Частное учреждение образования

«Колледж бизнеса и права»

Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»

Пояснительная записка

к курсовому проекту по предмету

«Конструирование программ и языки программирования»

КП Т.197007.401

Руководитель проекта (А.В.Кривошеина)

Обучающейся (С.Е.Казутина)

2024

СОДЕРЖАНИЕ

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.197007.401

Разраб.

Казутина С.Е.

Провер.

Кривошеина А.В.

Реценз.

Н. Контр.

Утверд.

Лит.

Листов

КБП

60

у

[Введение 4](#_Toc170335992)

[1 Описание задачи 6](#_Toc170335993)

[1.1 Анализ предметной области 6](#_Toc170335994)

[1.2 Постановка задачи 7](#_Toc170335995)

[2 Проектирование веб-приложения 8](#_Toc170335996)

[2.1 Проектирование модели 8](#_Toc170335997)

[2.2 Требования к веб-приложению 11](#_Toc170335998)

[2.3 Структура веб-приложения 13](#_Toc170335999)

[2.4 Проектирование макета веб-приложения 15](#_Toc170336000)

[2.5 Программно-технические средства, необходимые для разработки приложения 19](#_Toc170336001)

[2.6 Защита и сохранность данных 20](#_Toc170336002)

[2.7 Организация и ведение информационной базы (модели) 21](#_Toc170336003)

[3 Реализация веб-приложения 27](#_Toc170336004)

[3.1 Описание разделов веб-приложения 27](#_Toc170336005)

[3.2 Разработка административной части приложения 28](#_Toc170336006)

[3.3 Разработка клиентской части приложения 29](#_Toc170336007)

[3.4 Описание используемых функций и процедур 30](#_Toc170336008)

[3.5 Функциональное тестирование 31](#_Toc170336009)

[4 Применение 35](#_Toc170336010)

[4.1 Назначение веб-приложения 35](#_Toc170336011)

[4.2 Программно-аппаратное обеспечение сервера и клиента 36](#_Toc170336012)

[Заключение 37](#_Toc170336013)

[Список использованных источников 38](#_Toc170336014)

[Приложение А Текст программных модулей 39](#_Toc170336015)

[Приложение Б Результаты работы программы 50](#_Toc170336016)

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*

# Введение

В наше время медицинские центры сталкиваются с рядом сложностей при организации процесса заказа талонов и заполнения карточек. Если эту задачу выполнять вручную, без подходящей системы, возникает риск ошибок и неэффективности. В таком случае создание веб-приложения для автоматизации заказа талонов и заполнения карточек в медицинском центре «MedicalClinic» является решением.

Благодаря веб-приложению, пациенты могут безопасно и удобно заказывать талоны и создавать карточки, минимизируя время ожидания и устраняя возможные ошибки, связанные с ручным вводом информации. Веб-приложение предоставляет пользователям простой и интуитивно понятный интерфейс, где они могут выбирать доступные временные слоты, виды приемов и заполнять необходимые данные.

Автоматизированная система также облегчает работу персонала медицинского центра. Ресурсы, которые ранее были затрачены на ручную обработку и управление заказами талонов, теперь могут быть перераспределены на более важные задачи. Персонал может мониторить и управлять талонами через веб-интерфейс, а также быстро получить доступ к необходимой информации о пациентах.

Целью разработки веб-приложения «Создание веб-приложения для автоматизации заказа талонов и карточек в медицинском центре» является обеспечение эффективного и удобного процесса заказа талонов и создания карточек для пациентов.

Актуальность создания данного приложения заключается в возможности просмотра информации о доступных временных слотах, выборе видов приемов и заполнении необходимых данных пациентов. Приложение также позволяет минимизировать время ожидания и предотвращает возможные ошибки, связанные с ручным вводом информации.

Пояснительная записка состоит из четырех разделов, в которых расписаны шаги проектирования, разработки, а также описание алгоритмов и входных\выходных данных.

В первом разделе «Описание задачи» описываются цели создания приложения и проектирование содержимого приложения. Проводится исследование предметной области, задачи, которые планируется решать посредством приложения, определяется целевая аудитория, описываются алгоритмы получения промежуточных и итоговых показателей, обоснование необходимости компьютерной обработки информации или процессов.

Во втором разделе «Проектирование веб-приложения» описываются средства защиты будущего веб-приложения, накладываемые ограничения на интерфейс, общие требования к интерфейсу (расположение, размер, форма, шрифт и прочее для всех элементов управления), требования к аппаратным и операционным ресурсам (операционная система, объем памяти устройства, доступ к интернету и его скорость, дополнительные устройства (мышь, клавиатура, принтер и т.д.)).

В третьем разделе «Реализации веб-приложения» перечисляются инструменты разработки, необходимые для разработки веб-приложения, порядок авторизации пользователей, организацию данных, а также применяемые технологии.

В четвертом разделе «Применение» описывается назначение приложения. Анализируется программное обеспечение, которое необходимо для функционирования веб-приложения.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

В списке информационных источников перечисляются информационные источники, которые были использованы в процессе разработки веб-приложения.

В приложении А представлен текст программных модулей веб-приложения.

В приложении Б представлены формы входных и выходных документов.

В приложении В представлен результаты работы веб-приложения.

# Описание задачи

## Анализ предметной области

Создание веб-приложения для автоматизации заказа талонов и карточек в медицинском центре «MedicalClinic» является предметной областью решаемой задачи. Оно предназначено для упрощения и оптимизации процесса заказа талонов и заполнения карточек для пациентов.

Автоматизированное веб-приложение «MedicalClinic» разработано для удобного управления и контроля процесса заказа талонов и заполнения карточек в медицинском центре. Оно включает набор инструментов и функций, которые помогают пациентам и персоналу эффективно справляться с задачами, связанными с записью на прием, выбором медицинских услуг и заполнением необходимых данных.

Работа медицинского центра «MedicalClinic» включает в себя подбор медицинских услуг в соответствии с потребностями клиентов и предоставление полной информации о доступных процедурах и возможных рисках. Основная задача медицинского персонала заключается в умении предоставить квалифицированную помощь пациентам и провести процедуры в соответствии с их потребностями. Каждый медицинский работник имеет свой список услуг, свое расписание, а также пациентов, с которыми он работает. Поэтому важно организовать отображение только соответствующих данных для каждого медицинского работника.

Автоматизация процесса заказа талонов и заполнения карточек в медицинском центре «MedicalClinic» позволяет значительно упростить процедуру записи на прием и заполнения карточек пациентов. Это обеспечивает удобство для клиентов и экономит время медицинского персонала.

Веб-приложение предлагает удобный пользовательский интерфейс для пациентов и персонала медицинского центра, а также функциональность управления талонами, услугами, расписаниями и карточками в базе данных.

Потенциальными пользователями веб-приложения являются медицинские центры, их руководители, которые желают отслеживать результаты работы своего персонала, а также частные лица, занимающиеся медицинской деятельностью и заинтересованные в автоматизации процесса заказа талонов и создания карточек.

## Постановка задачи

Функции, которые должны быть автоматизированы в соответствии с проанализированными бизнес-процессами и задачами медцентра «MedicalClinic»:

* веб-приложение должно позволять пользователю вводить данные о клиенте, такие как ФИО, номер телефона, адрес проживания, участок для пациента и ФИО для медицинского специалиста;
* ввод данных о медицинских услугах, таких как название услуги, тип, описание, показания, подготовку к услуге, выбор клиента, который заказывает данную услугу;
* отображение выбранных данных на пользовательском интерфейсе;
* сохранение введенных данных в таблице базы данных;
* автоматическое формирование талона с участием выбранного пользователя, с возможностью выбора даты и времени приема, а также даты и времени создания талона;
* составление талона в формате PDF для печати;
* интерфейс веб-приложения должен быть максимально прост и интуитивно понятен любому пользователю, что обеспечивает комфортное взаимодействие пациентов с программным средством.

Существующие аналоги с указанием отличий, которые будут реализованы в разрабатываемом программном средстве:

* «Талон.бай» – ­­­онлайн-сервис для автоматизации заказа талонов и записи к врачам в медицинских центрах. Позволяет пользователям вводить данные о пациенте, такие как ФИО, идентификационный номер паспорта, номер телефона, адрес проживания, а также выбирать желаемого врача и дату приема. Обеспечивает удобный интерфейс и быструю запись к специалистам. В отличие от разрабатываемого приложения не предоставляет функцию полноное описание оказываемых медицинских услуг.
* «МедицинскаяКарта+» – веб-приложение для электронного учета медицинских карт пациентов в клиниках. Позволяет вводить данные о пациенте, историю болезни, проведенные процедуры и назначения врачей. Обеспечивает удобную навигацию по медицинским картам и быстрый доступ к информации. Однако, не включает функцию заказа талонов, которую предлагает разрабатываемое приложение.

# Проектирование веб-приложения

## Проектирование модели

Исходя из исследования предметной области можно выделить следующие сущности проекта.

Каждый пациент, который был зарегистрирован в медицинской клинике, характеризуется следующими данными:

* ФИО пациента;
* дата рождения;
* адрес проживания;
* номер участка;
* номер телефона.

Каждая услуга, предоставляемая в медцентре, характеризуется следующими данными:

* название услуги;
* тип;
* описание;
* врач, предоставляющий услугу;
* кабинет.

Каждый медицинский специалист, который предоставляет услуги в медицинской клинике, характеризуется следующими данными:

* ФИО специалиста;
* расписание работы специалиста.

Каждая запись в медицинской карточке пациента характеризуется следующими данными:

* данные пациента;
* данные специалиста;
* диагноз;
* лечение;
* дата приема.

Каждый медицинский работник, который работает в медицинской клинике, характеризуется следующими данными:

* ФИО персонала.

Каждый участок медицинской клиники следующими данными:

* номер участка.
* врач, закрепленный за участком.

Каждый талон, заказанный пациентом либо работником клиники, характеризуется следующими данными:

* данные пациента;
* данные врача;
* данные услуги;
* дата и время приема;
* дата и время заказа талона.

Каждый заказ, составленный агентом, характеризуется следующими данными:

* данные покупателя;
* данные недвижимости;
* данные продавца;
* номер заказа;
* дата составления заказа;
* номер договора.

При работе с пациентами, медицинский специалист должен иметь возможность решать следующие задачи:

* добавление новых записей в карточку пациенту во время приема;
* добавление новых талонов пациенту;
* просмотр записей в карточке пациента;
* просмотр своего расписания.

Диаграмма «Сущность-связь» представлена на рисунке 2.1.1.

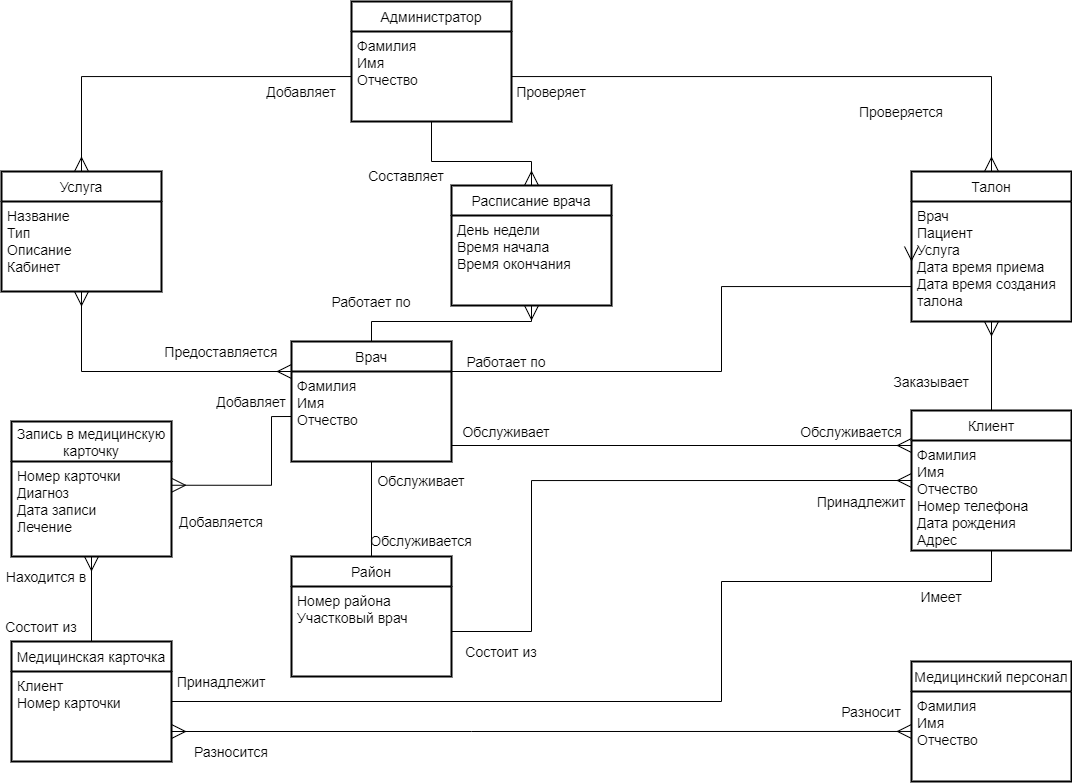


Рисунок 2.1.1 – Диаграмма «Сущность-связь» в нотации Мартина

Диаграмма вариантов использования (ДВИ) – это инструмент, используемый для моделирования функциональных требований системы. Они описывают взаимодействие внешних пользователей или других систем (актеров) с системой для достижения конкретных целей (вариантов использования). Данная диаграмма для разрабатываемого средства представлена на первом листе графической части.

Диаграмма классов – это диаграмма, которая используется для визуализации структуры системы путем отображения ее классов, атрибутов, методов и отношений между классами. Диаграмма классов для разрабатываемого средства представлена на втором листе графической части.

Диаграмма развертывания — это графическое представление архитектуры системы, которое показывает размещение компонентов на клиентской и серверной частях системы, а также связи между ними. Она предоставляет обзор структуры системы и помогает визуализировать развертывание компонентов на физических устройствах. Диаграмма развертывания представлена на третьем листе графической части.

Диаграмма последовательности – это еще один из типов диаграмм UML, который используется для представления взаимодействий между объектами в системе в хронологическом порядке, показывая, как объекты обмениваются сообщениями для выполнения функциональности. Диаграмма последовательности для функции составления заказа представлена на четвертом листе графической части.

Диаграмма состояний — это графическое представление поведения системы, которое моделирует различные состояния, переходы между ними и связи между состояниями. Она позволяет описать, как система изменяет свое состояние в ответ на внешние события или внутренние условия. Диаграмма состояний представлена на пятом листе графической части.

Алгоритмы, используемые в этом программном средстве, включают следующие шаги:

* ведение и редактирование базы данных пациентов: добавление, редактирование и удаление информации о пациентах, включая их персональные данные, медицинскую историю, текущие обращения и назначения.
* учет записей на прием к врачам: возможность пациентам заказывать талоны на прием к врачу, выбирать нужного специалиста, указывать предпочтительное время и дату приема;
* ведение расписания врачей: возможность просматривать и управлять записями пациентов.

## Требования к веб-приложению

Требования к стилистическому оформлению:

* веб-приложение должно иметь профессиональный и современный внешний вид, соответствующий медицинской тематике;
* цветовая палитра должна быть спокойной и гармоничной, с использованием нейтральных и пастельных оттенков;
* использование медицинских иконок и изображений будет соответствовать контексту и улучшит восприятие пользователей;
* интерфейс должен быть минималистичным и чистым, избегая излишней сложности и перегруженности элементами;
* стили и элементы должны быть единообразными на всех страницах приложения, чтобы обеспечить целостность и легкость восприятия.

Требования к графическому дизайну:

* графический дизайн должен быть согласован с общим стилем и цветовой палитрой приложения;
* использование простых и интуитивно понятных иконок и элементов управления поможет пользователям быстро ориентироваться;
* графические элементы должны быть оптимизированы для быстрой загрузки страниц и оптимального использования ресурсов.

Требования к шрифтовому оформлению:

* шрифты должны быть читабельными даже при малом размере и различных устройствах;
* рекомендуется использовать шрифты без засечек для лучшей читаемости;
* шрифты должны быть согласованы с общим стилем и дизайном приложения.

Требования к средствам просмотра:

* веб-приложение должно быть адаптивным и отзывчивым, чтобы корректно отображаться на различных устройствах, таких как компьютеры и ноутбуки;
* должна быть обеспечена возможность просмотра и использования приложения в разных браузерах, таких как Chrome, Firefox, Safari и т.д. Требования к контенту (наполнению):
* веб-приложение должно предоставлять пользователю возможность вести и редактировать базу данных пациентов, включая их персональные данные, медицинскую историю, текущие обращения и назначения;
* приложение должно предоставлять возможность пациентам заказывать талоны на прием к врачу, выбирать нужного специалиста, указывать предпочтительное время и дату приема;
* приложение должно предоставлять возможность просмотра и управления записями пациентов в расписании врачей.

Требования к системе управления контентом:

* приложение должно предоставлять возможность просмотра и управления записями пациентов в расписании врачей;
* система управления контентом должна обеспечивать безопасность данных и разграничение прав доступа к базе данных.

Требования к компоновке страниц:

* страницы приложения должны быть организованы логично и интуитивно понятно для пользователя;
* шапка, кнопки, формы и панели инструментов должны быть разработаны таким образом, чтобы обеспечить легкий и быстрый доступ к основным функциям приложения;
* размещение элементов интерфейса должно быть продуманным и удобным для использования на различных устройствах.

## Структура веб-приложения

Ниже приведено представление логической структуры веб-приложения для автоматизации работы медицинской клиники:

Исходные данные:

* справочник моделей;
* клиентская база;
* информация о талонах на услуги и приемы.

Разграничение прав доступа:

* аутентификация и авторизация пользователей;
* управление ролями и правами доступа.

Ведение и редактирование базы данных пациентов:

* добавление, редактирование и удаление информации о пациентах;
* управление персональными данными;
* ведение медицинской истории;
* запись текущих обращений и назначений.

