СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc132635760)

[1. ОБЩАЯ ЧАСТЬ 5](#_Toc132635761)

[1.1. Цель разработки 5](#_Toc132635762)

[1.2. Средства разработки 5](#_Toc132635763)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 7](#_Toc132635764)

[2.1. Постановка задачи 7](#_Toc132635765)

[2.1.1. Входные данные 7](#_Toc132635766)

[2.1.2. Выходные данные 7](#_Toc132635767)

[2.1.3. Подробные требования к проекту 8](#_Toc132635768)

[2.2. Внешняя спецификация 8](#_Toc132635769)

[2.2.1. Описание задачи 8](#_Toc132635770)

[2.2.2. Входные и выходные данные 11](#_Toc132635771)

[2.2.3. Методы 13](#_Toc132635772)

[2.2.4. Тесты 17](#_Toc132635773)

[2.2.5. Контроль целостности данных 17](#_Toc132635774)

[2.3. Проектирование 18](#_Toc132635775)

[2.3.1. Схема архитектуры приложения 18](#_Toc132635776)

[2.3.2. Логическая схема данных 19](#_Toc132635777)

[2.3.3. Физическая схема данных 20](#_Toc132635778)

[2.3.4. Структурная схема 22](#_Toc132635779)

[2.3.5. Функциональная схема 24](#_Toc132635780)

[2.3.6. Диаграмма классов 24](#_Toc132635781)

[2.3.7. Схема тестирования 27](#_Toc132635782)

[2.3.8. Схема пользовательского интерфейса 28](#_Toc132635783)

[2.4. Результат работы программы 28](#_Toc132635784)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 34](#_Toc132635785)

[3.1. Инструментальные средства 34](#_Toc132635786)

[3.2. Отладка программы 34](#_Toc132635787)

[3.3. Защитное программирование 35](#_Toc132635788)

[3.4. Характеристики программы 36](#_Toc132635789)

[ЗАКЛЮЧЕНИЕ 38](#_Toc132635790)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 41](#_Toc132635791)

ПРИЛОЖЕНИЕ Б. Текст программы

ПРИЛОЖЕНИЕ В. Сценарий и результаты тестовых испытаний

ПРИЛОЖЕНИЕ Г. Руководство пользователя

ПРИЛОЖЕНИЕ Д. Скрипт базы данных

# ВВЕДЕНИЕ

В условиях стремительного развития информационных технологий внедрение эффективных систем управления в медицинских учреждениях, особенно в стоматологической практике, становится более заметным и неотъемлемым процессом. Система управления пациентами в стоматологии охватывает важные аспекты взаимодействия между пациентами и медицинским персоналом, начиная от записи на прием и ведения истории болезни, и заканчивая учетом предоставленных услуг.

Цель курсового проекта заключается в разработке и анализе такой системы, которая не только оптимизирует процессы в стоматологической клинике, но также способствует повышению качества обслуживания пациентов. Необходимо производить анализ и выделять ключевые аспекты функционирования системы, ее структуру и воздействие на эффективность работы стоматологического учреждения.

Особое внимание будет уделяется процессам записи на прием, ведению истории болезни пациентов и учету предоставленных медицинских услуг. Путем анализа существующих проблем и предложений по их решению формируется представление о необходимых шагах для внедрения современных технологических решений в стоматологическую практику, что позволит улучшить качество медицинского обслуживания и оптимизировать рутинные процессы в учреждении.

# ОБЩАЯ ЧАСТЬ

## Цель разработки

Целью разработки информационной системы управления пациентами в стоматологии «Healthy smile» является упрощение работы с историями болезни пациентов, записи на приём к врачу и учётом предоставляемых услуг . Приложение призвано предоставить удобный и доступный интерфейс для управления историями болезни и услугами, позволяющий сотрудникам легко манипулировать данными, а пациентам записываться на приём.

* 1. Средства разработки

Для проектирования, разработки и тестирования программного обеспечения, а также оформления документации по проекту использовались программные средства, представленные в таблице 1.

Таблица 1 - Программные средства разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип средства | Название средства | Назначение |
| 1 | 2 | 3 | 4 |
| 1 | Операционная система | Microsoft Windows 10 | Организация взаимодействия программ и пользователя |
| 2 | Среда разработки веб-ресурса | Visual Studio 2022 | Разработка веб-приложения и сервера |
| 3 | Построение схем БД | SqlDBM | Построение логической и физической модели базы данных |
| 4 | СУБД | Microsoft SQL Server | Создание и манипуляция базой данных, обеспечение безопасности, надежности хранения и целостности данных, предоставления средств администрирования БД. |
| 5 | Интерфейс взаимодействия | Mircosoft SQL Server Management Studio 18 | Для взаимодействия СУБД |
| 6 | Браузер | Yandex | Взаимодействие с сервером |
| 7 | Текстовый редактор | Microsoft Offiice | Написание |

В качестве средств вычислительной техники при разработке ПО использовался ноутбук Lenovo 15ADA.  Характеристики представлены в таблице 2.

Таблица 2 - Технические характеристики устройств используемых при разработке ПО

|  |  |  |
| --- | --- | --- |
| № | Тип средства | Название средства |
| 1 | 2 | 3 |
| Ноутбук Lenovo 15ADA | | |
| 1 | Размер экрана: | 15,6 |
| 2 | Разрешение экрана: | 1920x1080 |
| 3 | Линейка процессора: | AMD Ryzen 5 3500U |
| 4 | Количество ядер процессора: | 4 |
| 5 | Оперативная память: | 8 ГБ |
| 6 | Тип видеокарты: | Встроенная |
| 7 | Видеокарта: | AMD Radeon Vega 8 |
| 8 | Конфигурация накопителей: | SSD |
| 9 | Общий объем всех накопителей: | 256 ГБ |

# 

# СПЕЦИАЛЬНАЯ ЧАСТЬ

## Постановка задачи

Разработать информационную систему управления заявками на техническое обслуживание «Tech Support». Приложение на базе Android, представляющее мобильное приложение для создания заявок на ремонт техники. Приложение для компьютера, представляет WPF приложение дающая возможность сотрудникам управлять заявками поступившими к ним.

Разработать информационную систему управления пациентами в стоматологии «Healthy smile». Приложение для компьютера, представляет сервер API и веб-ресурс, которые предоставляют возможность пациентам записываться на приём, а сотрудникам манипулировать данными о историях болезни, услугах и расписанием приёмов

## Входные данные

Входные данные представлены в следующем виде:

* Регистрация пользователя (Фамилия, Имя, Отчество, Адрес электронной почты, Пароль, Возраст, Пол);
* Авторизация пользователя (Адрес электронной почты, Пароль);
* Запись на приём (Врач, Услуга, дата, время);
* Изменение истории болезни (Жалоба, лечение)
* Добавление свободных дат приёма (дата, время)
* Добавление услуги (название, цена)
* Изменение свободных дат приёма (дата, время)
* Изменение услуги ( название, цена)

## Выходные данные

Выходные данные представлены в следующем виде:

* Список дат приёмов (дата, время, доступность);
* Список услуг (название, цена)
* Выбранная запись (услуга, врач, дата приёма, время приёма)
* Список историй болезни (пациент, жалоба, лечение, дата и время приёма
* Текущие записи (номер талона, услуга, стоимость, врач, дата приёма)

## Подробные требования к проекту

Подробные требования к проекту «информационная система управления пациентами в стоматологии «Healthy smile»», представлены в Приложении A. Техническое задание.

## Внешняя спецификация

## Описание задачи

При открытии веб-ресурса пользователю должно быть доступно окно авторизации. После ввода данных, пользователю открывается окно в зависимости от его роли.

Для роли админа открывается страница пользователя «Администратор», где расположены 2 вкладки

Во вкладке «Услуги» должен находиться весь список предоставляемых услуг, кнопки для изменения, удаления и добавления услуги. Для добавления данных необходимо нажать на кнопку «Добавить услугу», после чего заполнить форму на открывшемся окне и нажать кнопку «Подтвердить». Для изменения данных необходимо выбрать услугу из списка, после чего заполнить форму на открывшемся окне, нажать кнопку «Подтвердить». Для удаления данных необходимо выбрать услугу из списка и нажать кнопку «Удалить»

Во вкладке «Даты приёмов» должен находиться весь список занятых и свободных записей для приёма, кнопки для изменения, удаления и добавления даты. Для добавления данных необходимо нажать на кнопку «Добавить дату приёма», после чего заполнить форму на открывшемся окне и нажать кнопку «Подтвердить». Для изменения данных необходимо выбрать дату и время из списка, после чего заполнить форму на открывшемся окне, нажать кнопку «Подтвердить». Для удаления данных необходимо выбрать дату из списка и нажать кнопку «Удалить»

Если роль пользователя «Врач» открывается окно врача. На данном окне должна быть расположена вкладка «Истории болезней». Для изменения данных необходимо выбрать запись из списка, после чего заполнить форму на открывшемся окне, нажать кнопку «Подтвердить»

Если роль пользователя «Пациент» открывается окно пациента. На данном окне должны быть расположены 3 вкладки.

Во вкладке «Главная» должна отображаться заставка, для записи на прём необходимо нажать кнопку «Записаться на приём», затем из выпадающих списков выбрать услугу, врача, дату и время записи, нажать «Подтвердить»

Во вкладке «Услуги» должен отображаться прайс-лист всех предоставляемых услуг

Во вкладке «Профиль» должен отображаться список всех записей пациента к специалистам

## Входные и выходные данные

В таблицах под номерами 3-4 представлены входные и выходные данные приложения.

Таблица 3 - Входные данные приложения

| Поле | Ограничение | Тип | Форма ввода | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Авторизация | | | | |
| Почта пользователя | Должна иметь вид «\*@\*.\*» | Строка | Поле ввода | Поле ввода для почты пользователя при авторизации |
| Пароль пользователя | {6,10} | Строка | Поле ввода | Поле ввода для пароля пользователя при авторизации |
| Регистрация | | | | |
| Имя | {5,100} | Строка | Поле ввода | Поле ввода для имени пользователя при регистрации |
| Фамилия | {5,100} | Строка | Поле ввода | Поле ввода для фамилии пользователя при регистрации |
| Отчество | {5,100} | Строка | Поле ввода | Поле ввода для отчества пользователя при регистрации |
| Возраст | {5,100}, [18-116] | Строка | Поле ввода | Поле ввода для возраста пользователя при регистрации |
| Почта | Должна иметь вид «\*@\*.\*» | Строка | Поле ввода | Поле ввода для почты пользователя при регистрации |
| Пароль | {6,255}, Только буквы латинского алфавита | Строка | Поле ввода | Поле ввода для пароля пользователя при регистрации |
| Пол | {6,100}, Только буквы латинского алфавита | Строка | Выпадающий список | Выпадающий список для пола пользователя при регистрации |
| Создание и изменение услуги | | | | |
| Название услуги | Ограничения не предоставляются, буквы только латинского алфавита | Строка | Поле ввода | Поле ввода для названия услуги |
| Цена | Ограничения не предоставляются, буквы только латинского алфавита | Строка | Поле ввода | Поле ввода для цены |
| Создание и изменение даты приёма | | | | |
| Дата | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора даты |
| Время | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора времени |
| Запись на приём | | | | |
| Услуга | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список услуг для записи на приём |
| Дата | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список врачей для записи на приём |
| Врач | Ограничения не предоставляются | Строка | Выпадающий список | Выпадающий список для выбора врача для записи на приём |
| Изменение истории болезни | | | | |
| Жалоба | Ограничения не предоставляются | Строка | Поле ввода | Поле ввода жалобы при изменении истории болезни |
| Лечение | Ограничения не предоставляются | Строка | Поле ввода | Поле ввода лечение |

