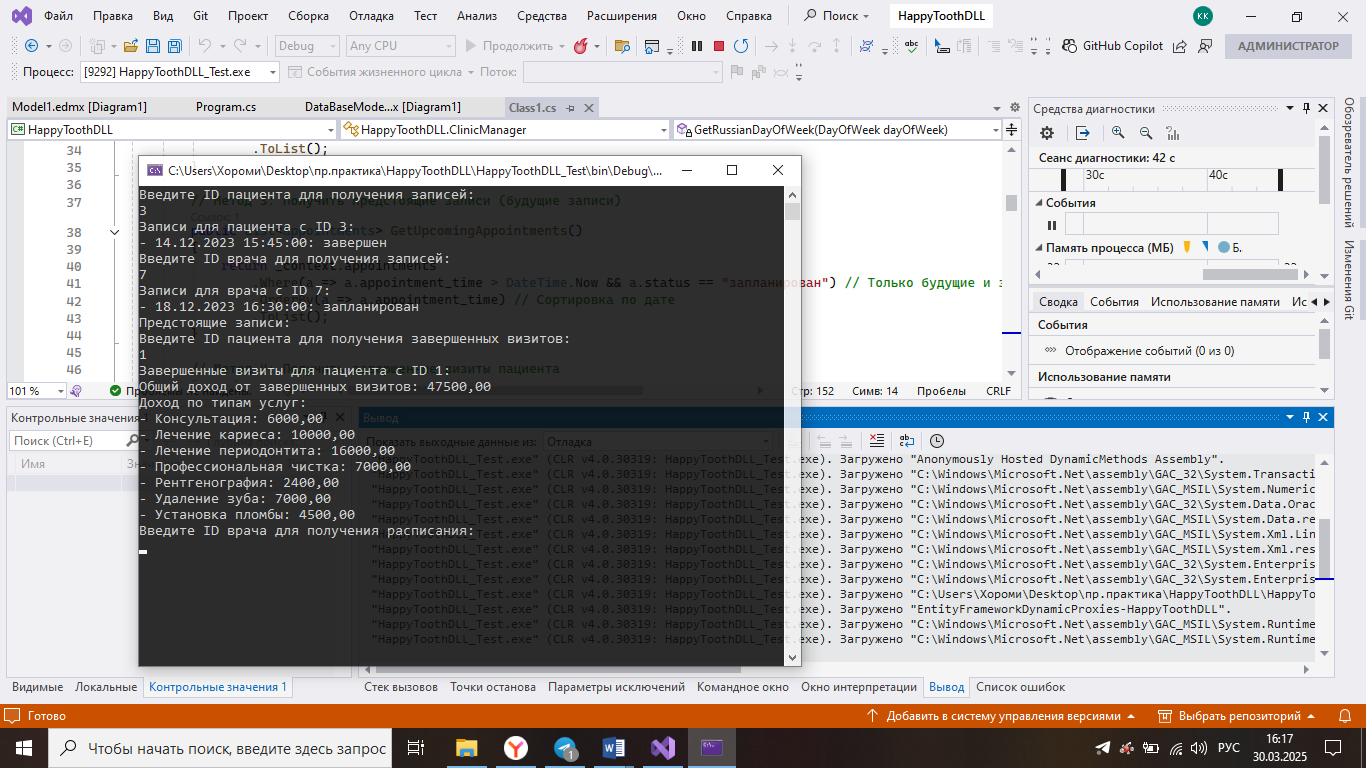


Рисунок 5.1 – Отладка библиотеки

Отладка настольного приложения (Рисунок 5.2):