Учет записей на прием к врачам:

* заказ талонов на прием к врачу;
* выбор нужного специалиста;
* указание предпочтительного времени и даты.

Ведение расписания врачей:

* просмотр и управление записями пациентов;
* добавление, редактирование и удаление записей.

Организация лаконичного интерфейса программного средства:

* создание меню, кнопочных форм, панелей инструментов;
* разработка удобного и понятного интерфейса.

Реализация иерархии классов:

* определение классов и связей между ними;
* разработка логики работы классов.

Работа с информацией:

* хранение данных в базе данных;
* запросы и обработка данных;
* взаимодействие с внешними системами.

Создание справочной системы:

* разработка документации и руководств;
* создание справочных ресурсов;
* предоставление информации и поддержка.

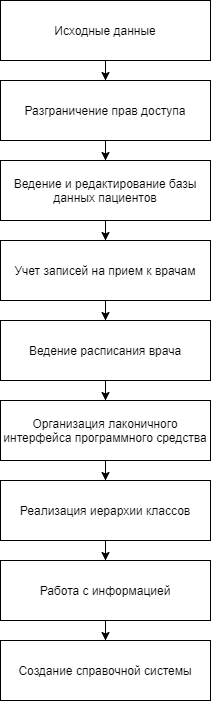
На рисунке 2.3.1 представлена структурная схема веб-приложения в виде диаграммы fire flow с указанием взаимосвязи структурных блоков:

Рисунок 2.3.1 – Структурная схема приложения в виде диаграммы fire flow

## Проектирование макета веб-приложения

Для разработки веб-приложения для автоматизации работы медицинской клиники рекомендуется использовать блочный тип дизайна. Блочный дизайн предоставляет гибкость и адаптивность, что является важным для веб-приложений, которые будут использоваться на различных устройствах и экранах. Блочный дизайн позволит легко управлять размещением элементов на странице, а также обеспечит хорошую читабельность контента и удобство использования.

Содержимое страницы авторизации:

* текстовое поле «Логин» будет позволять ввести логин аккаунта;
* текстовое поле «Пароль» будет позволять ввести пароль от аккаунта;
* кнопка «Войти» будет осуществлять проверку введенных данных и вход в аккаунт (переход в профиль пользователя);
* кнопка «Создать аккаунт»

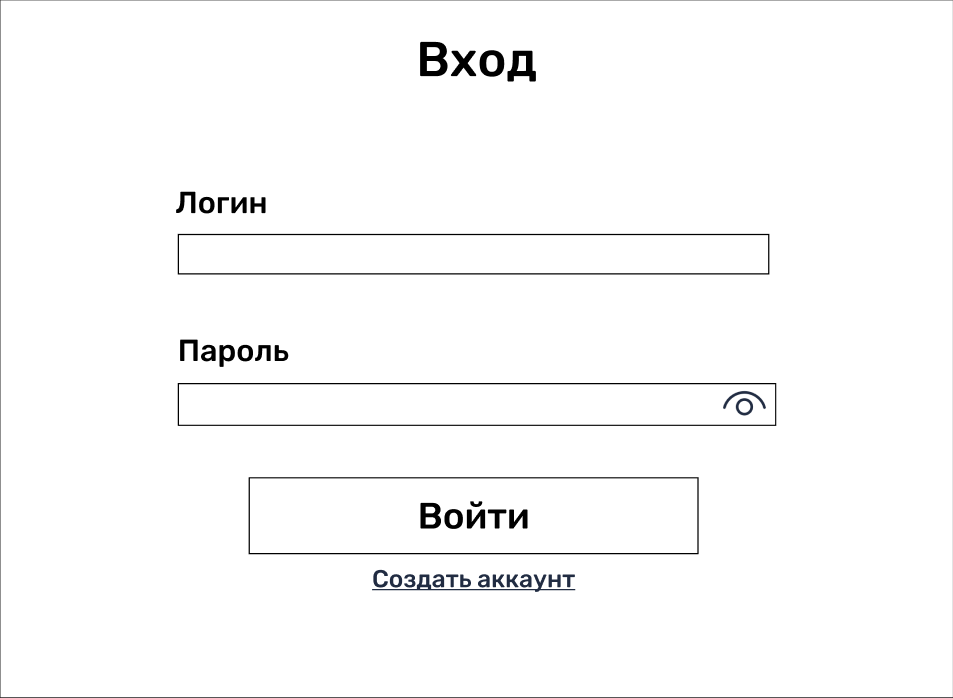
На рисунке 2.4.1 представлен прототип окна авторизации.

Рисунок 2.4.1 – Макет страницы авторизации

Содержимое главной страницы:

* шапка страницы с кнопками для перехода по основным страницам;
* блок с описанием медицинского цента;
* блок с описанием преимуществ медицинской клиники;
* блок с перечнем основных услуг;
* подвал страницы с перечнем всех услуг, контактной информацией, временем работы.

На рисунке 2.4.2 представлен макет главной страницы веб-приложения.



Рисунок 2.4.2 – Макет главной страницы

Содержимое страницы профиля пациента:

* шапка страницы с кнопками для перехода по основным страницам;
* текстовое поле «Имя», которое будет выводить имя зарегистрированного пользователя;
* текстовое поле «Фамилия», которое будет выводить фамилию зарегистрированного пользователя;
* текстовое поле «Отчество», которое будет выводить отчество зарегистрированного пользователя;
* текстовое поле «Дата рождения», которое будет выводить дату рождения зарегистрированного пользователя;
* текстовое поле «Номер телефона», которое будет выводить номер телефона зарегистрированного пользователя;
* текстовое поле «Адрес проживания», которое будет выводить адрес зарегистрированного пользователя;
* текстовое поле «Участок», которое будет выводить участок зарегистрированного пользователя;
* блок назначенных услуг выводящий информацию о талонах пользователя.

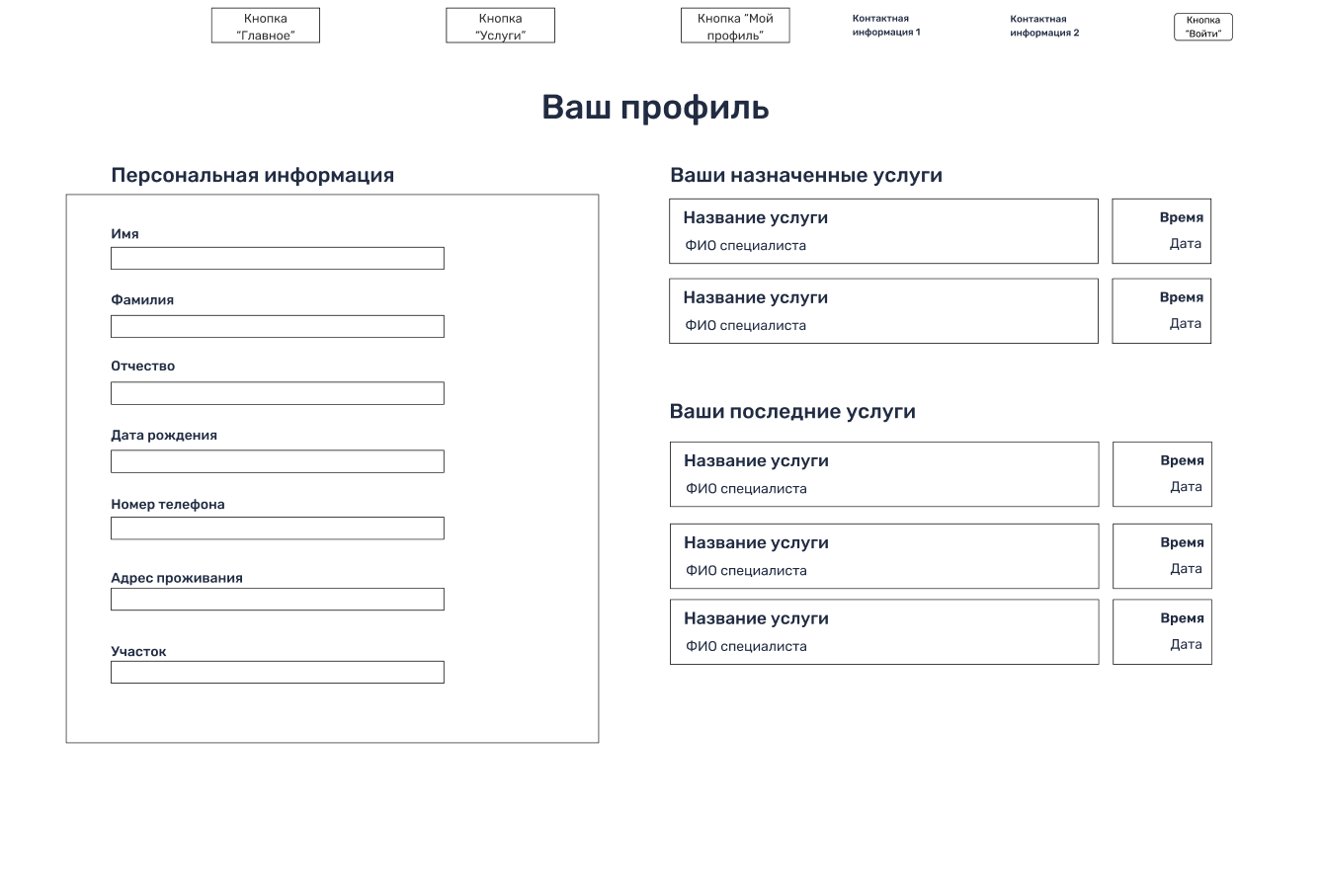
На рисунке 2.4.3 представлен макет страницы профиля пациента.

Рисунок 2.4.3 – Макет страницы профиля пациента

Содержимое страницы выбранной услуги:

* шапка страницы с кнопками для перехода по основным страницам;
* блок «Выбранная услуга», который будет выводить название выбранной услуги;
* блок «Описание услуги», который будет выводить описание выбранной услуги;
* блок «Календарь», который будет выводить календарь для записи на выбранную услугу.

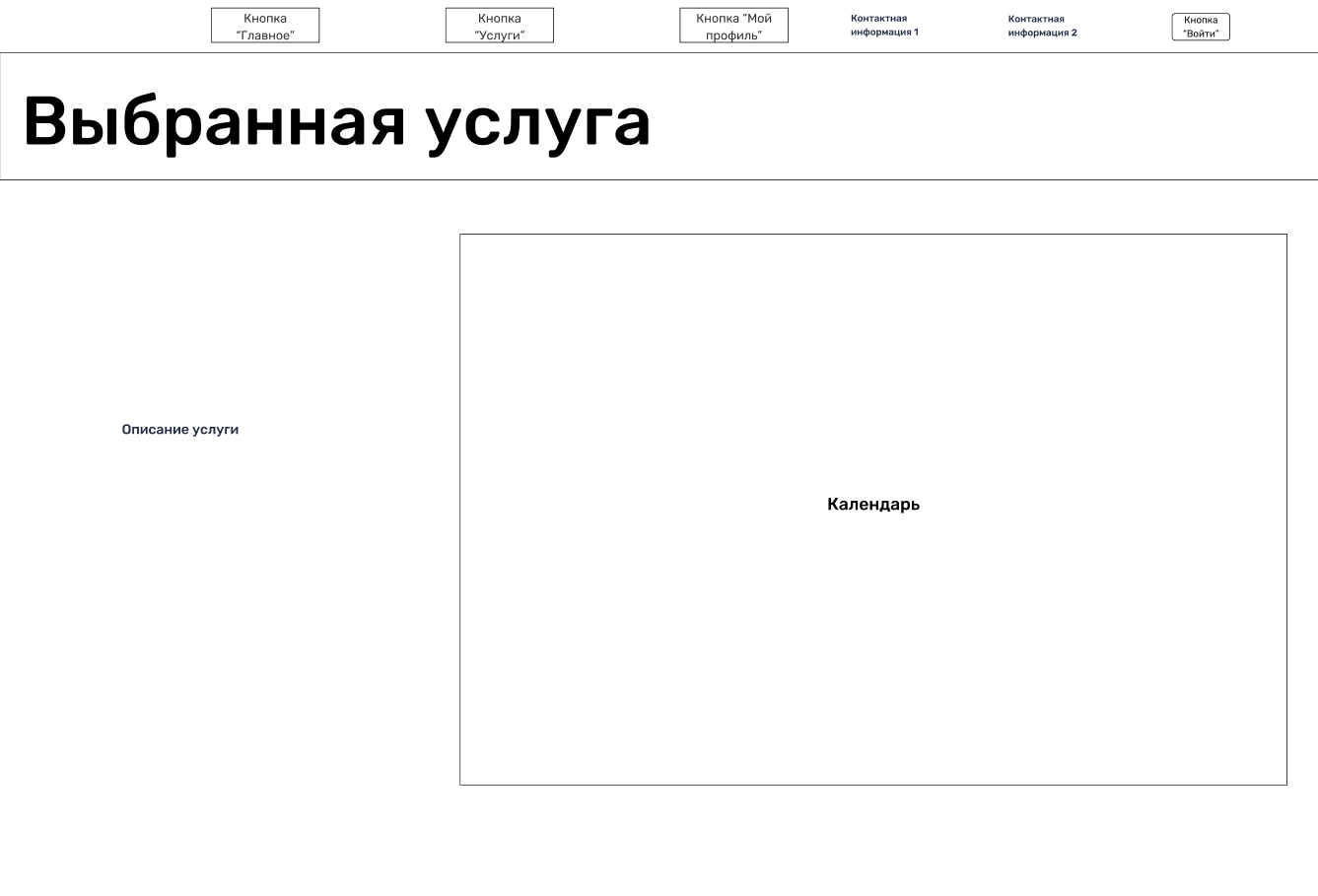
На рисунке 2.4.4 представлен макет страницы выбранной услуги.

Рисунок 2.4.4 – Макет страницы выбранной услуги

Содержимое страницы профиля медицинского специалиста:

* шапка страницы с кнопками для перехода по основным страницам;
* кнопка «Редактировать профиль», которая будет перенаправлять на страницу редактирования профиля специалиста;
* кнопка «Просмотр расписания», которая будет перенаправлять на страницу просмотра расписания специалиста;
* кнопка «Добавить запись в карточку», которая будет перенаправлять на страницу добавления записи в карточку пациента;
* кнопка «Редактировать запись в карточке», которая будет перенаправлять на список всех записей в карточках для дальнейшего редактирования;
* кнопка «Просмотреть участки», которая будет перенаправлять на страницу участков;
* кнопка «Записать пациента на услугу», которая будет перенаправлять на страницу создания талона;
* блок «Календарь», который будет выводить календарь для просмотра записей специалиста и создания новых записей.

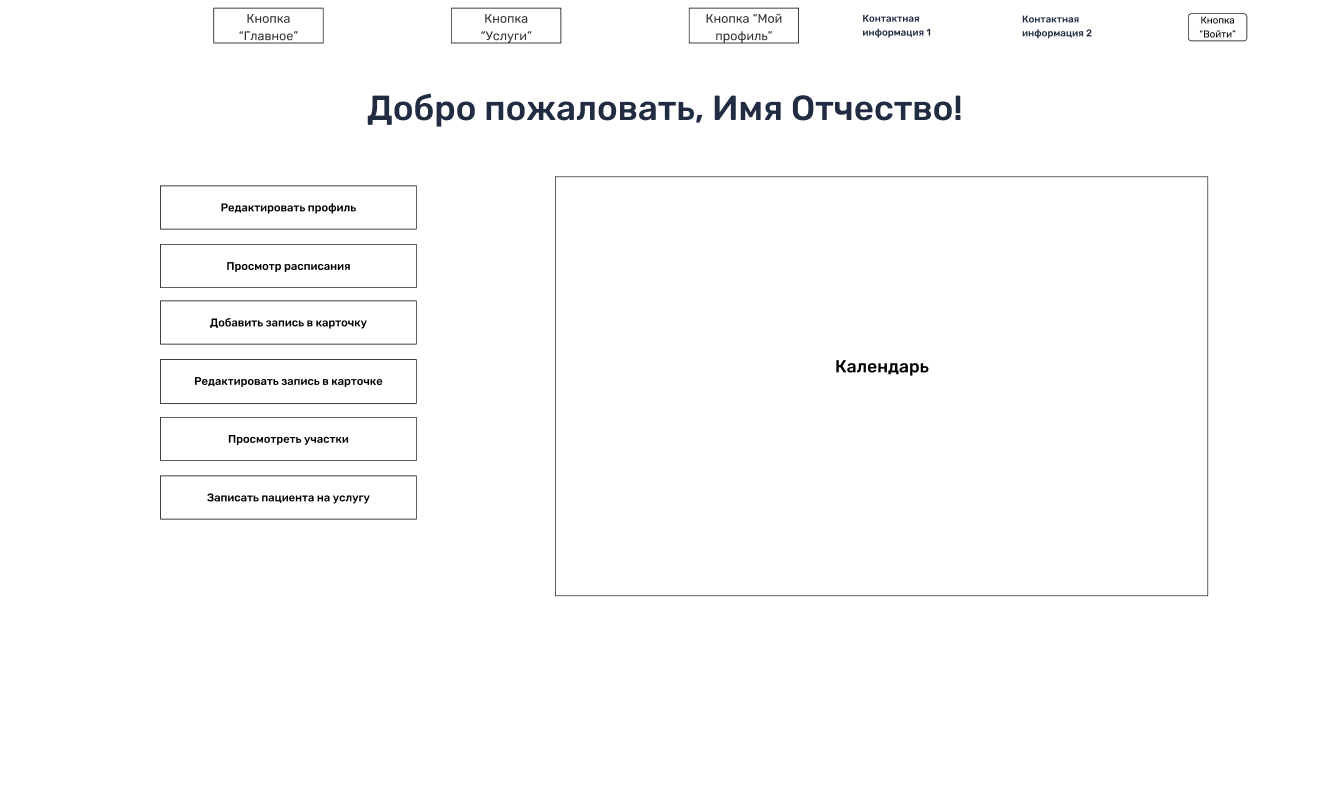
На рисунке 2.4.5 представлен макет страницы профиля специалиста.