Таблица 4 - Выходные данные приложения

| Поле | Ограничение | Тип | Описание |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Вывод списка историй болезни | | | |
| Список историй болезней | Не предусмотрено | Лист | Элемент представляет собой список историй болезни с жалобой, лечением, пациентом, датой приёма |
| Вывод списка услуг Админ | | | |
| Список списка услуг | Не предусмотрено | Лист | Элемент представляет собой список существующих услуг с названием и ценой |
| Вывод вариантов записи на приём | | | |
| Список услуг | Не предусмотрено | Лист | Элемент представляет собой список услуг с названием и ценой |
| Список врачей | Не предусмотрено | Лист | Элемент представляет собой список врачей с ФИО |
| Список дат | Не предусмотрено | Лист | Элемент представляет собой список датой, временем и доступностью вариантов записи |
| Вывод списка дат приёмов Админ | | | |
| Список дат приёмов | Не предусмотрено | Лист | Элемент представляет собой список датой, временем и доступностью существующих |
| Вывод списка приёмов пациента в профиле | | | |
| Список приёмов | Не предусмотрено | Лист | Элемент представляет собой список всех приёмов авторизированного пользователя |

## Методы

При разработке веб-приложения (WEB MVC) и API (WEB API) использовалось средство разработки Visual Studio, при помощи фреймворка ASP NET Core.

Приложения были разработаны с применением основных принципов объектно-ориентированного программирования. Использованы следующие парадигмы: инкапсуляция, которая ограничивает доступ компонентов программы друг к другу; наследование, необходимое для повторного использования кода и способствующее независимому расширению программного обеспечения и интерфейсов; полиморфизм, обеспечивающий обработку различных типов данных.

На рисунке 1 приведён пример инкапсуляции.

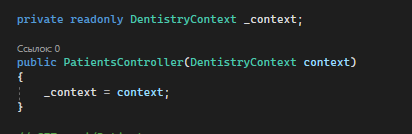


Рисунок - Пример инкапсуляции

Следующий рисунок 2 представляет наследование.

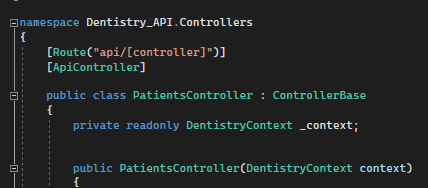


Рисунок - Пример наследования

Для разработки веб-приложения и API применялся архитектурный паттерн MVC (Model-View-Controller). В этой структуре пользователь через представление (View) вызывает методы контроллера. Контроллер, в свою очередь, обновляет модель в соответствии с действиями пользователя. Затем эта модель передается обратно в представление, которое может запрашивать данные из модели.

На рисунке 3 представлена модель.

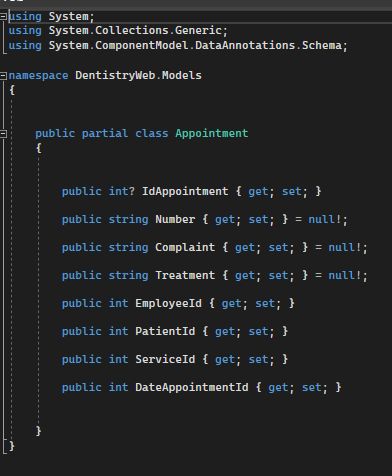


Рисунок - Пример модели из парадигмы MVC

На рисунке 4 представлено представление для модели.

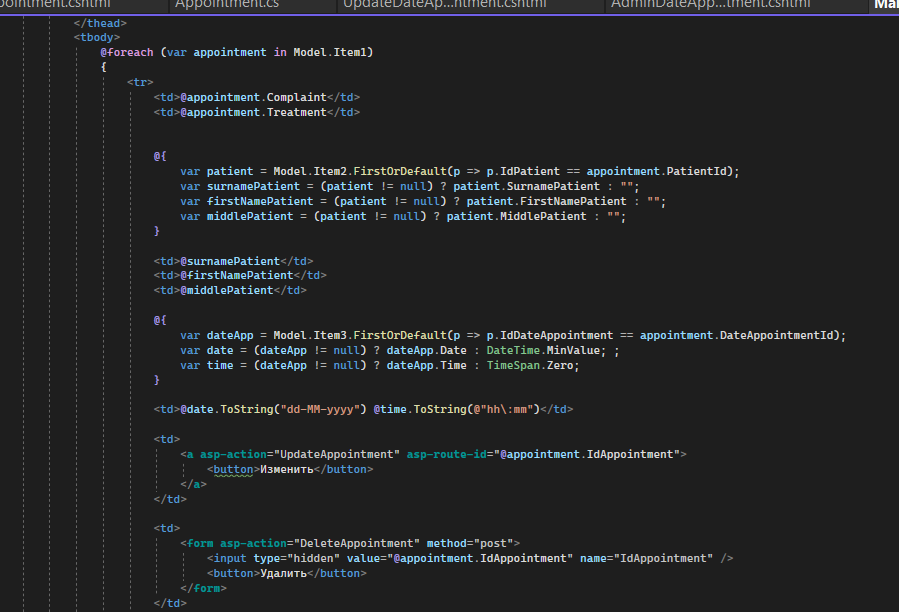


Рисунок – Представление из парадигмы MVC

На рисунке 5 представлен контроллер.



Рисунок - Пример контроллера из парадигмы MVC

Оптимизация и рефакторинг представляют собой ключевые этапы в процессе разработки программного обеспечения. Рефакторинг способствует улучшению структуры кода, устранению ошибок и повышению его качества. Эти изменения, в свою очередь, не только обеспечивают более эффективное выполнение кода, но также облегчают поддержку и возможность последующего расширения. С другой стороны, оптимизация направлена на обеспечение эффективной работы приложения, ускорение его выполнения и минимизацию потребления ресурсов. Это позволяет достичь более высокой производительности и оптимизировать использование ресурсов. На рисунках 6-7 приведен пример рефакторинга кода, осуществленного путем выделения функций в отдельные методы, что обеспечит возможность модернизации кода.

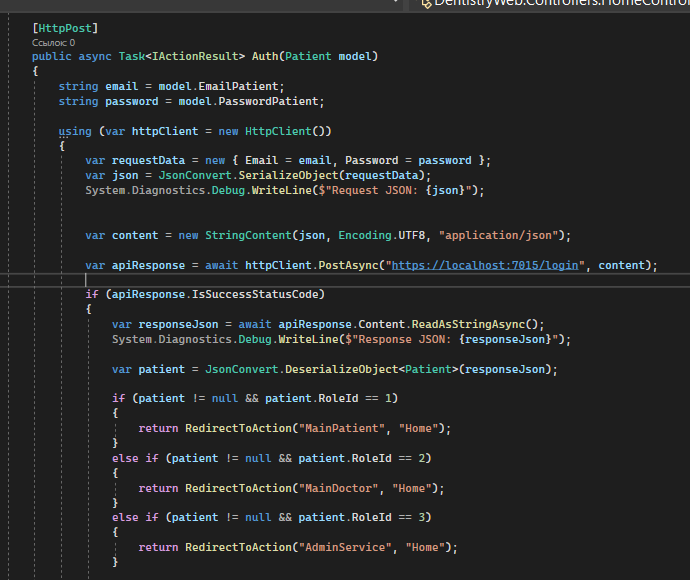


Рисунок - Пример кода до рефакторинга



Рисунок - Пример кода после рефакторинга

## Тесты

Для тестирования программы будет использоваться ручное тестирование по тестовым сценариям, также будет применено нагрузочное тестирование на разработанное API.

Сценарий и результаты тестовых испытаний представлены в Приложении В.

## Контроль целостности данных

В ходе разработки информационной системы были применены методы валидации входных данных, которые обеспечивают целостность этих данных. Этот подход включает контроль целостности данных, который определяет ситуации и действия приложения при выполнении задач, связанных с сохранением, изменением или удалением данных.

Таблица 5 - Контроль целостности данных

| № | Ситуация | Аномалия | Реакция программы |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 1 | Регистрация пользователя | Введена некорректная почта | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен существующий почта | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен некорректный пароль | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен некорректный телефон пользователя | Программа после попытки регистрации сообщает о провале регистрации. |
| Введен существующий телефон пользователя | Программа после попытки регистрации сообщает о провале регистрации. |
| 2 | Добавление и изменение модели | Введена некоренное название | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |
| Название не введено | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |
| Не выбран тип устройства | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |
| Не выбран бренд | Программа после попытки изменения данных сообщает о провале добавления или обновления модели |
| Выход за пределы ограничения в названии. | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |
| 3 | Добавление и изменение услуги | Введено некоренное название | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |
| Название не введено | Программа после попытки изменения данных сообщает о провале добавления или обновления модели. |

## Проектирование

## Схема архитектуры приложения

Приложение отправляет запросы на сервер, где находится база данных, с помощью API. Запросы и ответы осуществляются по определенному протоколу и формату данных, которые заранее согласовываются между приложением и сервером.

Приложение может отправлять запросы на чтение, изменение или удаление данных из базы данных через API. Соответственно, сервер обрабатывает запросы и отправляет обратно ответы с нужной информацией в формате, понятном приложению. Это позволяет приложению получать и обрабатывать данные из базы данных без необходимости обращения к ней напрямую.

Ниже на рисунке 8 представлена схема связи базы данных и приложения через API.

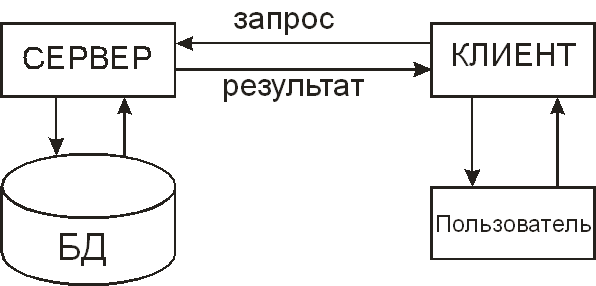


Рисунок - Схема архитектуры приложения

## Логическая схема данных

В базе данных находится следующие таблицы: роль, пациент, сотрудник, запись, клиника, дата записи, услуга - таблицы между собой имеют связь один ко многим

Ниже на рисунке 9 представлена логическая схема данных.