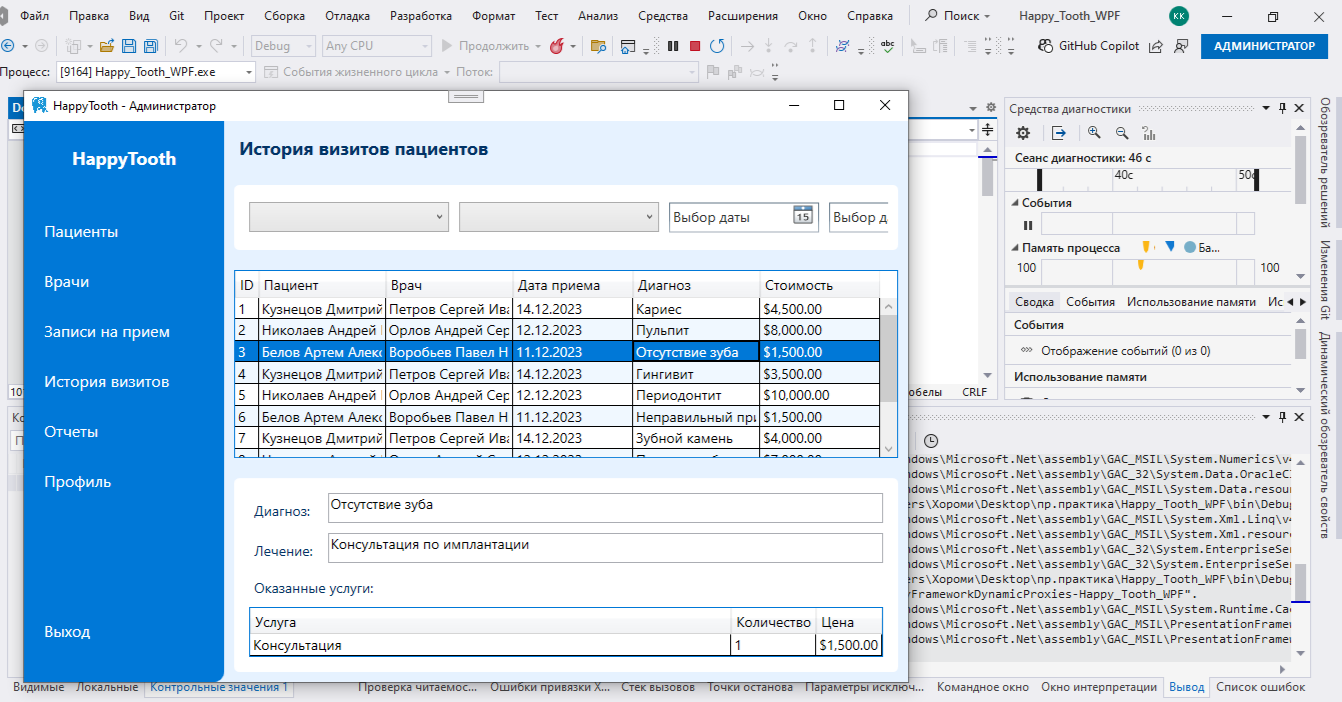


Рисунок 5.2 – Отладка настольного приложения

Отладка api приложения (Рисунок 5.3):

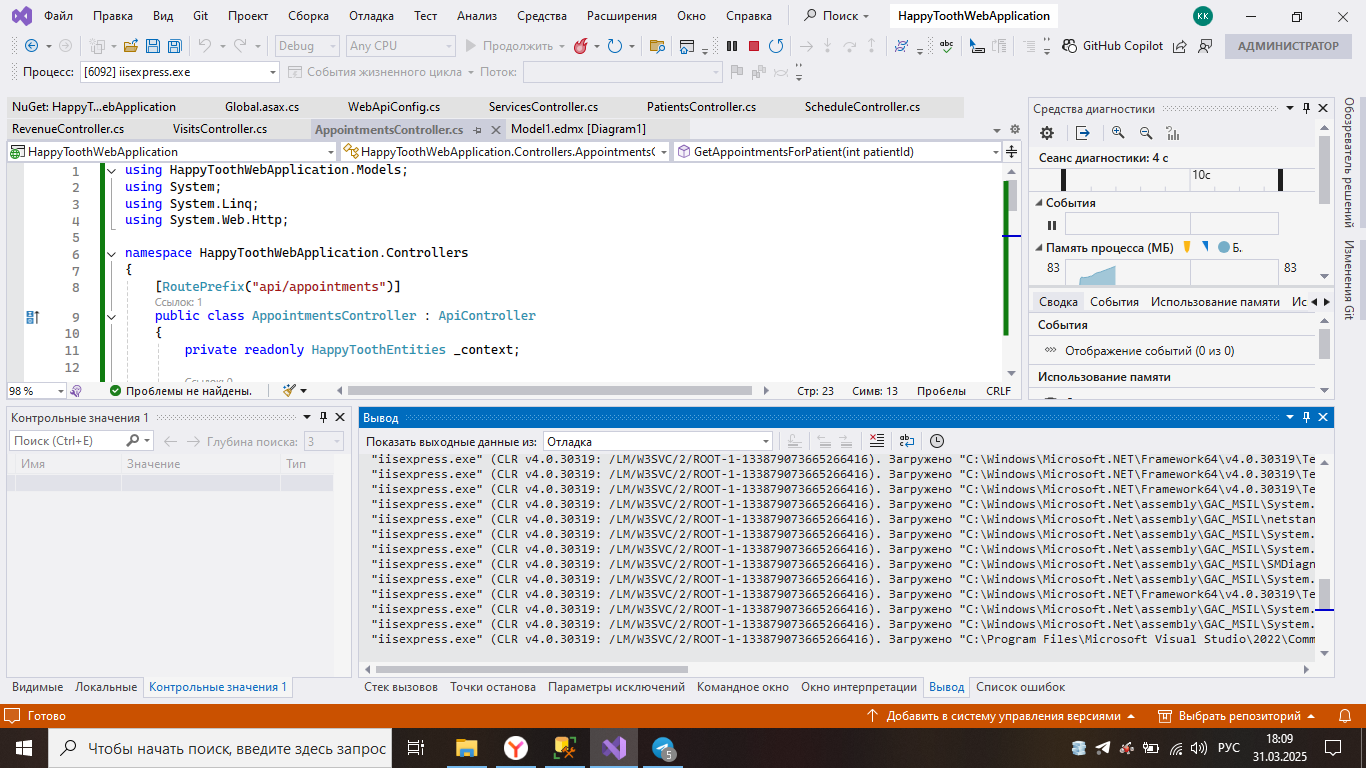


Рисунок 5.3 – Отладка api приложения

1. **ВЫПОЛНЕНИЕ ТЕСТИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ**

В процессе разработки программного обеспечения были протестированы следующие компоненты: библиотека классов, настольное приложение и API. Для каждого из этих компонентов были использованы различные методы тестирования, что обеспечило высокое качество и надежность системы.