Рисунок 2.4.5 – Макет страницы профиля специалиста

## Программно-технические средства, необходимые для разработки приложения

Анализируя предметную область и рассмотрев функциональные возможности приложения, было принято решение использовать следующие инструменты разработки.

В качестве интегрированной среды разработки будет использоваться Microsoft Visual Studio. Это позволит эффективно создавать, отлаживать и собирать приложение, а также использовать широкий спектр инструментов и функций, доступных в Visual Studio.

Для разработки приложения будет использоваться язык C# с помощью ASP.NET Core и Razor Pages. ASP.NET Core предоставляет мощный фреймворк для создания веб-приложений, а Razor Pages обеспечивает удобную модель программирования для создания интерфейса.

Для работы с базой данных будет использоваться SQL Server, а для управления базой данных — SQL Server Management Studio. SQL Server обладает мощными возможностями хранения и управления данными, а SQL Server Management Studio предоставляет удобный интерфейс для администрирования базы данных.

Для разработки веб-интерфейса приложения будут использоваться HTML, CSS и JavaScript. HTML обеспечивает структуру страницы, CSS — визуальное оформление, а JavaScript — динамическое взаимодействие с пользователем.

Для работы с данными в приложении будет использоваться Entity Framework, который обеспечивает удобное взаимодействие с базой данных SQL Server. Кроме того, для выполнения запросов к данным будет использоваться язык запросов LINQ.

Применение этих технологий обусловлено их совместимостью, эффективностью и удобством использования при разработке веб-приложений, требующих работы с данными и поддержки сети. Использование этих инструментов позволит создать функциональное и удобное в использовании веб-приложение с сетевой поддержкой и минимальными затратами времени и ресурсов.

## Защита и сохранность данных

В контексте разработки веб-приложения для автоматизации работы медицинской клиники, обеспечение защиты и сохранности данных является критически важным аспектом. Ниже представлено описание мер, которые можно принять для обеспечения безопасности информации.

Ограничение доступа к данным:

* разграничение прав доступа: веб-приложение должно иметь механизм аутентификации и авторизации, который позволяет управлять доступом пользователей к различным частям приложения и данным. Разные роли пользователей (администраторы, врачи, персонал, пациенты) должны иметь различные уровни доступа и функциональные возможности;
* управление доступом на уровне базы данных: должны быть применены механизмы управления доступом на уровне базы данных, такие как роли и разрешения, чтобы ограничить доступ к конкретным таблицам и данным.

Защита информации от несанкционированного использования:

* шифрование данных: критически важные данные, такие как персональные данные пациентов и медицинская история, должны быть зашифрованы в базе данных и при передаче по сети. Это поможет предотвратить несанкционированный доступ к данным, даже если злоумышленник получит физический или сетевой доступ к системе;
* защита от SQL-инъекций: необходимо применять параметризованные запросы или использовать ORM (Object-Relational Mapping) для взаимодействия с базой данных, чтобы предотвратить атаки SQL-инъекций.

Механизмы защиты:

* резервное копирование данных: регулярное создание резервных копий базы данных поможет защитить данные от потери в случае сбоев или чрезвычайных ситуаций;
* защита от SQL-инъекций: необходимо применять параметризованные запросы или использовать ORM (Object-Relational Mapping) для взаимодействия с базой данных, чтобы предотвратить атаки SQL-инъекций.
* мониторинг и журналирование: важно осуществлять мониторинг и журналирование активности пользователей, чтобы обнаруживать подозрительную активность или нарушения безопасности;
* обновление и патчи: регулярное обновление операционной системы, фреймворка ASP.NET Core и других компонентов приложения позволит исправить уязвимости безопасности и предотвратить эксплуатацию известных уязвимостей.

## Организация и ведение информационной базы (модели)

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами. Структура базы данных разрабатываемого программного средства включает девять таблиц.

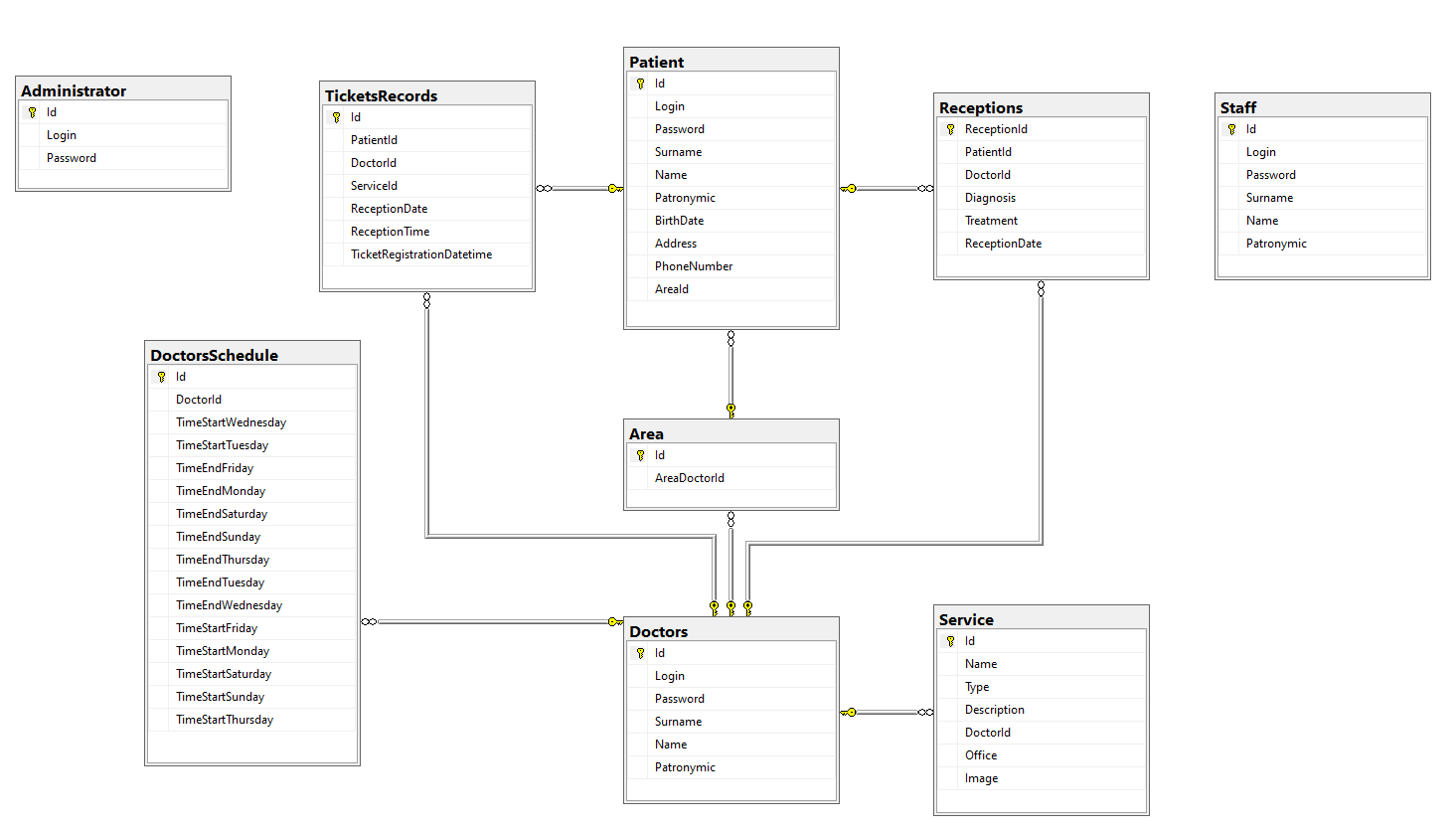
На рисунке 2.7.1 представлена схема базы данных.

Рисунок 2.7.1 – Схема базы данных

Структура данных таблиц и их краткое описание приводится в таблицах 2.7.1-2.7.9.

Таблица «Administrators» хранит информацию о всех администраторах. Структура приведена в таблице 2.7.1.

Таблица 2.7.1 – Структура таблицы «Administrator»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | INTEGER | 8 | Идентификационный номер |
| Login | nvarchar(60) | 120 | Логин пользователя |
| Password | nvarchar(100) | 200 | Пароль пользователя |

Таблица «Area» хранит информацию о всех районах. Структура приведена в таблице 2.7.2.

Таблица 2.7.2 – Структура таблицы «Area»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер |
| AreaDoctorId | int | 4 | Идентификационный номер врача участка |

Таблица «Doctors» хранит информацию о врач, структура приведена в таблице 2.7.3.

Таблица 2.7.3 – Структура таблицы «Doctors»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер |
| Login | nvarchar(60) | 120 | Логин врача |
| Password | nvarchar(100) | 200 | Пароль врача |
| Surname | nvarchar(60) | 120 | Фамилия врача |
| Name | nvarchar(60) | 120 | Имя врача |
| Patronymic | nvarchar(60) | 120 | Отчество врача |

Таблица «DoctorsSchedule» хранит информацию расписаний врачей, структура приведена в таблице 2.7.4.

Таблица 2.7.4 – Структура таблицы «DoctorsSchedule»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер расписания |
| DoctorId | int | 4 | Идентификационный номер врача |
| TimeStartWednesday | time(7) | 5 | Время начала приема в среду |
| TimeStartTuesday | time(7) | 5 | Время начала приема во вторник |
| TimeEndFriday | time(7) | 5 | Время окончания приема в пятницу |
| TimeEndMonday | time(7) | 5 | Время окончания приема в понедельник |
| TimeEndSaturday | time(7) | 5 | Время окончания приема в субботу |
| TimeEndSunday | time(7) | 5 | Время окончания приема в воскресенье |

Продолжение таблицы 2.7.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| TimeEndThursday | time(7) | 5 | Время окончания приема в четверг |
| TimeEndTuesday | time(7) | 5 | Время окончания приема во вторник |
| TimeEndWednesday | time(7) | 5 | Время окончания приема в среду |
| TimeStartFriday | time(7) | 5 | Время начала приема в пятницу |
| TimeStartMonday | time(7) | 5 | Время начала приема в понедельник |
| TimeStartSaturday | time(7) | 5 | Время начала приема в субботу |
| TimeStartSunday | time(7) | 5 | Время начала приема в воскресенье |
| TimeStartThursday | time(7) | 5 | Время начала приема в четверг |

Таблица «Patient» хранит информацию о пациентах, структура приведена в таблице 2.7.5.

Таблица 2.7.5 – Структура таблицы «Patient»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер |
| Login | nvarchar(60) | 120 | Логин пользователя |
| Password | nvarchar(100) | 200 | Пароль пользователя |
| Surname | nvarchar(60) | 120 | Фамилия пользователя |
| Name | nvarchar(60) | 120 | Имя пользователя |
| Patronymic | nvarchar(60) | 120 | Отчество пользователя |

Продолжение таблицы 2.7.5

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| BirthDate | datetime2(7) | 8 | Дата рождения пользователя |
| Address | nvarchar(100) | 200 | Адрес пользователя |
| PhoneNumber | nvarchar(MAX) | n\*2 | Номер телефона пользователя |
| AreaId | int | 4 | Идентификационный номер района |

Таблица «Receptions» хранит информацию о всех записях в карточках. Структура приведена в таблице 2.7.6.

Таблица 2.7.6 – Структура таблицы «Receptions»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| ReceptionId | int | 4 | Идентификационный номер записи |
| PatientId | int | 4 | Идентификационный номер пациента |
| DoctorId | int | 4 | Идентификационный номер врача |
| Diagnosis | nvarchar(MAX) | n\*2 | Диагноз |
| Treatment | nvarchar(MAX) | n\*2 | Лечение |
| ReceptionDate | datetime2(7) | 8 | Дата приема |

Таблица «Service» хранит информацию о всех услугах. Структура приведена в таблице 2.7.7.

Таблица 2.7.7 – Структура таблицы «Service»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер услуги |
| Name | nvarchar(60) | 120 | Название услуги |
| Type | nvarchar(60) | 120 | Тип услуги |
| Description | nvarchar(MAX) | n\*2 | Описание услуги |

Продолжение таблицы 2.7.7

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| DoctorId | int | 4 | Идентификационный номер врача |
| Office | nvarchar(60) | 120 | Кабинет |
| Image | int | 4 | Идентификационный номер изображения |

Таблица «Staff» хранит информацию о персонале, структура приведена в таблице 2.7.8.

Таблица 2.7.78 – Структура таблицы «Staff»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер сотрудника |
| Login | nvarchar(60) | 120 | Логин сотрудника |
| Password | nvarchar(100) | 200 | Пароль сотрудника |
| Surname | nvarchar(60) | 120 | Фамилия сотрудника |
| Name | nvarchar(60) | 120 | Имя сотрудника |
| Patronymic | nvarchar(60) | 120 | Отчество сотрудника |

Таблица «TicketsRecords» хранит информацию о талонах, структура приведена в таблице 2.7.9.

Таблица 2.7.9 – Структура таблицы «TicketsRecords»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| Id | int | 4 | Идентификационный номер талона |
| PatientId | int | 4 | Идентификационный номер пациента |
| DoctorId | int | 4 | Идентификационный номер врача |
| ServiceId | int | 4 | Идентификационный номер услуги |
| ReceptionDate | datetime2(7) | 8 | Дата приема |
| ReceptionTime | time(7) | 5 | Время приема |
| TicketRegistrationDatetime | datetime2(7) | 8 | Дата и время регистрации талона |

# Реализация веб-приложения

## Описание разделов веб-приложения

Разработанное веб-приложение для автоматизации работы медицинской клиники будет поддерживать различные профили пользователей, включая администратора, медицинского специалиста, пациента и персонал. Каждый профиль будет иметь свой функционал и доступ к определенным разделам приложения. Ниже приведено описание разделов веб-приложения, главной страницы и навигационного меню, учитывая специфику каждого профиля.

Главная страница административной части делится на 2 блока: блок «Редактирование» и блок «Создание». Блок «Редактирование» включает в себя 8 кнопок отвечающих за редактирование всех данных приложения. Блок «Создание» включает в себя 8 кнопок отвечающих за создание данных для приложения.

Главная страница медицинского специалиста тоже делится на 2 блока. Первый блок содержит кнопки для создания и редактирования талонов, записей в карточку, профиля. Второй блок содержит календарь, на котором он может просматривать как и забронированные записи путем нажатия на забронированное время, так и создать новые записи, нажимая на свободное время.

Главная страница пациента также делится на 2 блока. Первый блок содержит текстовые поля для вывода и редактирования персональных данных пациента. Второй блок содержит все забронированные пациентом талоны.

Главная страница медицинского персонала содержит кнопку для редактирования профиля, кнопку для просмотра участков и кнопку для создания талона.

Навигационное меню в веб-приложении представляет собой шапку, присутствующую на каждой странице. На шапке расположены такие элементы, как:

* кнопка «Главная», при нажатии на которую открывается главная страница;
* кнопка «Услуги», при нажатии на которую открывается страница услуг;
* кнопка «Мой профиль» позволяет пользователям просматривать их профиль, если они авторизованы, или перенаправляет их на страницу входа, если они не авторизованы;
* контактная информация медцентра;
* кнопка «Вход» перенаправляющая на страницу входа если пользователь не авторизован. Если пользователь авторизован, кнопка изменяется на «Выход», позволяющую пользователю выйти из аккаунта.

## Разработка административной части приложения

Для разработки административной части веб-приложения медицинской клиники необходимо учесть логическую (карту) и физическую структуру (каталог) серверной части приложения.

Логическая структура серверной части веб-приложения будет включать следующие компоненты:

* справочник моделей, содержащий информацию о моделях данных, используемых в приложении, такие как информация о пациентах, медицинская история, текущие обращения и назначения;
* клиентская база данных, где будет храниться информация о пациентах, включая их персональные данные и медицинскую историю;
* информация о талонах на услуги и приемы, которая будет отслеживать записи пациентов на прием к врачам.

Физическая структура серверной части веб-приложения будет организована в виде соответствующего каталога, где будут размещены файлы и компоненты, необходимые для работы приложения и обработки запросов.

Одним из важных аспектов административной части приложения является навигация, которая будет обеспечивать удобное перемещение по различным разделам и функциям. Для этого будут использоваться меню и гиперссылки.

На главной странице административной части приложения будет разделена на два блока: блок «Редактирование» и блок «Создание». Блок «Редактирование» будет содержать 8 кнопок, позволяющих редактировать различные данные приложения:

* кнопка «Редактирование специалиста» позволяет редактировать персональные данные врачей и их расписания;
* кнопка «Редактирование пациента» позволяет редактировать персональную информацию пациентов;
* кнопка «Редактирование услуги» позволяет редактировать информацию о услуге, ее типе и специалисте, который предоставляет данную услугу;
* кнопка «Редактирование участка» позволяет редактировать данные участка и специалиста, ведущего данный участок;
* кнопка «Редактирование приемов» позволяет редактировать записи в карточках пациентов;
* кнопка «Редактирование персонала» позволяет редактировать персональную информацию медицинского работника;
* кнопка «Редактирование администратора» позволяет редактировать данные имеющихся администраторов;
* кнопка «Редактирование записей на прием» позволяет редактировать талоны пациентов на услуги.

Блок «Создание» включает в себя 8 кнопок отвечающих за создание данных для приложения:

* кнопка «Добавление специалиста» позволяет добавить врача и его расписание;
* кнопка «Добавление пациента» позволяет добавить пациента;
* кнопка «Добавление услуги» позволяет добавить услугу;
* кнопка «Добавление участка» позволяет добавить участок;
* кнопка «Добавление приемов» позволяет добавить запись в карточку пациента;
* кнопка «Добавление персонала» позволяет добавить медицинского работника;
* кнопка «Добавление администратора» позволяет добавить администратора;
* кнопка «Добавление записи на прием» позволяет добавить талон пациенту на услугу.