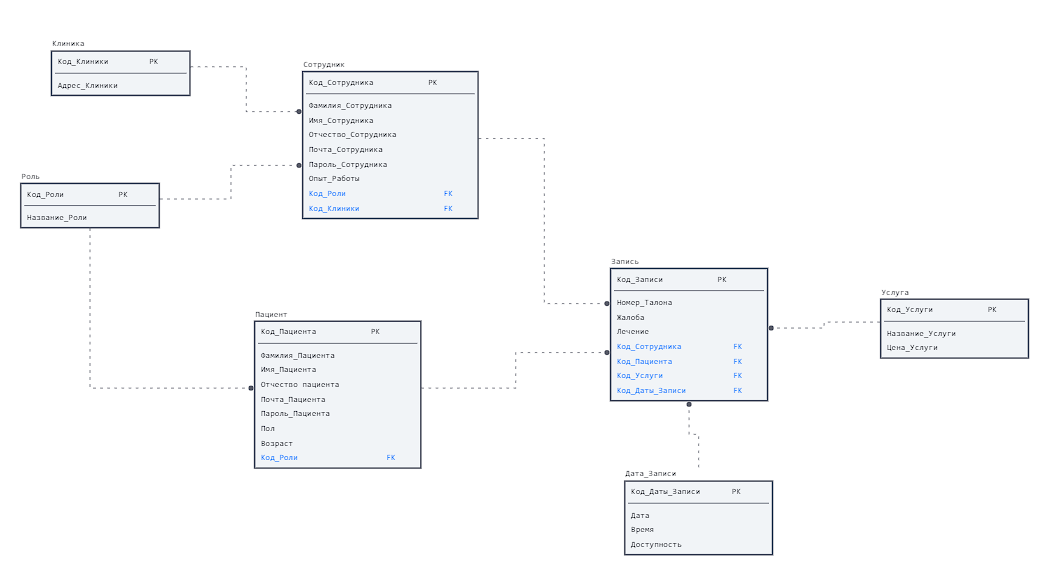


Рисунок - Логическая схема базы данных

## Физическая схема данных

Ниже на рисунке 10 представлена физическая схема базы данных

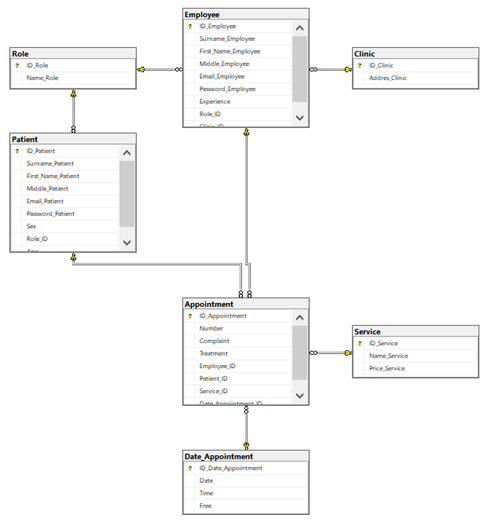


Рисунок - Физическая схема базы данных

Словарь данных представлен в таблице ниже.

Таблица 6 - Словарь данных

| Ключ | Наименование | Тип данных | Обязательность заполнения | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| [Clinic] | | | | |
| PK | [ID\_Clinic] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Addres\_Clinic] | Varchar(100) | Not null | Адрес клиники |
| [Role] | | | | |
| PK | [ID\_Role] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Name\_Role] | Varchar(50) | Not null | Название роли |
| [Employee] | | | | |
| PK | [ID\_Employee] | int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Surname\_Employee] | Varchar(50) | Not null | Фамилия сотрудника |
|  | [First\_Name\_Employee] | Varchar(50) | Not null | Имя сотрудника |
|  | [Middle\_Employee] | Varchar(50) | Not null | Отчество сотрудника |
|  | [Email\_Employee] | Varchar(50) | Not null | Почта сотрудника |
|  | [Password\_Employee] | Varchar(10) | Not null | Пароль сотрудника |
|  | [Experience] | Int | Not null | Опты работы |
| FK | [Role\_ID] | Int | Not null | Код роли |
| FK | [Clinic\_ID] | Int | Not null | Код клиники |
| [Service] | | | | |
| РК | [ID\_Service] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Name\_Service] | Varchar(50) | Not null | Название услуги |
|  | [Price\_Service] | Int | Not null | Цена услуги |
| [Patient] | | | | |
| PK | [ID\_Patient] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Surname\_Patient] | Varchar(50) | Not null | Фамилия пациента |
|  | [First\_Name\_Patient] | Varchar(50) | Not null | Имя пациента |
|  | [Middle\_Patient] | Varchar(50) | Not null | Отчество пациента |
|  | [Email\_Patient] | Varchar(50) | Not null | Почта пациента |
|  | [Password\_Patient] | Varchar(10) | Not null | Пароль пациента |
|  | [Sex] | Varchar(50) | Not null | Пол пациента |
|  | [Age] | Varchar(50) | Not null | Возраст пациента |
| FK | [Role\_ID] | Int | Not null | Код роли |
| [Date\_Appointment] | | | | |
| PK | [ID\_Date\_Appointment] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Date] | date | Not null | Дата |
|  | [Time] | time | Not null | Время |
|  | [Free] | bit | Not null | Доступность |
| [Appointment] | | | | |
| PK | [ID\_Appointment] | Int | Not null | Уникальный идентификатор таблицы с атрибутом auto increment |
|  | [Number] | Varchar(5) | Not null | Номер талона |
|  | [Complaint] | Varchar(200) | Not null | Жалоба |
|  | [Treatment] | Varchar(200) | Not null | Лечение |
| FK | [Employee\_ID] | Int | Not null | Код сотрудника |
| FK | [Patient\_ID] | Int | Not null | Код пациента |
| FK | [Service\_ID] | Int | Not null | Код услуги |
| FK | [Date\_Appointment\_ID] | Int | Not null | Когд даты |

## Структурная схема

На диаграмме классов представлена графическая модель, показывающая связи, зависимости и отношения между классами в программном проекте. Классы, как объекты программирования, могут иметь свойства, методы и поля.

Диаграмма классов приложения:

На рисунке 11 представлена схема приложения

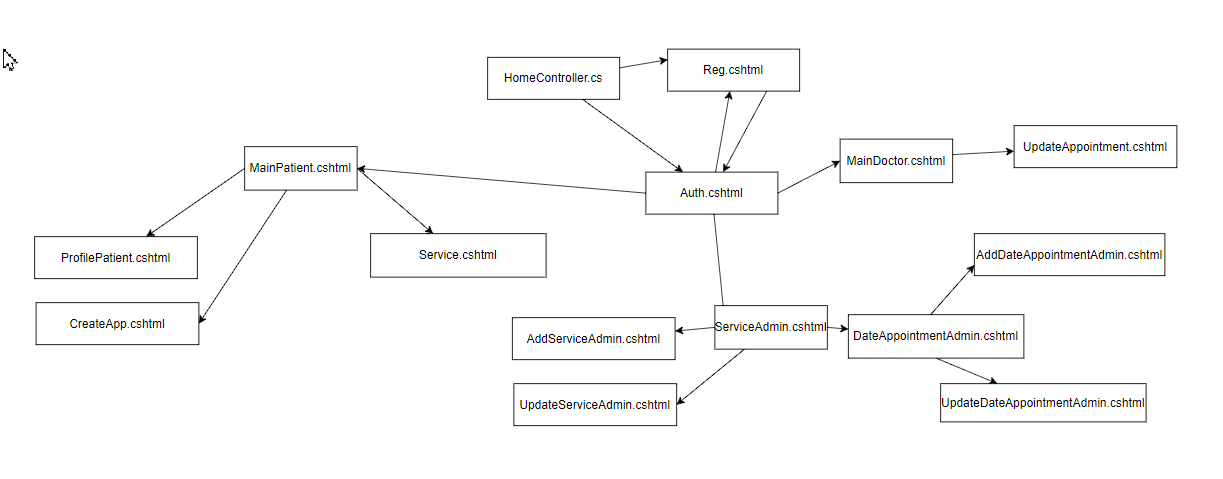


Рисунок – Структурная схема

Далее представлена таблица с описанием модулей

Таблица 7 - Модули приложения клиента

| Номер | Название модуля | Описание модуля |
| --- | --- | --- |
| 1 | 2 | 3 |
| 1 | HomeController.cs | Данный модуль содержит логику и разметку всех моделей и запросы к API |
| 2 | Reg.cshtml | Данный модуль содержит представление для регистрации |
| 3 | Auth.cshtml | Данный модуль содержит представление авторизации |
| 4 | MainDoctor.cshtml | Данный модуль содержит представление главной страницы доктора |
| 5 | UpdateAppointment.cshtml | Данный модуль содержит представление изменения истории болезни |
| 6 | ServiceAdmin.cshtml | Данный модуль содержит представление услуг админа |
| 7 | AddServiceAdmin.cshtml | Данный модуль содержит представление добавления услуг админа |
| 8 | UpdateServiceAdmin.cshtml | Данный модуль содержит представление изменения услуг админа |
| 9 | DateAppointmentAdmin.cshtml | Данный модуль содержит представление дат приёмов админа |
| 10 | AddDateAppointmentAdmin.cshtml | Данный модуль содержит представление добавления дат приёмов админа |
| 11 | UpdateDateAppointmentAdmin.cshtml | Данный модуль содержит представление изменения дат приёмов админа |
| 12 | MainPatient.cshtml | Данный модуль содержит представление главной страницы пациента |
| 13 | Service.cshtml | Данный модуль содержит представление услуг пациента |
| 14 | ProfilePatient.cshtml | Данный модуль содержит представление профиля пользователя |
| 15 | CreateApp.cshtml | Данный модуль содержит представление записи на приём |

## Функциональная схема

На функциональной схеме представлены функции, находящие на окнах приложений. Данная схема позволяет прослеживать связи между компонентами приложения и оценивать работу системы в целом.

На рисунке 12 представлена функциональная схема приложения

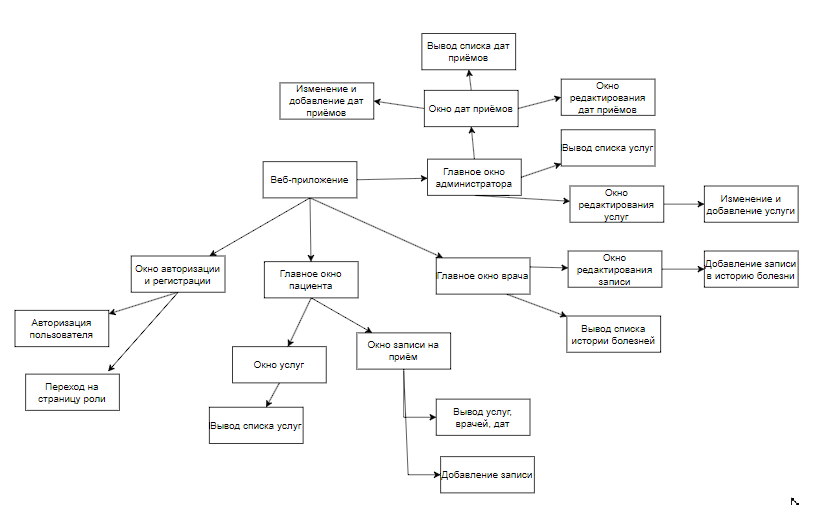


Рисунок – Функциональная схема

## Диаграмма классов

В таблице представлено описание классов проекта.

