ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА ПОДСИСТЕМЫ

«РЕГИСТРАТУРА КЛИНИКИ»

Л109. 25КП01. 029 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-21		08.12.2025	Е. В. Чугин
	(Группа)	(Подпись)	(Дата)	(И.О. Фамилия)
Преподаватель			09.12.2025	Ю.С. Маломан
		(Подпись)	(Дата)	(И.О. Фамилия)

Архангельск 2025

СОДЕРЖАНИЕ

Перечень сокращений и обозначений 3
Введение 4
1 Сбор и анализ требований 6
1.1 Назначение и область применения 6
1.2 Постановка задачи 6
1.3 Выбор состава программных и технических средств 9
2 Проектирование программного обеспечения 10
2.1 Проектирование интерфейса пользователя 10
2.2 Разработка архитектуры программного обеспечения 11
2.3 Проектирование базы данных 12
3 Разработка и интеграция модулей программного обеспечения 14
3.1 Разработка программных модулей 14
3.2 Реализация интерфейса пользователя 16
3.3 Разграничение прав доступа пользователей 18
3.4 Экспорт и импорт данных 20
4 Тестирование и отладка программного обеспечения 23
4.1 Структурное тестирование 23
4.2 Функциональное тестирование 25
5 Инструкция по эксплуатации программного обеспечения 27
5.1 Установка программного обеспечения 27
5.2 Инструкция по работе 27
Заключение 32
Список использованных источников 33

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

БД – база данных

ИС – информационная система

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

ER-модель – модель «сущность-связь»

IDE – интегрированная среда разработки

SQL – язык структурированных запросов

WPF – Windows Presentation Foundation (платформа для создания настольных приложений)

XAML – язык разметки WPF

ВВЕДЕНИЕ

Актуальность разрабатываемого проекта заключается в автоматизации процессов учета записей на приём, управления расписанием врачей и ведение электронных медицинских карт пациентов в медицинских учреждениях.

В современных условиях для эффективного управления медицинским учреждением важны оперативность и точность данных. Медицинские учреждения сталкиваются с проблемами, связанными с учетом записей на приём, управлением расписанием врачей, ведением медицинских карт пациентов и обеспечением безопасности персональных данных.

Разработка программного продукта «Регистратуры клиники» позволит значительно упростить процесс организации работы медицинского учреждения, улучшить взаимодействие с пациентами и врачами, повысить общую эффективность работы регистратуры и врачебного персонала [5].

Целью курсового проекта является разработка программного продукта «Регистратуры клиники», обеспечивающего возможность комплексного управления записями на приём, расписанием врачей, ведение медицинских карт пациентов и разграничение прав доступа для различных пользователей.

Для достижения поставленной цели требуется решить следующие задачи:

- провести сбор и анализ требований целевой аудитории (врачи, пациенты, администраторы, регистраторы);
- проанализировать информационные источники по предметной области;
- изучить существующие решения в области автоматизации работы медицинских учреждений;
- спроектировать архитектуру приложения;
- выбрать состав программных и технических средств для реализации проекта;

- спроектировать БД;
- создать БД в выбранной СУБД;
- реализовать разграничение прав доступа пользователей (Администратор, Регистратор, Врач, Пациент);
- обеспечить защиту персональных данных;
- разработать пользовательский интерфейс;
- реализовать функциональность записи на приём к врачу;
- реализовать функциональность управления расписанием врачей;
- реализовать функциональность ведения медицинских карт;
- реализовать функциональность импорта и экспорта данных;
- выполнить структурное тестирование ПО;
- выполнить функциональное тестирование ПО;
- разработать программную документацию.

В результате выполнения поставленных задач будет создан программный продукт «Регистратуры клиники», который значительно упростит процесс управления записями на приём, повысит эффективность работы медицинского персонала и улучшит качество обслуживания пациентов.

1 Сбор и анализ требований

1.1 Назначение и область применения

Основное назначение подсистемы: автоматизация процесса записи пациентов на приём к врачу, управление расписанием медицинского персонала и ведение электронных медицинских карт пациентов, что позволит снизить временные затраты на обработку запросов пациентов, уменьшить очереди в регистратуре и минимизировать человеческие ошибки при бронировании времени приёма, уменьшить затраты времени врачей на бумажное ведение медицинской карты.

ПО предназначено для использования в медицинских учреждениях различного типа: поликлиниках, больницах, частных медицинских центрах и специализированных клиниках. Гибкая архитектура системы позволяет настраивать её под различные типы медицинских учреждений, обеспечивая возможность адаптации функционала в соответствии с индивидуальными требованиями и рабочими процессами учреждения.

Применение системы позволит существенно повысить эффективность работы регистратуры, сократить время ожидания пациентов, улучшить качество обслуживания и обеспечить более точный учет медицинских данных. Электронное ведение медицинских карт обеспечивает быстрый доступ к истории болезни пациента, что критически важно для принятия обоснованных медицинских решений.

1.2 Постановка задачи

Необходимо разработать приложение, в котором буду реализованы следующие функциональные возможности:

- просмотр и редактирование расписание врачей;

- просмотр и редактирование данных о приёмах пациентов;
- просмотр и редактирование данных врачей и пациентов;
- формирование отчётов по работе медицинского учреждения;
- формирование отчётов о загрузке специалистов;
- система напоминаний о визитах.

В подсистеме должна быть обязательная авторизация и следующие пользователи:

- Пациент – авторизованный пользователь, который имеет доступ к функциям записи к врачу, переносу и отмене записи к врачу, а также просмотру медицинских записей о проведённом приёме;
- Врач имеет доступ к просмотру записанных пациентов на приём, и к просмотру и изменению информации в медицинской карточке пациента;
- Регистраторы в свою очередь имеют доступ к полному управлению записями, к расписанию врачей и данных пациентов, к формированию отчётов о нагрузке учреждения (нужно для более опционального расписание врачей и работы в мед. учреждении);
- Администратор имеет доступ ко всем функциям приложения, а также к изменению информации о пользователях.

Система обеспечивает разграничение прав доступа на основе ролей пользователей. Каждая роль имеет свой набор разрешений: пациенты могут управлять только своими записями, врачи работают со своими расписаниями и медицинскими картами пациентов, регистраторы имеют расширенные права на управление записями и расписанием, администраторы обладают полным доступом ко всем функциям системы, включая управление пользователями.

На рисунке 1 изображена диаграмма вариантов использование данного приложения.