Таким образом, административная часть приложения будет обеспечивать удобное редактирование и создание данных, а также навигацию с помощью меню и кнопок, обеспечивая пользователю лаконичный и понятный интерфейс для работы с приложением.

## Разработка клиентской части приложения

Для разработки клиентской части медицинского веб-приложения, которая будет предназначена для специалистов, пациентов и медицинского персонала, необходимо учесть логическую (карту) и физическую структуру (каталог) клиентской части приложения.

Логическая структура клиентской части приложения для медицинского специалиста будет включать следующие компоненты:

* кнопка «Редактировать профиль» позволяет специалисту изменить персональную информацию;
* кнопка «Просмотр расписания» позволяет специалисту просмотреть свое расписание;
* кнопка «Добавить запись в карточку» позволяет специалисту во время приема создать и заполнить запись в медицинской карточке пациента;
* кнопка «Редактировать запись в карточке» позволяет специалисту дополнить запись о приеме пациента или просмотреть другие записи в карточке;
* кнопка «Просмотреть участки» позволяет просмотреть все участки и список пациентов;
* кнопка «Записать пациента на услугу» позволяет специалисту создать талон для пациента;
* календарь для отслеживания записей.

Логическая структура клиентской части приложения для пациента будет включать следующие компоненты:

* текстовое поле «Имя», которое будет выводить имя зарегистрированного пользователя;
* текстовое поле «Фамилия», которое будет выводить фамилию зарегистрированного пользователя;
* текстовое поле «Отчество», которое будет выводить отчество зарегистрированного пользователя;
* текстовое поле «Дата рождения», которое будет выводить дату рождения зарегистрированного пользователя;
* текстовое поле «Номер телефона», которое будет выводить номер телефона зарегистрированного пользователя;
* текстовое поле «Адрес проживания», которое будет выводить адрес зарегистрированного пользователя;
* текстовое поле «Участок», которое будет выводить участок зарегистрированного пользователя;
* блок назначенных услуг выводящий информацию о талонах пользователя.

Логическая структура клиентской части приложения для медицинского персонала будет включать следующие компоненты:

* кнопка «Редактировать профиль» позволяет работнику изменить персональную информацию;
* кнопка «Просмотреть участки» позволяет просмотреть все участки и список пациентов;
* кнопка «Записать пациента на услугу» позволяет работнику создать талон для пациента.

Физическая структура клиентской части приложения будет организована в виде соответствующего каталога, где будут размещены файлы и компоненты, необходимые для работы клиентской части.

## Описание используемых функций и процедур

Веб-приложение для автоматизации работы медицинской клиники будет иметь следующие функции:

Ведение и редактирование базы данных пациентов:

* добавление новых пациентов с указанием их персональных данных, включая имя, фамилию, контактную информацию;
* редактирование информации о пациентах, включая изменение персональных данных, медицинской истории и текущих обращений;
* удаление информации о пациентах, если это необходимо.

Учет записей на прием к врачам:

* возможность пациентам заказывать талоны на прием к врачу через веб-приложение;
* выбор нужного специалиста из списка доступных врачей;
* уведомление врачей о новых записях и изменениях в расписании.

Ведение расписания врачей:

* просмотр расписания врачей для определенного временного периода;
* управление записями пациентов, включая добавление, изменение и удаление записей;
* уведомление врачей о новых записях и изменениях в расписании.

Для обеспечения лаконичного интерфейса программного средства, будут использованы следующие элементы управления:

* меню для навигации по различным разделам приложения, таким как «Главная», «Услуги», «Мой профиль»;
* кнопочные формы для выполнения различных функций, таких как добавление пациентов, редактирование записей и т.д.

Функции будут взаимосвязаны следующим образом:

* при добавлении нового пациента, информация будет сохраняться в базе данных пациентов;
* при заказе талона на прием, информация о записи будет добавлена в расписание соответствующего врача;
* при изменении или удалении записи пациента, соответствующая информация будет обновлена в базе данных и расписании врачей.

Для обеспечения работы с информацией и создания справочной системы приложения, будут разработаны соответствующие классы и функции для обработки и хранения данных, а также возможности поиска и просмотра информации о пациентах и расписании.

Листинги модулей с комментариями и указанием элементов управления, которые инициируют их выполнение, должны быть представлены в приложении А. На диаграмме вариантов использования представлен весь функционал пользователя и администратора.

## Функциональное тестирование

Функциональное тестирование является важной частью процесса обеспечения качества программного обеспечения. Оно направлено на проверку функциональности системы с целью убедиться, что она соответствует требованиям и ожиданиям пользователей. Прежде чем перейти к разработке тест-кейсов, необходимо провести анализ требований, понять функциональность приложения и определить основные сценарии использования. Это позволит сосредоточиться на наиболее важных и критических аспектах приложения. Подходящие тест-кейсы должны быть разработаны для каждого сценария использования, чтобы проверить, что функции приложения работают правильно и соответствуют ожиданиям пользователей. На таблицах 3.5.1-3.5.6 представлен набор тест-кейсов для программного средства.

Таблица 3.5.1 – Тест-кейс для функции регистрации пользователя в веб-приложении

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Открыть веб-приложение «MedicalClinic» | Веб-приложение «MedicalClinic» открыто и доступно | Пройден |
| Нажать кнопку «Войти» в шапке веб-приложения | Открыта страница входа | Пройден |
| Нажать кнопку «Аккаунт еще не создан» | Открыта страница регистрации | Пройден |
| Ввести в поле «Логин» и «Пароль» строки «qwerty», «11111111Aa» | Отображение введенного логина и пароля в поле | Пройден. Результат представлен в приложении Б на рисунке Б.1 |
| Нажать на кнопку «Зарегистрироваться» | Отображение страницы создания профиля | Пройден. Результат представлен в приложении Б на рисунке Б.2 |
| Ввести в поле «Фамилия» значение «Филимонов», в поле «Имя» значение «Виктор», в поле «Отчество» значение «Юрьевич», в поле «Дата рождения» значение «17.11.2005», в поле «Адрес» значение «ул. Казинца 78, кв. 54», в поле «Номер телефона» значение «+375 (44) 111-11-11» в поле «Участок» значение «1» | Отображение введенных данных | Пройден. Результат представлен в приложении Б на рисунке Б.3 |
| Нажать на кнопку «Зарегистрироваться» | Отображение страницы профиля пользователя | Пройден. Результат представлен в приложении Б на рисунке Б.4 |

Таблица 3.5.2 – Тест-кейс для функции авторизации пользователя в веб-приложении

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Открыть веб-приложение «MedicalClinic» | Веб-приложение «MedicalClinic» открыто и доступно | Пройден |
| Нажать на кнопке «Войти» в шапке веб-приложения | Открыта страница входа | Пройден |
| Ввести в поле «Логин» и «Пароль» строки «qwerty», «11111111Aa» | Отображение введенного логина и пароля в поле | Пройден. Результат представлен в приложении Б на рисунке Б.5 |
| Нажать на кнопку «Войти» | Отображение страницы профиля пользователя | Пройден. Результат представлен в приложении Б на рисунке Б.6 |

Таблица 3.5.3 – Тест-кейс для функции создания талона пользователем веб-приложения

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Открыть веб-приложение «MedicalClinic» | Веб-приложение «MedicalClinic» открыто и доступно | Пройден |
| Авторизоваться в веб-приложении | Пользователь авторизован, отображение страницы профиля пользователя | Пройден. Результат представлен в приложении Б на рисунке Б.6 |
| Нажать кнопку «Услуги» в шапке веб-приложения | Отображение страницы доступных услуг | Пройден. Результат представлен в приложении Б на рисунке Б.7 |
| Нажать на кнопку услуги «Биохимический анализ крови» | Отображение страницы услуги | Пройден. Результат представлен в приложении Б на рисунке Б.8 |

Продолжение таблицы 3.5.3

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Выбрать в календаре дату 29 июня и время 11:00 | Отображение страницы талона | Пройден. Результат представлен в приложении Б на рисунке Б.9 |

Таблица 3.5.4 – Тест-кейс для функции создания записи в карточке пациента для медицинского специалиста

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Результат теста |
| Открыть веб-приложение «MedicalClinic» | Веб-приложение «MedicalClinic» открыто и доступно | Пройден |
| Авторизоваться в веб-приложении | Пользователь авторизован, отображение страницы профиля пользователя | Пройден. Результат представлен в приложении Б на рисунке Б.10 |
| Нажать кнопку «Добавить запись в карточку» в профиле специалиста | Отображение страницы создание записи в карточку | Пройден. Результат представлен в приложении Б на рисунке Б.11 |
| Ввести в поле «ID пациента» значение «17», в поле «ID врача» значение «1», в поле «Диагноз» значение «Острый фарингит», в поле «Лечение» значение «Принимать обильное горячее питье (теплый чай, отвар из трав, вода с медом)», в поле «Дата приема» значение «20.06.2024» | Отображение введенных данных | Пройден. Результат представлен в приложении Б на рисунке Б.12 |
| Нажать на кнопку «Создать» | Отображение страницы всех приемов | Пройден. Результат представлен в приложении Б на рисунке Б.13 |

# Применение

## Назначение веб-приложения

Название разработанного программного средства – «MedicalClinic».

Назначение разрабатываемого веб-приложения для автоматизации работы медицинской клиники заключается в облегчении и оптимизации процессов управления данными о пациентах, записи на приемы и ведения расписания врачей. Оно предоставляет медицинскому персоналу и пациентам удобный доступ к необходимой информации и функциональности через веб-интерфейс.

Целью разработки данного веб-приложения является создание эффективного инструмента для медицинской клиники, который позволит автоматизировать и упростить процессы управления базой данных пациентов, записью на приемы и расписанием врачей. Главной задачей является повышение эффективности работы клиники, улучшение обслуживания пациентов и уменьшение ручной работы персонала.

Для установки веб-приложения используется установщик «MedicalClinic.exe».

Для запуска веб-приложение необходимо предусмотреть наличие SQL Server для базы данных. Затем пользователь может открыть веб-браузер и ввести URL-адрес, указывающий на главную страницу веб-приложения, чтобы получить доступ к функциональности и начать его использование.

Разработанное веб-приложение предназначено для автоматизации работы медицинской клиники. Оно позволяет управлять базой данных пациентов, записями на приемы и расписанием врачей. Веб-приложение позволяет пациентам удобно заказывать приемы, а медицинскому персоналу эффективно организовывать рабочие процессы.

Класс задач, решаемых данной системой, включает:

* управление базой данных пациентов: добавление, редактирование и удаление информации о пациентах;
* ведение медицинской истории пациентов: сохранение и обновление информации о медицинских обследованиях, процедурах, диагнозах и лечении;
* ведение расписания врачей: возможность просматривать и управлять записями пациентов в расписании врачей, оптимизируя распределение времени и предотвращая конфликты;
* разграничение прав доступа: обеспечение безопасности данных и ограничение доступа к конфиденциальной информации только авторизованным пользователям в соответствии с их ролями и полномочиями.

Область применения данного веб-приложения ограничена медицинской сферой и предназначена для использования в медицинских клиниках и учреждениях. Оно предоставляет инструменты для автоматизации работы медицинского персонала и облегчения взаимодействия с пациентами. Однако следует отметить, что веб-приложение не является заменой для медицинского образования и профессиональных медицинских консультаций. Оно служит вспомогательным средством для упрощения процессов управления данными и записями пациентов, но окончательные медицинские решения и диагнозы должны быть основаны на компетентности и опыте медицинского персонала.

## Программно-аппаратное обеспечение сервера и клиента

Для корректной работы веб-приложения для автоматизации работы медицинской клиники, требуется следующее программно-аппаратное обеспечение для сервера и клиента.

Сервер:

* операционная система. Рекомендуется использовать Windows Server для обеспечения стабильности и безопасности работы сервера;
* процессор. Мощный многоядерный процессор для обработки большого количества запросов от клиентов;
* память. Достаточно оперативной памяти для обработки и хранения данных, рекомендуется минимум 8 ГБ;
* хранилище. Достаточное пространство для хранения базы данных, файлов и резервных копий;
* база данных. Установленный и настроенный SQL Server для хранения и управления данными пациентов, расписаниями врачей и другой необходимой информацией;
* интернет-соединение. Стабильное и быстрое соединение для обеспечения доступа клиентов к приложению.

Клиент:

* операционная система. Может быть Windows, macOS или Linux, совместимая с браузером, который будет использоваться для доступа к веб-приложению;
* браузер. Рекомендуется использовать последнюю версию современного браузера, такого как Google Chrome или Microsoft Edge;
* процессор и память. Достаточно мощный процессор и оперативная память для обеспечения плавной работы веб-приложения;
* интернет-соединение. Стабильное и быстрое соединение для обеспечения доступа к серверу и передачи данных.

Требования к программно-аппаратному обеспечению обусловлены необходимостью обработки и хранения большого объема данных, обеспечения безопасности и стабильности работы приложения, а также удобства использования для клиентов. Правильная конфигурация сервера и клиента позволит обеспечить эффективность и надежность работы веб-приложения для медицинской клиники.

# Заключение

Веб-приложение для автоматизации работы медицинской клиники было разработано с использованием современных веб-технологий, таких как HTML, CSS, JavaScript и C#. Для разработки могли быть использованы такие инструменты и фреймворки, как ASP.NET, Entity Framework и SQL Server.

Веб-приложение выполняет все поставленные задачи. Оно предоставляет функционал для управления базой данных пациентов, включая добавление, редактирование и удаление информации о пациентах, их персональных данных, медицинской истории, обращениях и назначениях. Также реализована возможность пациентам заказывать талоны на прием к врачу, выбирать нужного специалиста и указывать предпочтительное время и дату приема. Приложение также позволяет вести расписание врачей и просматривать и управлять записями пациентов.

Преимущества разработанного веб-приложения:

* Автоматизация и упрощение работы медицинской клиники: Веб-приложение позволяет управлять и хранить информацию о пациентах и записях, сокращая необходимость вручную вести и обрабатывать бумажные документы;
* Улучшение доступности и удобства для пациентов: Пациенты могут заказывать талоны на прием онлайн и выбирать удобное для них время и дату, что упрощает процесс записи и экономит время;
* Разграничение прав доступа: Приложение предоставляет возможность управлять доступом к базе данных, обеспечивая безопасность и конфиденциальность информации.

Недостатки разработанного веб-приложения:

* Необходимость технической экспертизы: для разработки и поддержки веб-приложения может потребоваться наличие специалистов в области веб-разработки и баз данных, что может быть дорого и сложно для медицинской клиники;
* Зависимость от интернет-соединения: для использования веб-приложения требуется стабильное и быстрое интернет-соединение, что может быть проблематично в некоторых областях или в случае сбоев в сети.

В целом, разработанное веб-приложение предоставляет значительные преимущества для медицинской клиники, автоматизируя и упрощая процессы работы. Однако, необходимо учитывать некоторые недостатки и затраты на техническую поддержку и обеспечение стабильного интернет-соединения.

# Список использованных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» // Т.Г. Багласова, К.О. Якимович и др. – Минск : ЧУО «КБП» , 2022 . – 41 с.
2. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996 . –  Минск : Межгос. совет по стандартизации, метрологии и сертификации , 1995 . – 84 с.
3. Описание программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.402-2000 . – Введ. 01.09.2001 . – Минск : Межгос. совет по стандартизации, метрологии и сертификации , 2000 . – 14 с.
4. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. Минск : Межгос. совет по стандартизации, метрологии и сертификации , 2000 . – 14 с.
5. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.404-2000 . – Введ. 01.09.2001 . – Минск : Межгос. совет по стандартизации, метрологии и сертификации , 2000 . – 16 с.
6. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ Петербург, 2013. – 608 с.
7. Митин, А.И. Работа с базами данных Microsoft SQL Server: сценарии практических занятий / А. И. Митин. – М. : Директ-Медиа, 2020. – 142 с
8. Draw.io – бесплатное средство для создания блок-схем, инфографики, прототипов [Электронный ресурс]. – EL-BLOG.RU, 2023 – Режим доступа: https://el-blog.ru/draw-io/. – Дата доступа: 02.06.2024.
9. Обзор HTML5 и CSS3 [Электронный ресурс]. – W3Schools, 2023. – Режим доступа: https://www.w3schools.com/. – Дата доступа: 15.06.2024.
10. Обзор JavaScript [Электронный ресурс]. – Mozilla Developer Network, 2023. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide. – Дата доступа: 20.06.2024.
11. Обзор SQL [Электронный ресурс]. – SQLCourse, 2023. – Режим доступа: http://www.sqlcourse.com/. – Дата доступа: 25.06.2024.

# Приложение А

**(обязательное)****Текст программных модулей**

Код создания пациента.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.AspNetCore.Mvc.Rendering;

using medicalclinic.Data;

using medicalclinic.Models;

namespace medicalclinic.Pages.Patient

{

public class CreateModel : PageModel

{

private readonly medicalclinic.Data.medicalclinicContext \_context;

public CreateModel(medicalclinic.Data.medicalclinicContext context)

{

\_context = context;

}

public IActionResult OnGet()

{

return Page();

}

// Свойство, представляющее создаваемого пациента.

[BindProperty]

public medicalclinic.Models.Patient Patient { get; set; } = default!;

public async Task<IActionResult> OnPostAsync()

{

// Добавляем пациента в контекст базы данных.

\_context.Patient.Add(Patient);

// Сохраняем изменения в базе данных.

await \_context.SaveChangesAsync();

// Перенаправляем на страницу со списком пациентов (индекс).

return RedirectToPage("./Index");

}

}

}

Код создания талона.