**Библиотека классов:**

* Для тестирования библиотеки классов было создано консольное приложение. Это приложение позволяло вызывать методы библиотеки и проверять их функциональность в реальном времени. Консольное приложение обеспечивало удобный интерфейс для тестирования различных сценариев и проверки корректности работы библиотечных функций.

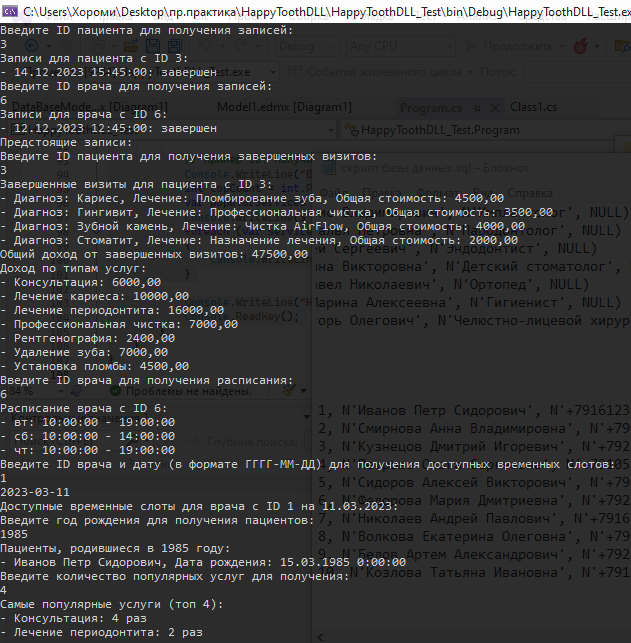


Рисунок 6.1 – Тестирование библиотеки

**API:**

* API было протестировано с использованием Postman. Этот инструмент позволил отправлять HTTP-запросы к API и проверять ответы на них. С помощью Postman были протестированы все конечные точки API, включая проверку корректности возвращаемых данных, статусов ответов и обработки ошибок. Однако, в процессе тестирования было выявлено, что не все методы API работают корректно. Некоторые конечные точки возвращали неожиданные результаты или ошибки, что требует дополнительного внимания и исправления.

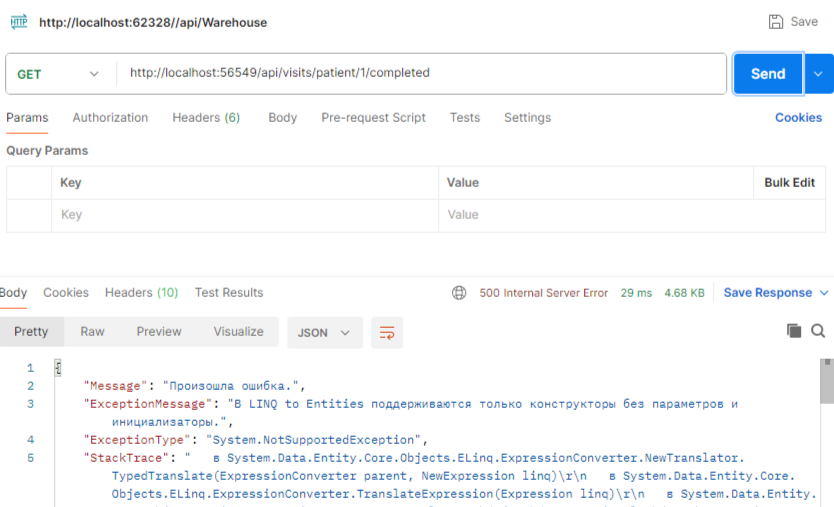
****

Рисунок 6.2 – Тестирование API

**Настольное приложение:**

* Настольное приложение прошло юнит-тестирование. Были написаны тесты для проверки отдельных компонентов и функций приложения. Юнит-тестирование позволило выявить ошибки на ранних этапах разработки и убедиться в том, что каждая часть приложения работает корректно и в соответствии с требованиями.

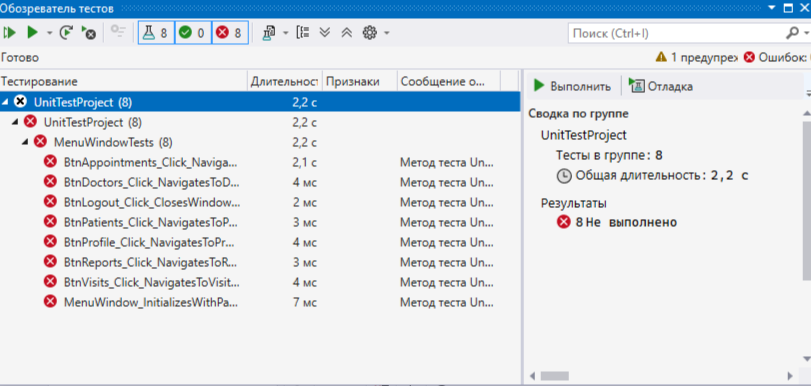


Рисунок 6.3 – Тестирование настольного приложения

Использование различных методов тестирования для каждого компонента системы обеспечило высокое качество программного обеспечения и минимизировало количество ошибок. Тем не менее, в процессе тестирования были выявлены проблемы с некоторыми методами API, которые требуют дополнительного внимания. Все тесты, которые были успешно выполнены, подтверждают готовность системы к развертыванию и использованию, но необходимо устранить выявленные недостатки перед окончательным запуском.