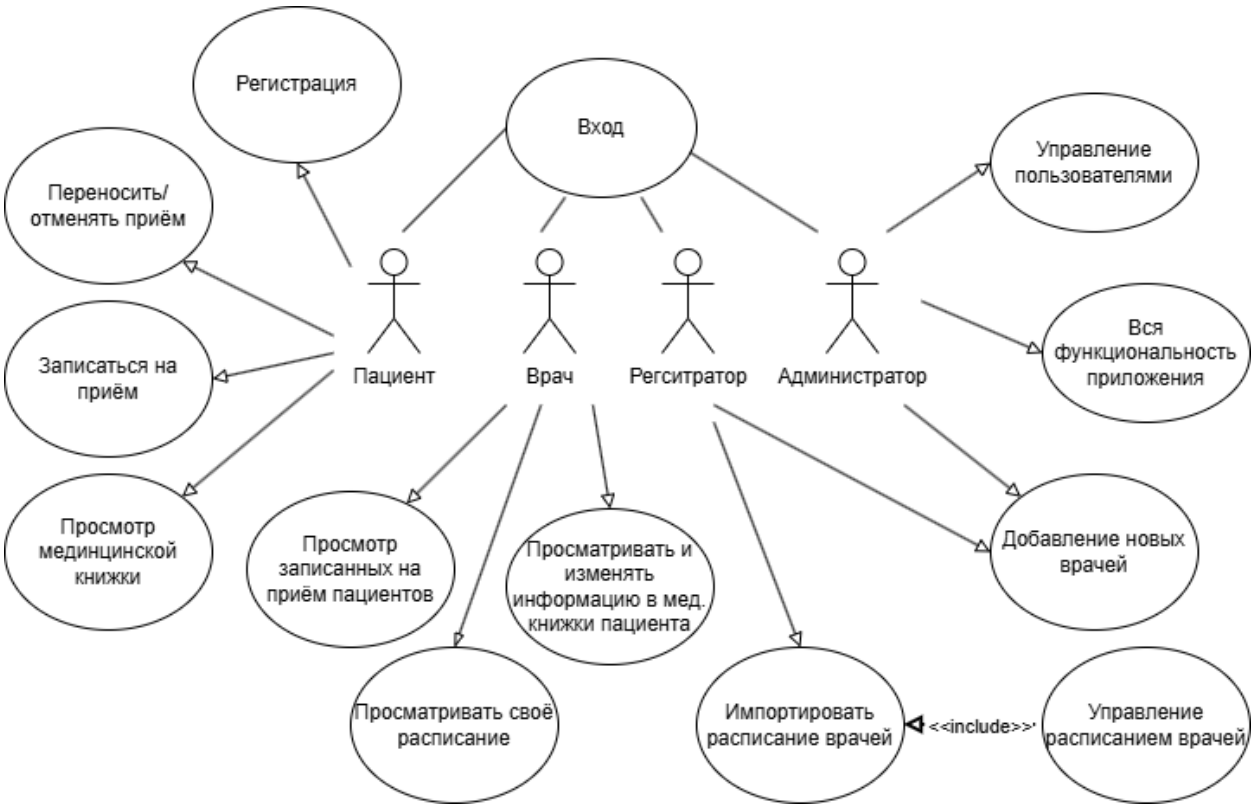

Рисунок 1 - Диаграмма вариантов использования

Работа с системой осуществляется по следующему алгоритму:

- пользователь входит в систему;
- система показывает доступных врачей и свободные временные окна;
- пациент выбирает специалиста и время;
- регистратор проверяет новые записи (при необходимости может корректировать расписание и импортировать данные пациентов);
- врач просматривает информацию о записанных пациентах и изучает их медицинские карты перед приёмом;
- во время приёма врач заполняет электронную медицинскую карту пациента, указывает диагноз и назначение;
- система автоматически формирует статистику о загруженности врачей;
- администратор анализирует отчёты о работе системы, настраивает параметры интеграции и оптимизирует алгоритмы распределение пациентов.

1.3 Выбор состава программных и технических средств

Работа с оконным приложением будет осуществляться на ПК и ноутбуках с ОС Windows (Windows 10 версии 1809 и новее, Windows 11).

В качестве СУБД выбрана Microsoft SQL Server, данная СУБД имеет прямую интеграцию с Visual Studio, удобна в использовании, обладает высокой производительностью, что позволяет эффективно обрабатывать данные о приёмах и пациентах в реальном времени [3].

Приложение будет разработано на C#, так как с помощью C#, можно эффективно создавать современные настольные приложения с использованием технологии WPF для пользовательского интерфейса и Entity Framework Core – библиотеки для объектно-реляционного отображения (ORM), которая обеспечивает эффективную работу с БД.

Для разработки оконного приложения будет использоваться IDE Visual Studio 2022, так как эта среда предлагает удобные инструменты для работы с C#, включая инструменты для работы с Git и средства отладки.

Для функционирования системы на стороне сервера необходимы следующие программные и технические средства:

- ОС Windows 10 и выше;
- процессор частотой 2 ГГц;
- свободная оперативная память 4 ГБ;
- свободное место на диске не менее 500 МБ;
- ПО для конфигурирования, управления и администрирования сервера БД: Sql Server Management Studio.

Для функционирования системы на стороне сервера необходимы следующие программные и технические средства:

- ОС Windows 10 и выше;
- процессор частотой 2 ГГц;
- свободная оперативная память 4 ГБ;
- свободное место на диске не менее 500 МБ.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

При создании настольного приложения «Регистратуры клиники» была выполнена предварительная проработка интерфейса на основе черновых макетов, подготовленных в инструменте figma. Эти схемы позволили сформировать представление о структуре клиентского приложения, расположении элементов управления и последовательности пользовательских действий [4].

Wireframe-макеты интерфейсов показаны на рисунке 2.

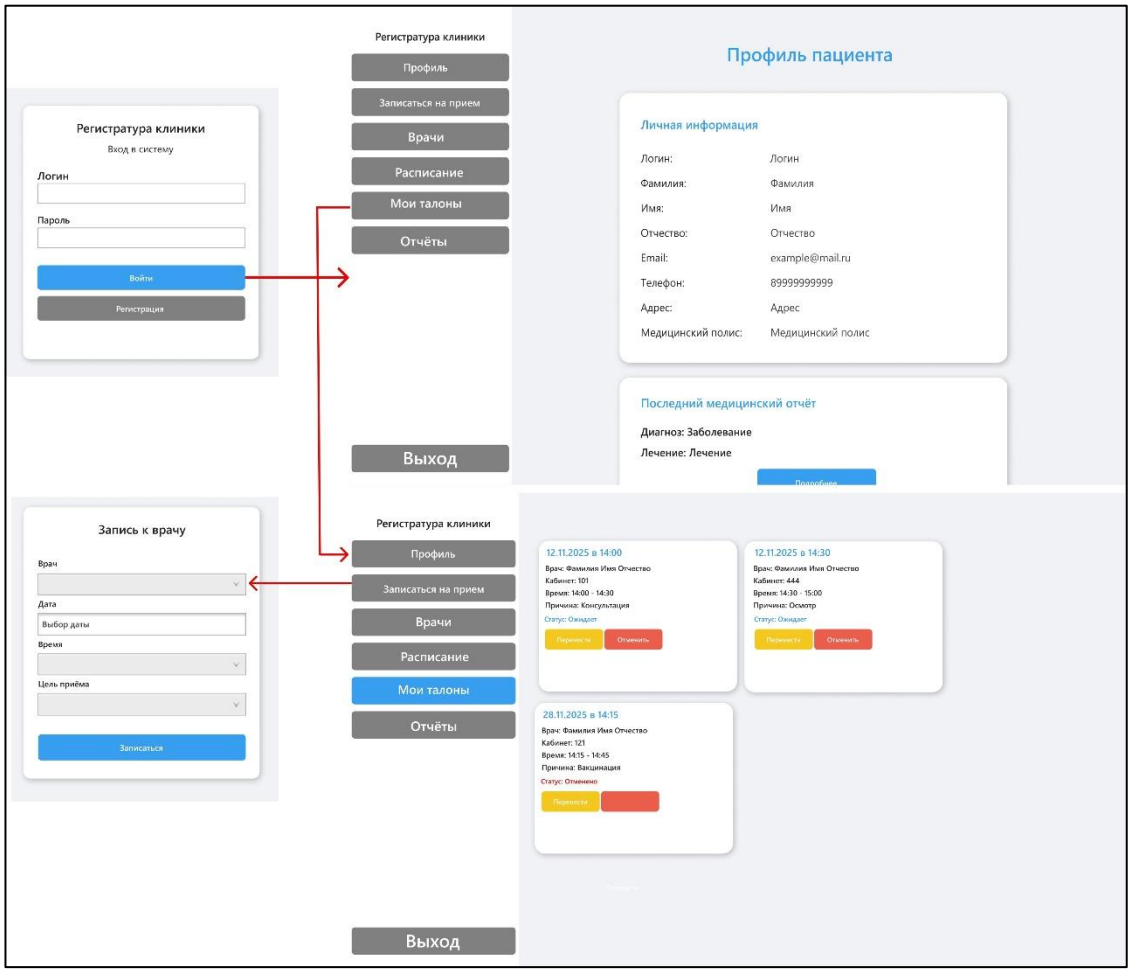

Рисунок 2 – Прототипы пользовательских окон подсистемы

Для обеспечения визуальной целостности приложения был подобран единый набор цветов и типографики, используемы во всех формах WPF.