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using medicalclinic.Models;

using medicalclinic.Data;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

namespace medicalclinic.Pages

{

// Модель страницы "ServiceInfoModel"

public class ServiceInfoModel : PageModel

{

// Приватное поле, содержащее контекст данных приложения

private readonly medicalclinicContext \_context;

// Публичные свойства модели

public medicalclinic.Models.Service Service { get; set; }

public List<medicalclinic.Models.TicketsRecords> Tickets { get; set; }

public List<medicalclinic.Models.DoctorsSchedule> Schedule { get; set; }

// Конструктор, инициализирующий контекст данных

public ServiceInfoModel(medicalclinicContext context)

{

\_context = context;

}

// Обработчик GET-запроса на страницу

public async Task<IActionResult> OnGetAsync(int? id)

{

// Если id равен null, возвращаем ответ NotFound()

if (id == null)

{

return NotFound();

}

// Загружаем информацию об услуге с указанным id, включая связанного врача

Service = await \_context.Service

.Include(s => s.Doctor)

.FirstOrDefaultAsync(m => m.Id == id);

// Если услуга не найдена, создаем новый объект Service с стандартными значениями

if (Service == null)

{

Service = new medicalclinic.Models.Service

{

Name = "Услуга не найдена",

Description = "Извините, но запрашиваемая услуга не найдена на сайте."

};

}

// Загружаем расписание врача, ответственного за данную услугу

Schedule = await \_context.DoctorsSchedule

.Where(s => s.DoctorId == Service.Doctor.Id)

.ToListAsync();

// Загружаем информацию о талонах, связанных с данной услугой

Tickets = await \_context.TicketsRecords

.Where(t => t.DoctorId == Service.Doctor.Id)

.ToListAsync();

// Сохраняем данные о расписании и талонах в объекте ViewData

ViewData["Schedule"] = Schedule;

ViewData["Tickets"] = Tickets;

// Возвращаем результат выполнения страницы

return Page();

}

}

}

Код создания записи в медкарте

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.AspNetCore.Mvc.Rendering;

using medicalclinic.Data;

using medicalclinic.Models;

namespace medicalclinic.Pages.Receptions

{

// Модель страницы "CreateModel"

public class CreateModel : PageModel

{

// Приватное поле, содержащее контекст данных приложения

private readonly medicalclinic.Data.medicalclinicContext \_context;

// Конструктор, инициализирующий контекст данных

public CreateModel(medicalclinic.Data.medicalclinicContext context)

{

\_context = context;

}

// Обработчик GET-запроса на страницу

public IActionResult OnGet()

{

// Возвращает страницу с пустой моделью Receptions

return Page();

}

// Публичное свойство модели, привязываемое к модели Receptions

[BindProperty]

public medicalclinic.Models.Receptions Receptions { get; set; } = default!;

// Обработчик POST-запроса на страницу

public async Task<IActionResult> OnPostAsync()

{

// Добавляем новую запись Receptions в контекст данных

\_context.Receptions.Add(Receptions);

// Сохраняем изменения в базе данных

await \_context.SaveChangesAsync();

// Перенаправляем пользователя на страницу "./Index"

return RedirectToPage("./Index");

}

}

}

Код страницы входа

// Подключение необходимых пространств имен

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using medicalclinic.Models;

using medicalclinic.Data;

using System.Security.Claims;

// Определение пространства имен и класса

namespace medicalclinic.Pages

{

// Класс модели страницы "LoginModel"

public class LoginModel : PageModel

{

// Приватное поле, содержащее контекст данных приложения

private readonly medicalclinicContext \_context;

// Конструктор, инициализирующий контекст данных

public LoginModel(medicalclinicContext context)

{

\_context = context;

}

// Публичное свойство модели, привязываемое к модели InputModel

[BindProperty]

public InputModel Input { get; set; }

// Вложенный класс, представляющий модель ввода данных для входа

public class InputModel

{

// Обязательное поле, длина которого от 3 до 60 символов

[Required]

[StringLength(60, MinimumLength = 3)]

public string Login { get; set; }

// Обязательное поле, длина которого от 6 до 100 символов

[Required]

[StringLength(100, MinimumLength = 6)]

public string Password { get; set; }

}

// Обработчик POST-запроса на страницу

public async Task<IActionResult> OnPostAsync()

{

// Проверка валидности модели

if (ModelState.IsValid)

{

// Поиск администратора, врача, пациента и сотрудника в базе данных

var admin = \_context.Administrator.FirstOrDefault(a => a.Login == Input.Login && a.Password == Input.Password);

var doctor = \_context.Doctors.FirstOrDefault(d => d.Login == Input.Login && d.Password == Input.Password);

var patient = \_context.Patient.FirstOrDefault(p => p.Login == Input.Login && p.Password == Input.Password);

var staff = \_context.Staff.FirstOrDefault(s => s.Login == Input.Login && s.Password == Input.Password);

// Проверка, является ли пользователь администратором

if (admin != null)

{

// Создание списка утверждений для аутентификации

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, admin.Login),

new Claim(ClaimTypes.Role, "Admin")

};

// Выполнение входа в систему с использованием куки-аутентификации

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme)));

// Перенаправление на страницу профиля администратора

return RedirectToPage("/AdminProfile");

}

// Проверка, является ли пользователь врачом

else if (doctor != null)

{

// Создание списка утверждений для аутентификации

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, doctor.Login),

new Claim(ClaimTypes.Role, "Doctor"),

new Claim("Id", doctor.Id.ToString())

};

// Выполнение входа в систему с использованием куки-аутентификации

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme)));

// Перенаправление на страницу профиля врача

return RedirectToPage("/DoctorProfile");

}

// Проверка, является ли пользователь пациентом

else if (patient != null)

{

// Создание списка утверждений для аутентификации

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, patient.Login),

new Claim(ClaimTypes.Role, "Patient"),

new Claim("Id", patient.Id.ToString())

};

// Выполнение входа в систему с использованием куки-аутентификации

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme)));

// Перенаправление на страницу профиля пациента

return RedirectToPage("/PatientProfile");

}

// Проверка, является ли пользователь сотрудником

else if (staff != null)

{

// Создание списка утверждений для аутентификации

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, staff.Login),

new Claim(ClaimTypes.Role, "Staff"),

new Claim("Id", staff.Id.ToString())

};

// Выполнение входа в систему с использованием куки-аутентификации

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme)));

// Перенаправление на страницу профиля сотрудника

return RedirectToPage("/StaffProfile");

}

else

{

// Добавление ошибки в модель состояния

ModelState.AddModelError(string.Empty, "Неверный логин или пароль.");

}

}

// Возвращение текущей страницы

return Page();

}

}

}

Код страницы регистарции

// Этот класс отвечает за обработку ввода пользователя и регистрацию нового пациента в системе.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.Cookies;

using medicalclinic.Models;

using medicalclinic.Data;

using System.Security.Claims;

namespace medicalclinic.Pages

{

public class RegisterModel : PageModel

{

// Приватное поле для контекста базы данных медицинской клиники

private readonly medicalclinicContext \_context;

// Конструктор, который принимает контекст базы данных в качестве параметра

public RegisterModel(medicalclinicContext context)

{

\_context = context;

}

// Свойство, связанное с данными входа пользователя

[BindProperty]

public InputModel Input { get; set; }

// Внутренний класс, описывающий данные входа пользователя

public class InputModel

{

[Required]

[StringLength(60, MinimumLength = 3)]

public string Login { get; set; }

[Required]

[StringLength(100, MinimumLength = 8)]

public string Password { get; set; }

}

// Метод, вызываемый при отправке формы регистрации

public async Task<IActionResult> OnPostAsync()

{

// Проверка валидности данных, введенных пользователем

if (ModelState.IsValid)

{

// Поиск существующих пользователей с таким же логином

var admin = \_context.Administrator.FirstOrDefault(a => a.Login == Input.Login);

var doctor = \_context.Doctors.FirstOrDefault(d => d.Login == Input.Login);

var patient = \_context.Patient.FirstOrDefault(p => p.Login == Input.Login);

var staff = \_context.Staff.FirstOrDefault(s => s.Login == Input.Login);

// Если логин уже занят, добавляем ошибку в модель

if (admin != null || doctor != null || patient != null || staff != null)

{

ModelState.AddModelError(string.Empty, "Логин уже занят.");

}

else

{

// Создание нового пациента и сохранение его в базе данных

var newPatient = new medicalclinic.Models.Patient { Login = Input.Login, Password = Input.Password };

\_context.Patient.Add(newPatient);

await \_context.SaveChangesAsync();

// Создание списка Claims для аутентификации пользователя

var claims = new List<Claim>

{

new Claim(ClaimTypes.Name, newPatient.Login),

new Claim("Id", newPatient.Id.ToString())

};

// Создание объекта ClaimsIdentity и настройка свойств аутентификации

var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

var authProperties = new AuthenticationProperties

{

};

// Вход пользователя в систему с использованием созданных Claims

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity), authProperties);

return RedirectToPage("/CreatePatient");

}

}

// Если данные невалидны, возвращаем страницу регистрации

return Page();