Таблица 10 - Классы и методы приложения.

| № | Наименование класса | Тип класса | Описание | |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | |
|  | HomeController.cs | public | Данный класс отвечает за логику приложения и запросы к API |
|  | Appointment.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Clinic.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | DateAppointment.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Employee.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Patient.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Role.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |
|  | Service.cs | Public | Данный класс отвечает за модель используемую при создания запроса. |

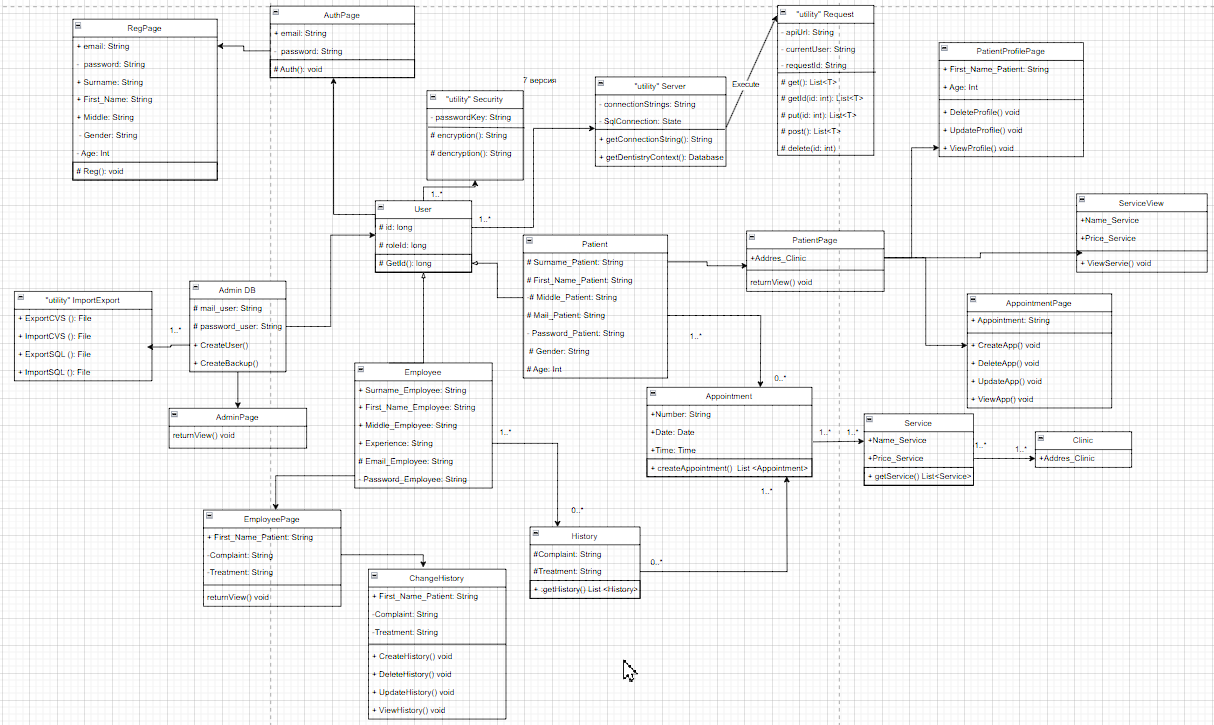


Рисунок – Диаграмма классов

## Схема тестирования

Схема тестирования подробно описана в приложении В Сценарий и результаты тестовых испытаний.

## Схема пользовательского интерфейса

Далее на рисунках 14-16 представлена схема пользовательского интерфейса, которая показывает, как осуществляется переход между окнами приложений.

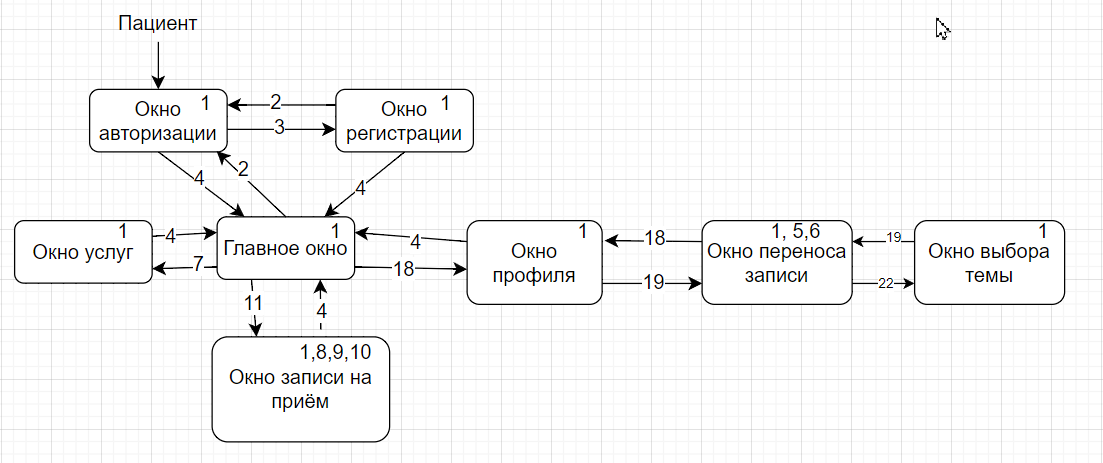


Рисунок – Схема интерфейса пользователя

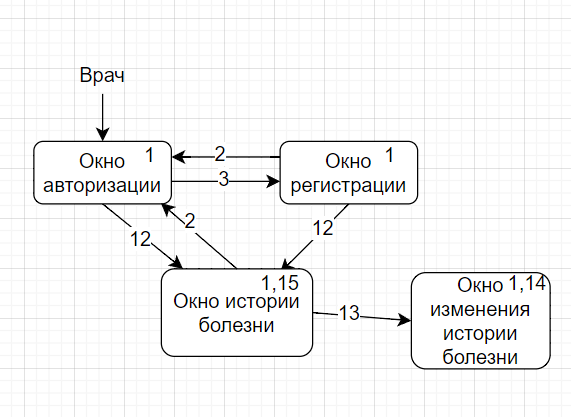


Рисунок – Схема интерфейса врача

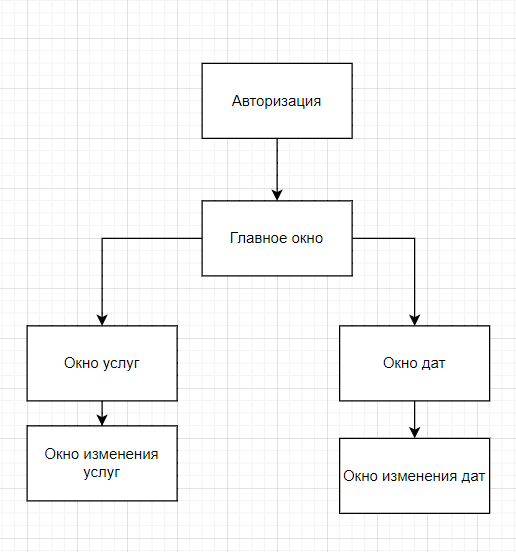


Рисунок – Схема интерфейса авдминистратора

## Результат работы программы

В результате выполнения поставленной задачи, было разработано приложение, которое является информационной системой управления пациента в стоматологии «Healthy smile»

На рисунках 17-29 представлен результат работы программы. Более подробно результат работы программы описан в приложении Г «Руководство пользователя».