1. **ОФОРМЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ**

Результаты тестирования были систематизированы и оформлены в виде отчетов, которые включают:

**Описание тестовых сценариев**

В данном разделе представлен подробный перечень всех проверенных функций и сценариев использования программного модуля. Например, были проверены сценарии авторизации, работы с товарами, формирования отчетов и другие ключевые функции, описанные в техническом задании.

**Отчеты были оформлены в виде онлайн документации:**

Все результаты тестирования, включая отчеты, были загружены в репозиторий на GitHub. Это обеспечивает удобное хранение, версионность и доступ к документам для всех участников проекта. Каждый отчет был размещен в папке репозитория, что позволяет легко находить нужные материалы и отслеживать изменения.

Таким образом, результаты тестирования были тщательно проанализированы и оформлены в удобном для использования виде, что позволяет эффективно работать с ними на всех этапах жизненного цикла проекта.

1. **РЕФАКТОРИНГ И ОПТИМИЗАЦИИ ПРОГРАММНОГО КОДА**

В процессе разработки программного обеспечения важным этапом является рефакторинг и оптимизация кода. Эти действия направлены на улучшение структуры, читаемости и надежности кода без изменения его функциональности. В рамках данного проекта были проведены следующие мероприятия по рефакторингу и оптимизации:

**Использование блоков try-catch:**

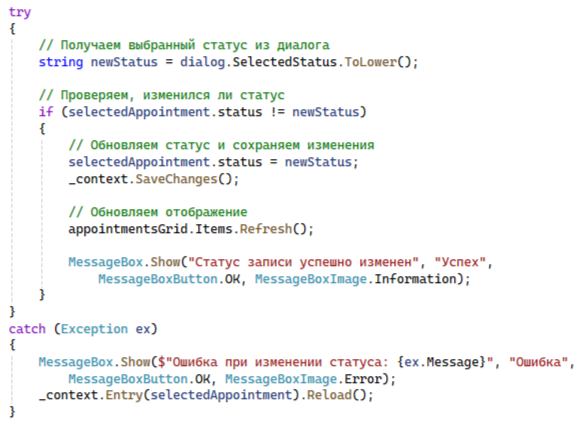
****

Рисунок 8.1 – Рефакторинг и оптимизация кода (блоки try-catch)

**Проверка на пустые значения:**

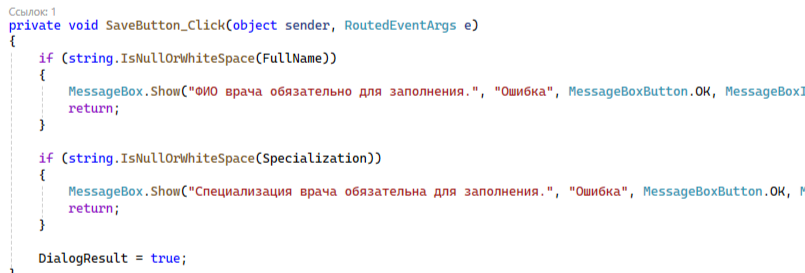
****

Рисунок 8.2 – Рефакторинг и оптимизация кода (пустые значения)

**Именование переменных по стилю CamelCase:**

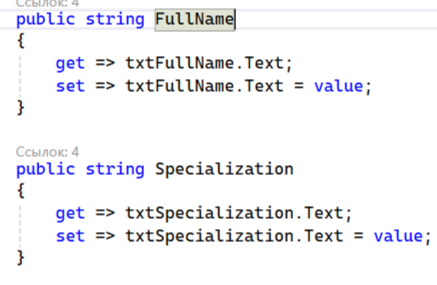
****

Рисунок 8.3 – Рефакторинг и оптимизация кода (CamelCase)

**Структуризация кода с использованием циклов и методов:**

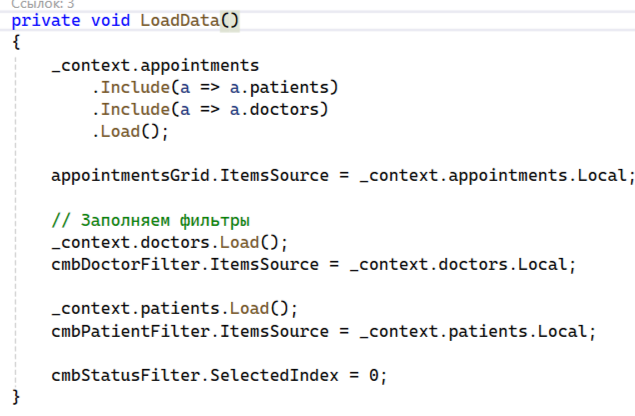
****

Рисунок 8.4 – Рефакторинг и оптимизация кода (циклы и методы)

**Добавление комментариев:**

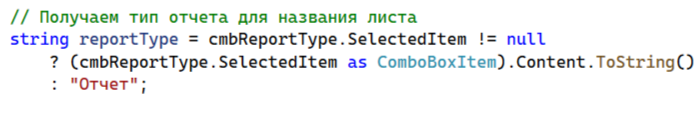
****

Рисунок 8.4 – Рефакторинг и оптимизация кода (комментирование)

1. **РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ МОБИЛЬНОГО УСТРОЙСТВА**

На данном этапе было разработано мобильное приложение, предназначенное для пациентов стоматологической клиники. Приложение предоставляет удобный интерфейс для управления записями на прием, просмотра истории посещений и работы с персональными данными.