}

}

}

# Приложение Б

**(справочное)****Результаты работы программы**

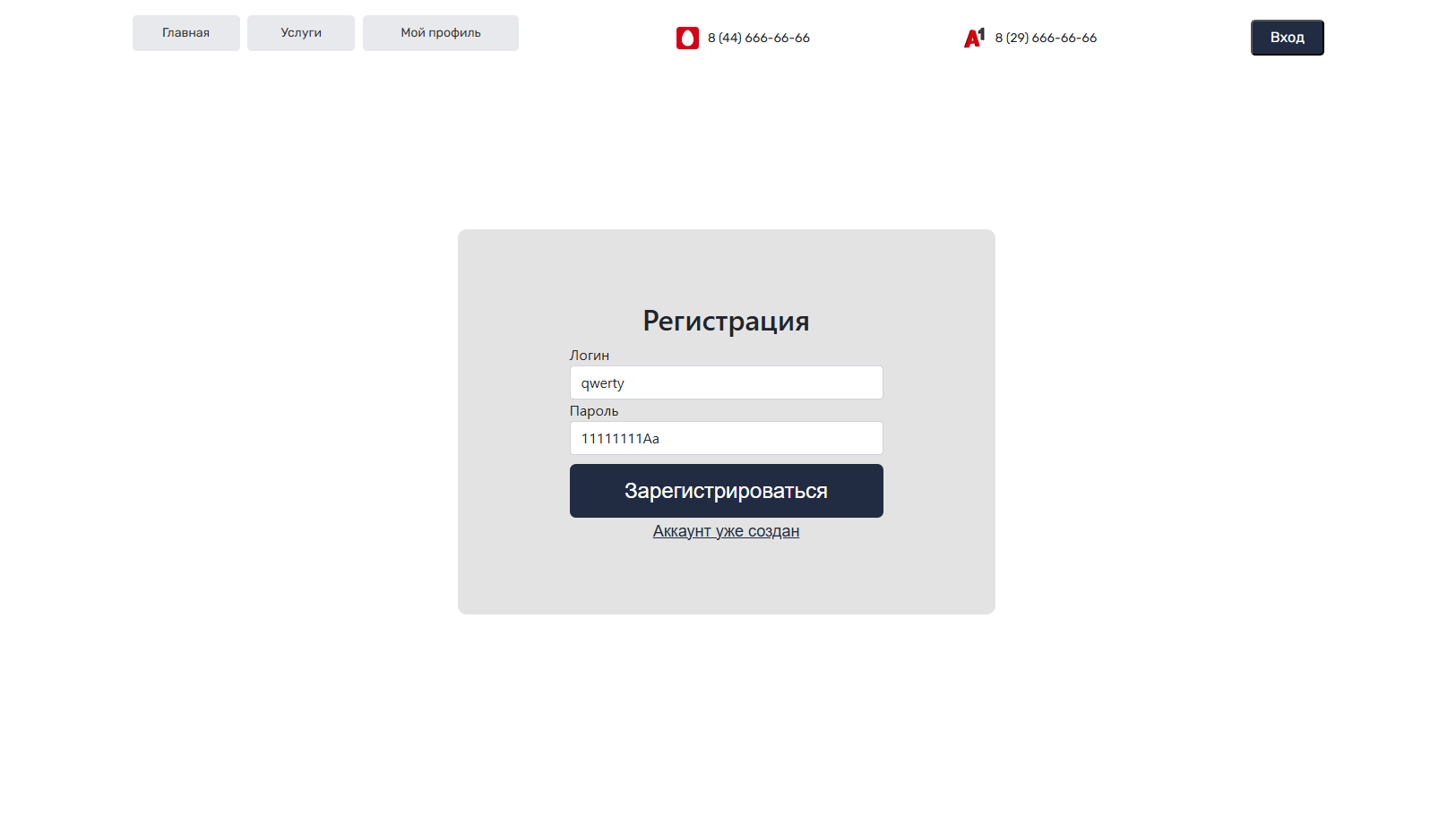


Рисунок Б.1 – Страница регистрации с заполненными полями

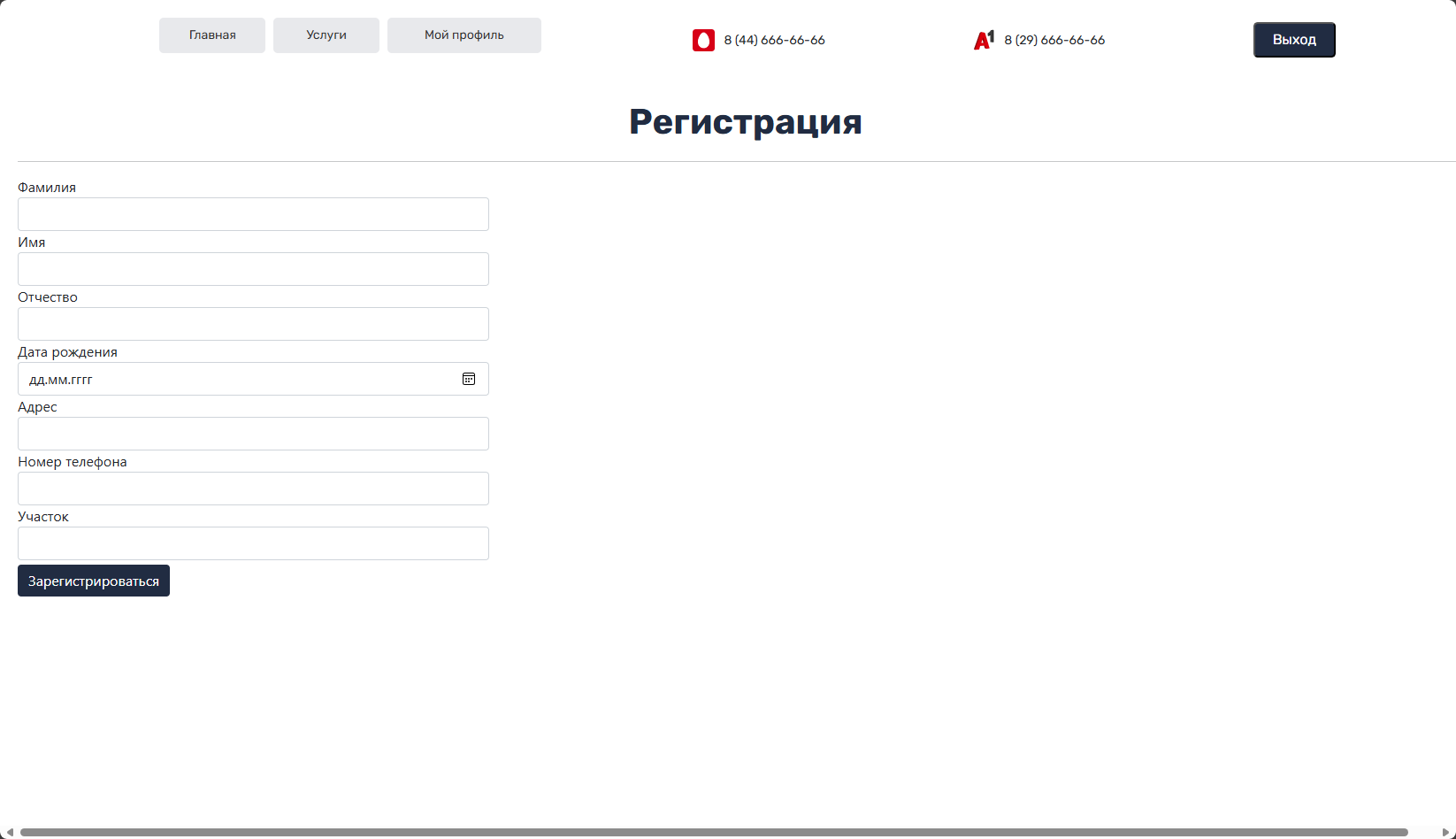


Рисунок Б.2 – Страница создания профиля

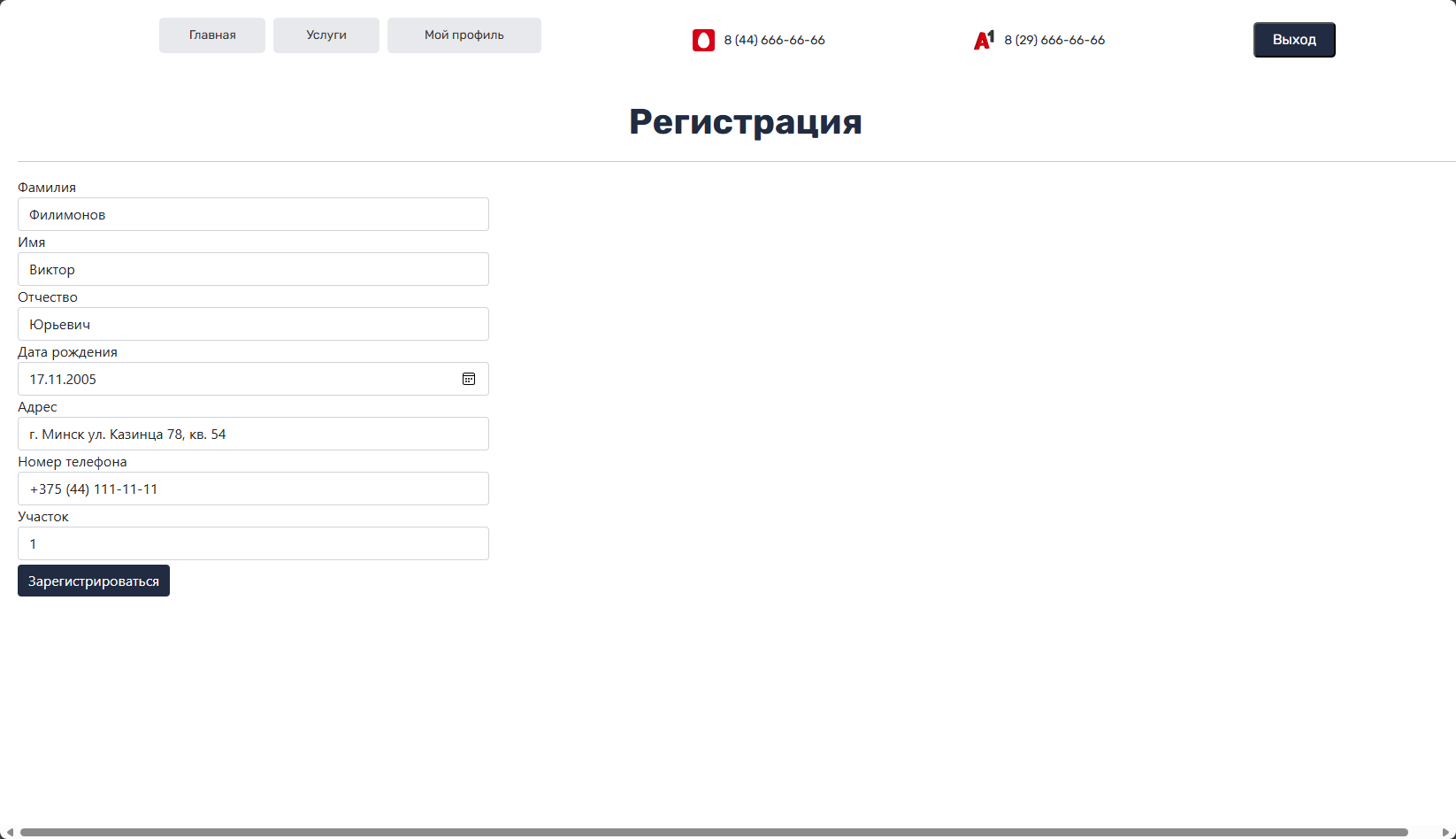


Рисунок Б.3 – Страница создания профиля с заполненными полями

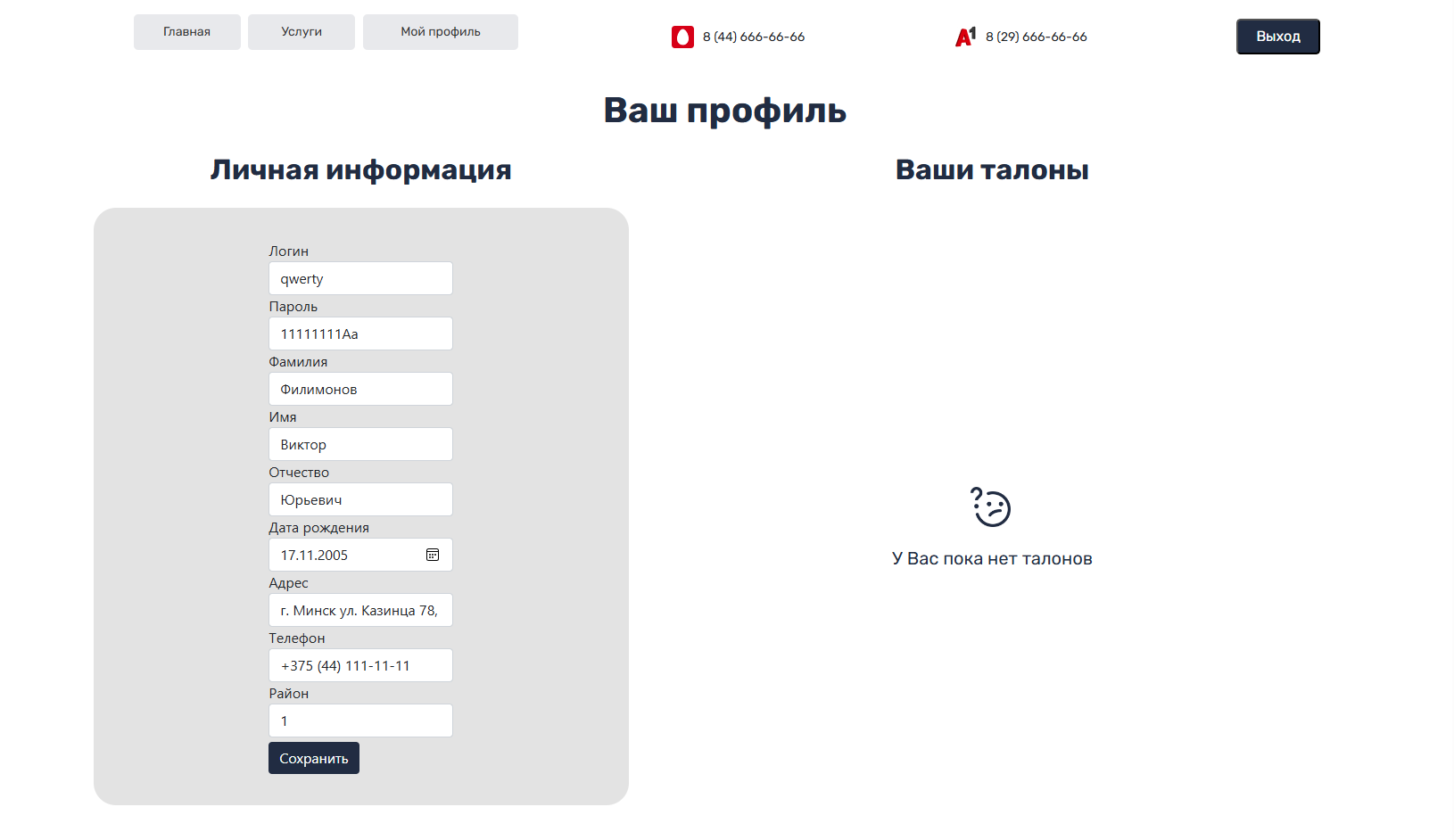


Рисунок Б.4 – Страница профиля пользователя

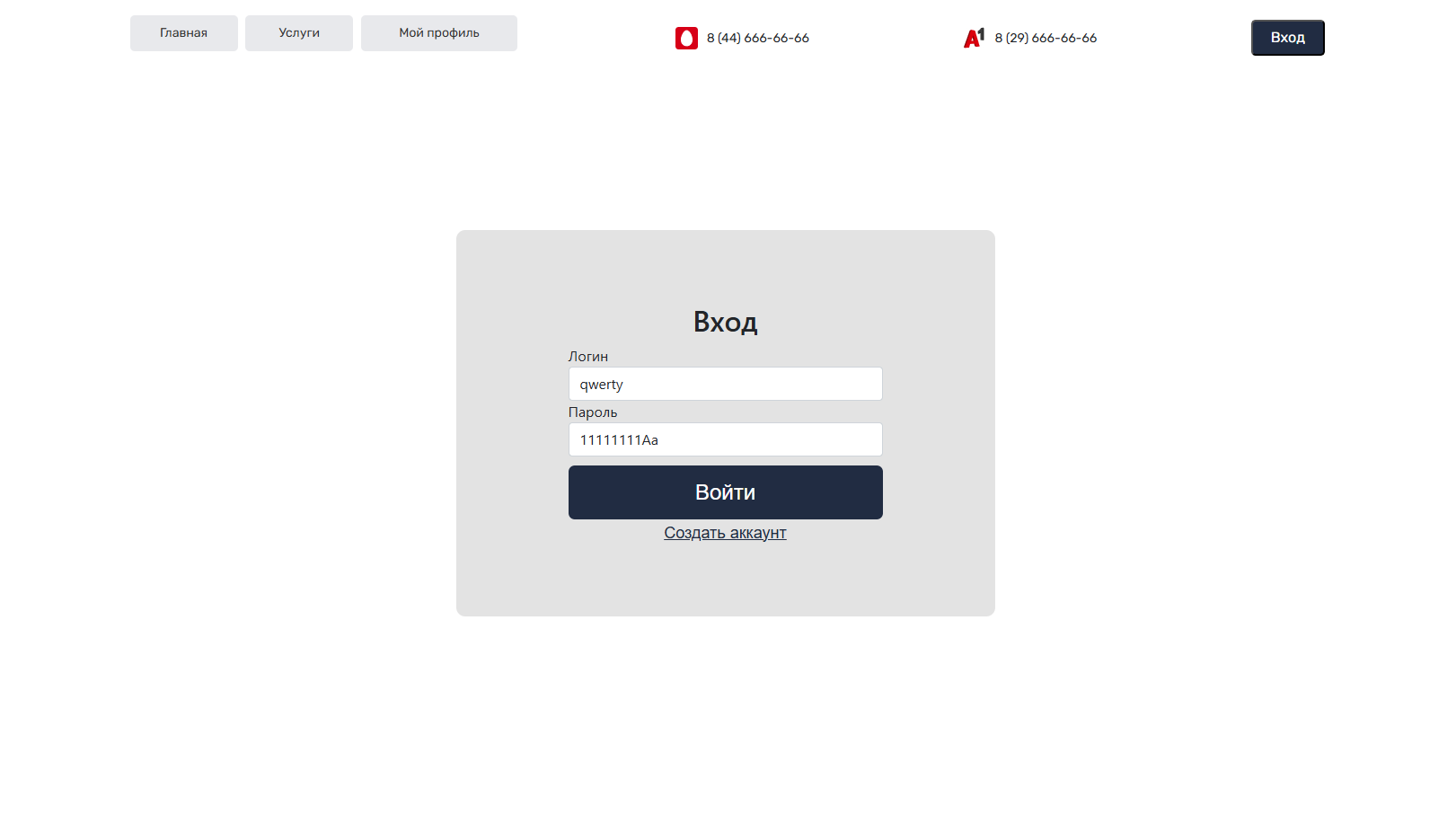


Рисунок Б.5 – Страница авторизации с заполненными полями

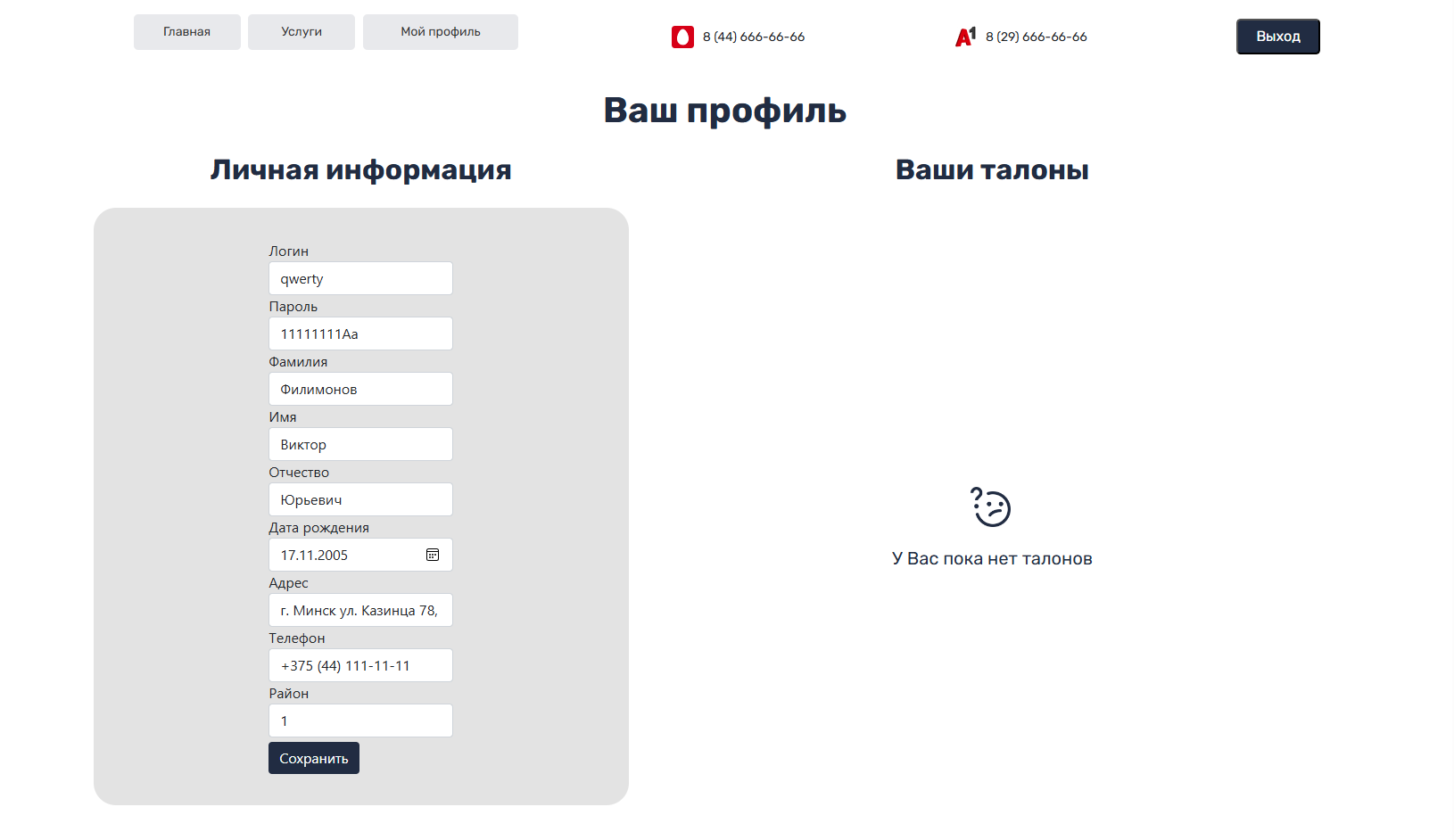


Рисунок Б.6 – Страница профиля пользователя

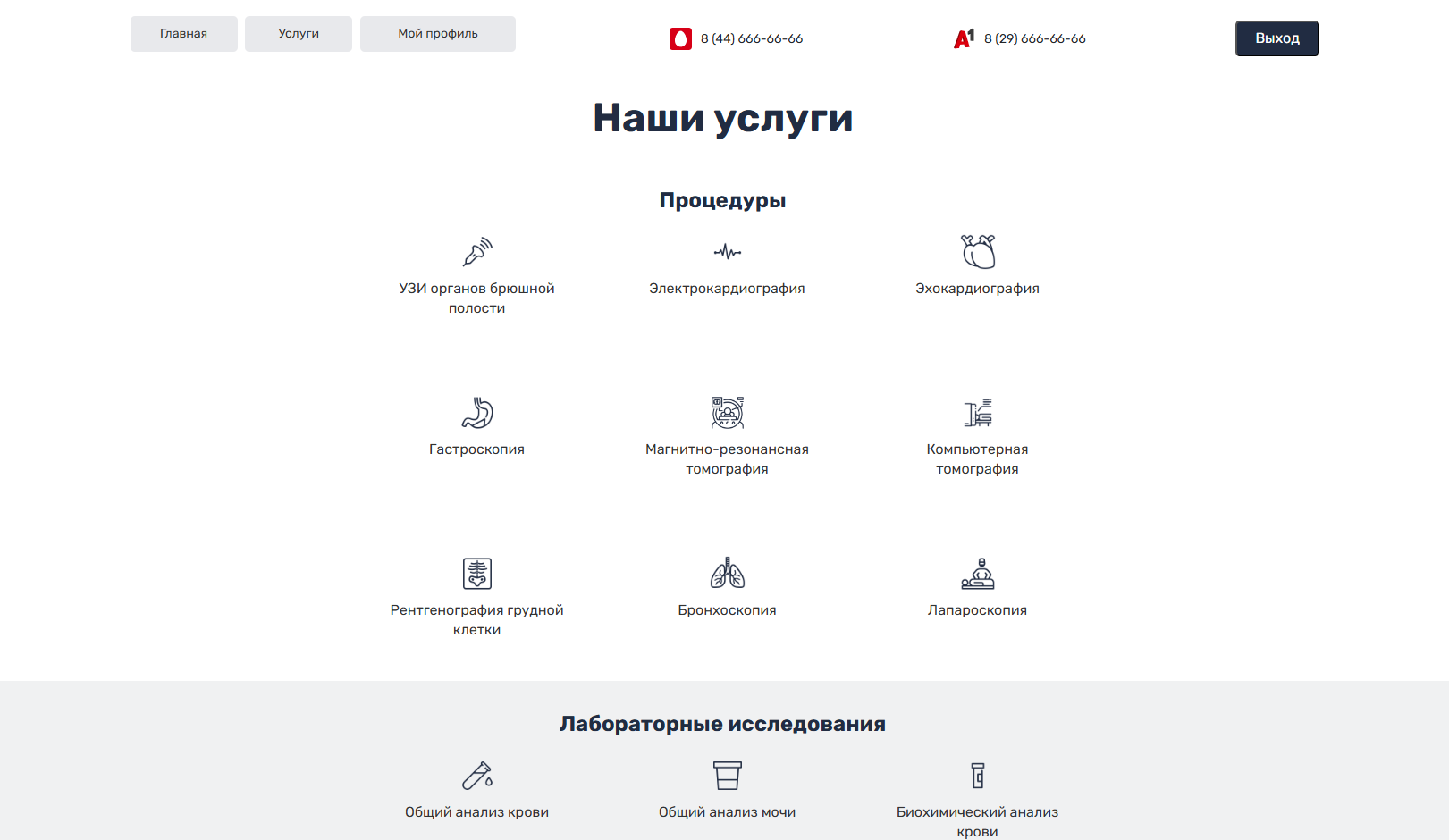


Рисунок Б.7 – Страница доступных услуг

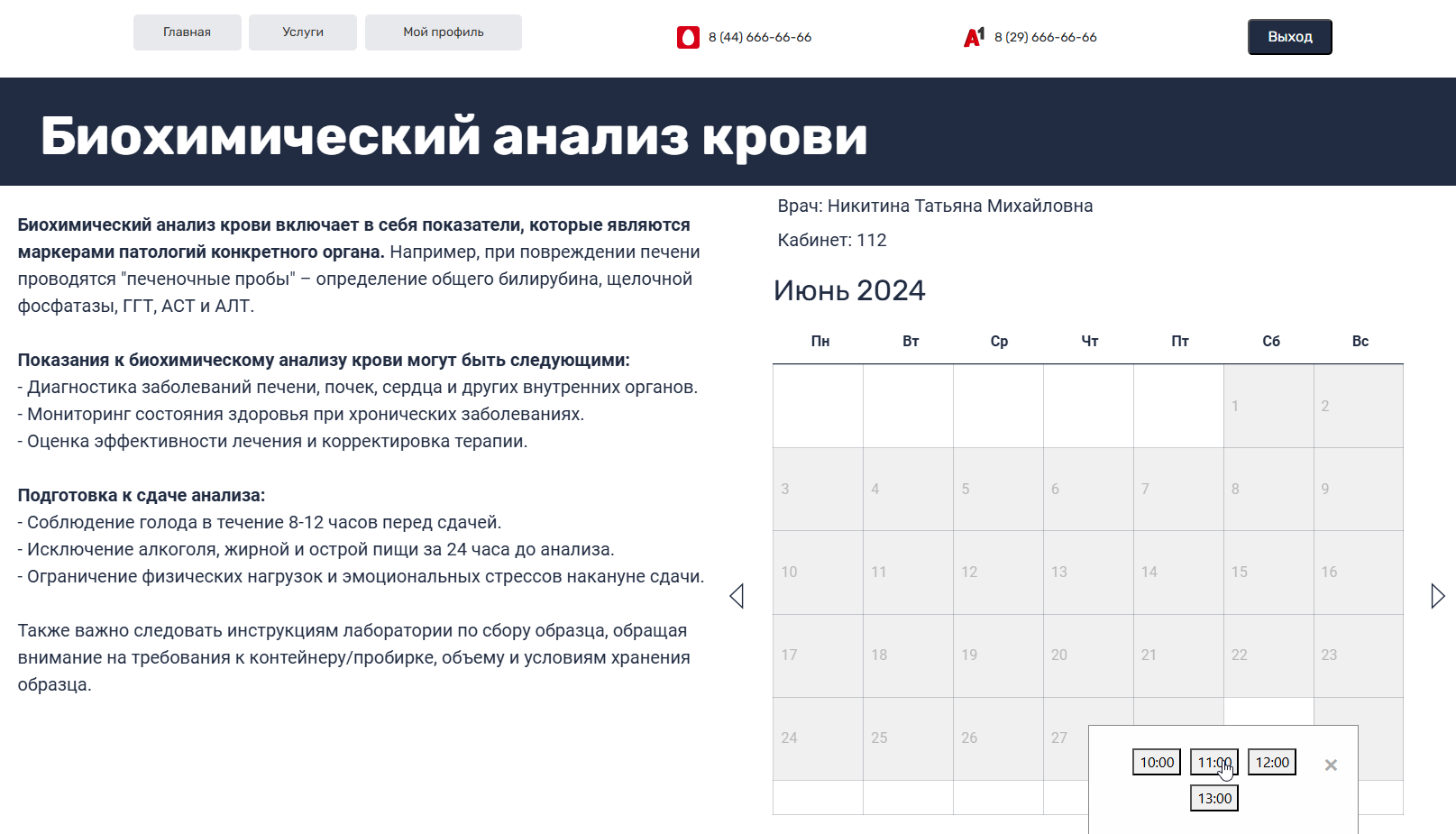


Рисунок Б.8 – Страница выбранной услуги

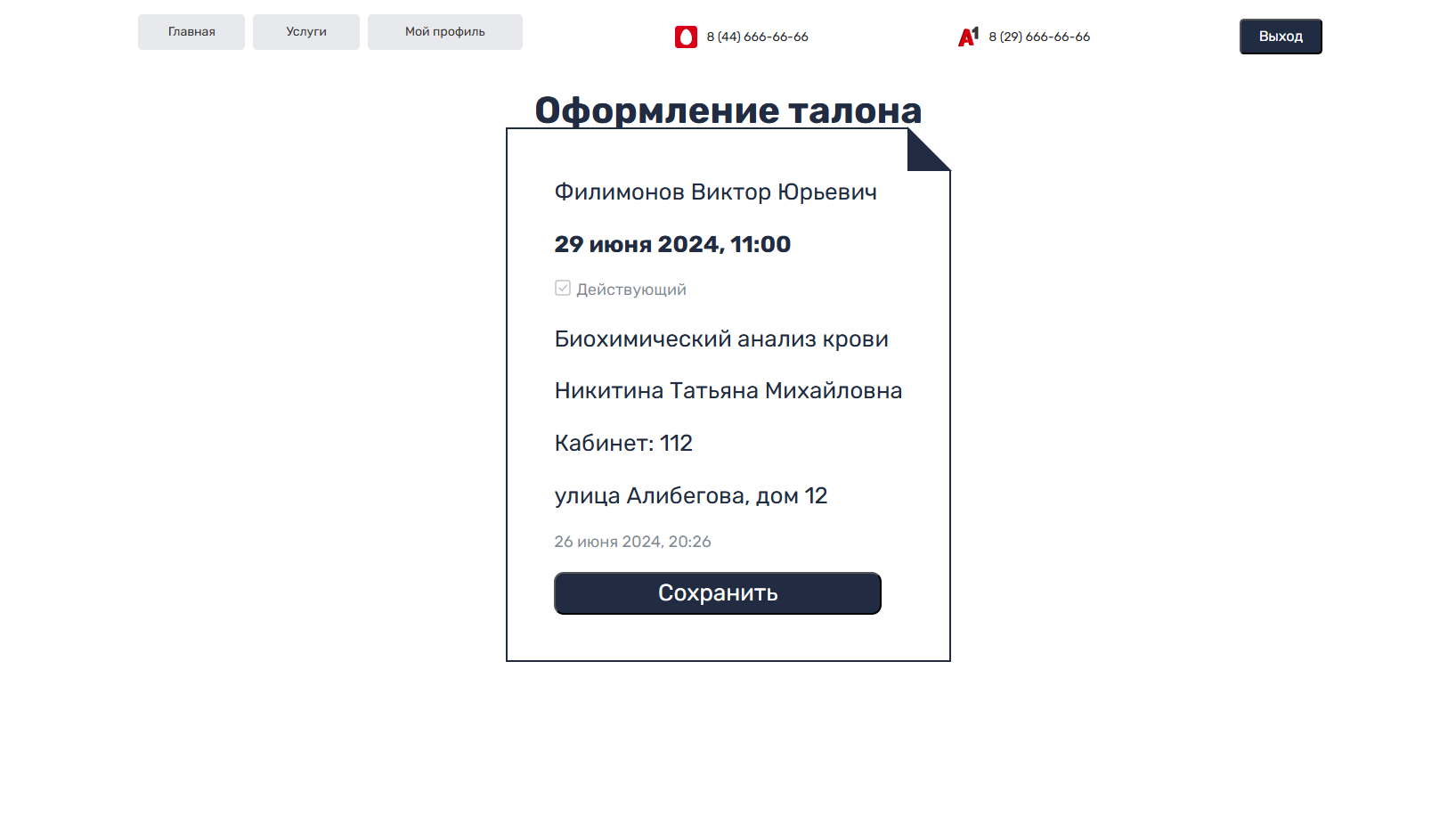


Рисунок Б.9 – Страница талона

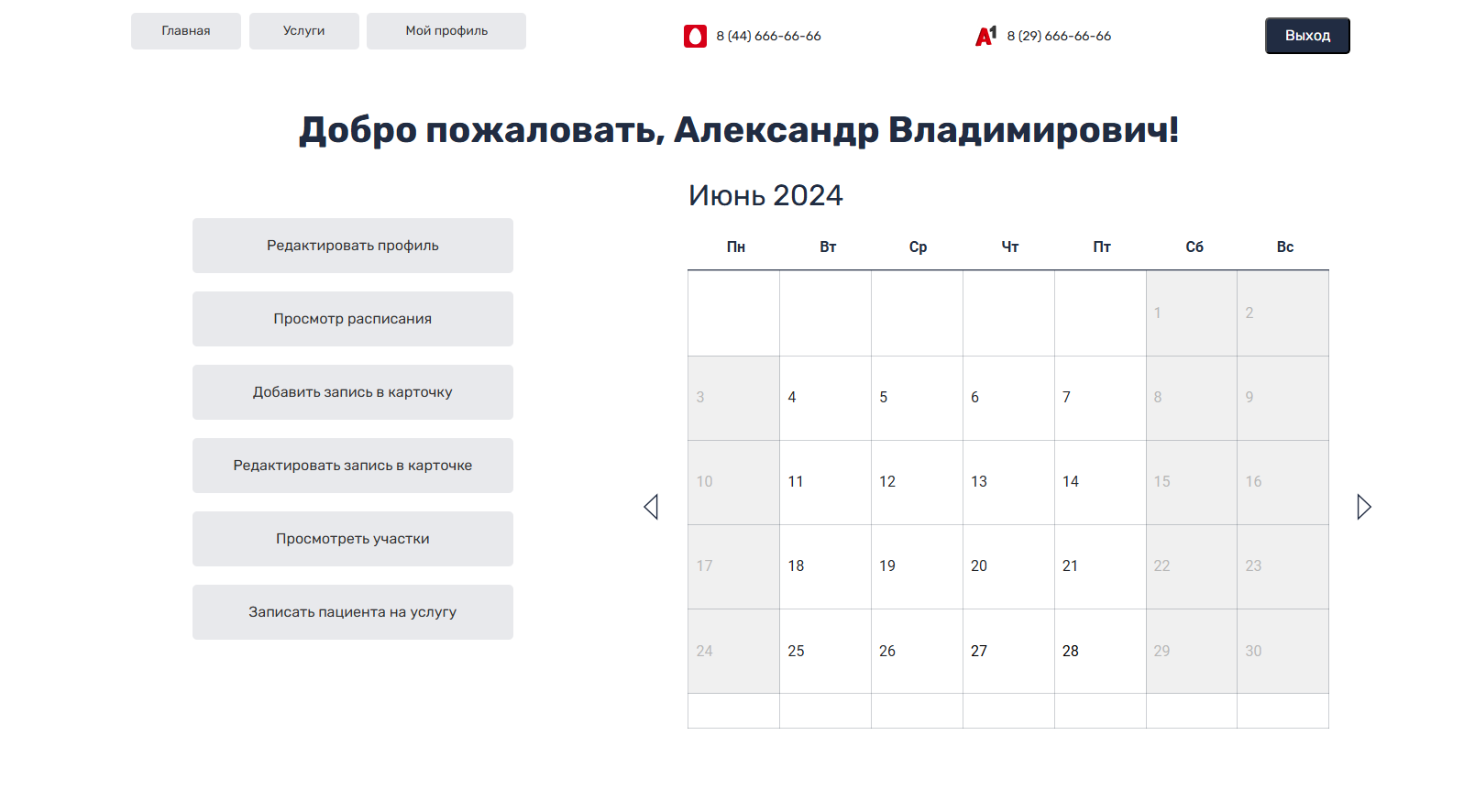


Рисунок Б.10 – Страница профиля специалиста

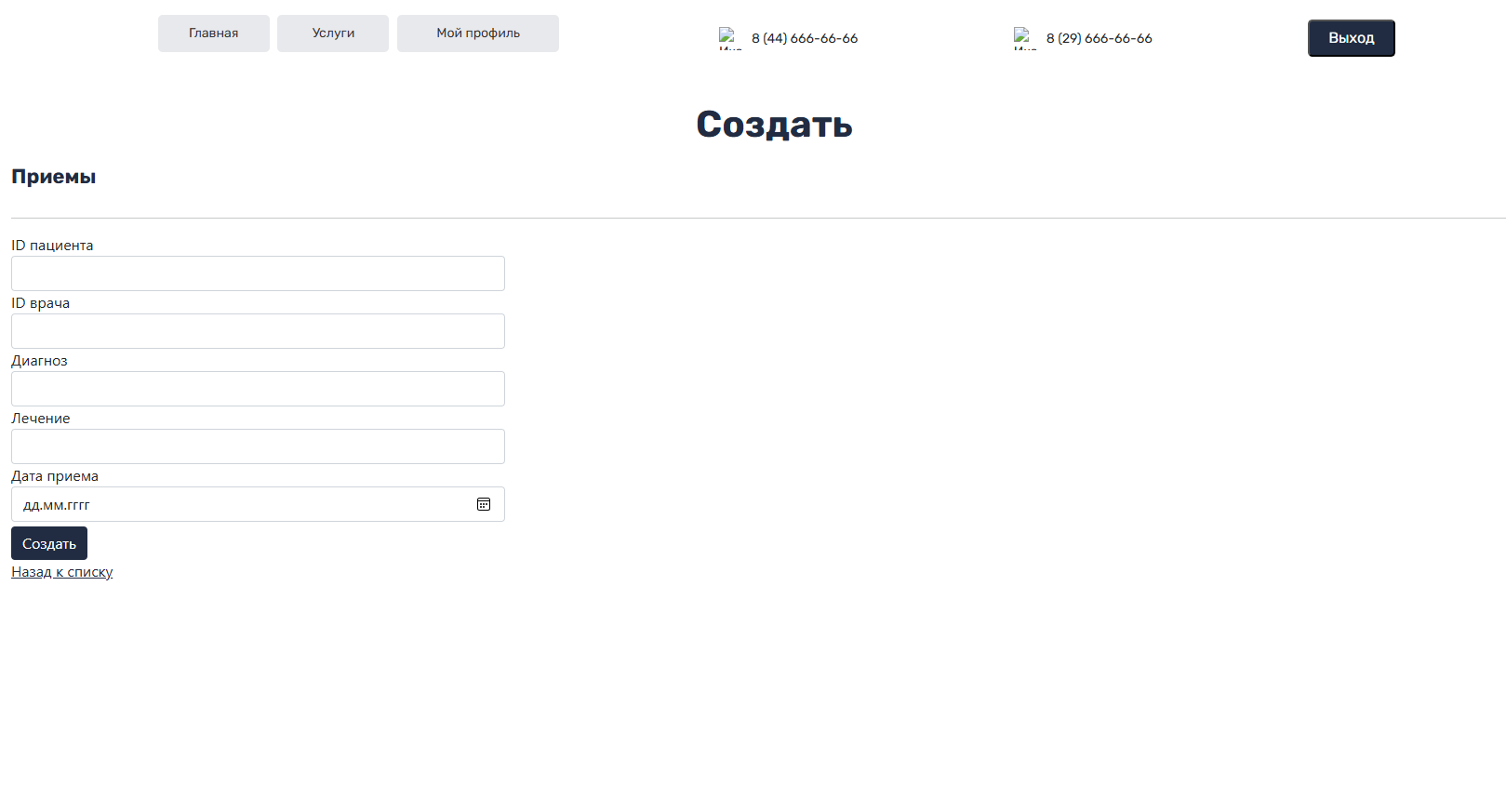


Рисунок Б.11 – Страница создания записи на прием