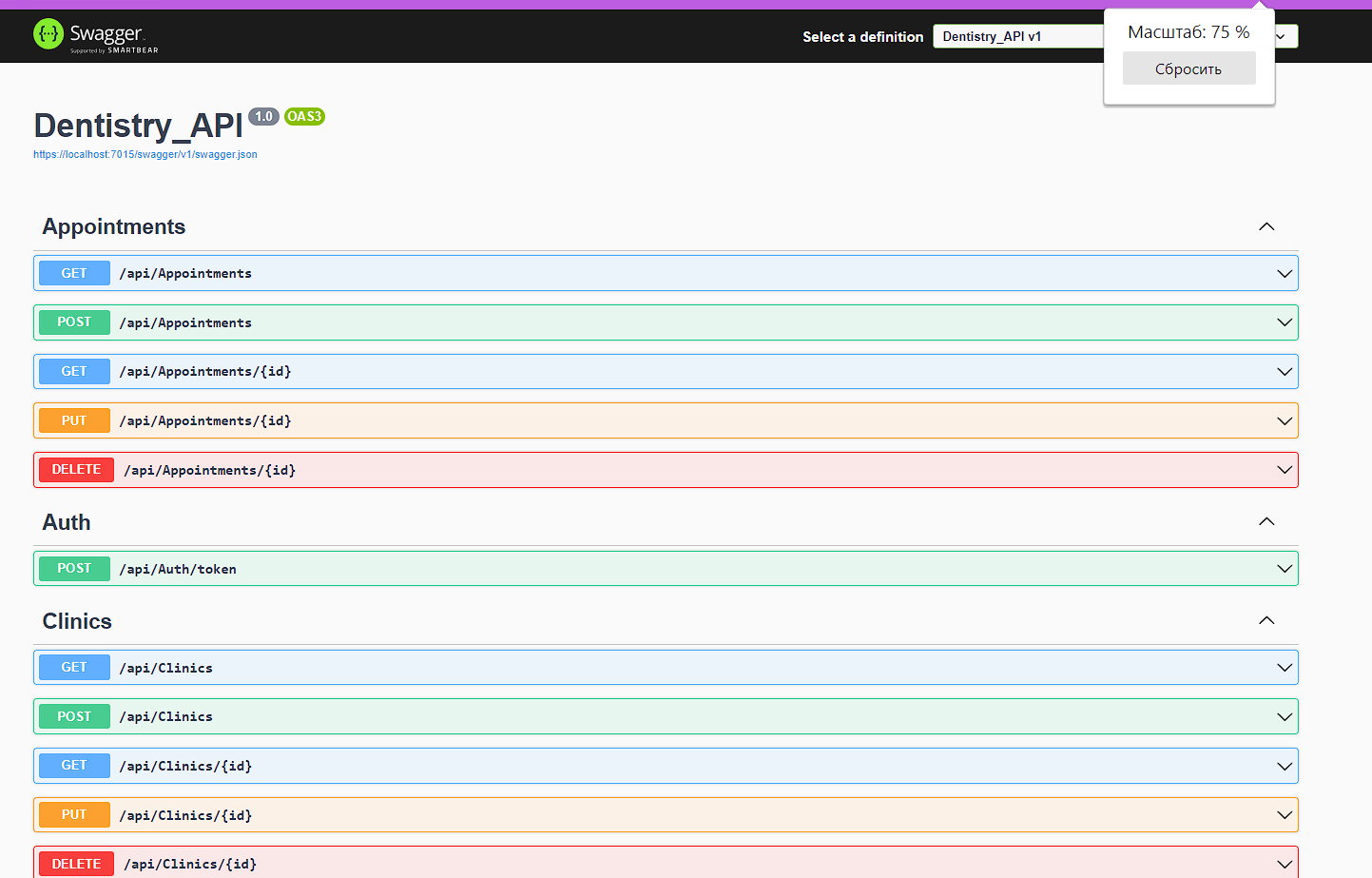


Рисунок – API

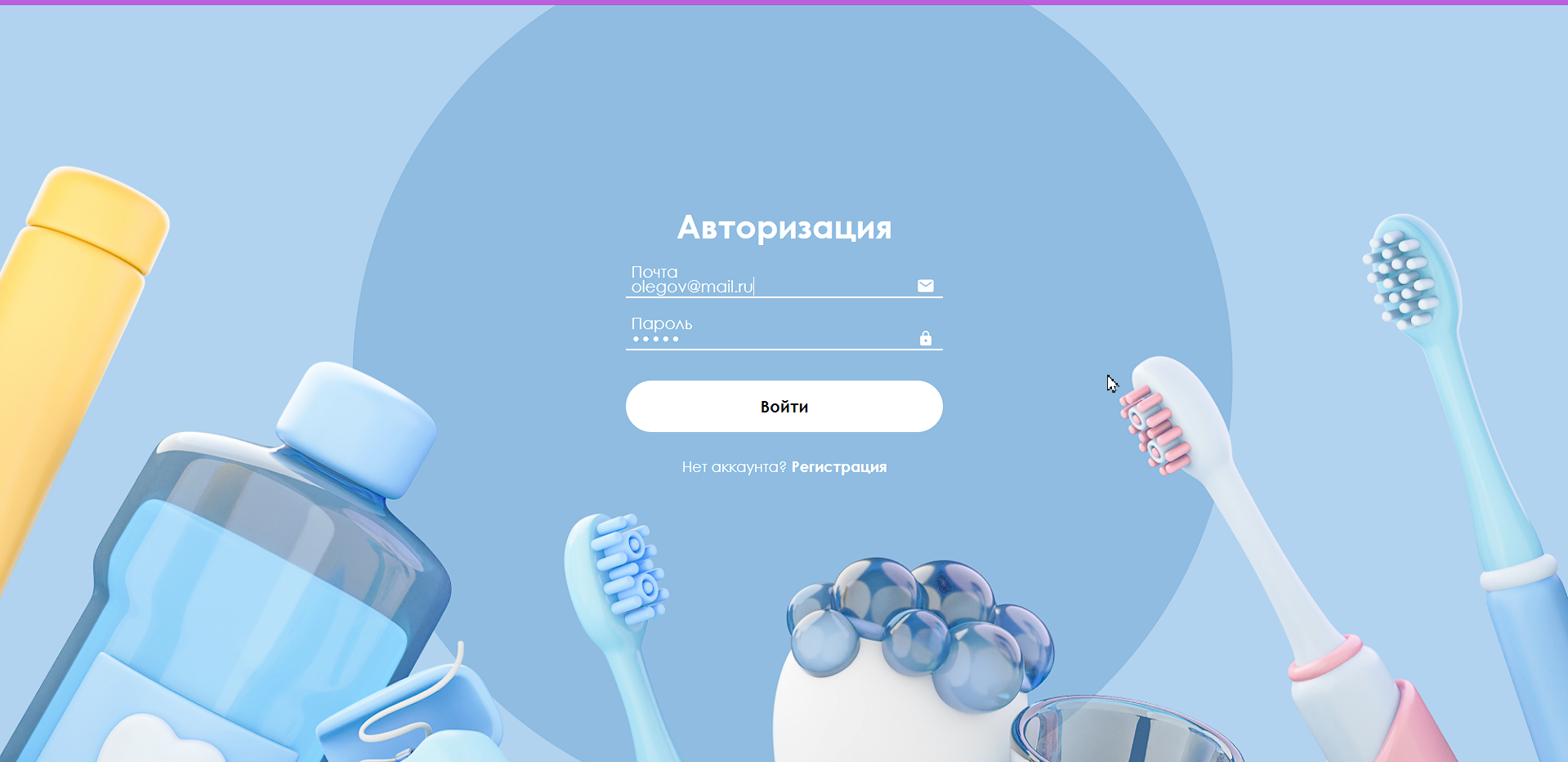


Рисунок – Авторизация

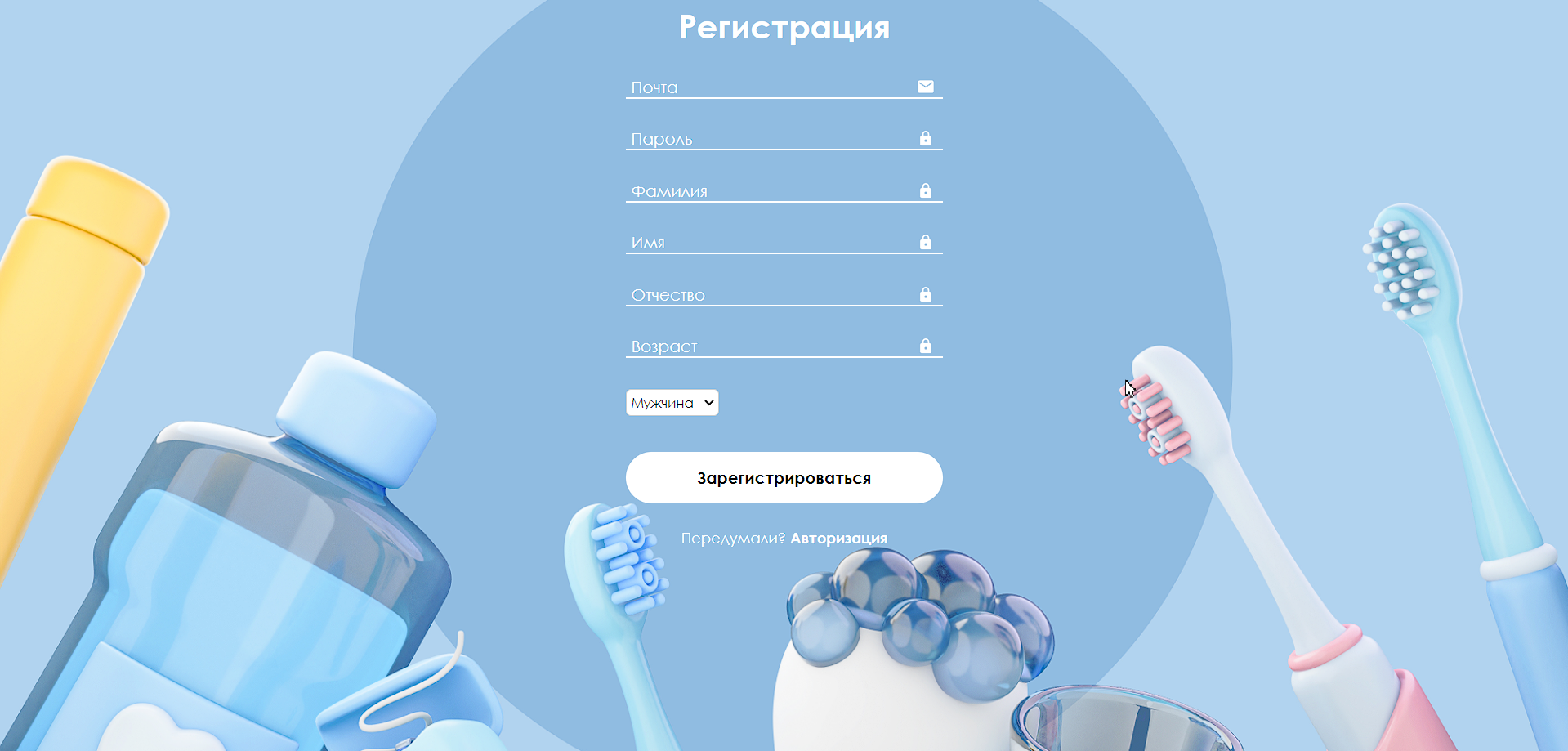


Рисунок – Регистрация

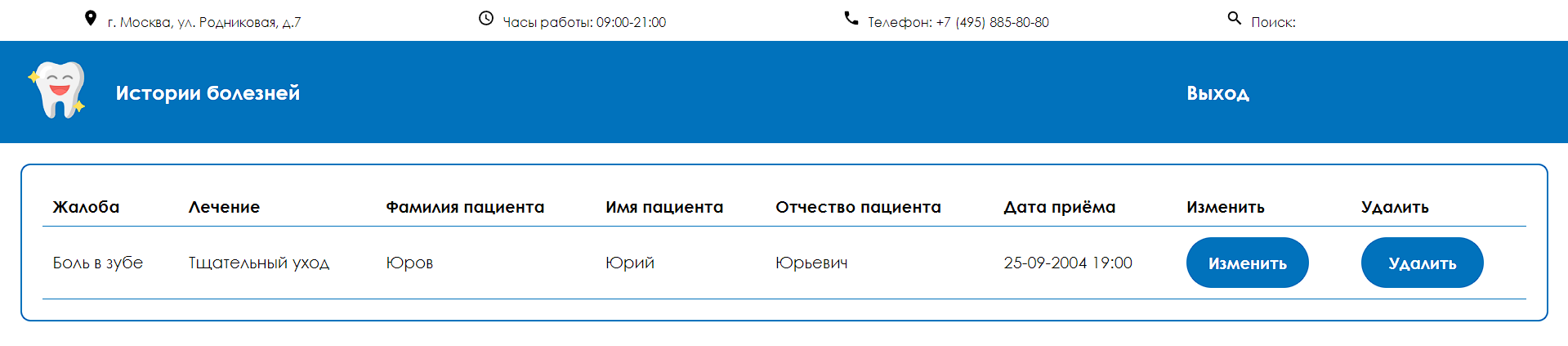


Рисунок – Главная страница Врача

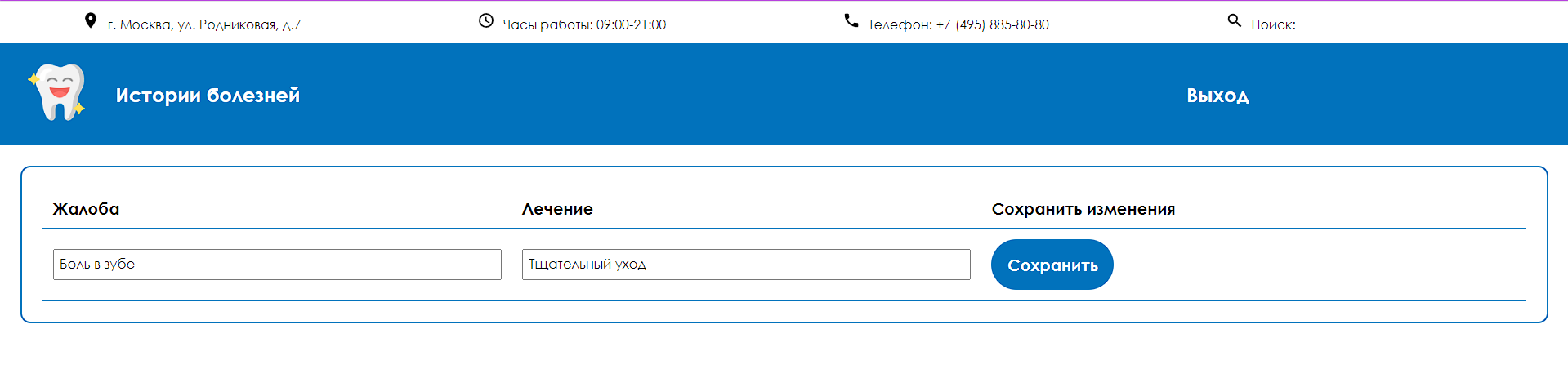


Рисунок – Изменение истории болезни

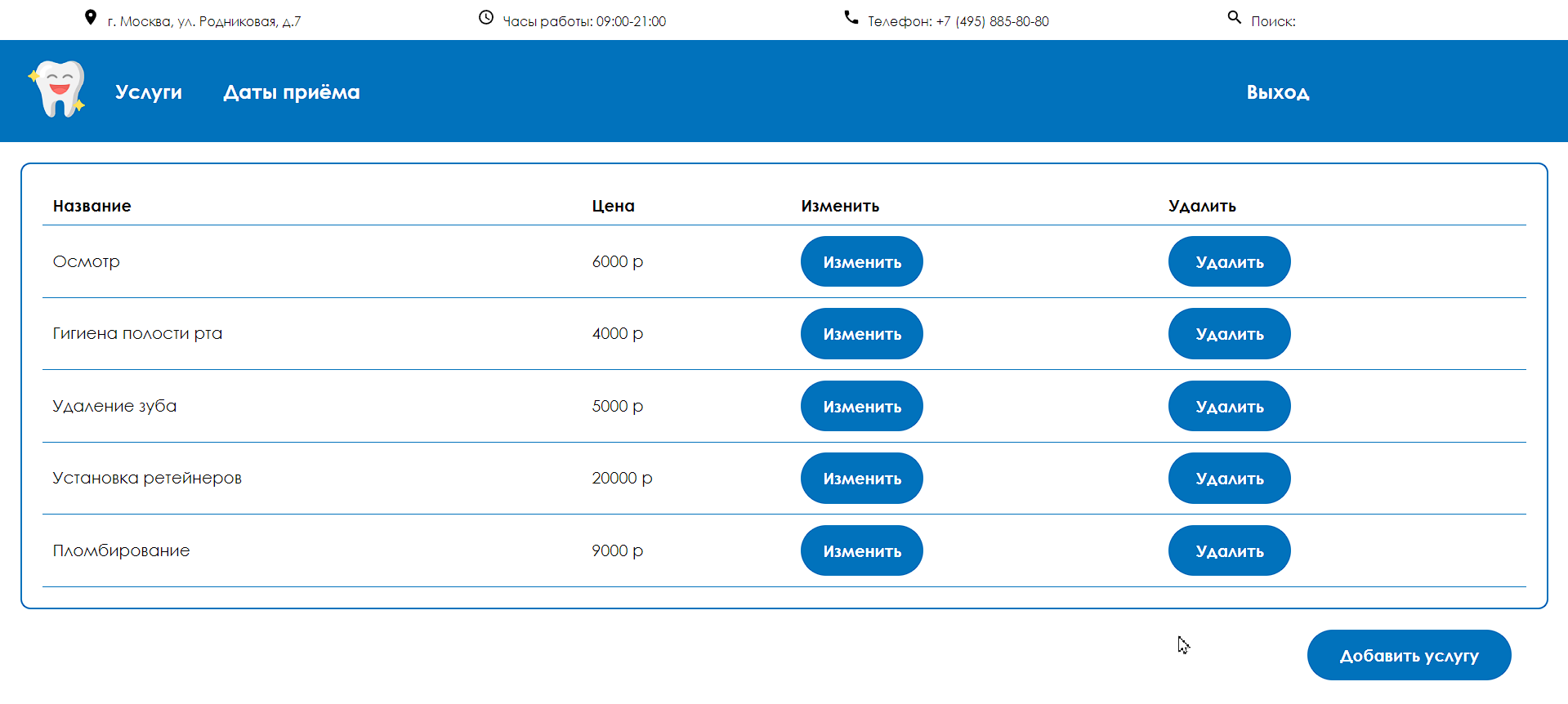


Рисунок – Окно услуг Администратора

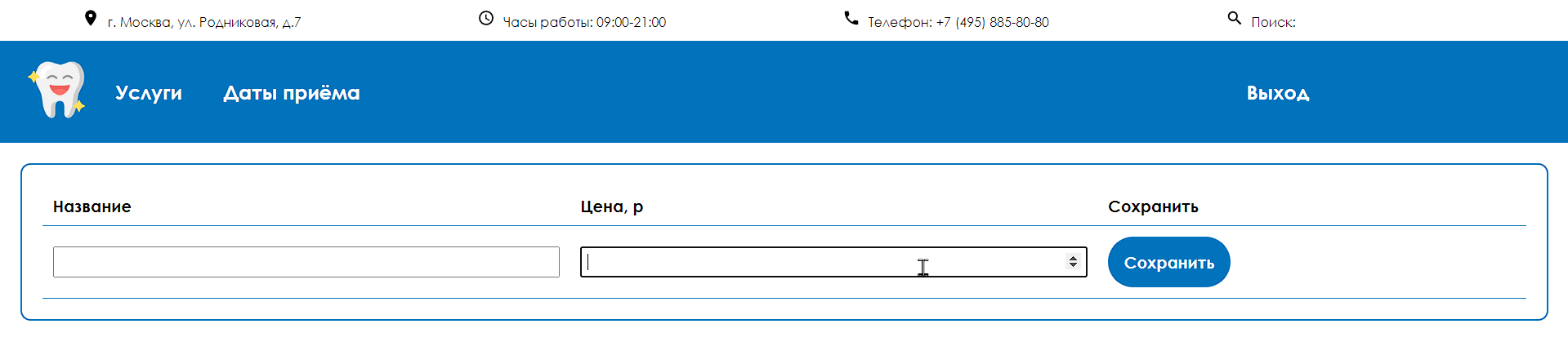


Рисунок – Добавление услуг

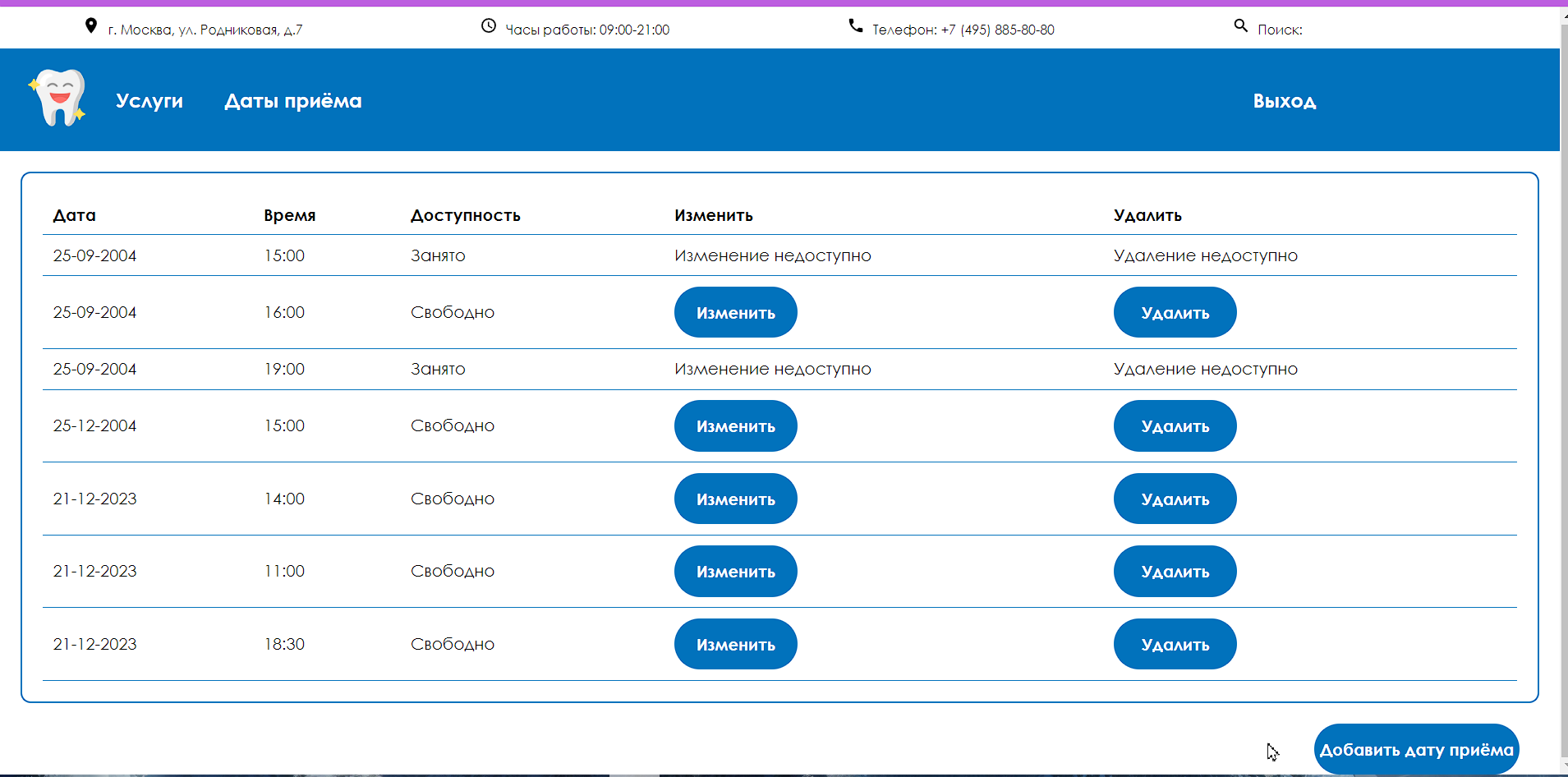


Рисунок – Окно дат приёмов Администратора

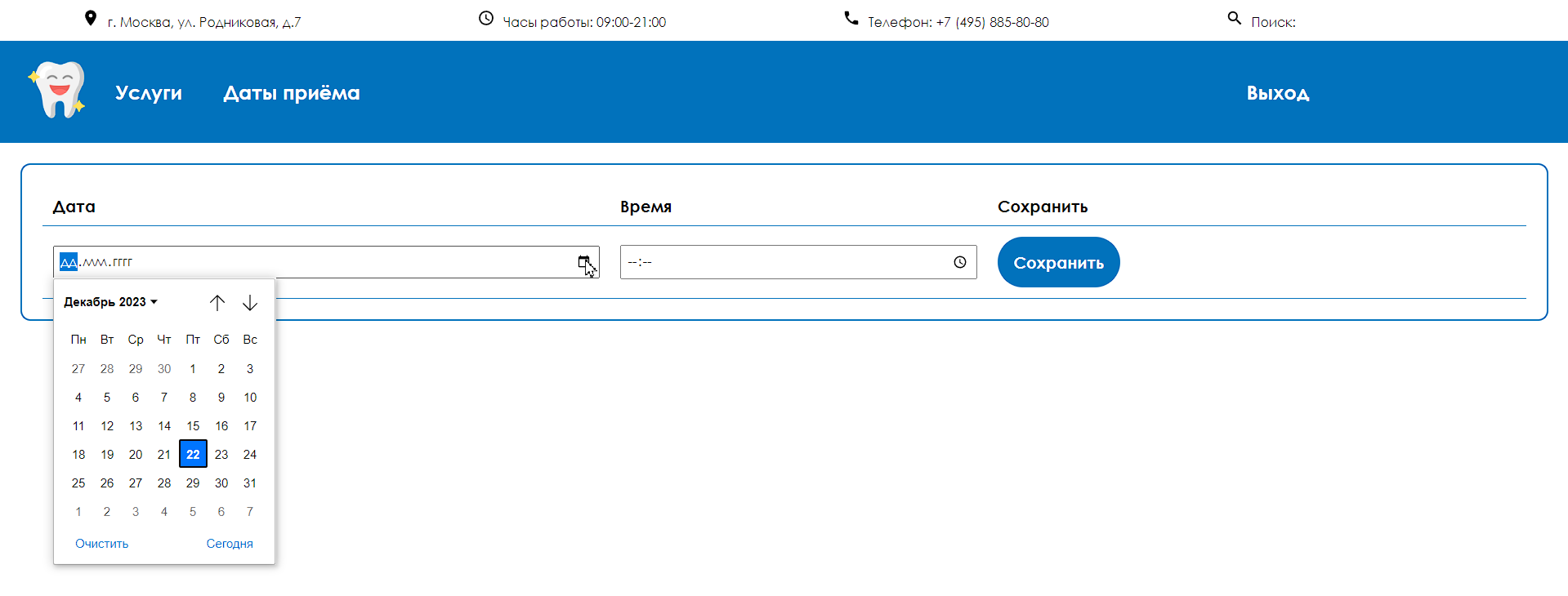


Рисунок – Добавление даты приёма

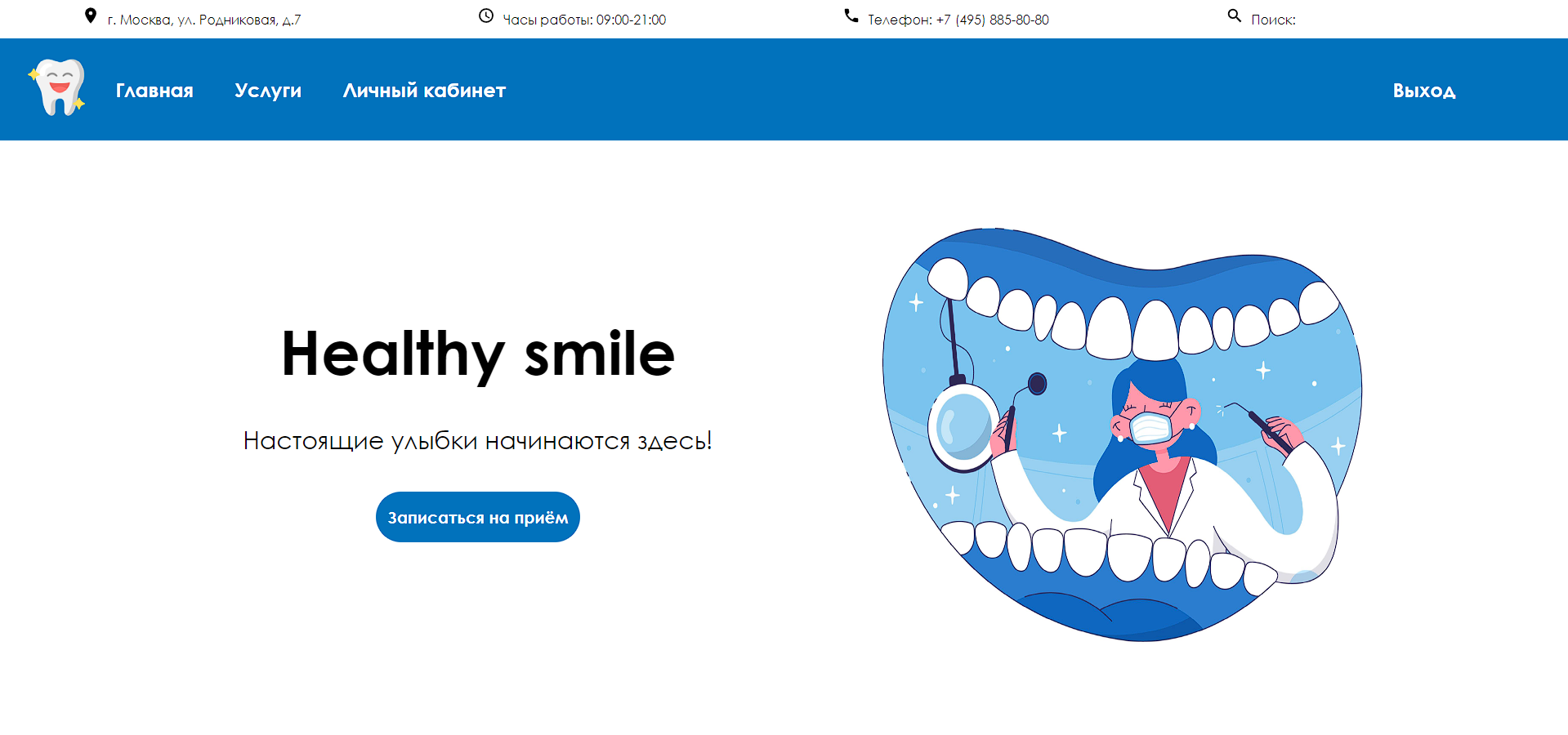


Рисунок – Главная страница Пациента

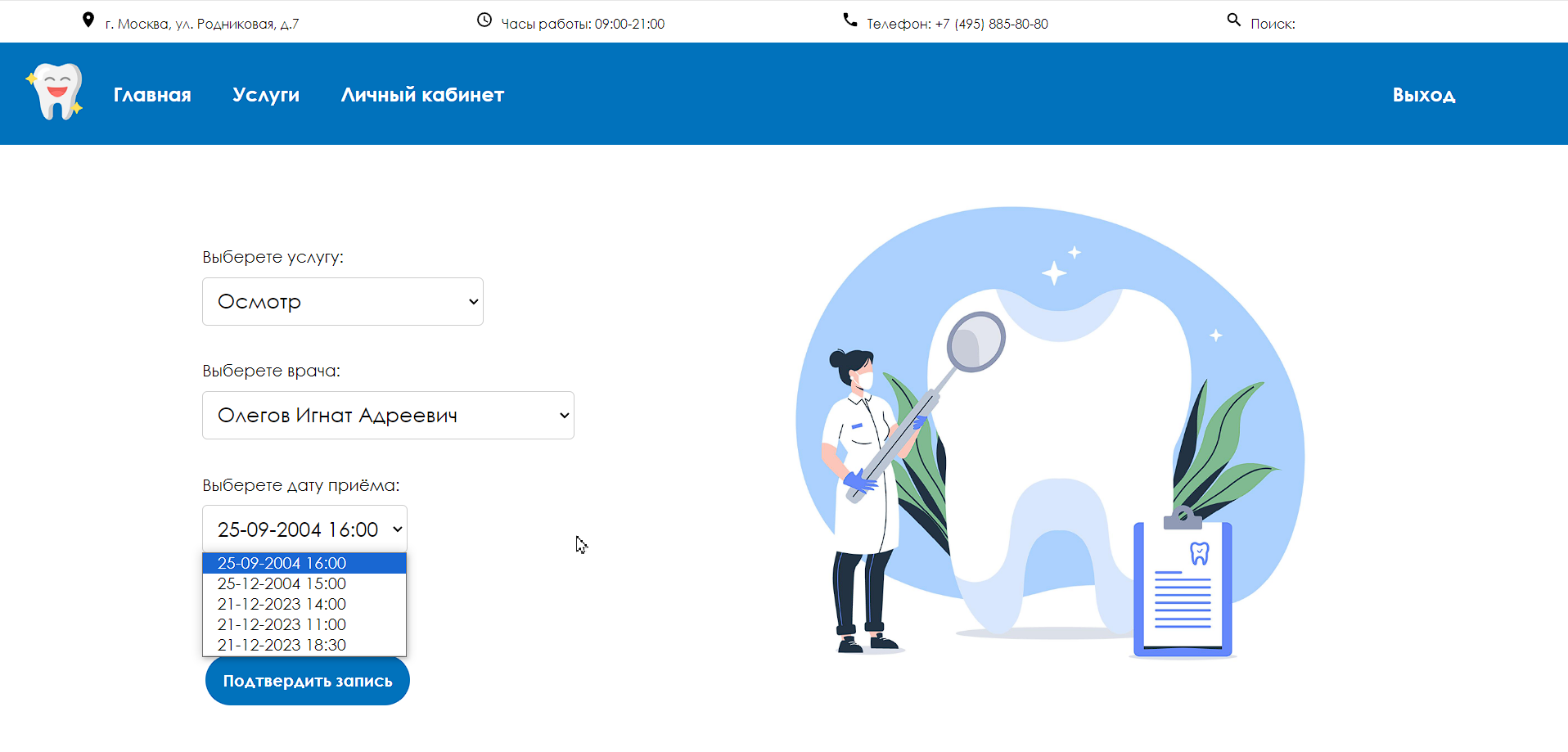


Рисунок – Страница записи пациента

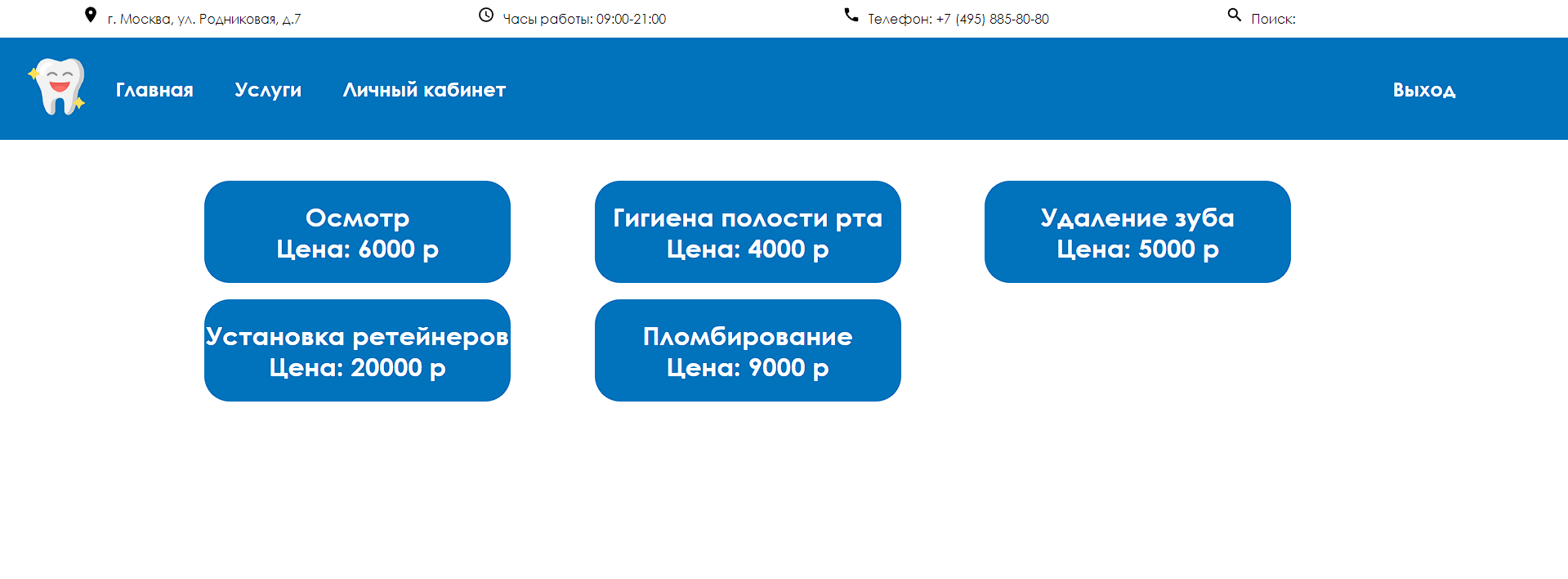


Рисунок – Страница услуг пациента

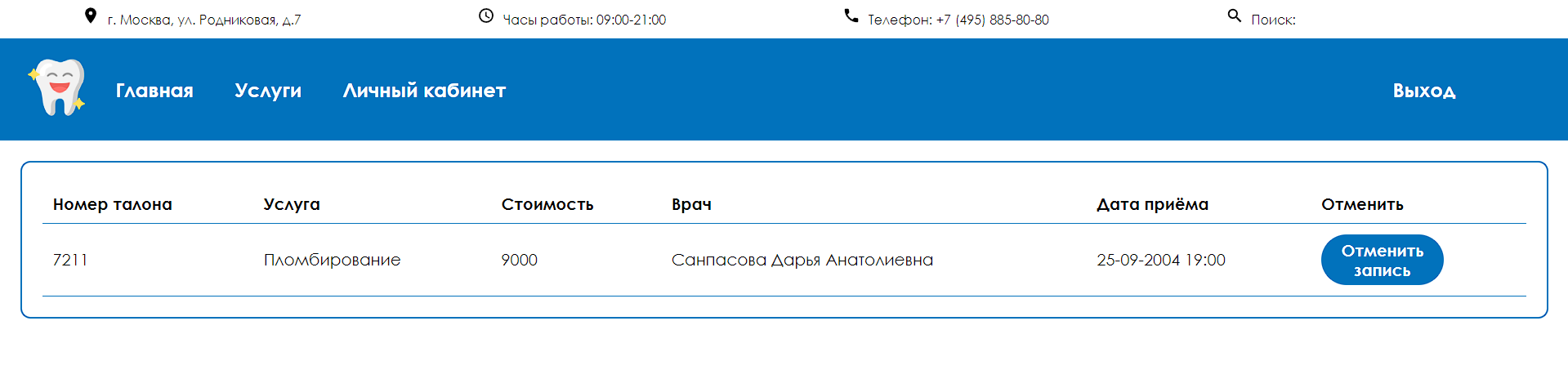


Рисунок – Страница профиля пациента

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Инструментальные средства

В качестве основы для базы данных было выбрано ядро MS SQL. Для формирования функций, таблиц и представлений применялась среда SQL Server Management Studio. MS SQL представляет собой широко используемую систему управления базами данных. Эта среда более богата функционалом, таким как транзакционность, полнотекстовый поиск и масштабирование для обработки больших объемов данных.

В качестве инструментов разработки а выбрана Microsoft Visual Studio 2022 для написания API и веб-приложения. Для разработки также использовался фреймворк ASP.NET, обеспечивающий преимущества архитектуры MVC. Эта архитектура предоставляет значительные преимущества по сравнению с другими средами разработки, поскольку может автоматически генерировать стандартные запросы к API. Visual Studio была выбрана также из-за возможности создания приложений для компьютеров под управлением операционной системы Windows.