Цветовая схема включает:

- базовый акцентный цвет: #2962FF;
- цвет основного фона: #F5F5F5;
- основной цвет текста: #212121;
- вспомогательный цвет текста: #424242.

Шрифтовое оформление:

- основной шрифт: Segoue UI;
- вспомогательный шрифт: Segoue UI Semibold.

Такой стиль помогает сформировать современный, лёгкий и понятный интерфейс, соответствующий требованиям обычного приложения для данных организаций [4].

2.2 Разработка архитектуры программного обеспечения

Архитектура системы построена на трёхслойной модели и включает несколько взаимосвязанных элементов: клиентское WPF-приложение, слой данных (DataLayer) и БД. Клиентское приложение напрямую взаимодействует с БД через слой данных, используя Entity Framework Core, что обеспечивает эффективный и безопасный обмен данными [2].

Основные компоненты архитектуры:

- WPF-клиент;
- Слой данных (DataLayer);
- СУБД SSMS;
- слой авторизации;
- сервис взаимодействия.

Диаграмма развёртывания элементов программного комплекса представлена на рисунке 3.

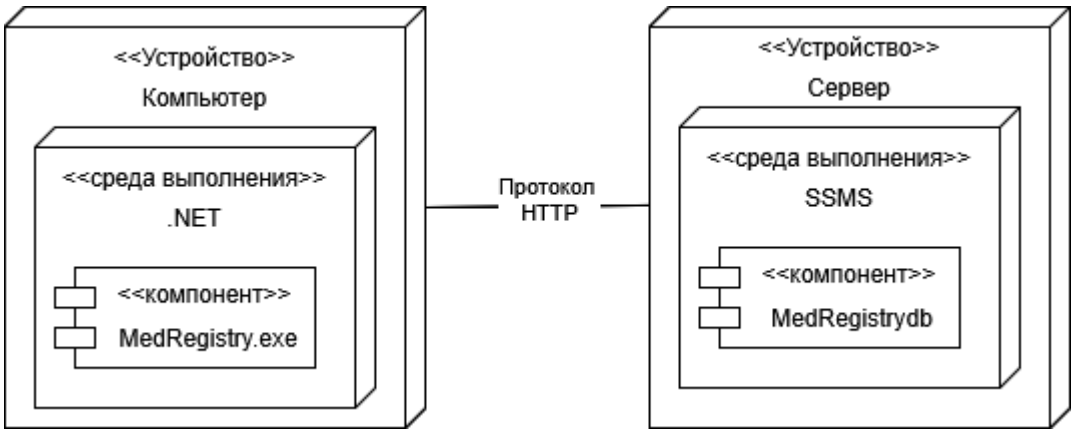

Рисунок 3 – Схема развёртывания компонент системы.

2.3 Проектирование базы данных

Для хранения данных, связанных с функционалом регистратуры клиники, была спроектирована реляционная БД. В БД представлены сущности для управления пользователями, ролями, их специальностями, расписанием, записями пациентов и медицинской документацией.

ER-диаграмма предметной области отражает связи между таблицами и демонстрирует их структуру. Физическая модель БД разработана на основе SSMS и включает следующие таблицы:

- Users – учётные записи пользователей системы;
- Roles – роли и права доступа;
- Doctors – информация о врачах;
- Specialization – медицинские специальности;
- Schedules – рабочее время врачей;
- Appointments – записи пациентов на приём;
- Patients – карточки пациентов;
- MedicalRecords – записи в медицинской карточке.

ER-диаграмма представлена на рисунке 4.

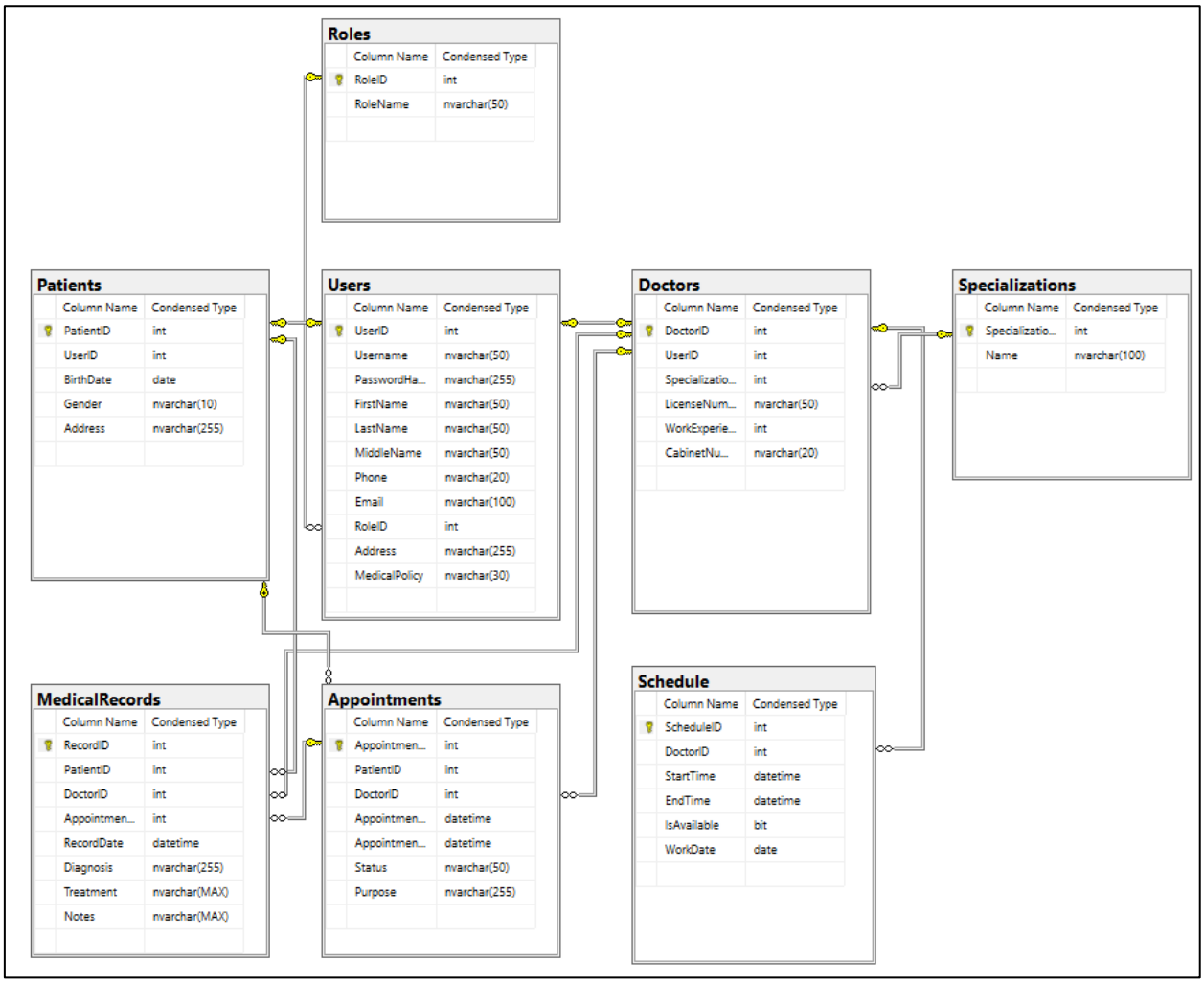

Рисунок 4 – Физическая структура базы данных

Такая модель обеспечивает целостность данных, минимизацию дублирования и логичную структуру хранения, подходящую для задач регистратуры и медицинской документации.

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

Для разработки программного продукта «Регистратуры клиники» используется среда разработки Microsoft Visual Studio 2022 с поддержкой .NET 8.0. Приложение построено на базе платформы WPF (Windows Presentation Foundation) для создания настольного приложения с графическим интерфейсом пользователя [2]. Результатом компиляции является исполняемый файл (exe), который может быть запущен на компьютерах с установленным .NET 8.0 Runtime.

Проект состоит из трёх основных слоев: DataLayer (слой данных), MedRegistry (клиентское приложение) и БД. Основной функционал реализован в клиентском приложении MedRegistry, которое напрямую взаимодействует с БД через Entity Framework Core.

Для работы с базой данных используется библиотека Entity Framework Core, которая обеспечивает объектно-реляционное отображение (ORM). В проекте DataLayer определены модели данных (User, Doctor, Patient, Appointment, Schedule, MedicalRecord и др.), которые представляют сущности предметной области. Контекст базы данных MedRegistryContext наследуется от DbContext и содержит наборы DbSet для каждой модели, что позволяет выполнять операции с данными через LINQ-запросы. Entity Framework Core автоматически генерирует SQL-запросы на основе LINQ-выражений и обеспечивает отслеживание изменений объектов для синхронизации с БД.

Получение информации из БД осуществляется посредством Entity Framework Core, код метода получения записей на приём, относящихся к определенному пользователю, представлен листингом 1.

Листинг 1 – Код метода LoadAppointments

```
private void LoadAppointments()
{
    // Создаем контекст базы данных для выполнения запросов
    using var db = new MedRegistryContext();

    // Формируем запрос к базе данных с подключением связанных
данных
    IQueryable<Appointment> query = db.Appointments
        .Include(a => a.Patient).ThenInclude(p => p.User)
        .Include(a => a.Doctor).ThenInclude(d => d.User);

    // Фильтруем записи в зависимости от роли пользователя
    if (_role == "Пациент")
    {
        // Для пациента возвращаем только его собственные
записи
        query = query.Where(a => a.Patient.UserId == _userId);
    }
    else if (_role == "Врач")
    {
        // Для врача возвращаем записи, где он назначен врачом
        query = query.Where(a => a.Doctor.UserId == _userId);
    }

    // Сортируем записи по дате и времени начала приёма
    var appointments = query.OrderBy(a =>
a.AppointmentStart).ToList();

    // Отображаем полученные записи на странице
    DisplayAppointments(appointments);
}
```

Метод выполняет запрос к БД с использованием Entity Framework Core для получения всех записей на приём с учетом роли пользователя. Для пациентов возвращаются только их собственные записи, для врачей – записи, где они назначены врачами. Метод использует загрузку связанных данных (Include) для получения информации о пациентах, врачах и пользователях.

Для работы с расписанием врачей разработан метод LoadSchedule, который загружает расписание с применением фильтрации по датам и врачам. Код метода представлен листингом 2.

Листинг 2 – Код метода LoadSchedule

```
public void LoadSchedule()
{
    try
    {
        // Создаем контекст базы данных для выполнения запросов
        using var db = new MedRegistryContext();

        // Загружаем все записи расписания с подключением
данных о врачах и пользователях
        _allSchedules = db.Schedules
            .Include(s => s.Doctor)
                .ThenInclude(d => d.User)
            .OrderBy(s => s.WorkDate)
            .ThenBy(s => s.Doctor.User.LastName)
            .ToList();

        // Применяем фильтры для отображения расписания
        ApplyFilters();
    }
    catch (Exception ex)
    {
        // Обрабатываем ошибку, если загрузка не удалась
        MessageBox.Show($"Ошибка при загрузке расписания:
{ex.Message}\n\nДетали: {ex.InnerException?.Message}",
            "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Error);
    }
}
```

Метод загружает все записи расписания из БД с подключением связанных данных о врачах и пользователях. Результаты сортируются по дате работы и фамилии врача, после чего применяются дополнительные фильтры через метод ApplyFilters.

3.2 Реализация интерфейса пользователя

Для реализации интерфейса пользователя использованы следующие технологии:

- XAML: для разметки и описания структуры интерфейса;
- C#: для программной логики и обработки событий;

- WPF: для создания компонентов интерфейса и управления состоянием приложения.

Для отображения записей на приём был разработан компонент AppointmentsPage, который представляет собой страницу с динамически генерируемыми карточками записей. Каждая карточка содержит информацию о дате, времени, враче, номере кабинета и причине визита. Компонент автоматически адаптирует отображаемую информацию в зависимости от роли пользователя: пациенты видят свои записи, врачи – записи с их участием.

Внешний вид страницы с записями на приём представлен на рисунке 5.

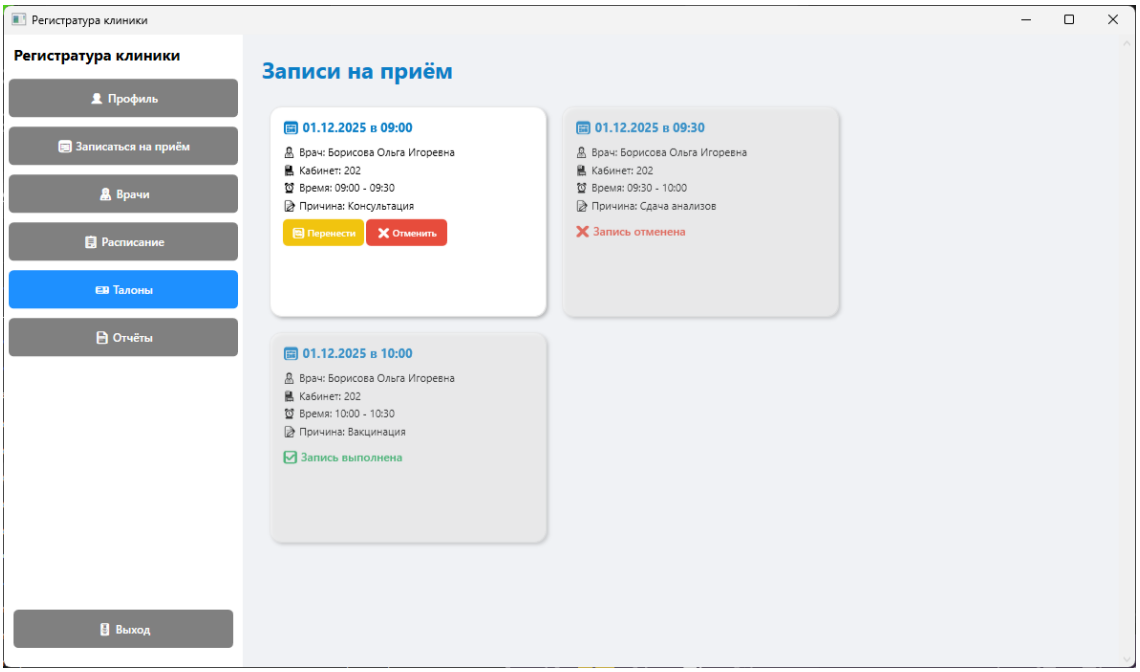

Рисунок 5 – Регистратуры клиники. Вид страницы «Талоны»

Для управления расписанием врачей разработана страница SchedulePage, которая позволяет администраторам и регистраторам просматривать, добавлять, редактировать и удалять записи расписания. Страница поддерживает фильтрацию по врачам и датам, а также импорт расписания из Excel-файлов.

Внешний вид страницы с расписанием врачей представлен на рисунке 6.

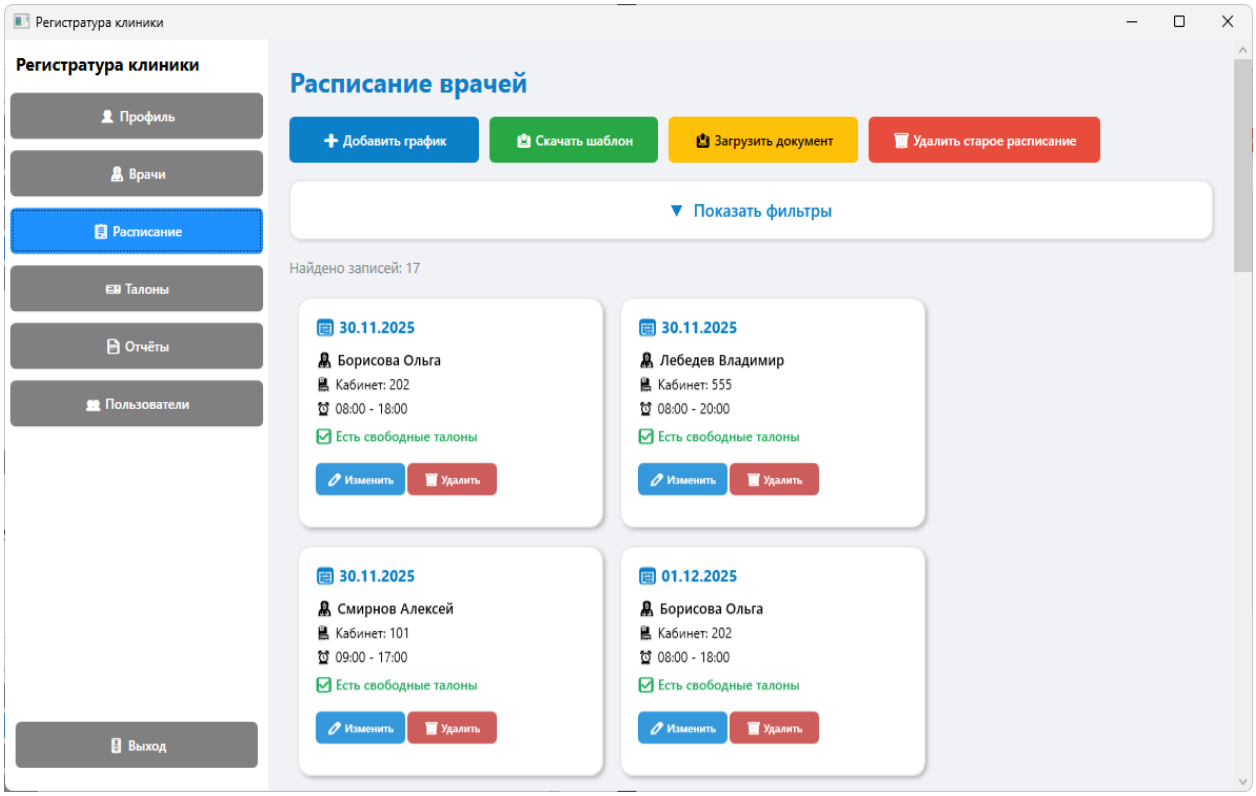

Рисунок 6 – Регистратуры клиники. Вид страницы «Расписание»

3.3 Разграничение прав доступа пользователей

Разграничение прав доступа реализовано при помощи таблицы Roles в БД. Система поддерживает следующие роли: Администратор, Регистратор, Врач и Пациент. Каждая роль определяет набор доступных функций и элементов интерфейса для пользователя. Права администратора у пользователя появляются при RoleName равном "Администратор", что позволяет получать доступ ко всем функциям системы, включая управление пользователями и полный контроль над данными.

Для аутентификации пользователей разработан метод Login_Click, который выполняет проверку учетных данных и определяет роль пользователя. Код метода представлен листингом 3.

Листинг 3 - Код метода аутентификации

```
private void Login_Click(object sender, RoutedEventArgs e)
{
    // Валидация входящих данных
    string username = UsernameBox.Text?.Trim();
    string password = PasswordBox.Password ?? "";

    if (string.IsNullOrEmpty(username) ||
string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Введите логин и пароль.", "Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    // Поиск пользователя в базе данных по логину и паролю
    var user = _db.Users
        .Include(u => u.Role)
        .FirstOrDefault(u => u.Username == username &&
u.PasswordHash == password);

    // Проверка наличия пользователя
    if (user == null)
    {
        MessageBox.Show("Неправильно введён логин или пароль",
"Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    // Получение роли пользователя и открытие главного окна
    string role = user.Role.RoleName;

    var main = new MainWindow(user.UserId, role);
    main.Show();
    this.Close();
}
```

Метод выполняет проверку учетных данных пользователя, извлекая логин и пароль из полей ввода. После успешной аутентификации определяется роль пользователя из связанной таблицы Roles, и открывается главное окно приложения с соответствующими правами доступа.

Проверка прав доступа реализована на уровне интерфейса и логики. Например, управление расписанием доступно только администраторам и

регистраторам, а страница управления пользователями – только администраторам. Код проверки прав доступа представлен листингом 4.

Листинг 4 - Код проверки прав доступа для страницы расписания

```
this.Loaded += (s, e) =>
{
    // Проверяем, имеет ли пользователь доступ к управлению
расписанием
    bool hasAccess = _role == "Администратор" || _role ==
"Регистратор";

    // Скрываем или показываем панель с кнопками в зависимости
от роли
    if (ButtonsPanel != null)
        ButtonsPanel.Visibility = hasAccess ?
Visibility.Visible : Visibility.Collapsed;
};
```

Метод проверяет роль пользователя при загрузке страницы и динамически скрывает или отображает элементы интерфейса в зависимости от прав доступа. Это обеспечивает безопасность системы и предотвращает несанкционированный доступ к функциям управления данными.

3.4 Экспорт и импорт данных

В приложении реализованы функции экспорта и импорта данных в формате Excel с использованием библиотеки ClosedXML. Экспорт данных позволяет сохранять информацию о расписании врачей, записях на приём и других данных в файлы Excel для дальнейшей обработки или архивирования. Импорт данных обеспечивает возможность массового добавления расписания врачей из предварительно подготовленных Excel-файлов.

Листинг 5 – Фрагмент кода метода экспорта шаблона расписания

```
using var workbook = new XLWorkbook();
```

```
var worksheet = workbook.Worksheets.Add("Расписание");
worksheet.Cell(1, 1).Value = "ID врача";
worksheet.Cell(1, 5).Value = "Дата работы (дд.мм.гггг)";
worksheet.Cell(1, 6).Value = "Время начала (чч:мм)";
worksheet.Cell(1, 7).Value = "Время окончания (чч:мм)";

using var db = new MedRegistryContext();
var doctors = db.Doctors.Include(d => d.User).ToList();
var doctorsSheet = workbook.Worksheets.Add("Справочник врачей");
int row = 2;
foreach (var doctor in doctors)
{
    doctorsSheet.Cell(row, 1).Value = doctor.DoctorId;
    doctorsSheet.Cell(row, 2).Value = $"{doctor.User?.LastName}
{doctor.User?.FirstName}";
    row++;
}
workbook.SaveAs(saveDialog.FileName);
```

Метод создает Excel-файл с листами "Расписание" и "Справочник врачей". Заполняются заголовки столбцов с информацией о расписании врачей, а также формируется справочник со списком всех врачей из БД для удобства заполнения шаблона.

Импорт данных из Excel реализован для расписания врачей. Метод выполняет чтение данных, валидацию (проверка формата даты, времени, существования врача), проверку на дубликаты и сохранение корректных записей. Ключевой фрагмент кода метода импорта представлен листингом 6.

Листинг 6 – Ключевой фрагмент кода метода импорта расписания

```
using var workbook = new XLWorkbook(openDialog.FileName);
var worksheet = workbook.Worksheets.FirstOrDefault(ws => ws.Name
== "Расписание");
var schedules = new List<Schedule>();
var errors = new List<string>();

int row = 2;
while (!worksheet.Cell(row, 1).IsEmpty())
{
    var doctorId = int.Parse(worksheet.Cell(row,
1).GetString());
    var dateStr = worksheet.Cell(row, 5).GetString();
    var startTime = TimeSpan.Parse(worksheet.Cell(row,
```

```
6).GetString());
    var endTime = TimeSpan.Parse(worksheet.Cell(row,
7).GetString());

    var schedule = new Schedule
    {
        DoctorId = doctorId,
        WorkDate =
DateOnly.FromDateTime(DateTime.ParseExact(dateStr, "dd.MM.yyyy",
null)),
        StartTime = workDate.Date.Add(startTime),
        EndTime = workDate.Date.Add(endTime),
        IsAvailable = true
    };

    schedules.Add(schedule);
    row++;
}

db.Schedules.AddRange(schedules);
db.SaveChanges();
```

Метод импорта выполняет чтение данных из Excel-файла, создает объекты Schedule и сохраняет их в базу данных с валидацией и обработкой ошибок.

4. Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

Структурное тестирование направлено на проверку корректности работы отдельных модулей и методов приложения. Для тестирования работы с записями на приём были разработаны тесты, использующие in-memory БД для изоляции тестируемого кода от реальной БД. Важный фрагмент кода тестовых методов представлен листингом 7.

Листинг 7 – Фрагмент кода тестовых методов для проверки работы с записями на приём

```
[Fact]
public void CreateAppointment_ValidData_ReturnsAppointment()
{
    using var context = GetInMemoryContext();
    var appointment = new Appointment
    {
        PatientId = 1,
        DoctorId = 1,
        AppointmentStart = DateTime.Now.AddDays(1),
        AppointmentEnd = DateTime.Now.AddDays(1).AddHours(1),
        Status = "Запланировано"
    };

    context.Appointments.Add(appointment);
    context.SaveChanges();

    var created = context.Appointments.FirstOrDefault(a =>
a.PatientId == 1);
    Assert.NotNull(created);
    Assert.Equal("Запланировано", created.Status);
}
[Fact]
public void GetAppointment_ById_ReturnsCorrectAppointment()
{
    // Arrange
    using var context = GetInMemoryContext();
    var appointment = new Appointment
    {
        AppointmentId = 1,
```

```
            PatientId = 1,
            DoctorId = 1,
            AppointmentStart = DateTime.Now.AddDays(1),
            AppointmentEnd = DateTime.Now.AddDays(1).AddHours(1),
            Status = "Запланировано"
        };
        context.Appointments.Add(appointment);
        context.SaveChanges();

        // Act
        var result = context.Appointments.Find(1);

        // Assert
        Assert.NotNull(result);
        Assert.Equal(1, result.AppointmentId);
        Assert.Equal(1, result.PatientId);
    }

    [Fact]
    public void
    UpdateAppointment_ChangeStatus_UpdatesSuccessfully()
    {
        // Arrange
        using var context = GetInMemoryContext();
        var appointment = new Appointment
        {
            AppointmentId = 1,
            PatientId = 1,
            DoctorId = 1,
            AppointmentStart = DateTime.Now.AddDays(1),
            AppointmentEnd = DateTime.Now.AddDays(1).AddHours(1),
            Status = "Запланировано"
        };
        context.Appointments.Add(appointment);
        context.SaveChanges();

        // Act
        var appointmentToUpdate = context.Appointments.Find(1);
        appointmentToUpdate.Status = "Завершено";
        context.SaveChanges();

        // Assert
        var updated = context.Appointments.Find(1);
        Assert.Equal("Завершено", updated.Status);
    }
```

Тестовые методы используют паттерн AAA (Arrange-Act-Assert), где в секции Arrange подготавливаются тестовые данные, в секции Act выполняется тестируемое действие, а в секции Assert проверяется корректность результата.

Тесты подтвердили корректность работы методов создания и получения записей на приём [1].

В рамках структурного тестирования были проверены следующие методы работы с записями на приём:

- создание записи на приём (CreateAppointment);
- получение записи на приём по идентификатору (GetAppointment);
- обновление записи на приём (UpdateAppointment);
- удаление записи на приём из базы данных (DeleteAppointment).

Результаты структурного тестирования представлены в таблице 1.

Таблица 1 – Результаты структурного тестирования

№ п/п	Действие	Ожидаемый результат	Полученный результат
1	CreateAppointment	Запись успешно записана и создана в базе данных	Запись создана, статус соответствует ожидаемому
2	GetAppointment	Запись по идентификатору найдена и вывелась	Возврат корректной записи с указанным ID
3	UpdateAppointment	Обновление статуса записи на приём	Запись успешно обновлена

Все тесты структурного тестирования были успешно пройдены. Методы работы с записями на приём корректно выполняют операции создания, чтения, обновления и удаления данных в БД. Тестирование подтвердило правильность работы слоя доступа к данным и корректность взаимодействия с Entity Framework Core.

4.2 Функциональное тестирование

Функциональное тестирование направлено на проверку работы приложения с точки зрения пользователя и соответствия требованиям. В рамках функционального тестирования были проверены следующие

сценарии:

- Просмотр расписания врачей;
- управление расписанием врачей;
- разграничение прав доступа в зависимости от роли пользователя;
- экспорт и импорт данных в формате Excel;
- фильтрация записей по различным критериям.

Результаты функционального тестирования представлены в таблице 2.

Таблица 2 – Результаты функционального тестирования

№ п/п	Действие	Ожидаемый результат	Полученный результат
1	Просмотр расписания врачей	Отображение списка расписания врачей	Список расписания отображается корректно
2	Создание записи на приём	Создание новой записи в базе данных	Запись успешно создана
3	Редактирование записи на приём	Обновление записи в базе данных	Запись успешно обновлена
4	Отмена записи на приём	Изменение статуса записи	Статус изменён корректно
5	Управление расписанием врачей	Создание записи расписания в базе данных	Расписание успешно добавлено
6	Экспорт расписания в Excel	Создание Excel-файла с данными	Файл успешно создан
7	Импорт расписания из Excel	Добавление записей в базу данных	Данные успешно импортированы
8	Проверка прав доступа	Скрытие недоступных элементов интерфейса	Права доступа работают корректно

Все проверенные сценарии были успешно пройдены. Приложение корректно обрабатывает ввод данных пользователем, выполняет валидацию и обеспечивает надежное хранение информации в БД. Тестирование показало, что программный продукт готов к использованию и соответствует предъявленным требованиям.

5. Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

Для установки ИС «Регистратура клиники» на стороне сервера необходимо: убедиться, что сервер соответствует минимальным системным требованиям Microsoft SQL Server 2019 Express; установить Microsoft SQL Server 2019 Express; установить Microsoft SQL Server Management Studio 18; запустить Microsoft SQL Server Management Studio 18, осуществить подключение к серверу и выполнить SQL-скрипт по созданию БД (файл «Скрипт по созданию бд.sql»).

Для установки приложения на стороне клиента необходимо: установить клиентское приложение; установить .NET 8.0 Runtime (если не установлен); при необходимости настроить строку подключения к БД в файле конфигурации приложения.

5.2 Инструкция по работе

Для запуска приложения требуется запустить исполняемый файл приложения. Пользователю откроется окно авторизации (рисунок 7), в котором необходимо указать учетные данные пользователя или перейти к регистрации новой учетной записи. Регистрация и авторизация — базовые шаги, которые обеспечивают защищенный доступ к системе и разделение прав доступа между сотрудниками.

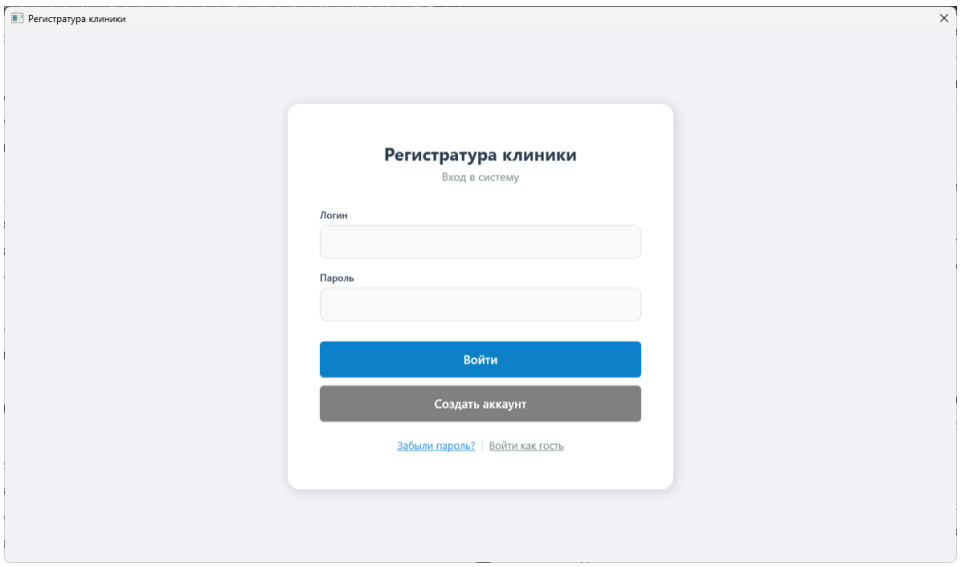

Рисунок 7 – Регистратура клиники. Вид окна «Авторизация»

После успешной авторизации открывается главное окно приложения с боковым меню навигации. Доступные пункты меню зависят от роли пользователя в системе.

Для роли администратора доступны все пункты меню: «Профиль», «Врачи», «Расписание», «Талоны», «Отчёты», «Пользователи».

На странице «Пользователи» администратор может управлять всеми пользователями системы и их ролями (рисунок 8). На странице «Врачи» доступно управление информацией о врачах. На странице «Талоны» отображаются все записи на приём с возможностью редактирования и изменения статуса. На странице «Отчёты» доступен просмотр и создание электронных медицинских карт пациентов.

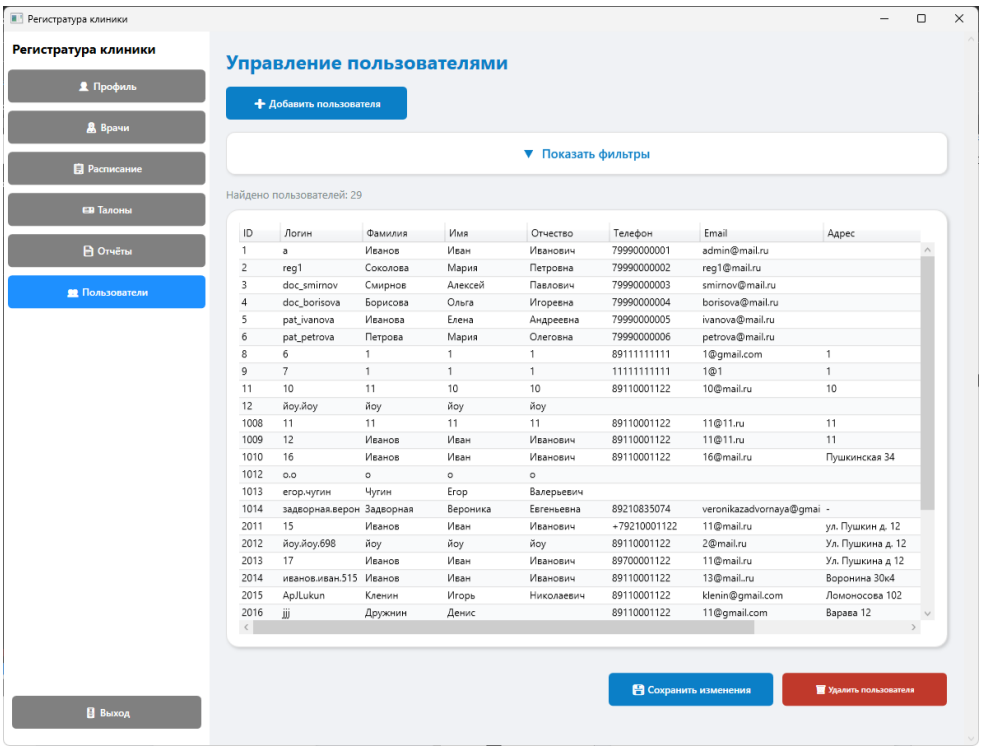

Рисунок 8 – Регистратура клиники. Вид страницы с пользователями

Для роли регистратора доступны пункты меню: «Профиль», «Врачи», «Расписание», «Талоны», «Отчёты» (рисунок 9).

Регистратор может создавать записи на приём для пациентов, изменять статусы записей, просматривать и изменять список врачей (рисунок 9), а также их расписание, и работать с медицинскими картами пациентов. Управление пользователями и врачами для регистратора недоступно.

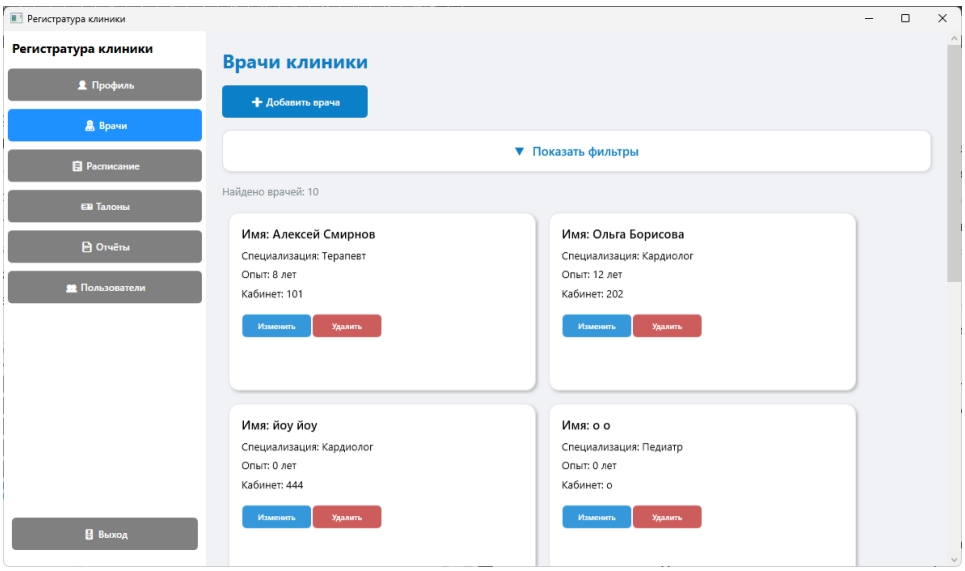

Рисунок 9 – Регистратура клиники. Вид страницы списка врачей

Для роли врача доступны пункты меню: «Профиль», «Врачи», «Расписание», «Талоны», «Отчёты».

Врач видит только свои записи на приём на странице «Талоны» и может создавать медицинские записи о приёме (рисунок 10), просматривать историю медицинских карт пациентов, создавать повторные записи для пациентов.

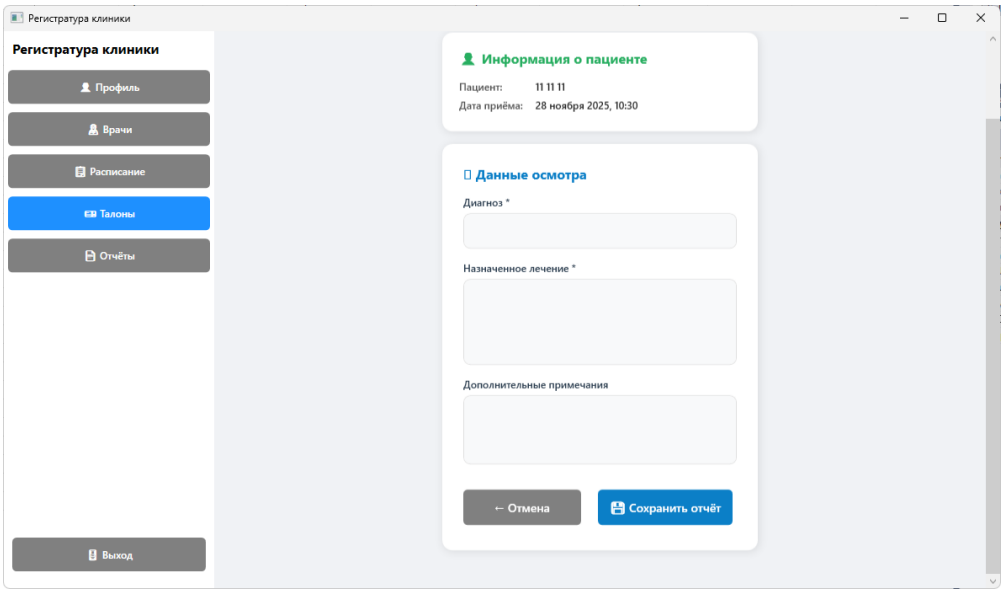

Рисунок 10 – Регистратура клиники. Вид страницы записи о приеме

Для роли пациента доступны пункты меню: «Профиль», «Записаться на приём», «Врачи», «Расписание», «Талоны».

Пациент может записаться на приём через кнопку «Записаться на приём» (рисунок 11), просматривать свои записи на приём с возможностью переноса и отмены, просматривать список врачей и их расписание, редактировать свои личные данные на странице «Профиль».

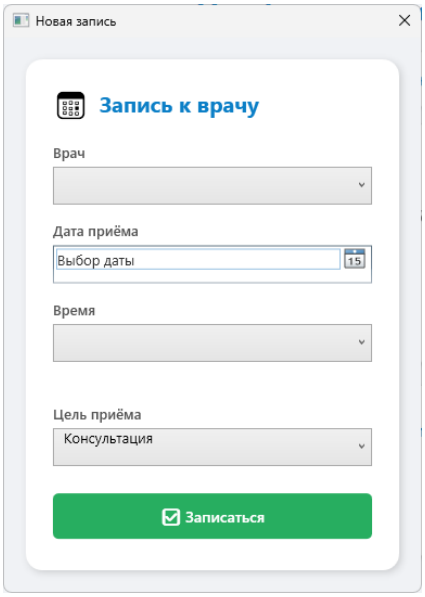

Рисунок 11 – Регистратура клиники. Вид страницы записи на прием

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были приобретены и закреплены навыки проектирования, разработки и документирования информационных систем медицинского назначения. Была выполнена разработка полноценной многопользовательской информационной системы «регистратуры клиники», обеспечивающий автоматизацию ключевых процессов медицинского учреждения [2, 5].

Поставленная цель – создания ИС для записи, управления расписанием врачей и ведения медицинской документации – достигнута. В процессе выполнения работы были решены следующие задачи:

- изучена предметная область, связанная с автоматизацией работы регистратуры;
- сформирована программная спецификация;
- разработаны диаграммы вариантов использования;
- построены концептуальная, логическая и физическая модели БД;
- создана БД в SSMS с учётом требований целостности;
- разработаны основные серверные методы для взаимодействия с данными;
- реализовано WPF-клиентское приложение;
- настроен механизм авторизации и дифференциации прав доступа;
- добавлена поддержка конфигурируемого подключения к БД;
- реализованы функции просмотра, редактирования, сортировки и фильтрации медицинских данных;
- подготовлено руководство пользователей.

Таким образом, разработанная система соответствует заданным требованиям и может рассматриваться как готовое решение для применения в медицинских учреждениях.

Проект успешно выполнен.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading (дата обращения: __.11.2025). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

2. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – URL: https://znanium.ru/catalog/product/2178802 (дата обращения: __.11.2025). – Режим доступа: по подписке. – Текст : электронный.

3. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2024. – 368 с. – Текст : электронный. – URL: https://znanium.ru/catalog/product/2096940 (дата обращения: __.11.2025). – Режим доступа: по подписке.

4. Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – Текст : электронный. – URL: https://ibooks.ru/bookshelf/386796/reading (дата обращения: __.11.2025). – Режим доступа: для зарегистрир. пользователей.

5. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. — Москва : КУРС : ИНФРА-М, 2024. — 336 с. – URL: https://znanium.ru/catalog/product/2083407 (дата обращения: __.11.2025). – Режим доступа: по подписке. – Текст : электронный.