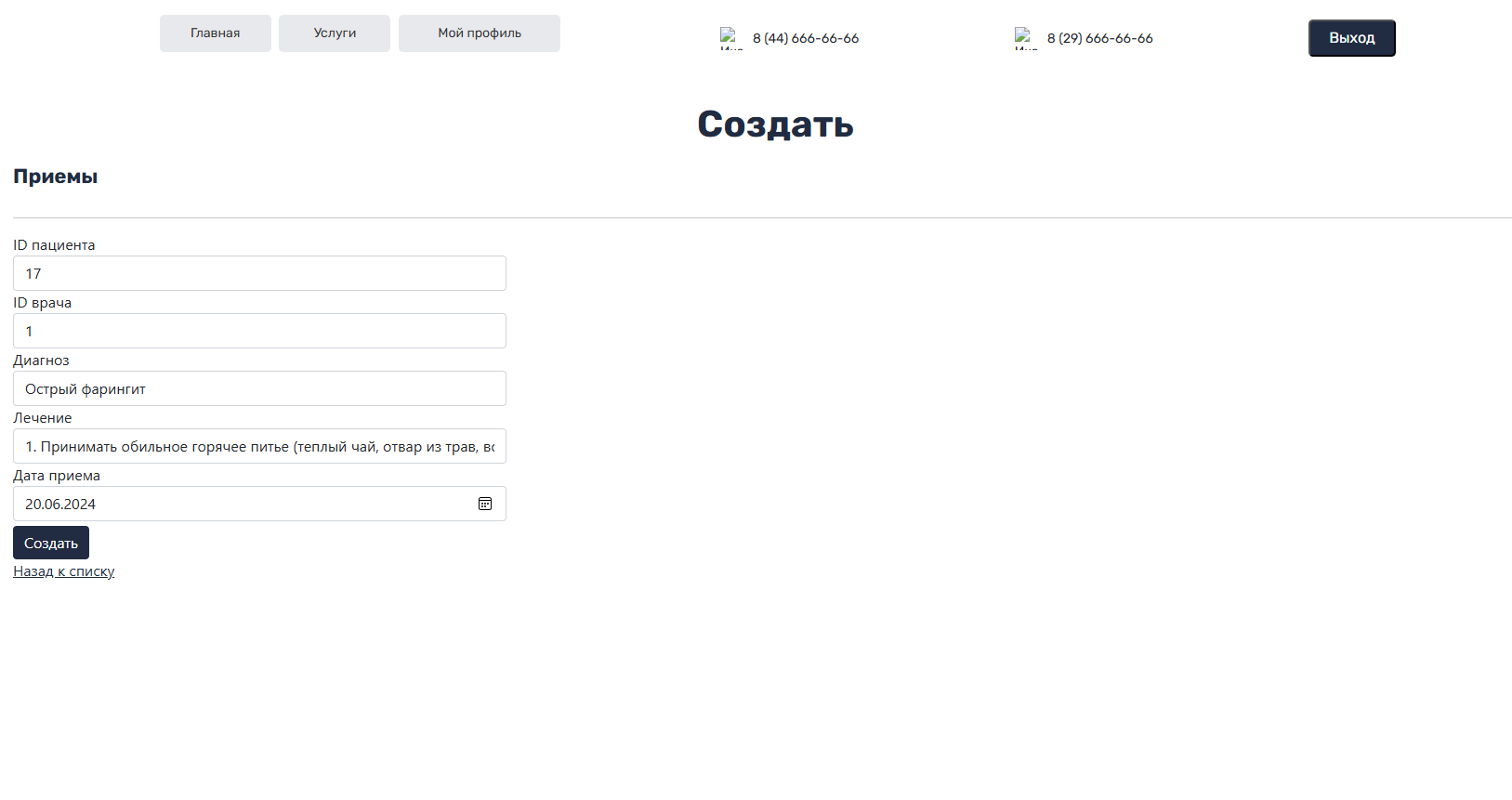


Рисунок Б.12 – Страница создания записи на прием с заполненными полями

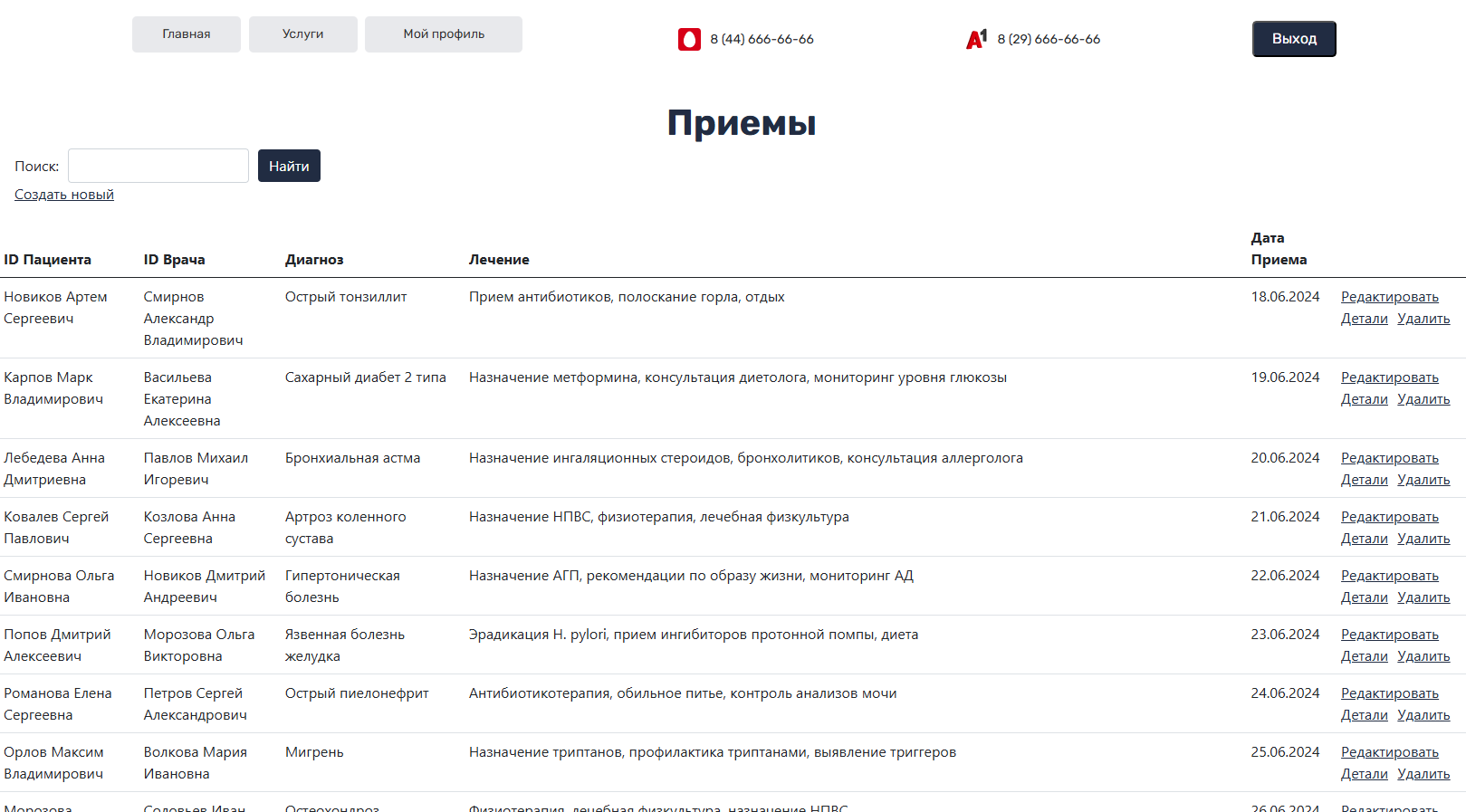


Рисунок Б.13 – Страница приемов

КБП

У

Т. Контр.

Разраб. Разраб.

Казутина С.Е.

Провер. Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 5

Инв.№подл.

Подп. и дата

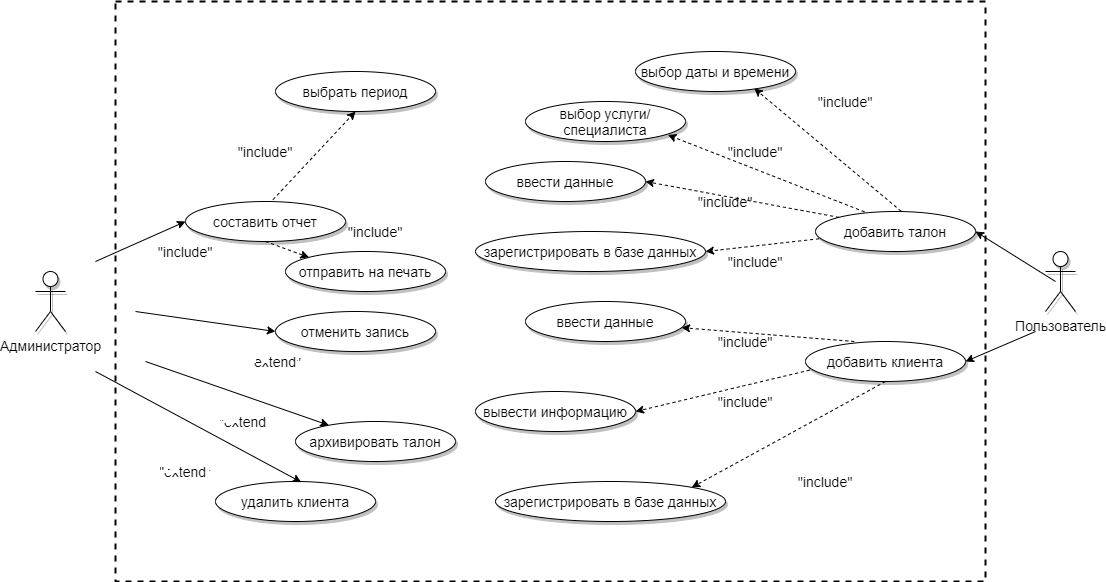
Взам.инв.№

Инв.№дубл.

Подп. и дата

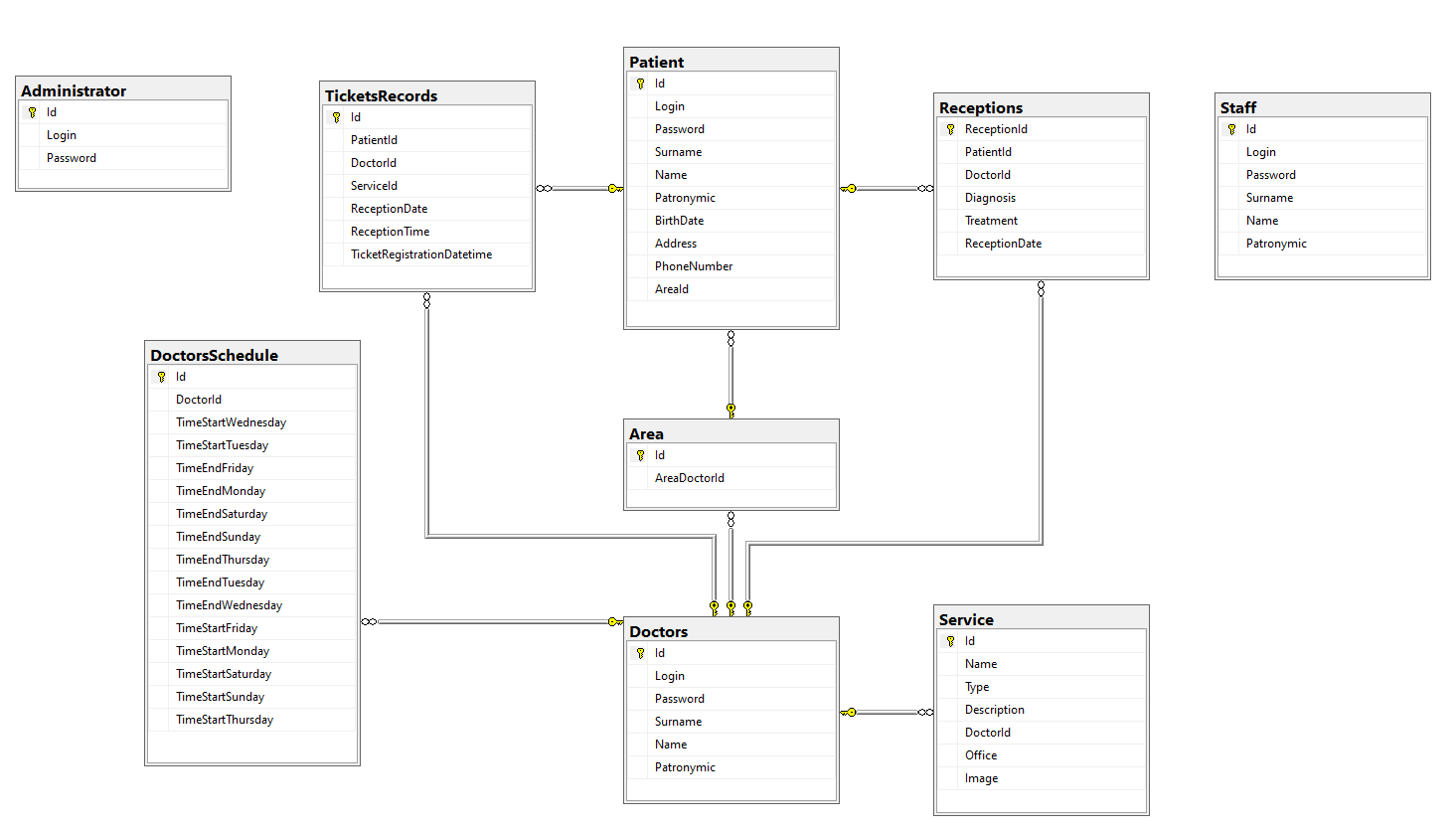
КП Т.197007.401 ГЧ Т.ХХХХХХ.401 ГЧ

*КП Т.197007.401 ГЧ*



*Диаграмма вариантов использования*

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*



КБП

У

Т. Контр.

Разраб. Разраб.

Казутина С.Е.

Провер. Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 5

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

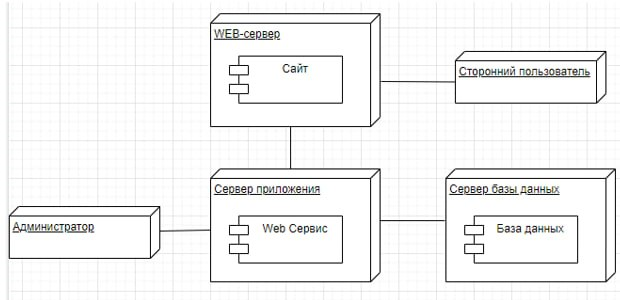
Подп. и дата

КП Т.197007.401 ГЧ Т.ХХХХХХ.401 ГЧ

*КП Т.197007.401 ГЧ*

*Диаграмма классов*

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*



*Диаграмма развертывания*

КБП

У

Т. Контр.

Разраб. Разраб.

Казутина С.Е.