## Отладка программы

Отладка приложения и API была осуществлена с помощью встроенных средств отладки в Visual Studio. Отладка использовалась при включенном приложении и приостанавливалась, доходя до точки остановки.

В ходе работы над проектом наиболее частой ошибкой был «Bad Request», возникающий из-за некорректных запросов к API

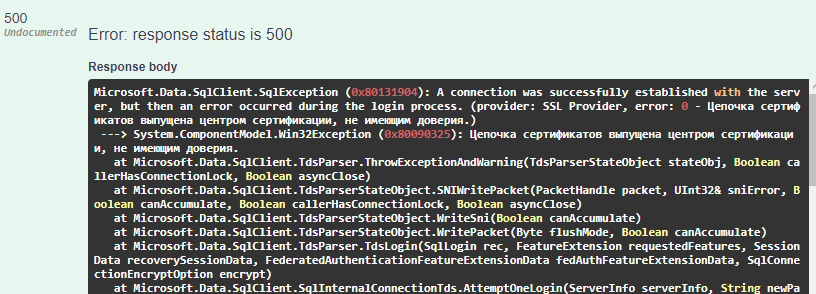


Рисунок - Bad Request

Для отслеживания отправляемых запросов в веб-приложении использовался вывод отправляемого на сервер json в консоль дебага. Со стороны пользователя же выводилось сообщение о неверном вводе

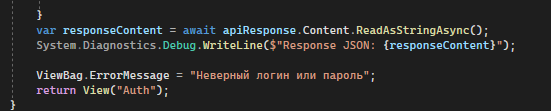


Рисунок – Код вывода json

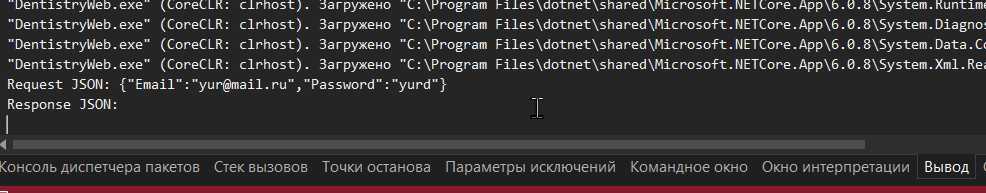


Рисунок – Вывод в консоль

## Защитное программирование

Использование механизма try-catch в программе и API обеспечивает выявление ошибок во время выполнения кода и принятие соответствующих мер по их обработке, что улучшает надежность и безопасность системы. В случае возникновения ошибок этот механизм позволяет изолировать проблемный участок кода и обрабатывать ошибку локально, не прерывая выполнение всего скрипта или приложения. Такой подход способствует улучшению производительности и снижению рисков сбоев.

Далее представлен пример его использования:

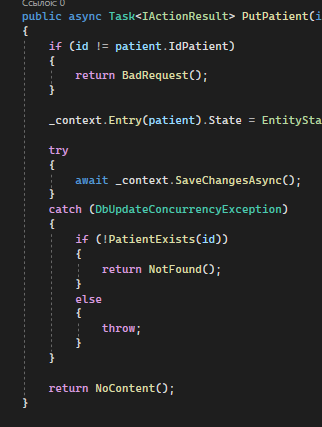


Рисунок - Блок try catch

Валидация была выполнена на стороне приложения для функции регистрации, для полей почты, пароля и имени пользователя. Ниже представлен скриншот с валидацией.

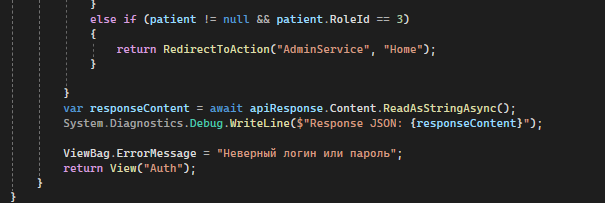


Рисунок 34 - Валидация данных

Так же для безопасности использовалось хеширования пароля

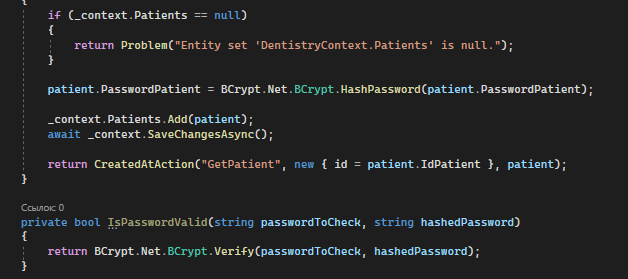


Рисунок - Методы хеширования пароля

## Характеристики программы.

Веб-приложение может быть запущено на операционных системах Windows 10 и выше при наличии браузера.

Характеристики программы представлены в приложении Б «Текст программы» в таблице 1 «Модули».

# ЗАКЛЮЧЕНИЕ

За время выполнения курсового проекта по теме "Управление пациентами в стоматологии" было успешно разработано приложение, предоставляющее эффективные средства для записи на приём, внесения изменений в историю болезни и добавления услуг, а также управления свободными датами для приёма.

В ходе проекта был осуществлен анализ требований и потребностей в области стоматологии, что послужило основой для создания информационной системы, обеспечивающей удобное и надежное взаимодействие между пациентами и медицинским персоналом. Была построена структура базы данных, учитывающая особенности хранения информации о пациентах, их истории болезни, услугах и свободных датах для записи.

Использование MS SQL Server и Visual Studio с применением фреймворка ASP.NET Core позволило эффективно реализовать функциональности, необходимые для управления пациентами в стоматологии. В приложении успешно реализованы механизмы записи на приём, изменения истории болезни, добавления услуг и управления доступными датами для приёма.

Процесс разработки включал в себя создание пользовательского интерфейса, обеспечивающего интуитивную навигацию и удобство использования для пациентов и медицинского персонала. Реализованные функциональности также поддерживают стандарты безопасности и надежности данных, что обеспечивает конфиденциальность информации о пациентах.

В ходе тестирования приложения были применены различные методы, включая ручное тестирование и проверку функциональности. Обнаруженные ошибки были успешно устранены, и приложение готово к применению на практике.

Разработанное приложение не только соответствует поставленным требованиям, но и предоставляет практические возможности для удобного и эффективного управления пациентами в стоматологии. Полученный опыт работы с базой данных, веб-технологиями и функциональными возможностями приложения представляет собой ценный ресурс для дальнейшей профессиональной деятельности и развития в области разработки информационных систем для здравоохранения.

# СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19404- 79 ЕСПД. Пояснительная записка. ПЕРЕИЗДАНИЕ Января 2010 г.
2. ГОСТ 7.80-2000 СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления.
3. ГОСТ Р 7.0.5-2008 БИБЛИОГРАФИЧЕСКАЯ ССЫЛКА. Общие требования и правила составления.
4. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
5. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. Сайт Microsoft [Электронный ресурс] <https://learn.microsoft.com/ru-ru/aspnet/core/data/ef-mvc/sort-filter-page?view=aspnetcore-7.0> (дата обращения)
8. Учебник HTML [Электронный ресурс] <https://www.basicweb.ru/html/razmetka.php> (дата обращения: 15.10.2023)
9. Официальный сайт Microsoft [Электронный ресурс] <URL:https://docs.microsoft.com/ru-RU/sql/ssms/download-sql-server> management-studio-ssms?view=aps-pdw-2016 (Дата обращения 18.03.2023)
10. Сайт FreePick [Электронный ресурс] <https://ru.freepik.com/free-vector/flat-dental-care-concept-illustration_15292171.htm#query=стоматология%20улыбка&position=1&from_view=search&track=ais&uuid=3a837ae1-a1ac-452f-af06-cfbd9701dbf9> (дата обращения: 03.11.23)
11. Сайт форум Stack Overflow [Электронный ресурс] URL: <https://stackoverflow.com/> (дата обращения 23.11.2023)
12. Сайт Основные понятия Grid Layot CSS [Электронный ресурс] <https://developer.mozilla.org/ru/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout> (дата обращения: 19.11.2023)