**Технологии разработки мобильного приложения:**

Платформа:

* Visual Studio 2022 - Xamarin.Forms (Android)

База данных:

* SQLite (локальная)

Библиотеки для работы с базой данных:

* SQLite-net-pcl,
* SQLiteNetExtensions

**Особенности реализации:**

Пользовательский интерфейс:

* Интуитивно понятный и отзывчивый интерфейс
* Адаптивный дизайн для различных размеров экранов
* Удобная навигация между разделами

Работа с данными:

* Полноценная система CRUD-операций
* Сложные запросы для получения связанных данных
* Автоматическое заполнение тестовыми данными при первом запуске

Безопасность:

* Хранение учетных данных в защищенном виде
* Разграничение прав доступа

Производительность:

* Оптимизированные запросы к базе данных
* Минимальное потребление ресурсов устройства

**Скриншоты выполнения мобильного приложения:**

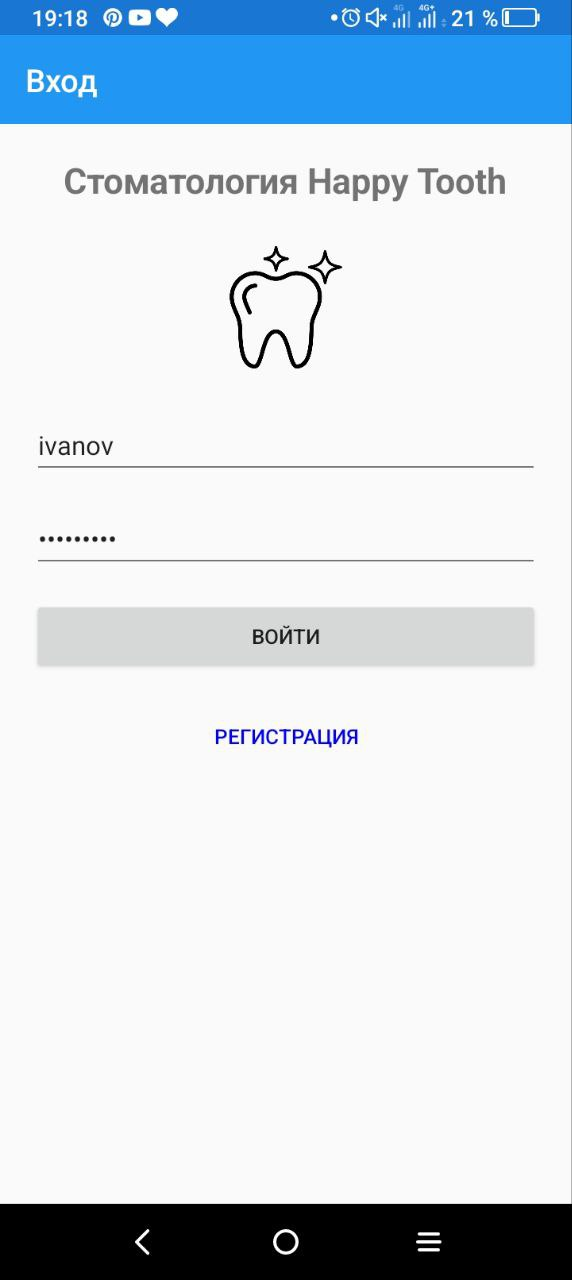


Рисунок 9.1 – Окно авторизации

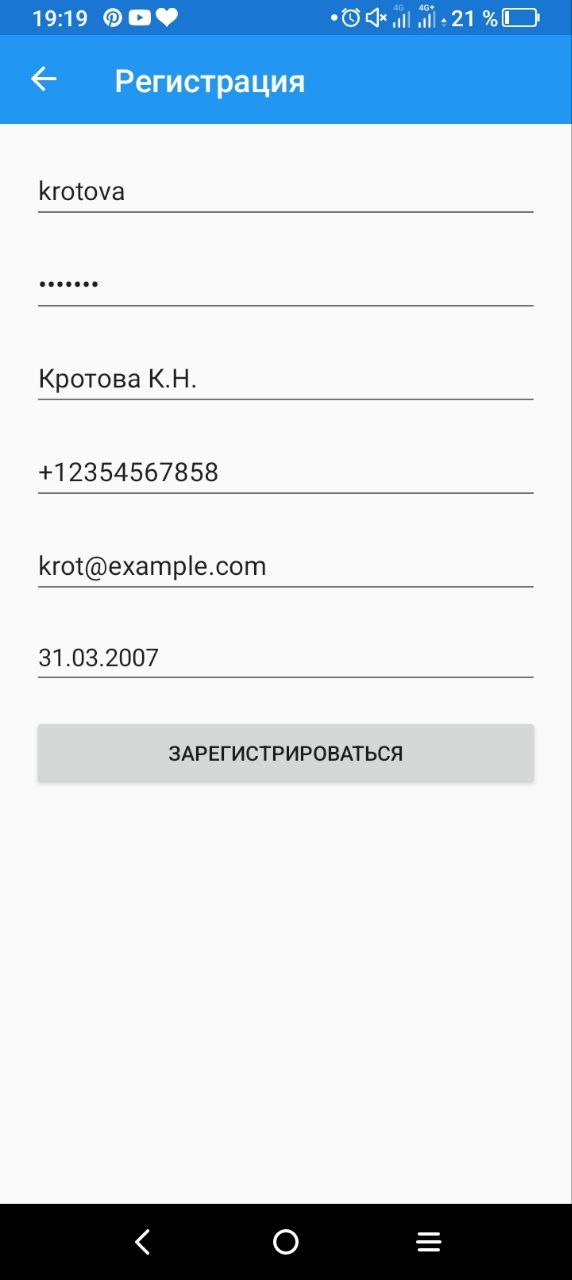


Рисунок 9.2 – Окно регистрации нового пациента

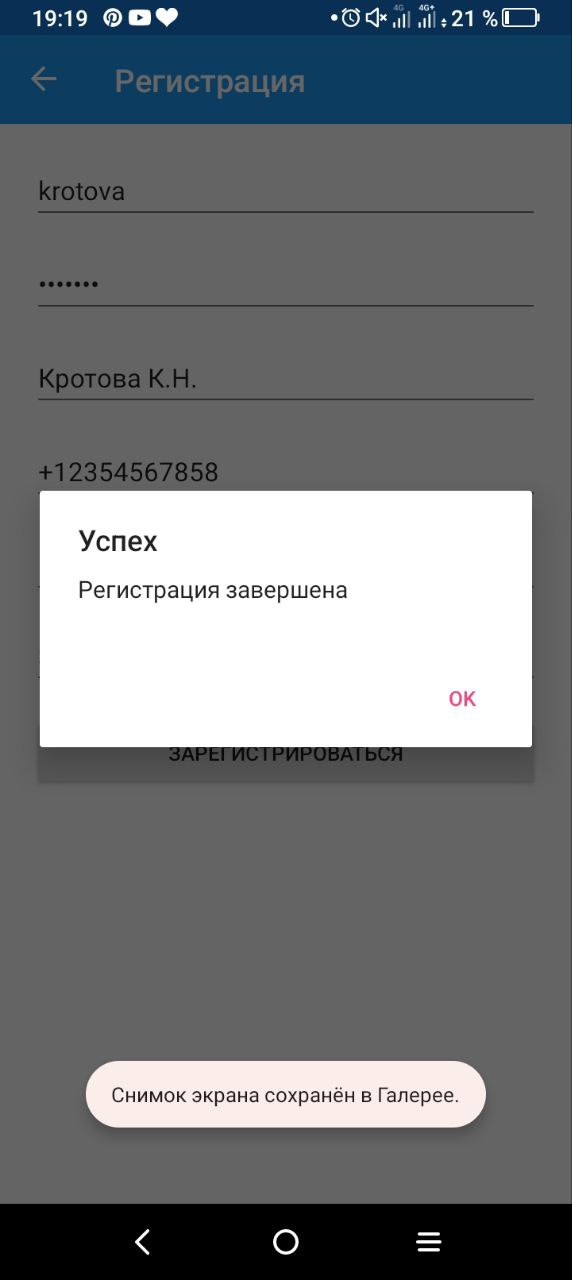


Рисунок 9.3 – Уведомление об успешной регистрации

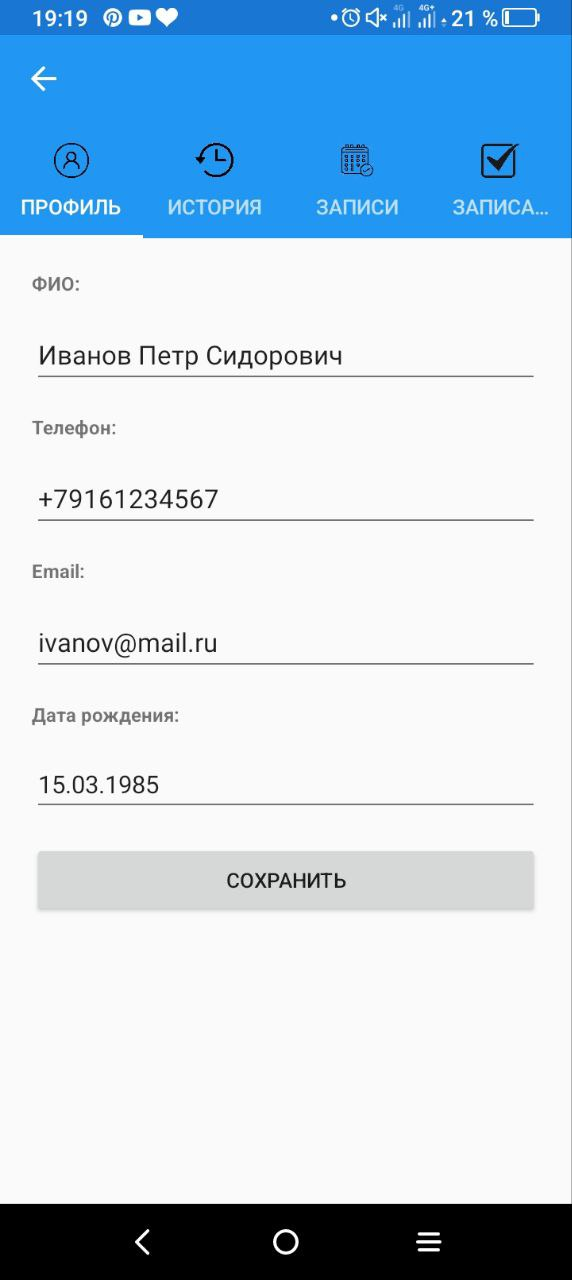


Рисунок 9.4 – Личный профиль пациента

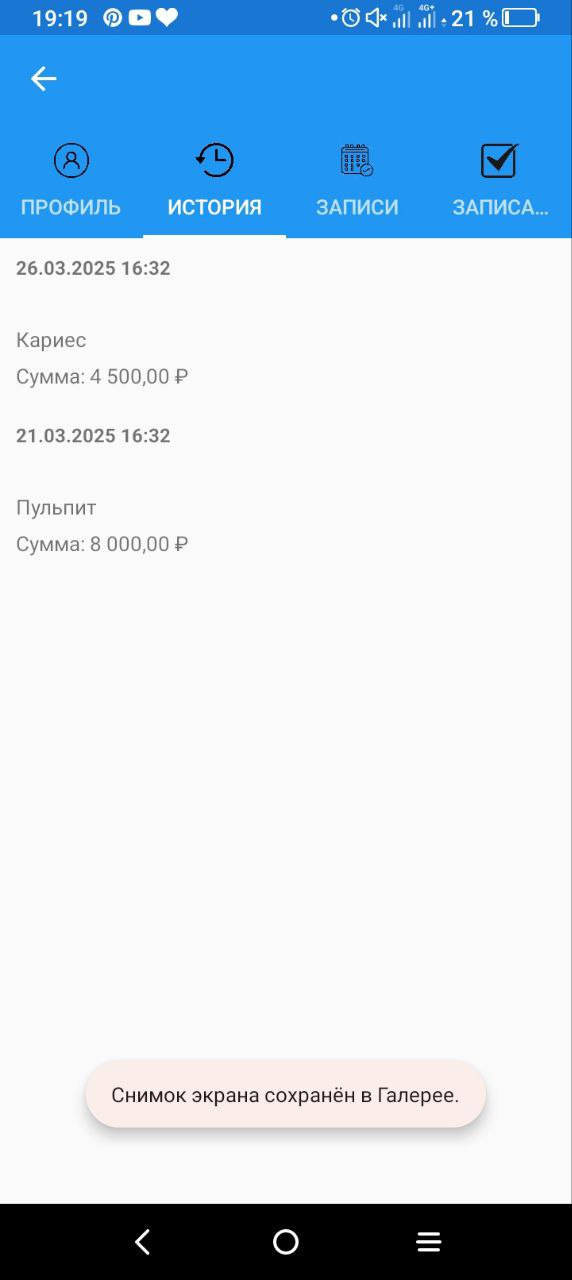


Рисунок 9.5 – История посещений пациента

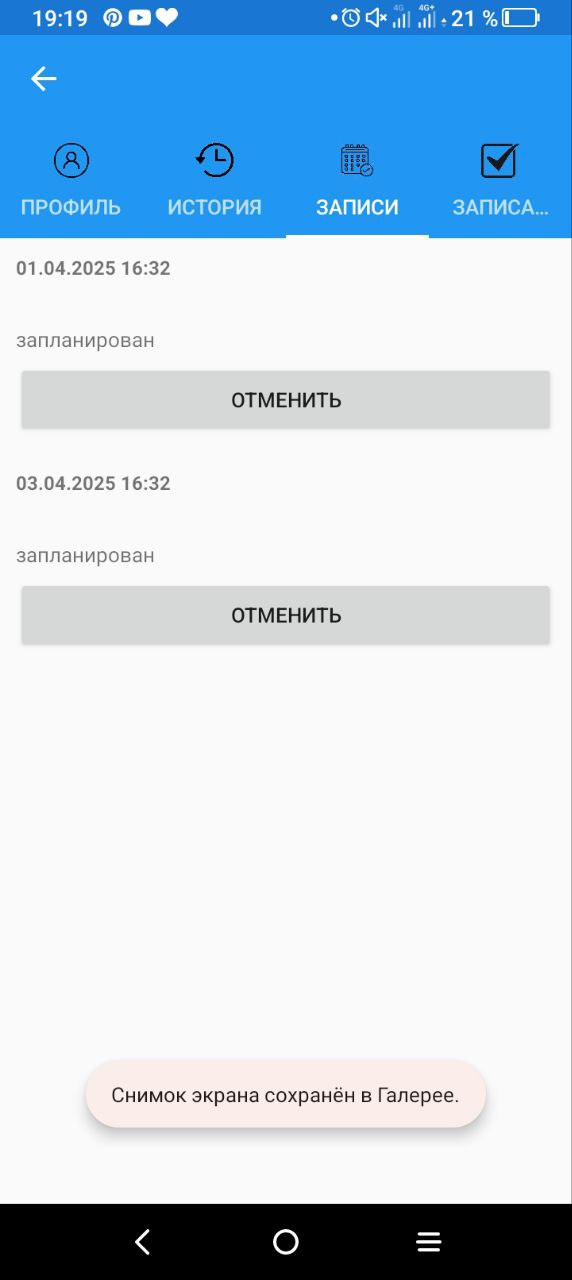


Рисунок 9.6 – Запланированные посещения пациента

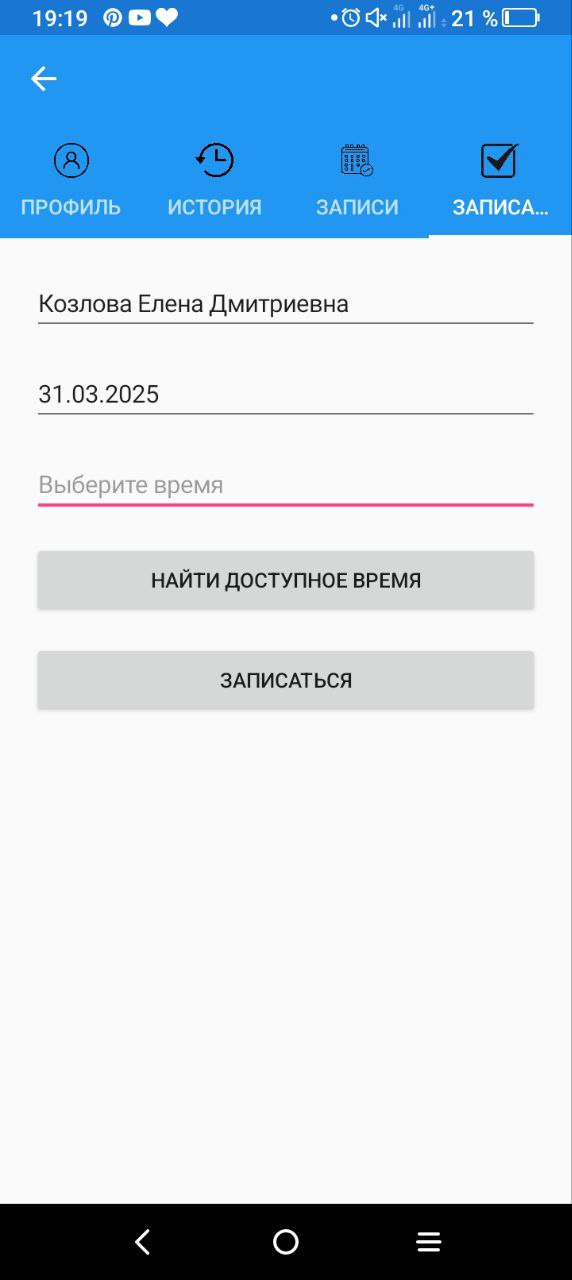


Рисунок 9.7 – Окно записи на прием