Провер. Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 3

Листов 5

Инв.№подл.

Подп. и дата

Взам.инв.№

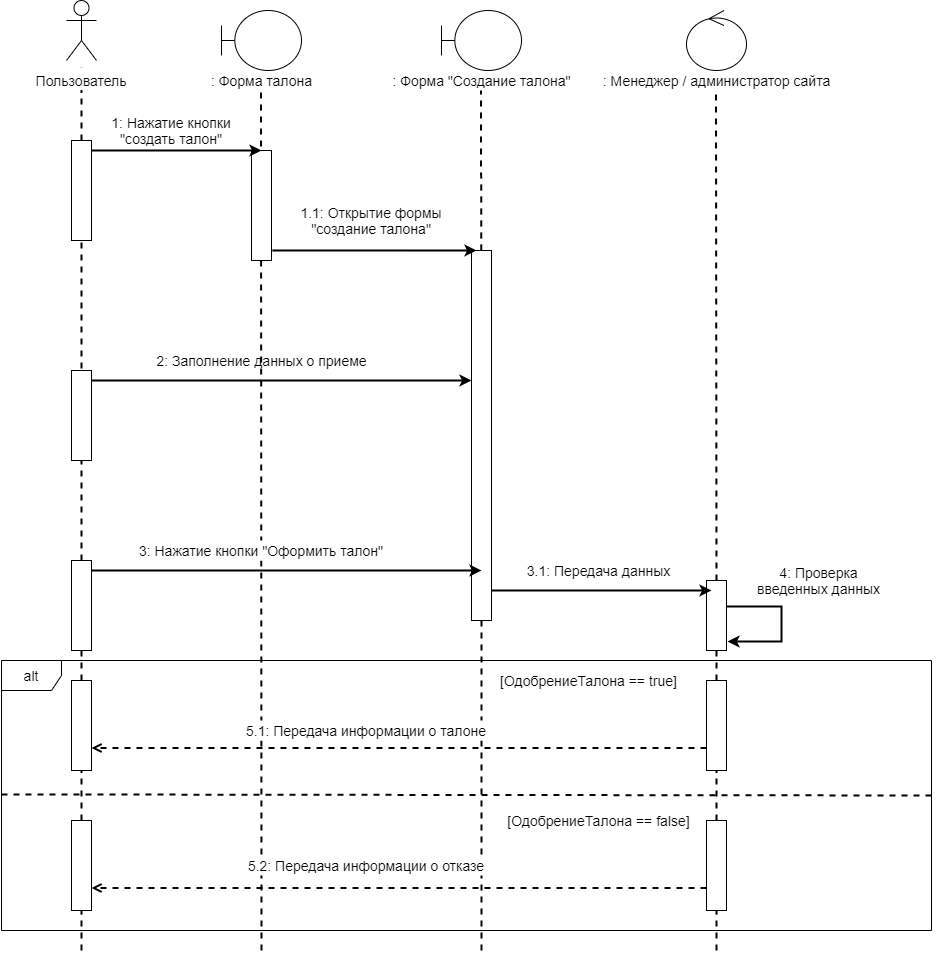
Инв.№дубл.

Подп. и дата

КП Т.197007.401 ГЧ Т.ХХХХХХ.401 ГЧ

*КП Т.197007.401 ГЧ*

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*



*Диаграмма последовательности (для функции создания пользователем талона)*

КБП

У

Т. Контр.

Разраб. Разраб.

Казутина С.Е.

Провер. Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 4

Листов 5

Инв.№подл.

Подп. и дата

Взам.инв.№

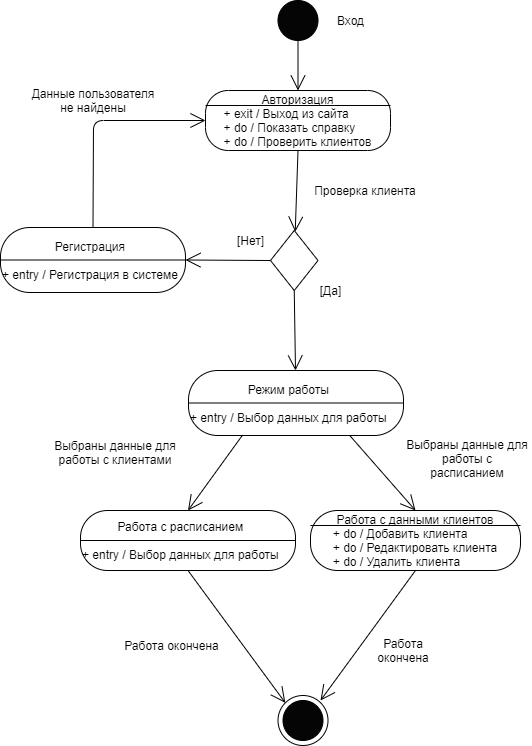
Инв.№дубл.

Подп. и дата

КП Т.197007.401 ГЧ Т.ХХХХХХ.401 ГЧ

*КП Т.197007.401 ГЧ*

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*



*КП Т.197007.401 ГЧ*

*Диаграмма состояний (для административной части)*

КБП

У

Т. Контр.

Разраб. Разраб.

Казутина С.Е.

Провер. Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 5

Листов 5

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.197007.401 ГЧ Т.ХХХХХХ.401 ГЧ

*Создание веб-приложения для автоматизации заказа талонов и карточек в медцентре «MedicalClinic»*