**Тестирование мобильного приложения**

В процессе тестирования мобильного приложения были разработаны и выполнены тест-кейсы, охватывающие основные функциональные сценарии. Все тест-кейсы загружены в репозиторий GitHub для обеспечения прозрачности процесса тестирования и дальнейшего аудита.

В ходе проверки выявлены следующие критические ошибки:

* Отмена записи не работает
* Функция записи к врачу не работает

**Тестирование проводилось на:**

* Эмуляторах Android (версии 10, 11, 12)
* Физических устройствах (TCL 40R 5G)

1. **ЗАКЛЮЧЕНИЕ.**

Производственная практика ПМ01. «Разработка модулей программного обеспечения для компьютерных систем» проходила с 20 марта по 21 апреля 2025 г. в организации ООО «Электрон-Софт».

В ходе практики я получила ценный опыт в области разработки программных модулей, который позволил мне закрепить теоретические знания и приобрести практические навыки, необходимые для работы в данной сфере.

Производственная практика позволила закрепить полученные теоретические знания на учебной практике и приобрести необходимые навыки в области осуществлении разработки программных модулей.

Полученные знания и опыт будут использованы в дальнейшей профессиональной деятельности.

1. **ПРИЛОЖЕНИЯ К ОТЧЕТУ: ССЫЛКА НА РЕПОЗИТОРИЙ ОТЧЕТ В ЭЛЕКТРОННОМ ВИДЕ, ПРЕЗЕНТАЦИЯ ДЛЯ ВЫСТУПЛЕНИЯ, ПРОЕКТ ПРОГРАММНОГО МОДУЛЯ, ПРОЕКТ МОБИЛЬНОГО ПРИЛОЖЕНИЯ И ДР. МАТЕРИАЛЫ.**

Все материалы, программные модули и документация, созданные в рамках проекта, были систематизированы и загружены в репозиторий на GitHub.

Ссылка на репозиторий:

