**Содержание**

[Введение 4](#_Toc167025626)

[1 Общая часть 6](#_Toc167025627)

[1.1 Структура предприятия, место и функции информационного отдела 6](#_Toc167025628)

[1.2 Анализ бизнес-процессов и постановка задачи 12](#_Toc167025629)

[1.3 Анализ средств разработки 13](#_Toc167025630)

[2 Специальная часть 15](#_Toc167025631)

[2.1 Проектирование архитектуры решения 15](#_Toc167025632)

[2.2 Проектирование базы данных 17](#_Toc167025633)

[2.3 Проектирование программных модулей решения 33](#_Toc167025634)

[2.4 Реализация технического решения 35](#_Toc167025635)

[2.4.1 Разработка пользовательского интерфейса решения 35](#_Toc167025636)

[2.4.2 Разработка программных модулей решения 57](#_Toc167025637)

[2.5 Тестирование разработанного решения 63](#_Toc167025638)

[3 Организационно-экономическая часть 68](#_Toc167025639)

[3.1 Расчет сметы затрат на разработку программы 68](#_Toc167025640)

[3.2 Расчет экономической эффективности внедрения программы 77](#_Toc167025641)

[4 Мероприятия по технике безопасности и охране окружающей среды 80](#_Toc167025642)

[4.1 Организация безопасности труда на предприятии 80](#_Toc167025643)

[4.2 Техника безопасности при работе с вычислительной техникой 81](#_Toc167025644)

[Заключение 82](#_Toc167025645)

[Список использованных источников 83](#_Toc167025646)

# **Введение**

Автоматизация учета данных регистратуры поликлиники — это внедрение информационной системы, которая позволяет автоматизировать процессы записи на прием к врачу, учета посещений пациентов, выписки направлений и других медицинских документов, а также управления расписанием работы врачей.

Актуальность применения автоматизации учета данных регистратуры поликлиники заключается в том, что современные медицинские учреждения все чаще стремятся автоматизировать свои процессы, что позволяет значительно упростить и ускорить работу медицинского персонала, а также повысить качество и безопасность предоставляемых медицинских услуг.

Практическая значимость подобных приложений заключается в следующем:

1. Удобство хранения и доступа к медицинским данным пациентов. Автоматизация учета данных регистратуры поликлиники позволяет хранить все данные пациента в одном месте, что облегчает их поиск и доступность для врачей и медицинского персонала.
2. Улучшение качества предоставляемых медицинских услуг. Благодаря автоматизации учета данных регистратуры поликлиники врачи имеют доступ к полной и актуальной информации о пациенте, что помогает им принимать обоснованные решения и предоставлять более качественное лечение.
3. Сокращение времени на обработку и анализ медицинских данных. Автоматизация процессов сбора, хранения и анализа данных позволяет сократить время, затрачиваемое на рутинные операции, и увеличить эффективность работы медицинского персонала.
4. Улучшение безопасности данных. Автоматизация учета данных регистратуры поликлиники позволяет установить ограничения доступа к конфиденциальной информации, что повышает безопасность медицинских данных пациентов.

Основой любой электронной системы является база данных.

База данных (БД) — это структурированная коллекция данных, которая используется для хранения информации о пациентах и их медицинской истории. Любая база данных создается при помощи систем управления базами данных.

Система управления базами данных (СУБД) — это специализированное программное обеспечение, предназначенное для управления и организации данных в базах данных.

Во время прохождения производственной практики в ГБУЗ «Районная больница г. Верхнеуральск», было дано задание, которое заключалось в разработке системы, которая позволит вести учет данных пациентов в электронном виде.

В соответствии с полученным заданием на производственной практике была определена цель дипломного проекта: «Автоматизация учета данных регистратуры поликлиники».

Исходя из цели работы был описан следующий круг задач дипломного проекта:

1. Изучение требований заказчика и проведение анализа предметной области — данные пациента и их учет.
2. Проектирование модели базы данных.
3. Создание пользовательского интерфейса учета данных регистратуры поликлиники.
4. Реализация функциональных возможностей электронной системы.
5. Тестирование и отладка приложения для обеспечения его корректной работы и соответствия требованиям.
6. Оформление технической документации в виде пояснительной записки на дипломный проект.

# **1 Общая часть**

# **1.1 Структура предприятия, место и функции информационного отдела**

Районная больница города Верхнеуральск – осуществляет свою деятельность в соответствии с предметом и целями деятельности, согласно общепринятым стандартам качества оказываемой медицинской помощи определенными в соответствии с законами и иными нормативными правовыми актами Российской Федерации.

Учреждение функционирует как амбулатория, медицинский центр, поликлиника для взрослых, диагностический центр, больница и родильный дом, где ведут прием врачи различных специализаций.

Поликлиника для взрослых ГБУЗ Районная больница г. Верхнеуральск пользуется современным оборудованием. Высококвалифицированные специалисты, которые качественно обслуживают своих пациентов, а также приемлемая стоимость на услуги приносит мед центру ГБУЗ Районная больница г. Верхнеуральск высокую популярность. На web-сайте центра вам предоставляется информация о ценах на лечение и возможность записаться на прием онлайн или по номеру телефона. Онлайн-записи к врачу обрабатываются круглыми сутками. Лечение наших пациентов благодаря хорошим результатам ультрасовременной диагностики выполняется на достойном уровне. ГБУЗ Районная больница г. Верхнеуральск предоставляет индивидуальный подход к каждому пациенту, нет очередей и созданы все комфортные условия для пациентов.

Целями ГБУЗ Районная больница г. Верхнеуральск являются:

* обеспечение высокого качества и доступности оказываемой медицинской помощи;
* соблюдение требований Территориальной программы государственных гарантий оказания гражданам бесплатной медицинской помощи;
* соблюдение стандартов и порядков оказания медицинской помощи;
* повышение профессионального уровня работников;
* создание благоприятных условий работникам для выполнения своих должностных обязанностей;
* соблюдение Кодекса медицинского работника;
* эффективное расходование заработанных финансовых средств.

В больнице работают специалисты, занимающиеся информационно-технической деятельностью, а именно: инженер-программист, оператор ЭВМ, системный администратор.

Основные функции для специалистов информационно-технической деятельности являются:

* выполнение работ по разработке, тестированию, отладки, внедрению и сопровождению программного обеспечения;
* ведение технологической документации;
* организация доступа сотрудников к информационным ресурсам, представленным в электронном виде;
* обеспечение бесперебойной работы сетевого оборудования, компьютерной техники коллективного использования, оборудования передачи данных, устройств ввода и вывода данных (принтер, сканер, микрофон и так далее).

Инженер-программист — это человек, который применяет принципы разработки программного обеспечения при проектировании, разработке, обслуживании, тестировании и оценке компьютерного программного обеспечения.

Инженер-программист обязан:

1. Принимать участие в проведении работ по разработке, внедрению, опытной эксплуатации, наладке, модернизации, сопровождению.
2. На основе анализа математической модели и алгоритмов решения задач мониторинга и управления технологических процессов своевременно разрабатывать программное обеспечение, проводить его тестирование и отладку.
3. Разрабатывать технологию решения задачи по всем этапам обработки информации.
4. Осуществлять освоение программного обеспечения автоматизированных систем.
5. Разрабатывать информационное обеспечение систем управления.
6. Выполнять работу по отладке программного обеспечения, проводить корректировку разработанной программы на основе анализа выходных данных.
7. Определять объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программы их функциональному назначению.
8. Разрабатывать инструкции по работе с программным обеспечением, оформлять необходимую техническую документацию в соответствии со стандартами и документами.
9. Обеспечивать объективность и достоверность информации, материалов и отчетности, а также в пределах своей компетенции, конфиденциальность информации, составляющую коммерческую или служебную тайну.



Рисунок 1 – Организационная структура

Для автоматизации деятельности предприятия используются персональные компьютеры, ноутбуки и моноблоки для сотрудников, сервера для обеспечения доступа к базе данных и общим сетевым ресурсам, мультифункциональным устройствам для работы с документами (принтер, сканер, МФУ), высокоскоростной доступ в Интернет. Удаленные здания подключаются с помощью VPN, который шифрует все данные.

Виртуальная частная сеть (VPN) — технология, которая позволяет установить безопасное подключение к сети Интернет.

При подключении к Интернету через VPN, программное обеспечение создает безопасное соединение между вашим устройством и удаленным VPN-сервером, шифруя ваши данные просмотра веб-страниц и скрывая ваш IP-адрес.

Для более удобного просмотра всех комплектующих установленных в устройство можно просмотреть в таблице 1.

Таблица 1 – Технические характеристики ПК, ноутбука и моноблока.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Характеристики устройства** | | |
| **ПК** | **Ноутбука** | **Моноблока** |
| Материнская плата | GIGABYTE B450M H | 15-dk2050ur | Intel® 300 серии H310 |
| Процессор | AMD Ryzen 3 | Intel Core i5 10300H | Intel Core i3, Core i5, Core i7  H310 |
| Видеокарта | Встроенная | NVIDIA GeForce GTX 1650 Ti - 4 ГБ | Intel® UHD Graphics Engine 600 серии |
| Оперативная память | DDR4 8G 16G 2666MHz | DDR4 8G 3200MHz | DDR4 8G 16G 2400/2666MHz |
| Жесткий диск | 250 - 500 ГБ | - | 1 — 6 Т |
| Тверд. Тел. накопитель | 512 ГБ | 512 ГБ | 120 — 512 ГБ |
| Блок питания | Sven 350W | Li-Ion | 150-200W |
| Монитор | Samsung SyncMaster S22A200B | - | - |

Совокупность программ и сопровождающей их документации, предназначенная для решения задач на ПК, называется программным обеспечением.

Программное обеспечение делится на:

* системное программное обеспечение;
* прикладное программное обеспечение;
* инструментальное программное обеспечение.

Системное программное обеспечение – совокупность программ и программных комплексов для обеспечения работы компьютера и сетей.

Прикладное программное обеспечение – комплекс взаимосвязанных программ для решения задач определенного класса или конкретной предметной области.

Инструментальное программное обеспечение – программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ.

В организации используются следующее системное программного обеспечение, представленное в таблицах 2 и 3.

Таблица 2 – Используемые архиваторы.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Winrar | Архиватор файлов для 32- и 64-разрядных операционных систем Windows, позволяющий создавать, изменять и распаковывать множество архивов |
| 7-Zip | Свободный файловый архиватор с высокой степенью сжатия данных. Поддерживает несколько алгоритмов сжатия и множество форматов данных, включая собственный формат 7z c высокоэффективным алгоритмом сжатия LZMA |
| Ark | Программа для работы с архивами различных форматов в среде KDE. С помощью Ark архивы можно просматривать, извлекать, создавать и изменять. Она может обрабатывать различные форматы (tar, gzip, bzip2, rar, zip, lha) |

Таблица 3 – Используемые операционные системы.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| OS Windows 7/8/10/11 | Операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT |
| OS AstraLinux (OREL) версии 12.2.40 и выше | Операционная система на базе ядра Linux, которая внедряется в России в качестве альтернативы Microsoft Windows. Обеспечивает степень защиты обрабатываемой информации до уровня государственной тайны «особой важности» включительно |

Прикладное программное обеспечение, используемое в организации:

* Microsoft Word - текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов;
* Microsoft Excel - программа для работы с электронными таблицами, созданная корпорацией Microsoft;
* LibreOffice Writer - текстовый процессор и визуальный редактор HTML, входящий в состав офисного пакета LibreOffice;
* LibreOffice Calc - табличный процессор, входящий в состав офисного пакета LibreOffice. Является ответвлением табличного процессора OpenOffice Calc;
* Adobe Acrobat — пакет программ, выпускаемый с 1993 года компанией Adobe Systems и предназначенный для создания и просмотра электронных публикаций в формате PDF;
* ABBYY FineReader - программа для оптического распознавания символов, разработанная международной компанией ABBYY. Программа позволяет переводить изображения документов в электронные редактируемые форматы;
* ANYDesk - Приложение для удаленного рабочего стола, распространяемое компанией AnyDesk Software GmbH. Проприетарное программное обеспечение обеспечивает независимый от платформы удаленный доступ к персональным компьютерам и другим устройствам, на которых запущено основное приложение.

Инструментальное программное обеспечение, используемое в организации:

* Visual Studio - линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств;
* Mircosoft SQL Server Management Studio - Утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера;
* Медицинская информационная система «БАРС.Здравоохранение»;
* Visual Studio Code - Текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

# **1.2 Анализ бизнес-процессов и постановка задачи**

Проблема, выявленная в результате анализа деятельности регистратуры поликлиники, заключается в неэффективном учете и обработке данных. В настоящее время процессы учета информации о посещениях пациентов, записи на прием к врачам, выписке направлений и других административных процедурах осуществляются вручную, что часто приводит к ошибкам, задержкам и неэффективному использованию ресурсов.

Для решения данной проблемы в будущем планируется внедрение системы автоматизации учета данных регистратуры поликлиники. Это позволит ускорить процессы обработки информации, снизить вероятность ошибок, улучшить качество обслуживания пациентов и оптимизировать рабочие процессы сотрудников регистратуры.

Основанием для разработки проекта автоматизации учета данных регистратуры поликлиники послужило стремление к повышению эффективности работы медицинского учреждения, улучшению качества обслуживания пациентов и сокращению временных затрат на административные процессы.

Автоматизация учета данных также позволит лучше контролировать информацию, предотвращать потерю данных и обеспечить более точный анализ статистической информации для принятия управленческих решений.

# **1.3 Анализ средств разработки**

Для разработки автоматизация учета данных регистратуры поликлиники была выбрана среда разработки Visual Studio и Radzen Blazor Studio. Выбор пал именно на эти среды, потому что имеется большое количество документации и электронных ресурсов, которые позволяют решить трудности возникшие на момент разработки. Visual Studio поддерживает универсальный и гибкий язык программирования, а также файлы совместимые с системными. Удобный интерфейс, а также проведение тестирования с использованием точек остановок. Radzen Blazor Studio предоставляет возможность создания веб-приложений на платформе Blazor, что позволяет создать современное и эффективное приложение с использованием технологии WebAssembly.

СУБД была выбрана SQLite — это компактная и легковесная СУБД, которая хранит базы данных в одном файле, что удобно для развертывания и поддержки. SQLite обладает хорошей производительностью и поддерживает стандарт SQL, что обеспечит надежное хранение данных для вашего проекта. Также она была выбрана, потому что хорошо интегрируется с проектами от Microsoft, к примеру .Net Framework.

Технология доступа к данным пала на SQLiteStudio, потому что она является гибкой и легко переносимой вместе с проектом, что позволяет быстро вносить какие-либо изменения или дополнения в проекте, а также в самой базе данных. Также она обладает высокой надежностью, удобным интерфейсом, большим набором инструментов.

Язык программирования использовал C#, потому что имеется опыт в работе с ним, а также он легок в изучение. Наличие обширной документации на сайтах, которые помогают при возникновении вопросов. Он также имеет интеграцию с платформой .NET и обладает высокой надежностью и много поточностью.

# **2 Специальная часть**

# **2.1 Проектирование архитектуры решения**

В приложении есть несколько пользователей: администратор, регистратор, доктор и пациент.

Пациент обращается к регистратору с целью пройти осмотр, регистратор осуществляет проверку наличия учетной записи пользователя в системе и осуществляет запись к нужному доктору, если учетной записи пользователя нету, то регистратор начинает добавлять данные пациента в систему, а после чего записывает к нужному доктору. Также если есть какие-то ошибки в уже имеющейся учетной записи пациента, регистратор может их изменить.

Администратор осуществляет операции по добавлению, редактированию и удалению пользователей системы.

После того как запись пациента к нужному врачу состоялась, доктор проводит осмотр пациента и заполняет данные в систему. Также имеется возможность изменения данных проведенного осмотра или удаление осмотра.



Рисунок 2 – Диаграмма Use-Case

Ход реализации программы:

Определение логики работы системы.

Создание проекта в Radzen Blazor Studio, определение цвета компонентов для разработки и стиля текста.

Создание базы данных в SQLite, построение таблиц, для хранения данных об пользователях, проведенных осмотрах и так далее.

Проектирование интерфейса, продумывание визуальной части системы.

Реализация функциональных возможностей для работы с данными (добавление, редактирование, удаление, экспорт, отправка сообщений и так далее).

# **2.2 Проектирование базы данных**

Входная информация – это информация, которая подается на вход в систему, процесс или программу. Она может быть представлена в виде текста, чисел, изображений, звуковых файлов и так далее. Входная информация служит для выполнения определенных операций, вычислений или обработки данных.

Входные данные разработанного приложения: логин и пароль, с помощью которых осуществляются вход в систему, данные о пациенте (фамилия, имя, отчество, паспортные данные, полис и так далее), которые необходимы для регистрации пациента в системе, данные вводимые пользователем при осмотре пациента.

Таблица 4 – «Gender» (Пол)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор пола |
| Title | Строковый (NVARCHAR(10)) | Название пола |

Таблица 5 – «Diagnosis» (Диагноз)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор диагноза |
| Title | Строковый (NVARCHAR(250)) | Название диагноза |

Таблица 6 – «Visit» (Посещение)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор посещения |
| DateVisit | Дата (DATE) | Дата посещения |
| TimeVisit | Время (TIME) | Время посещения |
| Complaint | Строковый (NVARCHAR(500)) | Жалоба |
| Appointments | Строковый (NVARCHAR(500)) | Назначение лечения |
| Status | Строковый (NVARCHAR(20)) | Статус осмотра |

Таблица 7 – «Schedule» (Расписание)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор расписания |
| DateStart | Дата (DATE) | Дата начала работы |
| TimeStart | Время (TIME) | Время начала работы |
| TimeEnd | Время (TIME) | Время окончания работы |
| Duration | Целочисленный (INTEGER) | Длительность приема |
| Cabinet | Строковый (NVARCHAR(30)) | Кабинет |

Таблица 8 – «Doctor» (Доктор)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор доктора |
| Surname | Строковый (NVARCHAR(50)) | Фамилия доктора |
| Name | Строковый (NVARCHAR(50)) | Имя доктора |
| Forename | Строковый (NVARCHAR(50)) | Отчество доктора |
| Phone | Строковый (NVARCHAR(20)) | Телефон доктора |

Таблица 9 – «Specialization» (Специализация)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор специализации |
| Title | Строковый (NVARCHAR(100)) | Название специализации |

Таблица 10 – «AspNetUserClaims» (Утверждения к пользователю)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор утверждения |
| ClaimType | Текстовый (TEXT) | Тип утверждения |
| ClaimValue | Текстовый (TEXT) | Значение утверждения |

Таблица 11 – «AspNetRoles» (Роли)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор роли |
| ConcurrencyStamp | Текстовый (TEXT) | Штамп параллелизма |
| Name | Текстовый (TEXT) | Имя роли |
| NormalizedName | Текстовый (TEXT) | Нормализованное имя |

Таблица 12 – «AspNetRoleClaims» (Утверждения к роли)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор утверждения |
| ClaimType | Текстовый (TEXT) | Тип утверждения |
| ClaimValue | Текстовый (TEXT) | Значение утверждения |

Таблица 13 – «AspNetUserLogins» (Логины пользователей)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| LoginProvider | Текстовый (TEXT) | Идентификатор поставщика логина |
| ProviderKey | Текстовый (TEXT) | Идентификатор ключа провайдера |
| ProviderDisplayName | Текстовый (TEXT) | Отображаемое имя поставщика |

Таблица 14 – «AspNetUserTokens» (Токены пользователей)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| UserId | Текстовый (TEXT) | Идентификатор пользователя |
| LoginProvider | Текстовый (TEXT) | Идентификатор поставщика логина |
| Name | Текстовый (TEXT) | Идентификатор имени токена |
| Value | Текстовый (TEXT) | Значение токена |

Таблица 15 – «\_\_EFMigrationsHistory» (История миграций)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| MigrationId | Текстовый (TEXT) | Идентификатор миграции |
| ProductVersion | Текстовый (TEXT) | Версия продукта |

Таблица 16 – «AspNetUsers» (Пользователи)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Текстовый (TEXT) | Идентификатор пользователя |
| AccessFailedCount | Целочисленный (INTEGER) | Количество неудачных попыток доступа |
| ConcurrencyStamp | Текстовый (TEXT) | Штамп параллелизма |
| Email | Текстовый (TEXT) | Электронная почта |
| EmailConfirmed | Целочисленный (INTEGER) | Подтвержденный адрес электронной почты |
| LockoutEnabled | Целочисленный (INTEGER) | Включена блокировка |
| LockoutEnd | Текстовый (TEXT) | Завершение блокировки |
| NormalizedEmail | Текстовый (TEXT) | Нормализованная электронная почта |
| NormalizedUserName | Текстовый (TEXT) | Нормализованное имя пользователя |
| PasswordHash | Текстовый (TEXT) | Хэш пароля пользователя |
| PhoneNumber | Текстовый (TEXT) | Номер телефона пользователя |
| PhoneNumberConfirmed | Целочисленный (INTEGER) | Подтвержденный номер телефона |
| SecurityStamp | Текстовый (TEXT) | Защитный штамп |
| TwoFactorEnabled | Целочисленный (INTEGER) | Включен двухфакторный режим |
| UserName | Текстовый (TEXT) | Имя пользователя |

Таблица 17 – «Patient» (пациент)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| Id | Целочисленный (INTEGER) | Идентификатор пациента |
| Surname | Строковый (NVARCHAR(50)) | Фамилия пациента |
| Name | Строковый (NVARCHAR(50)) | Имя пациента |
| Forename | Строковый (NVARCHAR(50)) | Отчество пациента |
| PlaceResidence | Строковый (NVARCHAR(255)) | Место жительства пациента |
| Address | Строковый (NVARCHAR(100)) | Адрес пациента |
| Phone | Строковый (NVARCHAR(20)) | Телефон пациента |
| Email | Строковый (NVARCHAR(50)) | Электронная почта |
| DateOfBirth | Дата (DATE) | Дата рождения |
| NumberCard | Целочисленный (INTEGER) | Номер карты |
| MHIpolicy | Строковый (NVARCHAR(19)) | Полюс ОМС |
| VMIpolicy | Строковый (NVARCHAR(19)) | Полюс ДМС |
| PassportSeries | Строковый (NVARCHAR(5)) | Серия паспорта |
| PassportNumber | Строковый (NVARCHAR(6)) | Номер паспорта |
| DateOfIssue | Дата (DATE) | Дата получения паспорта |

Таблица 18 – «AspNetUserRoles» (Роли пользователей)

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип данных** | **Комментарий** |
| UserId | Текстовый (TEXT) | Идентификатор пользователя |
| RoleId | Текстовый (TEXT) | Идентификатор роли |

Выходная информация – это информация, которая предоставляется или передается системой, или устройством наружу или другой системе, или устройству. Она может быть представлена в различных формах, таких как текст, изображения, звук или видео.

Выходная информация разработанного приложения организована в виде вывода данных о пациенте (рисунок 4).

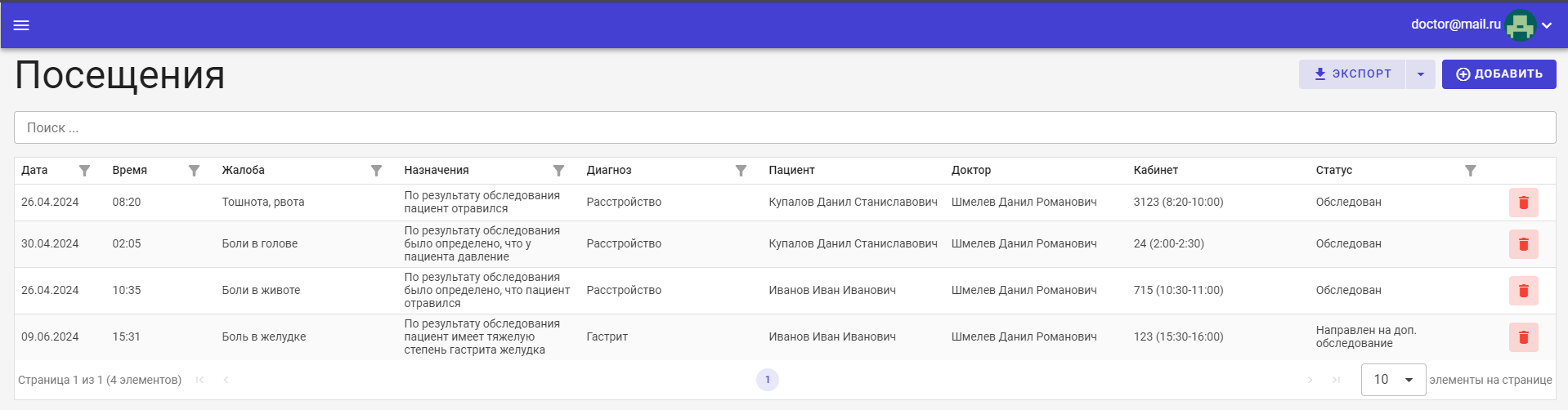


Рисунок 4 – Вывод осмотра пациента

Модель базы данных разработана на основе анализа требований заказчика и изучения предметной области – данные пациента и их учет.

Модель базы данных – это абстрактная структура, которая определяет способ организации и хранения данных в базе данных. Она определяет структуру данных, схему таблиц, связей между таблицами и правила целостности данных. Модель базы данных помогает описать и организовать данные для их эффективного использования и обработки. В зависимости от потребностей и особенностей проекта, могут быть использованы различные модели баз данных, такие как реляционная, иерархическая, сетевая, объектно-ориентированная, графовая и др.

Предметная область – область знаний или деятельности, которая имеет определенные характеристики, термины, понятия, методы и принципы, связанные с определенной областью знания или практики. Предметная область может быть узкой (например, физика элементарных частиц) или широкой (например, медицина) и обычно определяет предмет исследования или работы в определенной области. Например, в области информационных технологий предметная область может включать такие области, как программирование, базы данных, сети, компьютерная безопасность и т. д.

В разработанной модели базы данных сущностями являются созданные таблицы:

* «Visit» (Посещения);
* «Diagnosis» (Диагноз);
* «Record» (Запись);
* «Schedule» (Расписание);
* «Doctor» (Доктор);
* «Gender» (Пол);
* «Patient» (Пациент);
* «Specialization» (Специализация);
* «AspNetRoles» (Роли);
* «AspNetRoleClaims» (Утверждения к роли);
* «AspNetUserRoles» (Роли пользователей);
* «AspNetUserClaims» (Утверждения к пользователю);
* «AspNetUserLogins» (Логины пользователей);
* «AspNetUserTokens» (Токены пользователей);
* «AspNetUsers» (Пользователи);
* «\_\_EFMigrationsHistory» (История миграции).

Следует отметить то, что каждая из вышеперечисленных сущностей (таблиц) имеет свои атрибуты (свойства):

1. Таблица «Посещения» необходима для хранения данных о посещениях пациентов нужных докторов с указанием жалоб и направлений.
2. Таблицы «Диагноз» необходима для хранения диагнозов.
3. Таблица «Запись» необходима для хранения данных записанного пациента к нужному врачу на определенное время.
4. Таблица «Расписание» необходима для хранения данных о времени работы каждого врача с указанием кабинета, длительности приема.
5. Таблица «Доктор» необходима для хранения данных о докторе, его специализации и контактных данных.
6. Таблица «Пол» необходима для хранения данных о гендерном поле человека.
7. Таблица «Пациент» необходима для хранения данных о пациенте, которые используются для регистрации и осуществления записи на осмотр.
8. Таблица «Специализация» необходима для хранения данных о специализации доктора.
9. Таблица «Роли» необходима для хранения данных о ролях пользователей.
10. Таблица «Утверждения к роли» необходима для хранения данных об утверждениях, которые присвоены роли.
11. Таблица «Роли пользователей» необходима для хранения данных о пользователе и присвоенных ему ролей.
12. Таблица «Утверждения к пользователю» необходима для хранения данных об утверждениях, которые присвоены пользователю.
13. Таблица «Логины пользователей» необходима для хранений данных о пользователях, которые совершили вход путем использования учетных данных внешних провайдеров (VK, Google и так далее).
14. Таблица «Токены пользователей» необходима для хранения данных о токенов обновления внешних входов для пользователей, которые совершили вход при помощи внешних провайдеров.
15. Таблица «Пользователи» необходима для хранения данных о пользователях приложения. Она содержит информацию о пользователе имя, пароль, почта и так далее.
16. Таблица «История миграции» необходима для хранения данных истории миграции базы данных.

Миграции — это механизм, позволяющий изменять схему базы данных с течением времени, не теряя данных.

Модель базы данных представлена в виде ER-диаграмма (рисунок 5), которая была создана в Microsoft Visio

ER-диаграмма, представленная на рисунке 5, позволяет описать предметную область приложения, сущности базы данных и связи между ними.

ER-диаграмма – это визуальная модель, используемая для описания структуры и взаимодействия между сущностями (сущности-связи) в системе. ER-диаграмма широко используется в области проектирования баз данных для определения сущностей, их атрибутов и связей между ними. Она позволяет разработчикам легко понять структуру данных и связи между ними, что помогает в создании эффективной базы данных. ER-диаграмма состоит из сущностей (объектов), атрибутов (свойств объектов) и связей (отношений между объектами).



Рисунок 5 – ER-модель базы данных

Вышеперечисленные объекты (таблицы) были созданы в системе управления базами данных SQLite. Для реализации автоматического построения схемы базы данных было использовано стороннее приложение DBeaver.

На рисунке 6 представлена физическая модель базы данных.

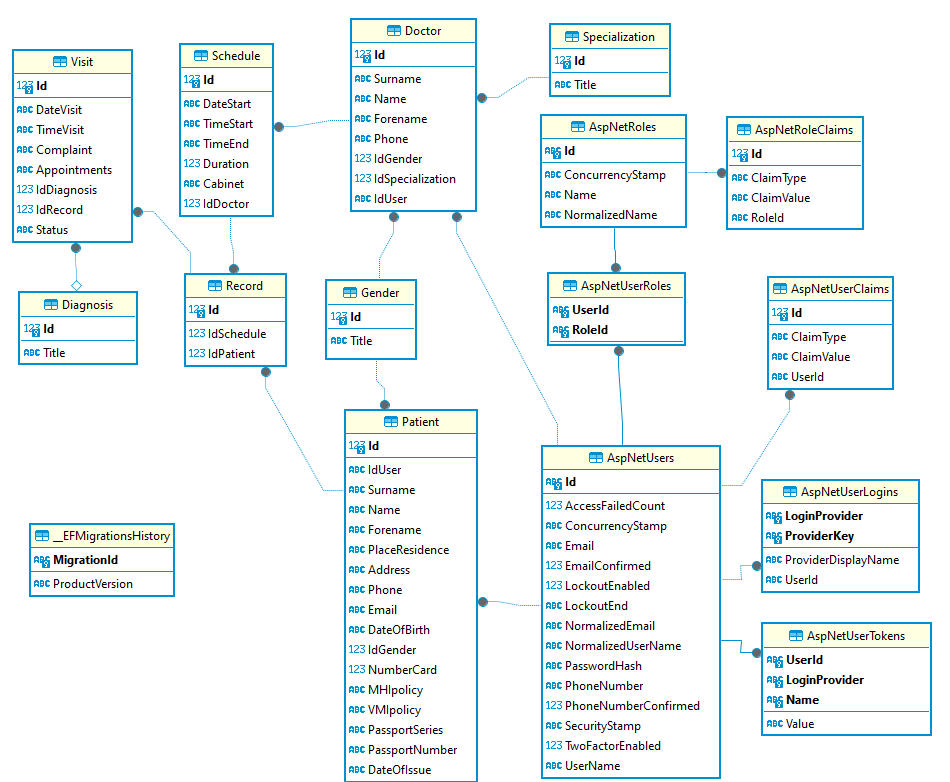


Рисунок 6 – Физическая модель базы данных

Далее приводится описание всех зависимостей между основными сущностями с подробным их описанием.

Таблицы «Диагноз» и «Посещение» находится в отношении «один-ко- многим» (рисунок 7), так как один и тот же диагноз может быть поставлен при неоднократном посещении доктора.



Рисунок 7 – Связь «Visit (Посещения)» с «Diagnosis (Диагноз)»

Таблицы «Запись» и «Посещение» находятся в отношении «один-ко- многим» (рисунок 8), так как одна запись может проводиться при неоднократном посещение доктора.

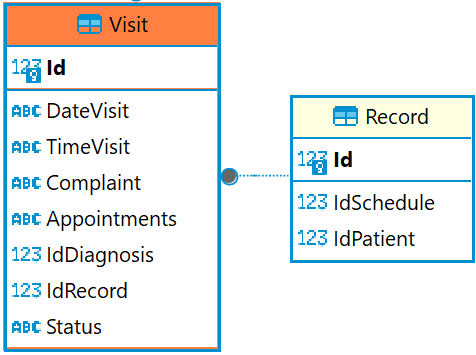


Рисунок 8 – Связь «Visit (Посещения)» с «Record (Запись)»

Таблицы «Запись» и «Пациент» находятся в отношении «один-ко- многим» (рисунок 9), так как один пациент может осуществить несколько записей к доктору.

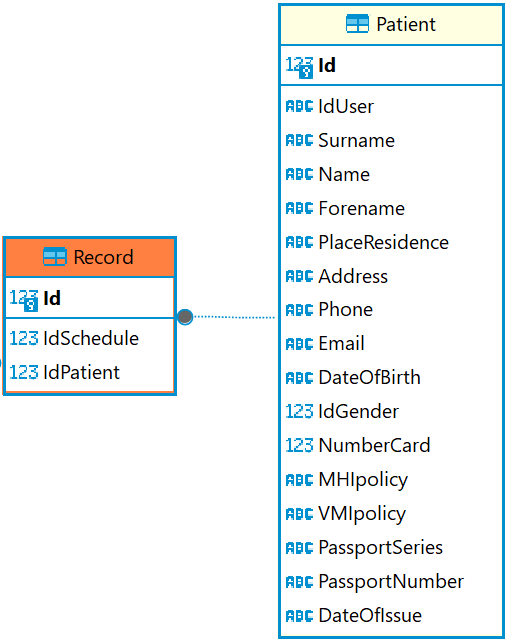


Рисунок 9 – Связь «Record (Запись)» с «Patient (Пациент)»

Таблицы «Запись» и «Расписание» находятся в отношении «один-ко- многим» (рисунок 10), так как одна запись может быть назначена на разное время.

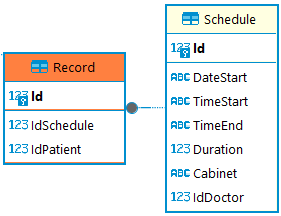


Рисунок 10 – Связь «Record (Запись)» с «Schedule (Расписание)»

Таблицы «Расписание» и «Доктор» находятся в отношение «один-ко-многим» (рисунок 11), так как один доктор имеет несколько временных промежутков для приема пациентов.



Рисунок 11 – Связь «Schedule (Расписание)» с «Doctor (Доктор)»

Таблицы «Специализация» и «Доктор» находятся в отношение «один-ко-многим» (рисунок 12), так как одна специализация может быть у нескольких докторов.

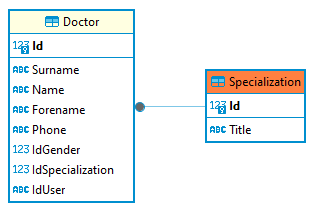


Рисунок 12 – Связь «Specialization (Специализация)» с «Doctor (Доктор)»

Таблицы «Доктор» и «Пол» находятся в отношении «один-ко-многим» (рисунок 13), так как у множества врачей может быть одинаковый пол.

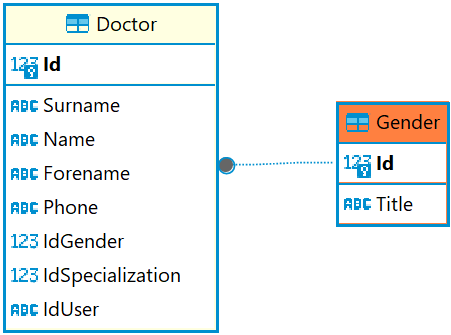


Рисунок 13 – Связь «Doctor (Доктор)» с «Gender (Пол)»

Таблицы «Пациент» и «Пол» находятся в отношении «один-ко-многим» (рисунок 14), так как один и тот же пол может быть у нескольких пациентов.

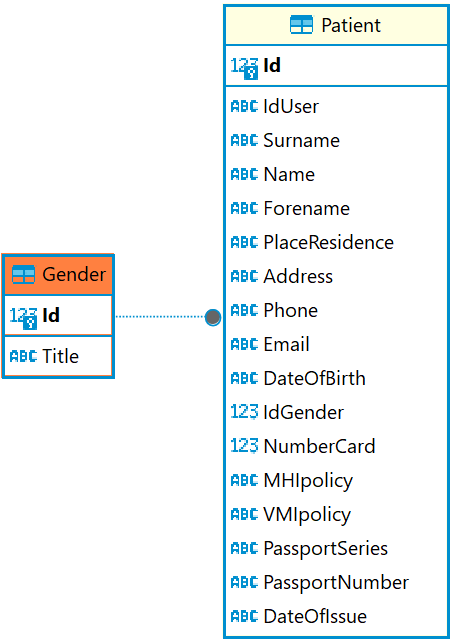


Рисунок 14 – Связь «Gender (Пол)» с «Patient (Пациент)»

Таблицы «Пользователь» и «Пациент» находятся в отношении «один-к-одному» (рисунок 15), так как у каждого пациента своя учетная запись.

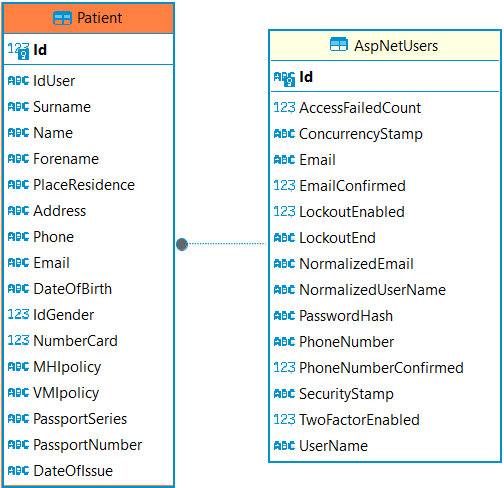


Рисунок 15 – Связь «Patient (Пациент)» с «AspNetUsers (Пользователь)»

Таблицы «Пользователь» и «Доктор» находятся в отношении «один-к-одному» (рисунок 16), так как у каждого доктора своя учетная запись для входа.

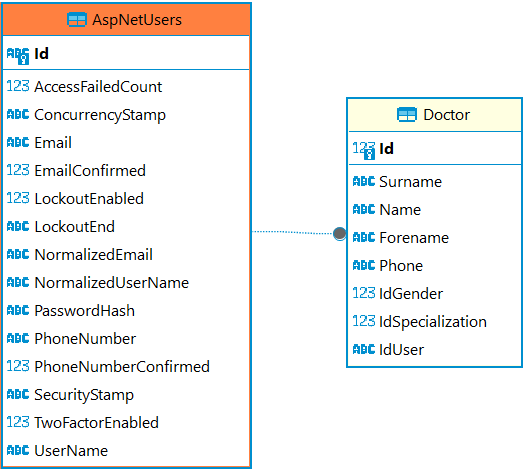


Рисунок 16 – Связь «Doctor (Доктор)» с «AspNetUsers (Пользователь)»

Таблицы «Утверждения к роли» и «Роли в сети» находятся в отношении «один-ко-многим» (рисунок 17), так как одна роль может иметь несколько утверждений.

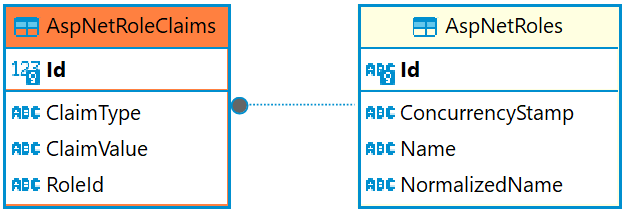


Рисунок 17 – Связь «AspNetRoleClaims (Утверждения к роли)» с «AspNetRoles (Роли)»

Таблицы «Роли» и «Роли пользователя» находятся в отношении «один-к-одному» (рисунок 18), так как одна роль может быть только у одного пользователя.

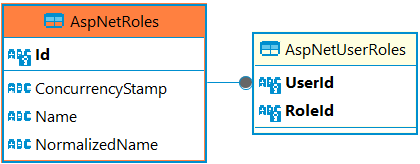


Рисунок 18 – Связь «AspNetRoles (Роли)» с «AspNetUserRoles (Роли пользователей)»

Таблицы «Пользователи» и «Роли пользователя» находятся в отношении «один-к-одному» (рисунок 19), так как один пользователь может иметь только одну роль.

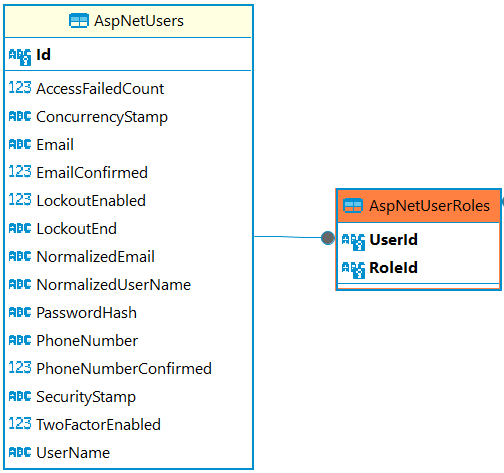


Рисунок 19 – Связь «AspNetUsers (Пользователи)» с «AspNetUserRoles (Роли пользователей)»

Таблицы «Пользователи» и «Утверждения к пользователю» находятся в отношении «один-ко-многим» (рисунок 20), так как один пользователь может иметь несколько утверждений.

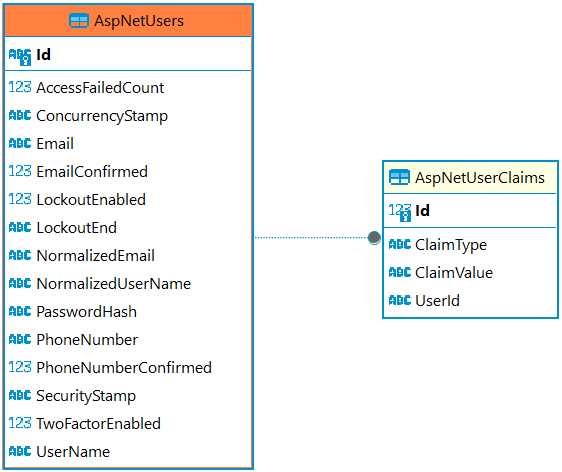


Рисунок 20 – Связь «AspNetUsers (Пользователи)» с «AspNetUserClaims (Утверждения к пользователю)»

Таблицы «Пользователи» и «Токены пользователей» находятся в отношении «один-к-одному» (рисунок 21), так как один пользователь может иметь только один токен для входа.

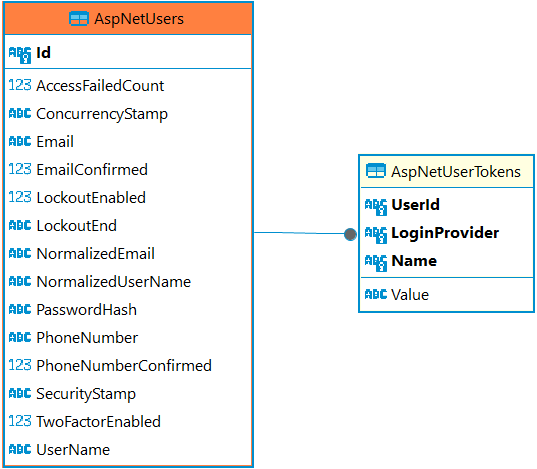


Рисунок 21 – Связь «AspNetUsers (Пользователи)» с «AspNetUserTokens (Токены пользователей)»

Таблицы «Пользователи» и «Логины пользователей» находятся в отношении «один-ко-многим» (рисунок 22), так как один пользователь может иметь несколько внешних провайдеров для входа.

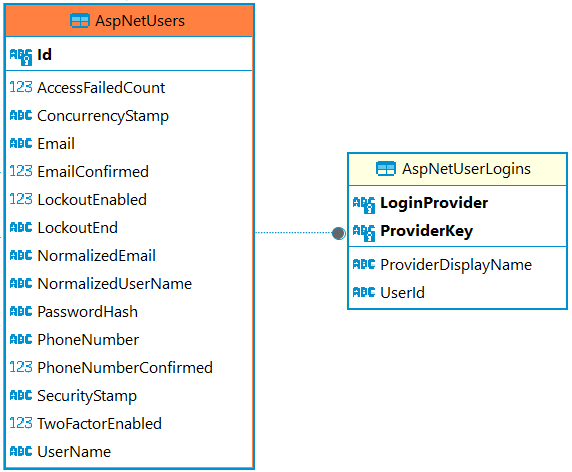


Рисунок 22 – Связь «AspNetUsers (Пользователи)» с «AspNetUserLogins (Логины пользователей)»

При проектировании базы данных необходимо провести процесс нормализации для обеспечения корректной работы всех данных в базе и для создания логической структуры. Нормализация выполняется следующим образом:

1. Идентификация всех сущностей предметной области, определение их атрибутов, ограничений и так далее.
2. Создание таблиц для каждой сущности, которая содержит столбцы для атрибутов.
3. Определение первичных ключей для каждой таблицы с целью ее идентификации.
4. Создание внешних ключей для связи таблиц. Они позволяют устанавливать отношения между таблицами. Каждый внешний ключ ссылается на первичный ключ в другой таблице.
5. Установка ограничений целостности данных. Использование ограничений целостности данных, такие как ограничения: «NOT NULL», «UNIQUE», «FOREIGN KEY» и «PRIMARY KEY», чтобы гарантировать целостность и согласованность данных.

Таким образом, нормализация базы данных стала более гибкой, поддерживаемой и непротиворечивой, что позволило уменьшить избыточность данных, улучшить их структурирование и обеспечить целостность информации.

# **2.3 Проектирование программных модулей решения**

Алгоритм — это логические инструкции или шаги, которые определяют порядок выполнения определенной задачи или решения определенной проблемы в программном обеспечении. Он описывает, каким образом должны быть выполнены определенные действия или операции, чтобы достичь желаемого результата.

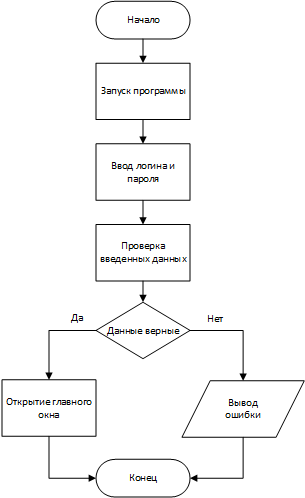
Алгоритмы ПО являются основой для разработки программ и являются важным компонентом в различных областях, таких как компьютерная наука, информационные технологии, искусственный интеллект, анализ данных и других.

Основные функции программы:

* авторизация в систему (блок-схема 1);
* осуществление операций по добавлению, изменению, удалению данных пациента (блок-схема 2).

Блок-схема 1 включает в себя логику и алгоритмы, отвечающие за авторизацию пользователя в систему. После того как был открыт сайт в браузере, пользователю необходимо ввести свой логин и пароль, как только пользователь ввел данные, отправляется запрос в базу данных с целью узнать наличие данной учетной записи в системе, если учетной записи в системе нету или пользователь не заполнил поля, то отобразится соответствующее сообщение об ошибке, иначе откроется главное окно системы.

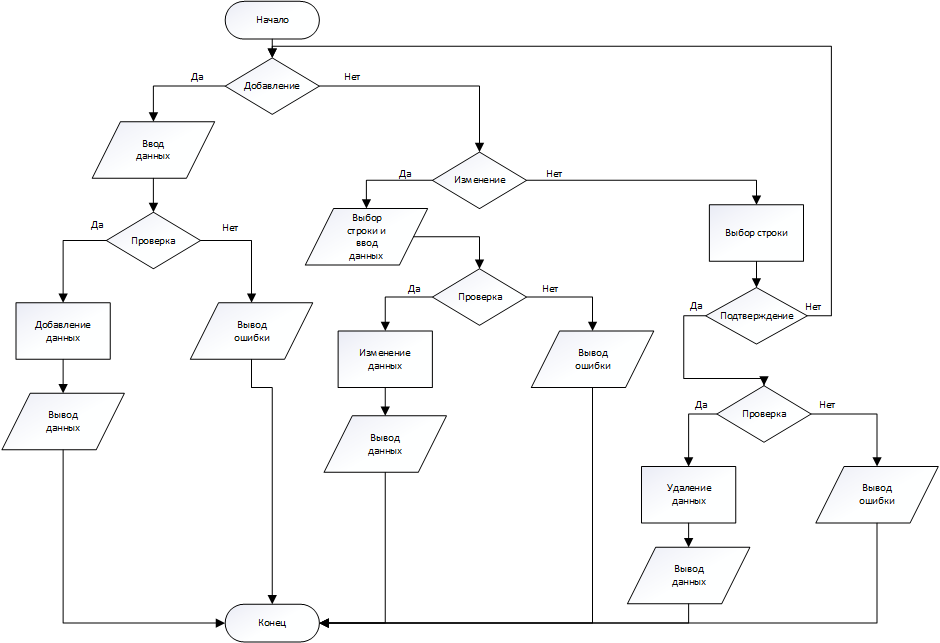
Блок-схема 1 – Авторизация в систему



Блок-схема 2 включает в себя логику и алгоритмы, отвечающие за операции для работы с данными пациента. На данной блок-схеме представлено управление операциями:

* добавление осуществляется путем ввода соответствующих данных в необходимые поля, как только данные введены происходит проверка их корректности и в зависимости от проверки, данные либо сохранятся, либо появиться сообщение об ошибке;
* редактирование происходит путем двойного нажатия по выбранной строке, а затем замена введенных ранее данных на новые, после происходит проверка на корректность вводимых изменений;
* удаление происходит путем нажатия кнопки, которая имеется у каждой строки, после чего отобразиться окно для подтверждения.

Блок-схема 2 – Операции для работы с данными пациента



# **2.4 Реализация технического решения**

# **2.4.1 Разработка пользовательского интерфейса решения**

Работа с сайтом начинается с запуска окна для авторизации (рисунок 23). Данная форма содержит поля для ввода логина (имя пользователя) и пароля, а также кнопку для авторизации.

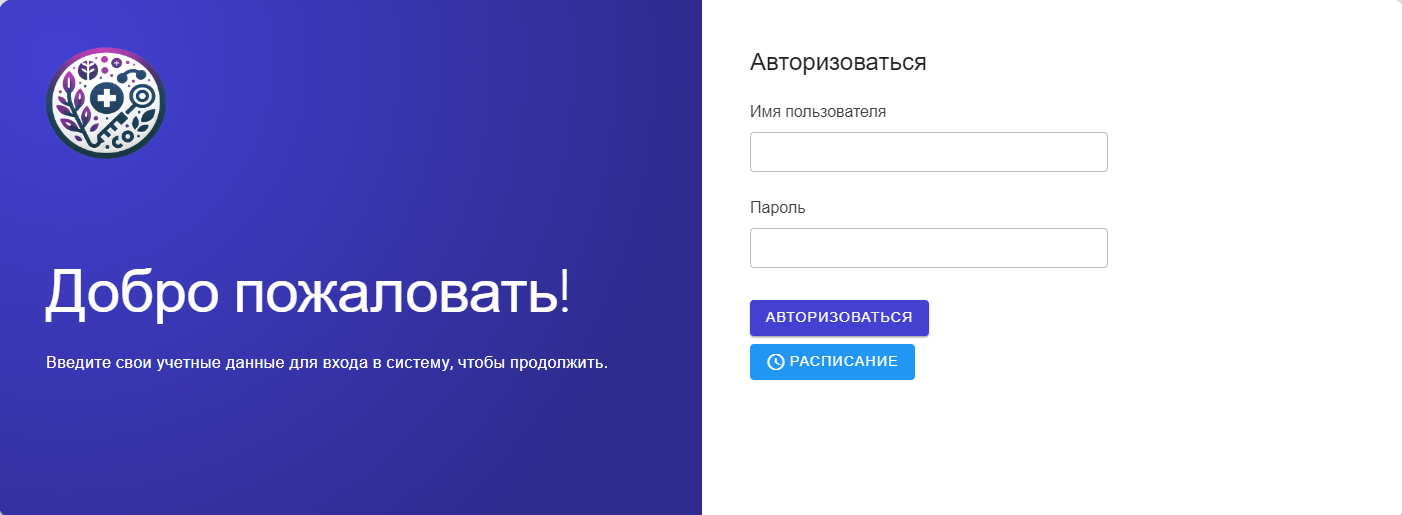


Рисунок 23 – Форма авторизации

Если пользователь не имеет учетных данных для авторизации, то он может просмотреть расписание работы докторов. Для этого необходимо нажать на кнопку «Расписание», которое отобразит модальное окно (рисунок 24). На данном окне есть возможность поиска, сортировки и фильтрации данных.

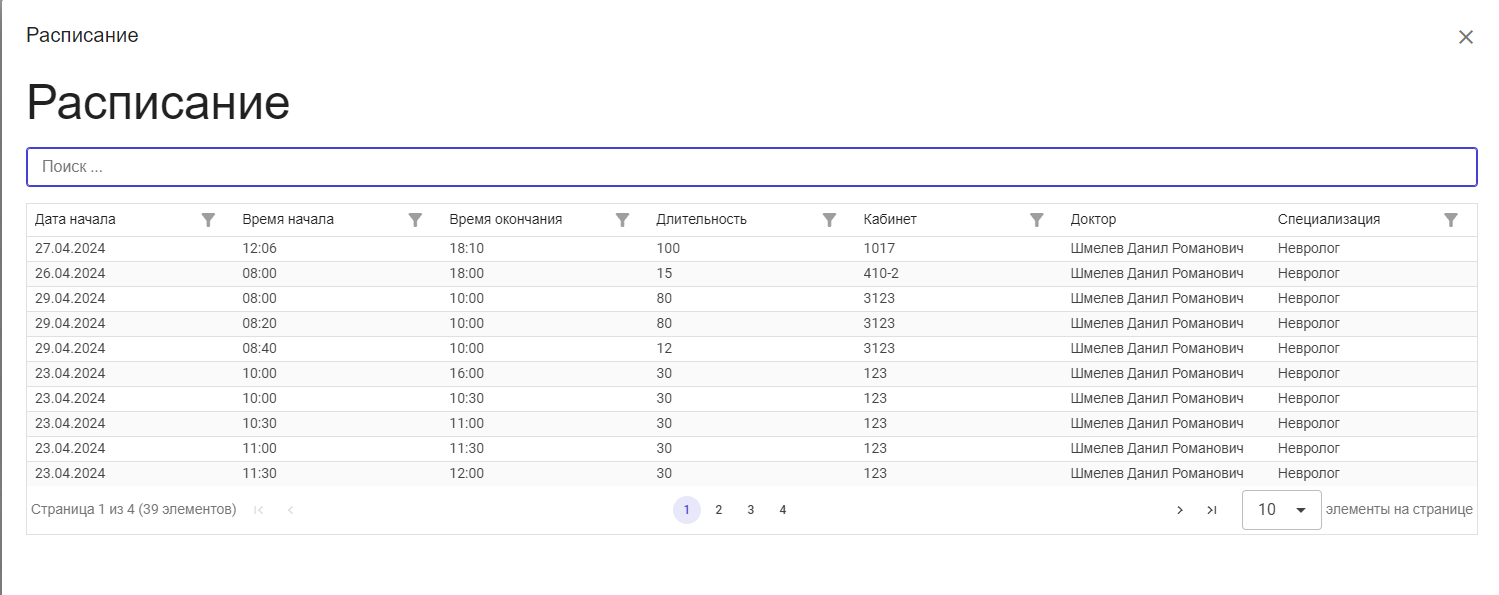


Рисунок 24 – Расписание работы докторов

Перед тем как осуществить вход необходимо правильно заполнить поля. При неверном вводе отобразиться текст (рисунок 25), указывающий на причину, по которой авторизация не может быть выполнена.

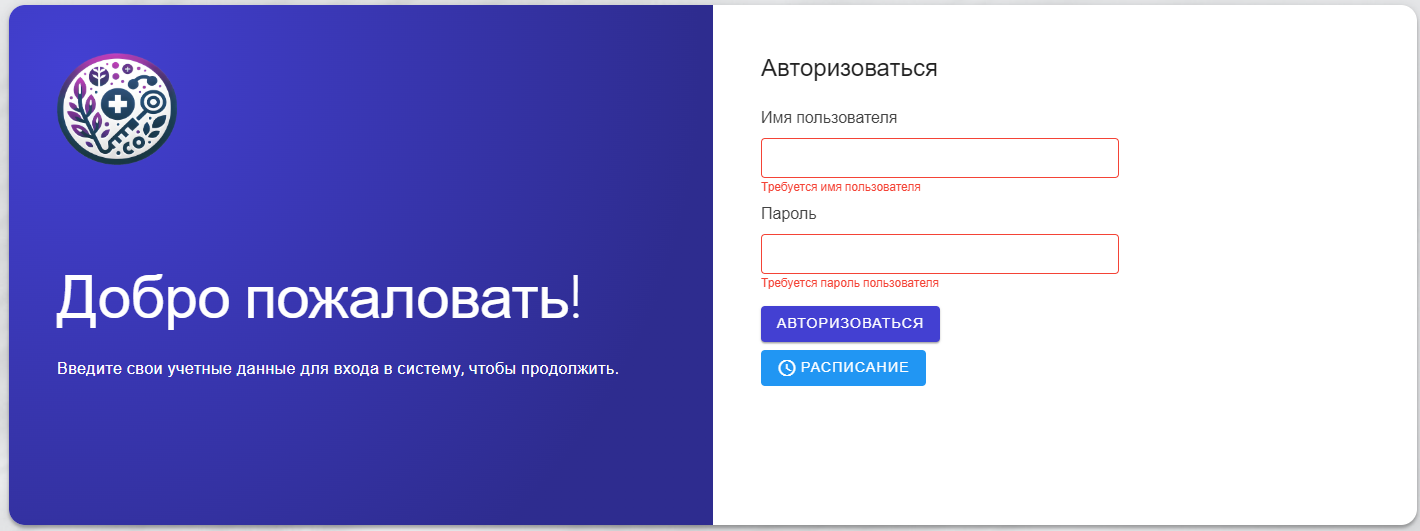


Рисунок 25 – Поля для ввода данных являются пустыми

После успешного ввода логина и пароля в зависимости от роли пользователя отобразиться страница (рисунок 26), которая содержит боковую панель навигации (рисунок 27), адрес расположения поликлиники, а также в заголовке имеется меню для управления аккаунтом (рисунок 28). И что их них что?

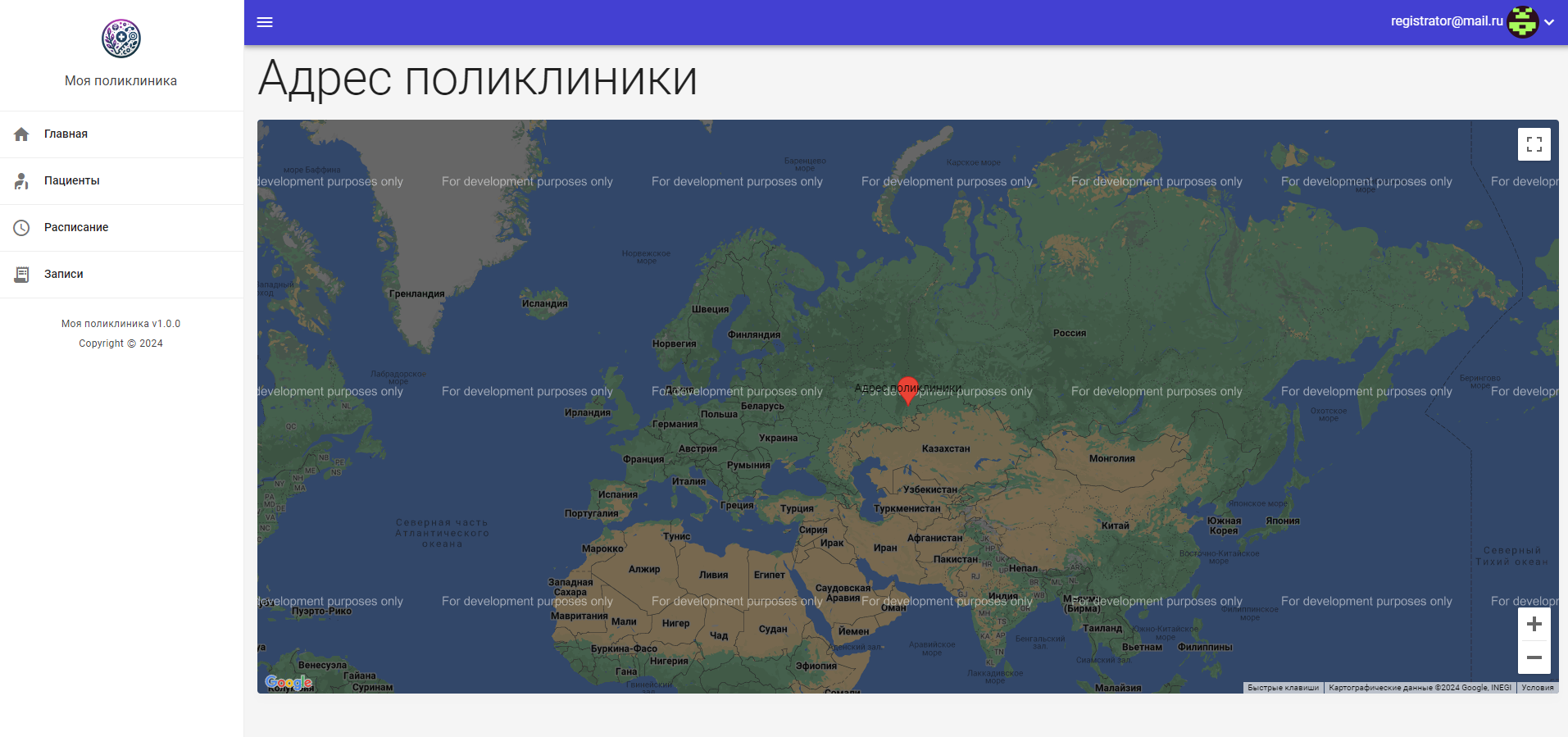


Рисунок 26 – Начальная страница после входа

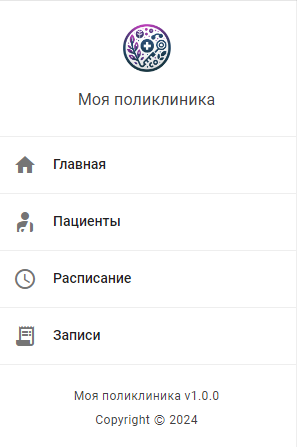


Рисунок 27 – Боковая панель навигации

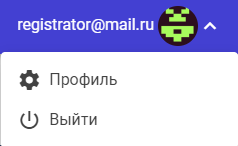


Рисунок 28 – Функции меню в заголовке страницы

При нажатии на кнопку «Профиль» отобразиться страница, позволяющая изменять свой пароль для входа (рисунок 29).



Рисунок 29 – Смена пароля пользователя

Если пользователь оставил поля незаполненными или указал неверные данные в полях, то появиться текст об ошибке.

При нажатии на кнопку «Выход» (рисунок 28) осуществится переход на страницу для авторизации (рисунок 23).

Если авторизоваться под пользователем с ролью «регистратор» можно увидеть доступные ему функции, которые расположены в боковой панели навигации, а также в меню, расположенном в заголовке.

Почти каждая страница будет содержать следующие функции:

* кнопку «Экспорт», которая позволяет выгружать данные в файлы Excel или CSV (рисунок 30);
* кнопку «Добавить», которая вызывает модальное окно для добавление нового пациента;
* текстовое поле для поиска;
* таблицу, которая отображает данные;
* кнопку удаления у каждой записи;
* ячейку для вывода отображаемой таблицы, сортировку и фильтрацию.

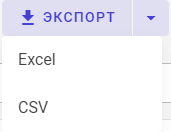


Рисунок 30 – кнопка «Экспорт»

Работа с данными пациента осуществляется на странице «Пациент» (рисунок 31). Переход на данную страницу происходит путем нажатия по элементу меню «Пациенты», расположенном в боковой панели навигации.

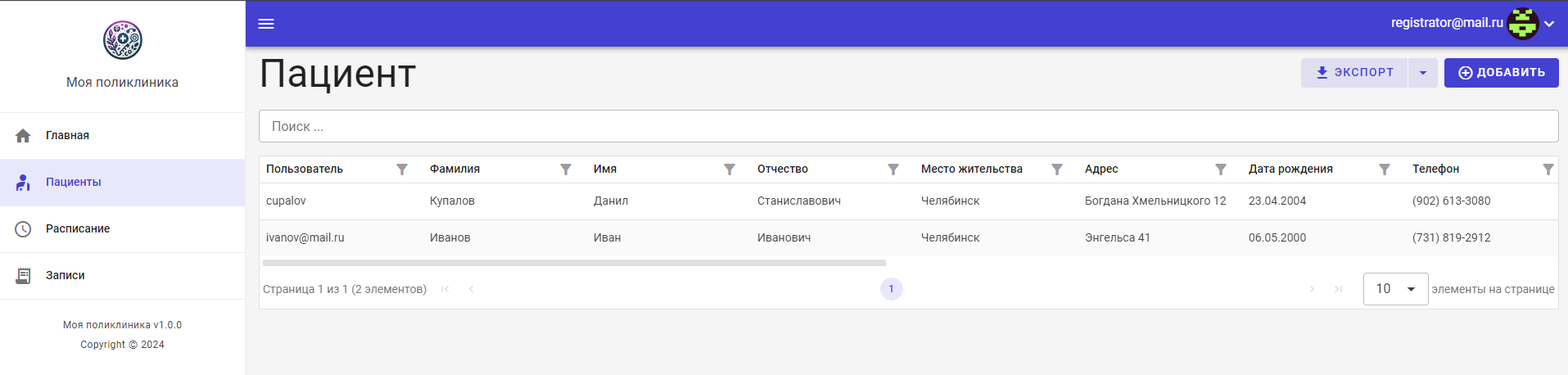


Рисунок 31 – Страница «Пациент»

Добавление нового пациента осуществляет при помощи нажатия кнопки «Добавить», после чего откроется модальное окно с вводом данных (рисунок 32).



Рисунок 32 – Окно добавления пациента

Поля, которые содержат символ «\*» являются обязательными к заполнению. Ввод данных по типу: номер телефона, паспорт, серия и так далее осуществляется с использованием маски. Каждое поле имеет пример того, что требуется ввести пользователю и что от него требуется.

После заполнения полей, необходимо осуществить нажатие на кнопку «Сохранить» в результате чего произойдет проверка введенных данных на корректность и если поля заполнены верно, то пользователя перенесет на страницу, с которой осуществлялось открытие окна добавления.

При неверном вводе данных при сохранение появиться текст о том, что данные указаны неверно (рисунок 33).

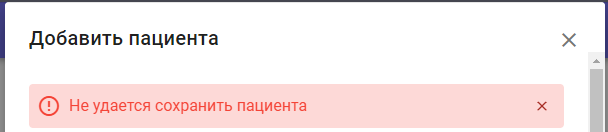


Рисунок 33 – Некорректный ввод данных при добавлении пациента

Редактирование происходит путем двойного нажатия по строке, а после чего открывается модальное окно, в котором изменяются ранее введенные данные об пациенте (рисунок 34). Для сохранения данных необходимо осуществить нажатия на кнопку «Сохранить», после которой произойдет проверка полей на корректность и при успешной проверке страница для редактирования закроется.

Обработка ошибок и ограничения полей точно такие-же как и при добавлении.

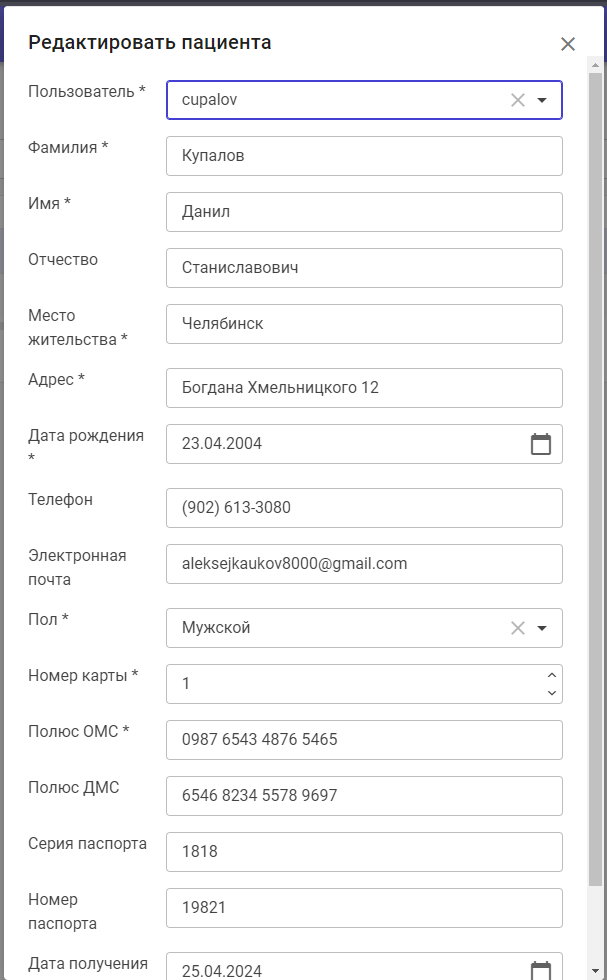


Рисунок 34 – Окно редактирования пациента

Удаление данные об пациенте осуществляется путем нажатия кнопки с иконкой урны у соответствующей строки (рисунок 35).

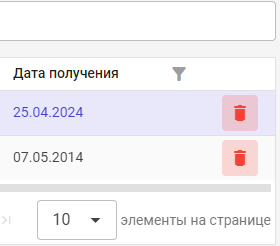


Рисунок 35 – Удаление данных пациента

После нажатия кнопки удаления появиться окно подтверждения действий (рисунок 36).

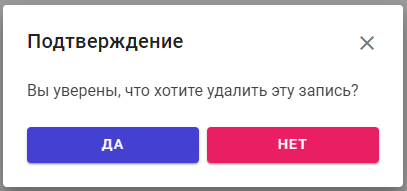


Рисунок 36 – Подтверждение удаления

Если пациент назначен на посещение к врачу, то при удалении его записи появиться сообщение.

Просмотр расписания приема докторов осуществляется на странице «Расписание» (рисунок 37). Переход на данную страницу осуществляется путем нажатия по элементу меню «Расписание», расположенном в боковой панели навигации.

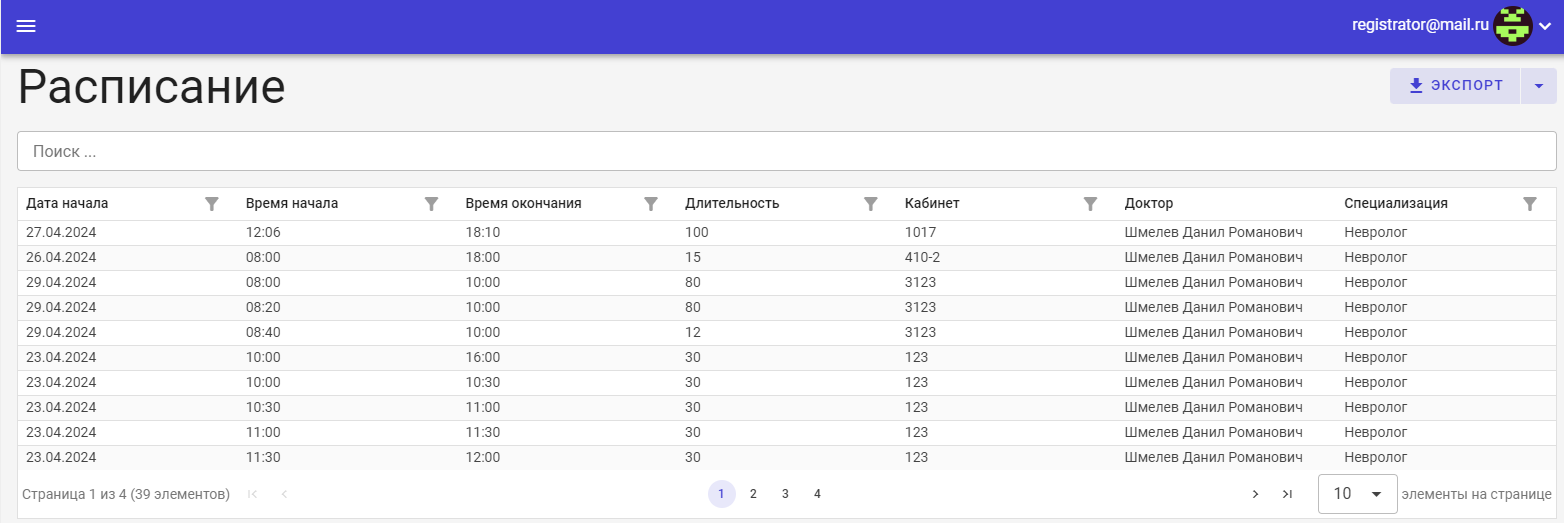


Рисунок 37 - Страница «Расписание»

Данная страница имеет возможность экспорта данных, поиска, сортировки, фильтрации и пагинации.

Для просмотра записей пациента необходимо перейти на страницу «Запись» (рисунок 38). Переход на данную страницу осуществляется путем нажатия по элементу меню «Записи», расположенном в боковой панели навигации.

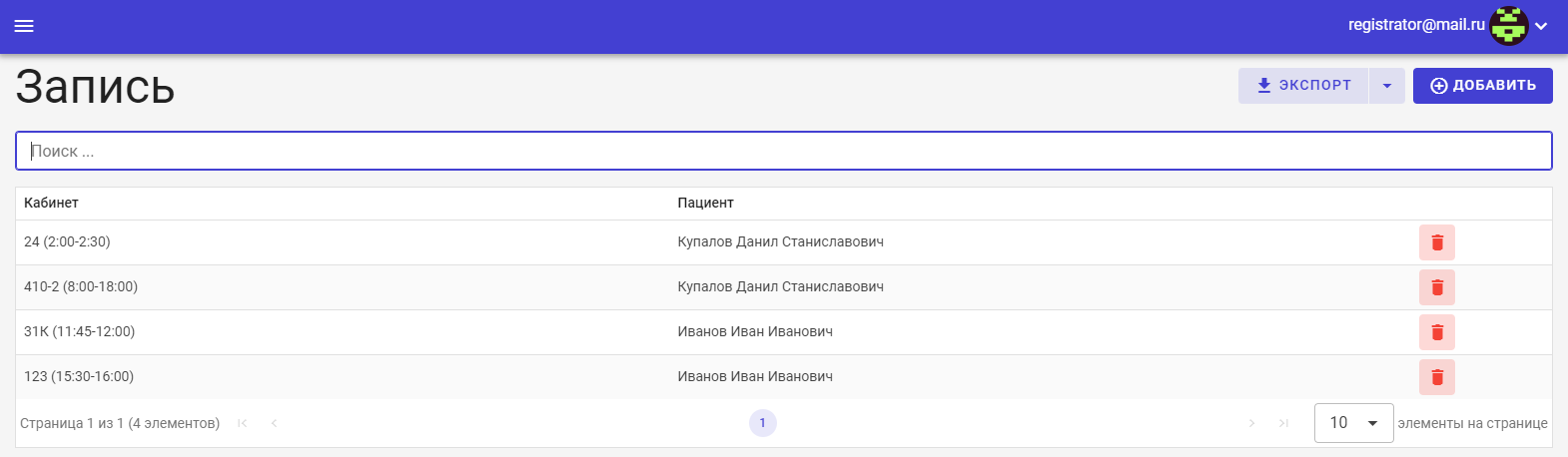


Рисунок 38 – Страница «Запись»

Добавление новой записи осуществляет при помощи нажатия кнопки «Добавить», после чего откроется модальное окно с вводом данных (рисунок 39).

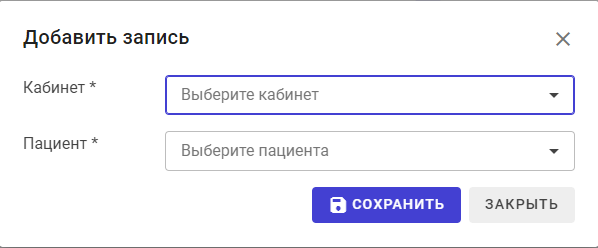


Рисунок 39 – Добавление записи

Оба поля являются обязательными к заполнению, если одно из полей не было заполнено, то отобразиться текст с ошибкой.

Редактирование записи происходит путем двойного нажатия по строке, после чего открывается модальное окно (рисунок 40).

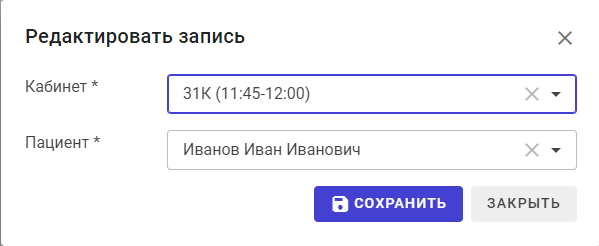


Рисунок 40 – Редактирование записи

Оба поля являются обязательными к заполнению и если одно из полей было не заполнено, то появиться текст с ошибкой.

При добавлении и редактирование, если у пациента была указана почта, то ему придет сообщение, которое будет содержать данные о записи.

Удаление записи происходит путем нажатия кнопки удалить у соответствующей строки с данными, которая вызовет окно подтверждения (рисунок 36), иначе отобразиться ошибка.

Пользователь вошедший в систему под ролью «доктор» будет иметь просмотра расположения поликлиники, просмотр расписания работы докторов, осуществлять операции по добавлению, редактированию и удалению посещений, а также изменять свой пароль.

Если перейти на страницу с расписанием, то функционал останется такой-же как и пользователя с ролью «регистратор».

Проведение осмотра происходит на странице «Посещения» (рисунок 41), переход на данную страницу осуществляется путем нажатия на элемент меню «Посещения», расположенного в боковой панели навигации.

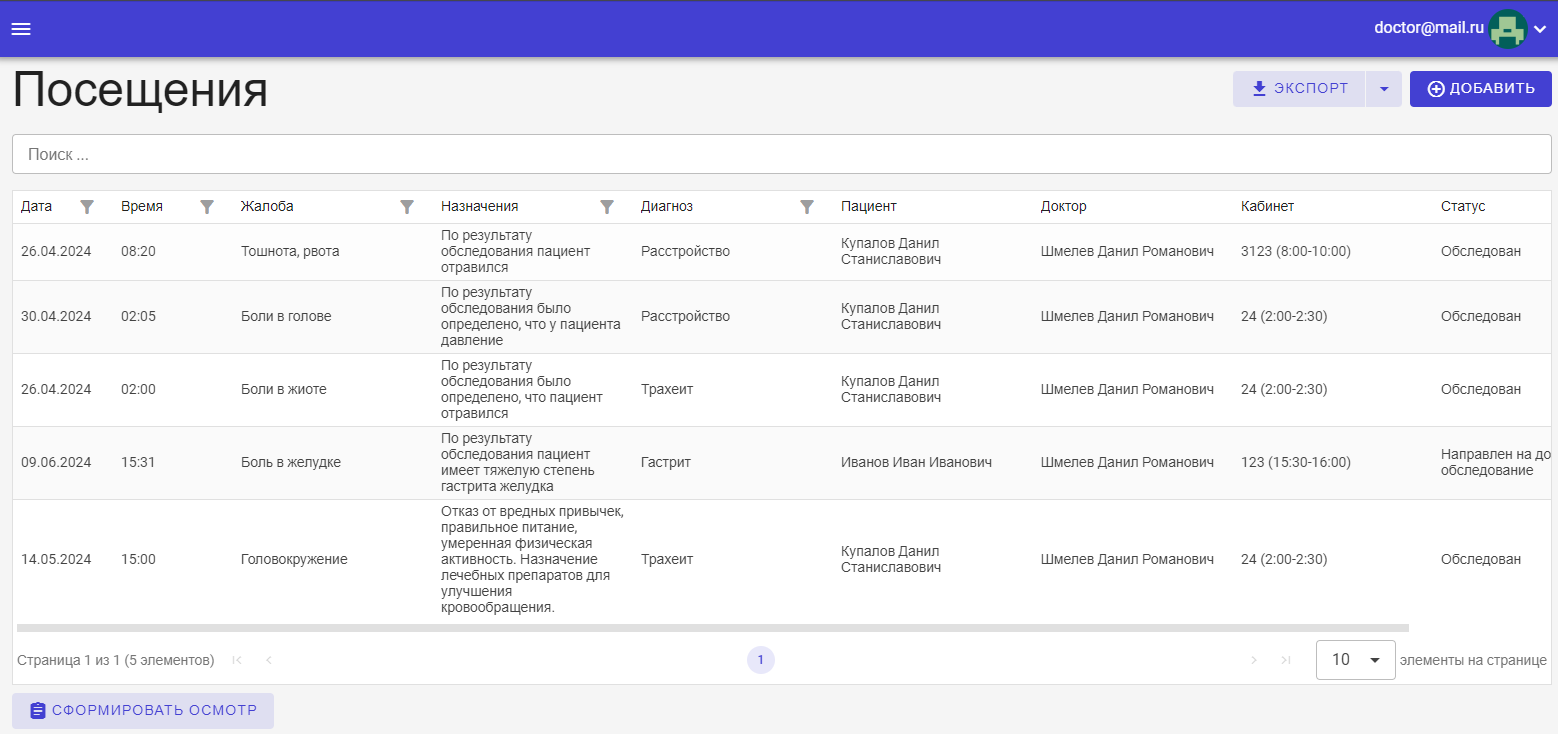


Рисунок 41 – Страница «Посещения»

Данное окно содержит следующий функционал: операции по управлению данными (добавление, удаление и редактирование), поиск, фильтрацию, сортировку, формирования документа об осмотре и экспорт.

Добавление осуществляется путем нажатия на кнопку «Добавить», после чего открывается модальное окно для ввода данных (рисунок 42). Поля, содержащие символ «\*» являются обязательными к заполнению, при неверном заполнение полей – появиться текст с ошибкой.

После заполнения полей необходимо осуществить нажатие на кнопку «Сохранить», в результате чего произойдет проверка полей на корректность и при верном заполнение окно закроется и отобразится страница «Посещения».

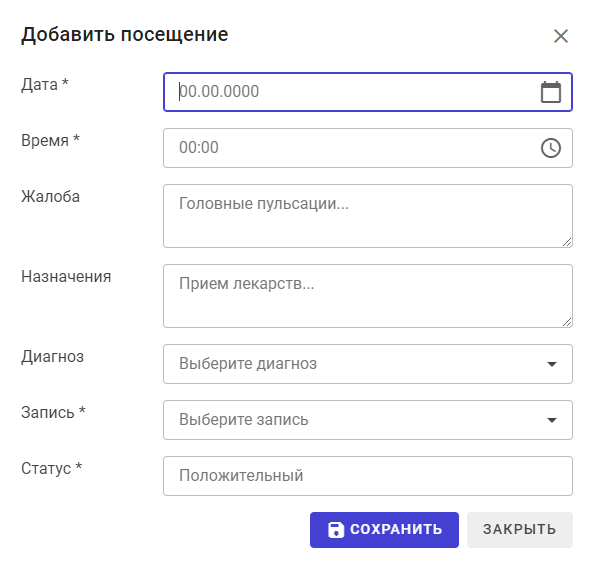


Рисунок 42 – Добавление посещения

Редактирование происходит путем двойного нажатия по записи, после которого откроется модальное окно с возможностью изменять ранее введенные данные (рисунок 43), при неверном вводе данных появиться текст с ошибкой.

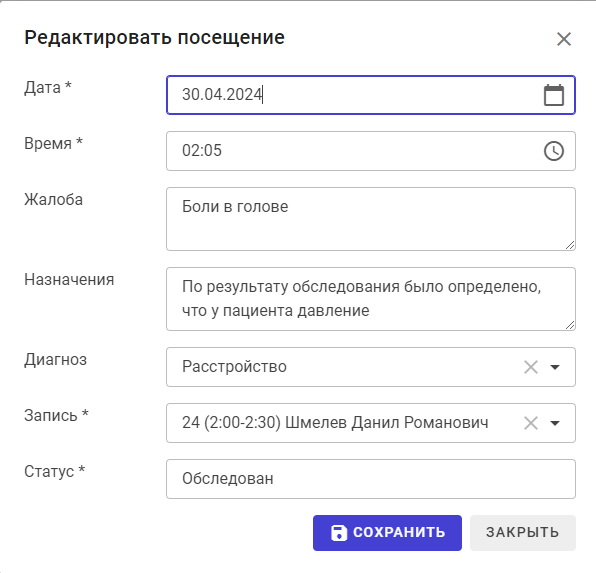


Рисунок 43 – Редактирование посещения

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение (рисунок 36).

Пользователь вошедший под ролью «администратор» будет иметь полный функционал, операции по работе с данными (добавление, редактирование, удаление), экспорт, поиск, сортировку, фильтрацию, изменять пароль себе и другим пользователям.

Переход на страницу с диагнозами происходит путем нажатия на элемент «Диагнозы», который расположен в боковой панели навигации (рисунок 44).



Рисунок 44 – Страница «Диагноз»

Добавление нового диагноза происходит путем нажатия кнопки «Добавить», после чего открывает модальное окно для добавления (рисунок 45).

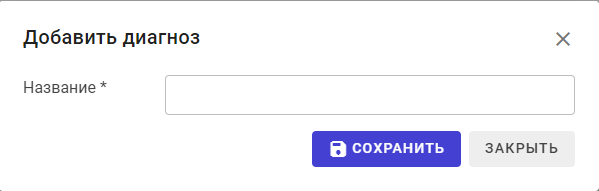


Рисунок 45 – Добавление диагноза

Если в поле для ввода не были введены данные, а нажатие на кнопку «Сохранить» осуществилось, то отобразиться текст с ошибкой.

Редактирование происходит путем двойного нажатия на строку, после которого откроется модальное окно для внесения изменений.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение (рисунок 35).

Для просмотра имеющихся докторов в системе, а также для их добавления и редактирования имеется отдельная страница «Доктор», переход на нее осуществляется путем нажатия элемента меню «Доктора» в боковой панели навигации (рисунок 46).

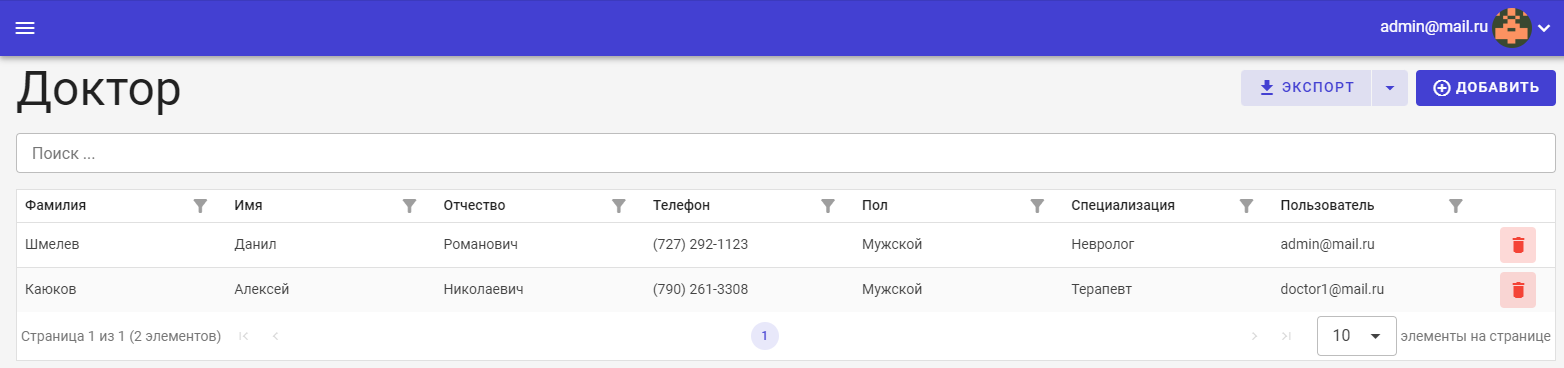


Рисунок 46– Страница «Доктор»

Создание нового доктора происходит путем нажатия кнопки «Добавить», после которого откроется модальное окно для ввода данных (рисунок 47).

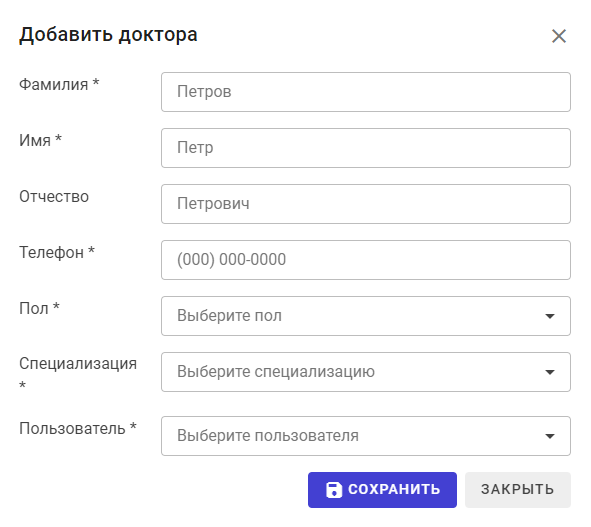


Рисунок 47 – Добавление доктора

Поля, имеющие символ «\*» являются обязательными к заполнению. При неверном вводе данных в поля и сохранение отобразиться текст с ошибкой.

Редактирование данных доктора происходит путем двойного нажатия по нужной строке, после чего откроется модальное окно для редактирования.

Удаление происходит путем нажатия по кнопке «Удалить» у соответствующей строки и осуществить подтверждение (рисунок 36).

Формирования времени приема докторов происходит на странице «Расписание», переход на которое осуществляется нажатием на соответствующий элемент в боковой панели навигации (рисунок 48).

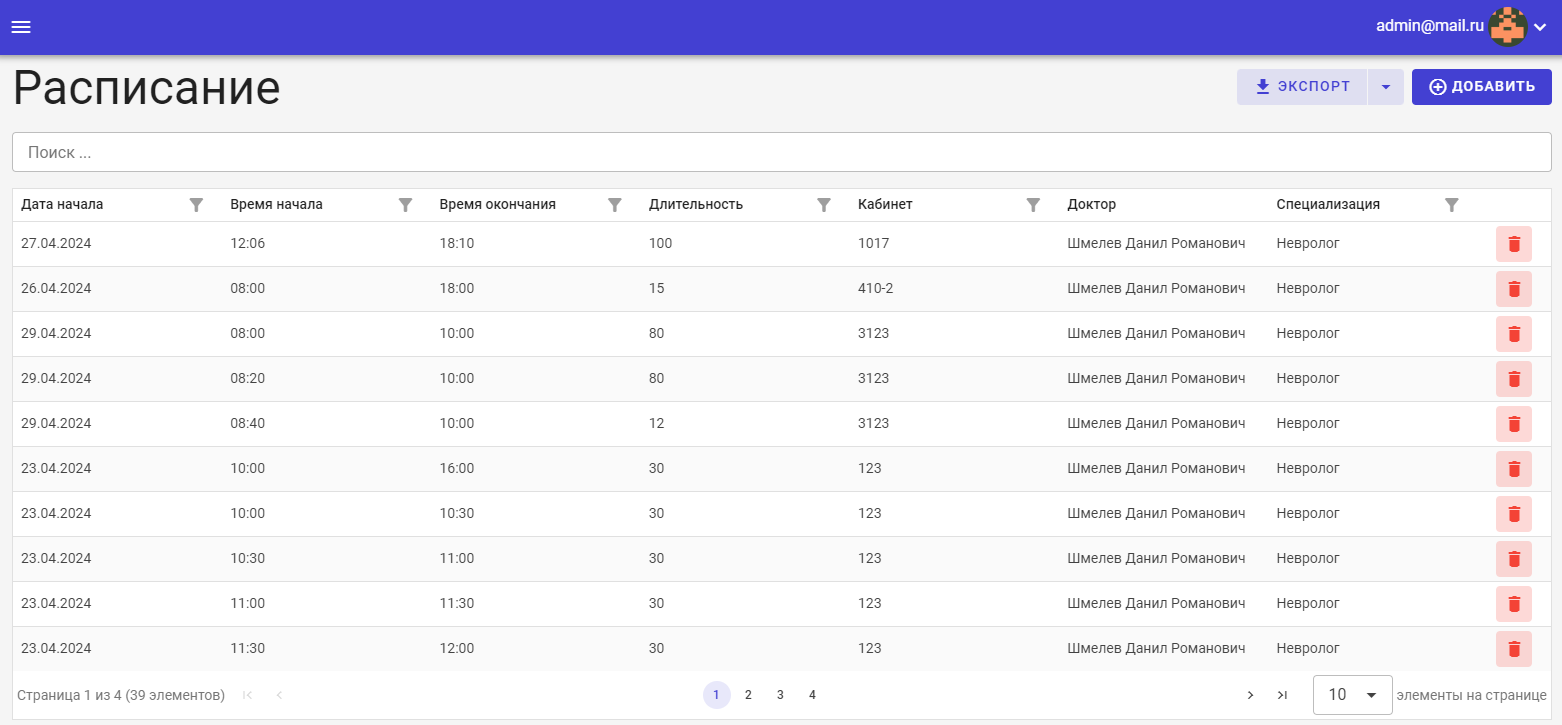


Рисунок 48 – Страница «Расписание»

Добавление нового расписания происходит путем нажатия кнопки «Добавить», после чего открывается модальное окно для ввода данных (рисунок 48).

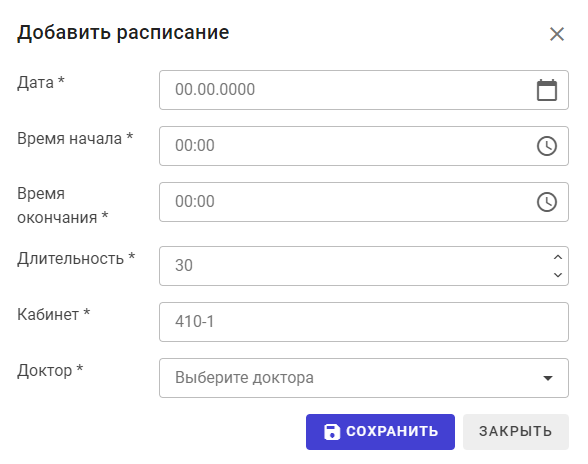


Рисунок 49– Добавление расписания

Поля, содержащие символ «\*» являются обязательными к заполнению, при некорректном вводе данных появиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной строке, после чего открывается модальное окно, в котором можно изменять ранее введенные данные.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение (рисунок 36).

Просмотр и осуществление операций по добавлению, удалению, редактированию данных о специализациях происходит на странице «Специализация» (рисунок 50), переход на которое осуществляется из боковой панели навигации.

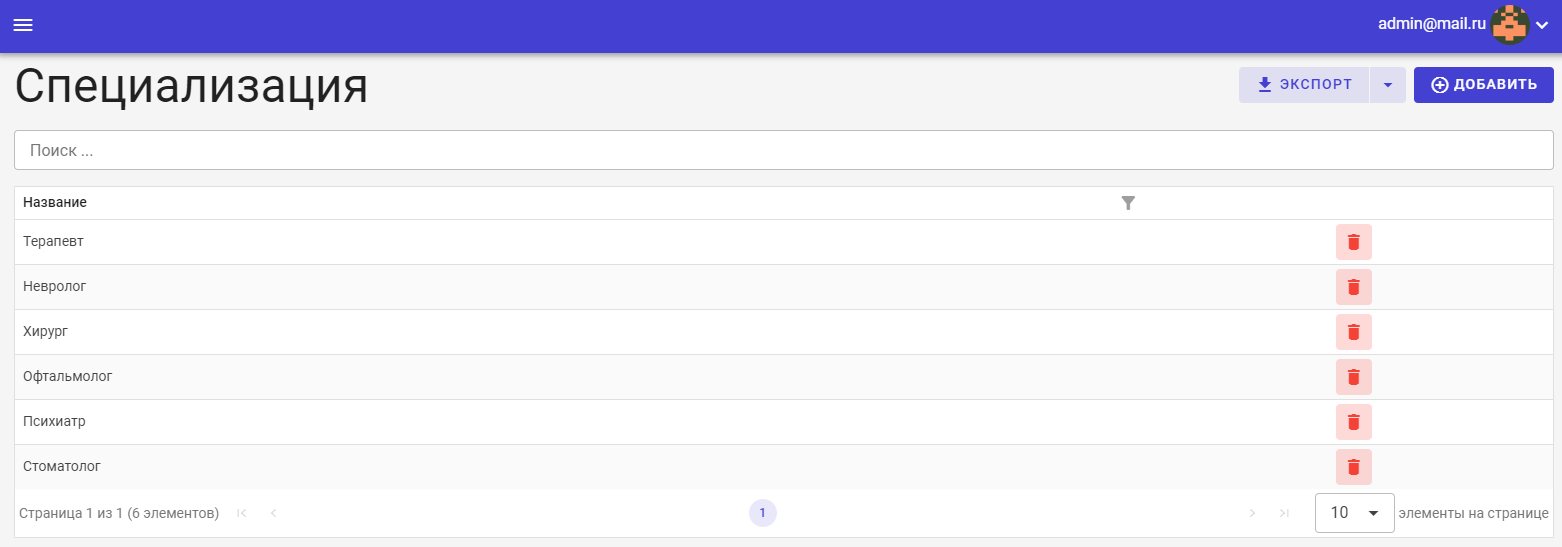


Рисунок 50 – Страница «Специализация»

Создание новой специализации происходит путем нажатия на кнопку «Добавить», после которого отрывается модальное окно для ввода названия.

Если поле для ввода названия специализации осталось пустое или такая специализация уже имеется, то появиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения названия специализации.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение (рисунок 36).

Просмотр и осуществление различных операций с утверждений к роли осуществляется путем открытия страницы «Утверждения к роли», переход на которое осуществляется при помощи боковой панели навигации (рисунок 51).



Рисунок 51 – Страница «Утверждения к роли»

Добавление нового утверждения к роли осуществляется путем нажатия на кнопку «Добавить», которая отобразит модальное окно для ввода данных.

Поле «Роль» в данном случае является обязательным. При некорректном вводе данных в поля отобразиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр ролей и осуществление операций с ними происходит на странице «Роли» (рисунок 52), переход на которое осуществляется из боковой панели навигации.

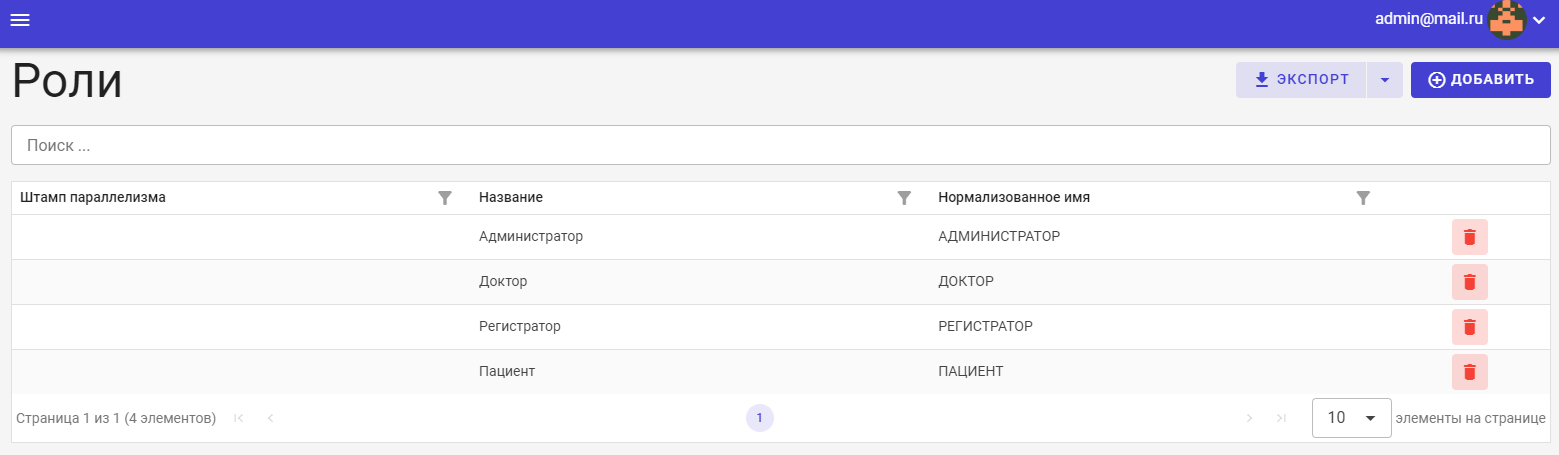


Рисунок 52 – Страница «Роли»

Добавление новой роли осуществляется путем нажатия на кнопку «Добавить», после которого откроется модально окно для ввода данных (рисунок 53).

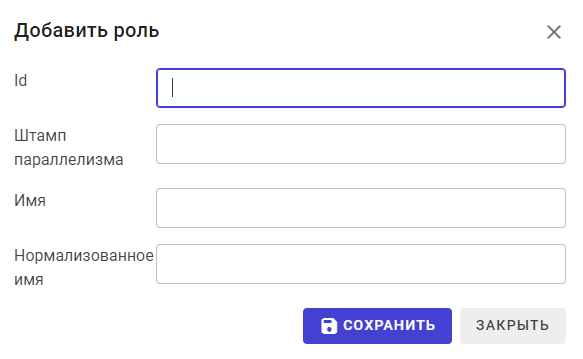


Рисунок 53 – Добавление роли

Редактирование роли происходит путем двойного нажатия по нужной строке, после которого откроется модальное окно для изменения ранее введенных данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр и осуществление различных операций с утверждений к роли осуществляется путем открытия страницы «Утверждения к пользователю» (рисунок 54), переход на которое осуществляется при помощи боковой панели навигации.

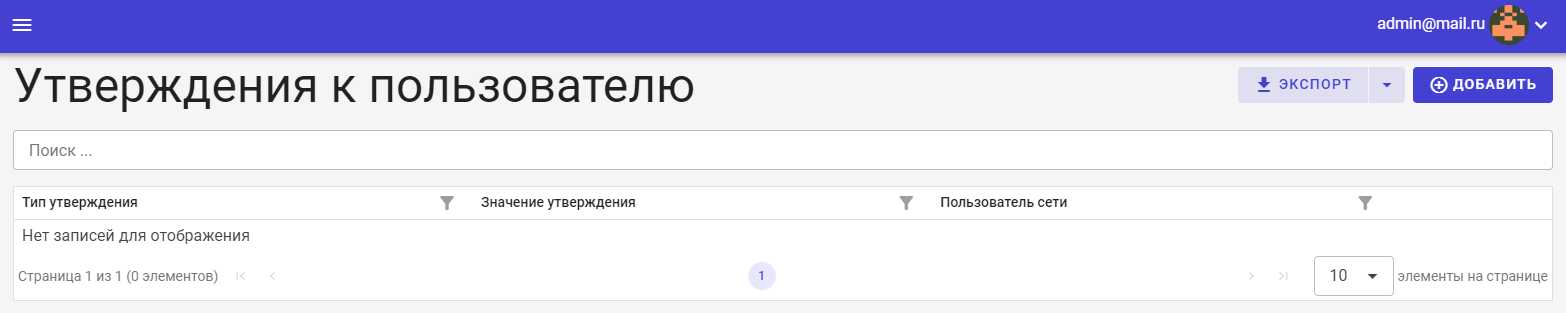


Рисунок 54 – Страница «Утверждения к роли»

Добавление нового утверждения к пользователю осуществляется путем нажатия на кнопку «Добавить», которая откроет модальное окно (рисунок 55).

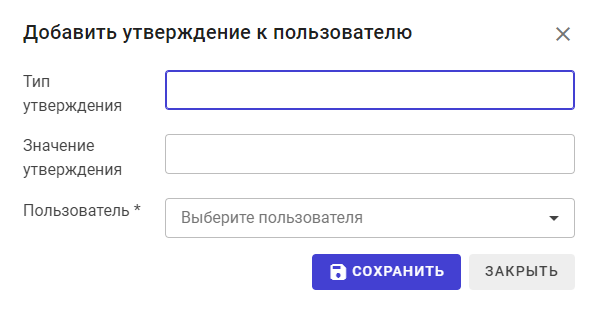


Рисунок 55 – Добавление утверждения к пользователю

Поле «Пользователь» в данном случае является обязательным. При некорректном вводе данных в поля отобразиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр и осуществление различных операций с логинами пользователей осуществляется путем открытия страницы «Логины пользователей» (рисунок 56), переход на которое осуществляется при помощи боковой панели навигации.

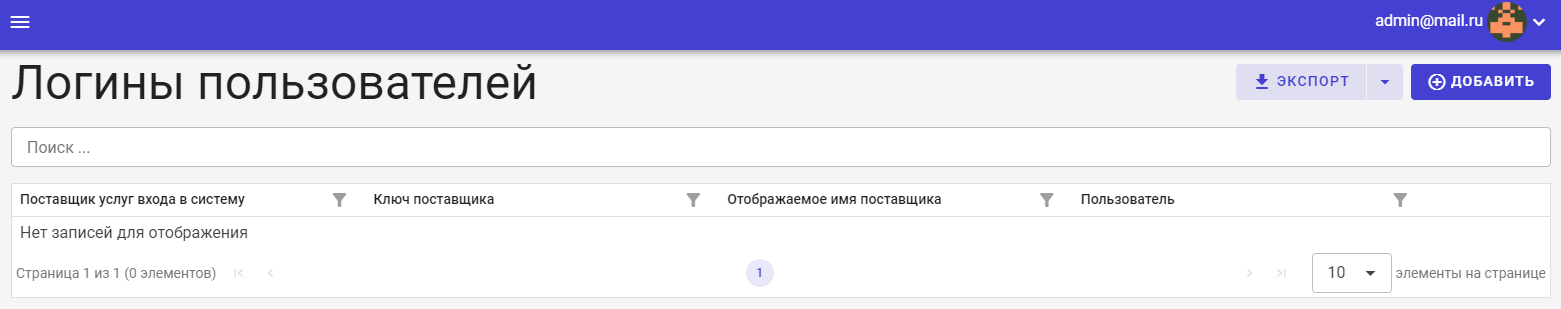


Рисунок 56 – Страница «Логины пользователей»

Добавление нового логина пользователя осуществляется путем нажатия на кнопку «Добавить», которая отобразит модальное окно (рисунок 57).

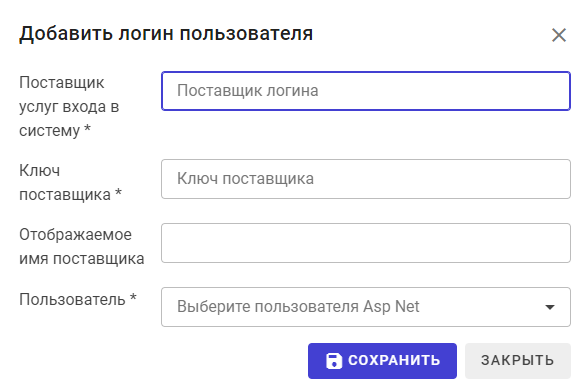


Рисунок 57 – Добавления логина пользователя

Поля, содержащие символ «\*» являются обязательными к заполнению. При некорректном вводе данных в поля отобразиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр ролей у пользователей и осуществление различных операций на странице «Роли пользователей» (рисунок 58), переход к которой осуществляется через боковую панель навигации.

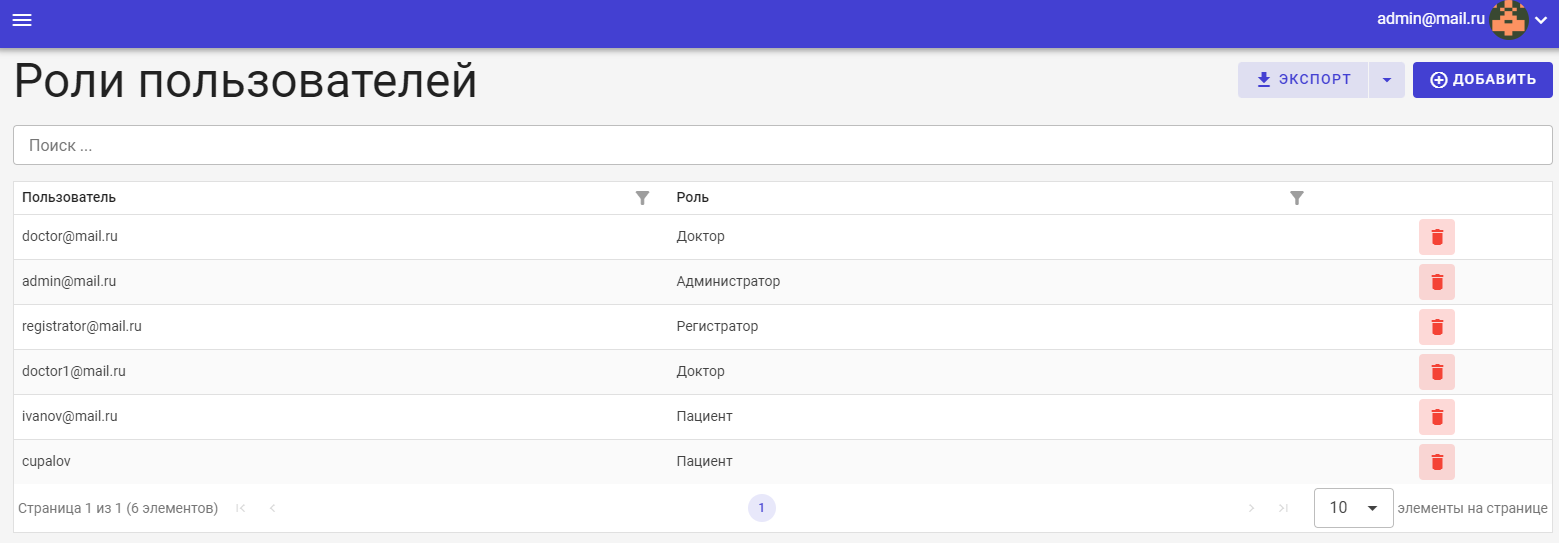


Рисунок 58 – Страница «Роли пользователей»

Добавление новой роли к пользователю происходит путем нажатия кнопки «Добавить» и заполнения обязательных полей (рисунок 59).

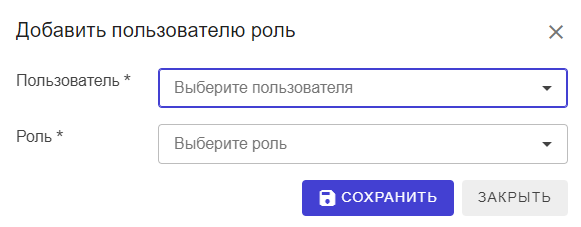


Рисунок 59 – Добавление роли к пользователю

При неверном вводе данных после нажатия кнопки для сохранения появиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр информации о пользователях, а также осуществление различных операций с ними осуществляется на странице «Пользователи» (рисунок 60), переход на нее происходит из боковой панели навигации.

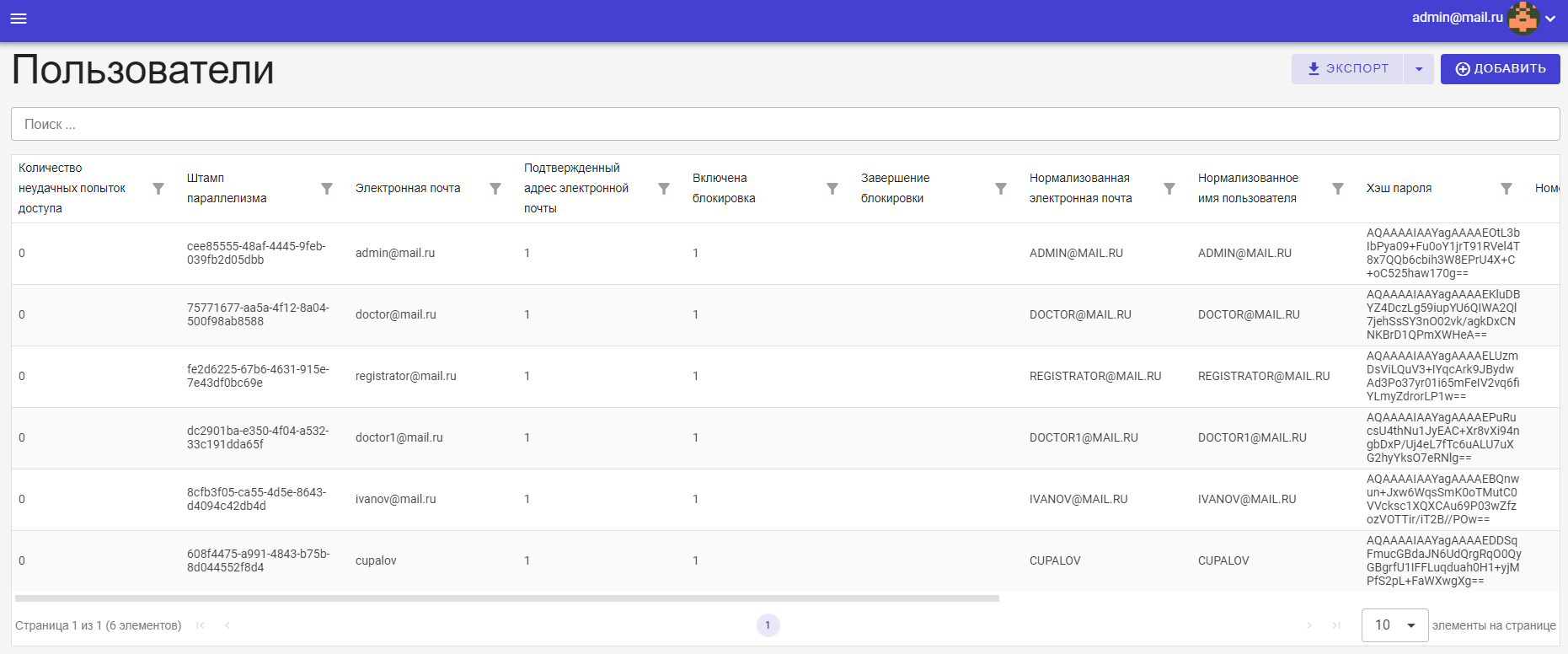


Рисунок 60 – Страница «Пользователи»

Добавление нового пользователя осуществляется путем нажатия кнопки «Добавить», которая открывает модальное окно для ввода данных нового пользователя (рисунок 61).

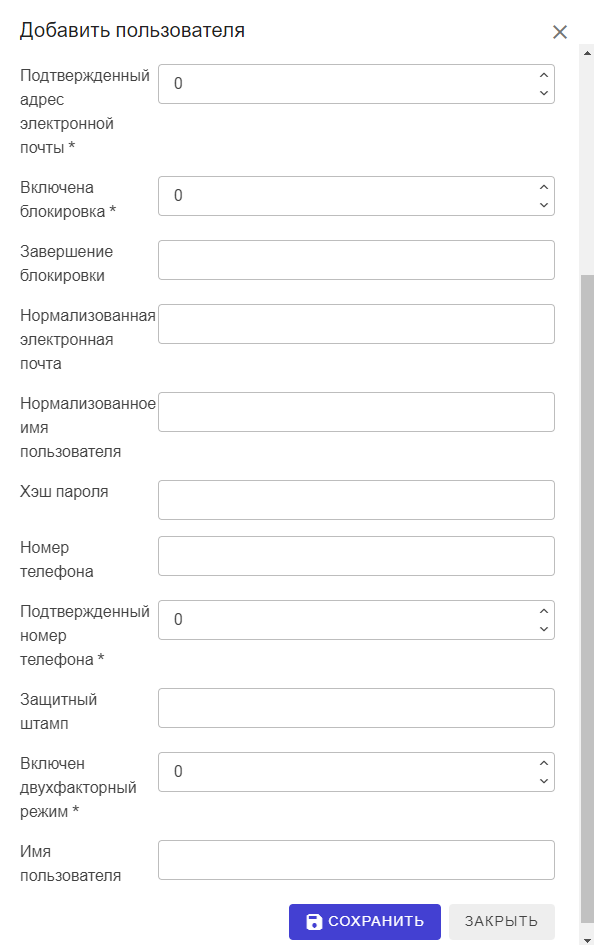


Рисунок 61 – Добавление пользователя

Редактирование данных пользователя осуществляется путем двойного нажатия по строке, после чего отобразиться модальное окно с ранее введенными данными.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр и осуществление различных операций с токенами пользователей осуществляется путем открытия страницы «Токены пользователей» (рисунок 62), переход на которое осуществляется при помощи боковой панели навигации.

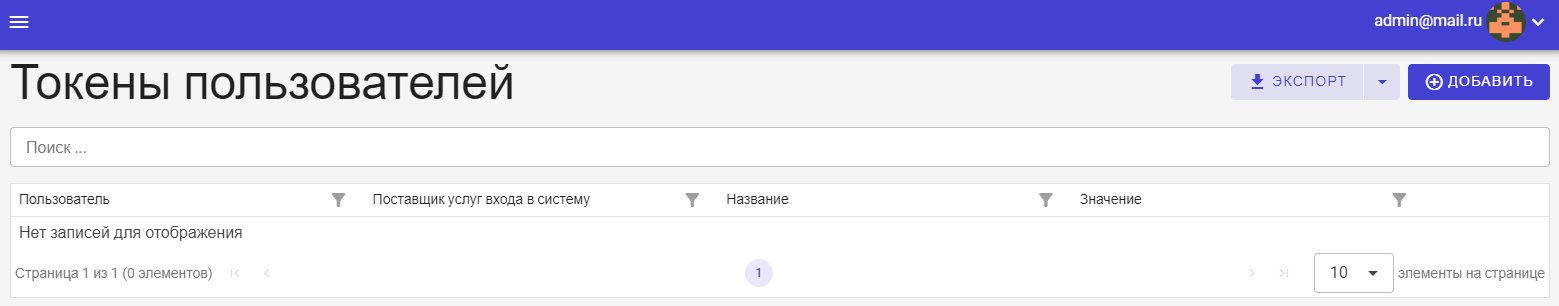


Рисунок 62 – Страница «Токены пользователей»

Добавление нового токена пользователя осуществляется путем нажатия на кнопку «Добавить», которая отобразит модальное окно для ввода данных.

Поля, содержащие символ «\*» являются обязательными к заполнению. При некорректном вводе данных в поля отобразиться текст с ошибкой.

Редактирование происходит путем двойного нажатия по нужной записи в таблице, после чего отобразиться модальное окно с возможностью изменения данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр всех пользователей осуществляется на странице «Пользователи приложения» (рисунок 63), переход на данную страницу может быть выполнен как через боковую панель навигации, так и через меню расположенном в заголовке.

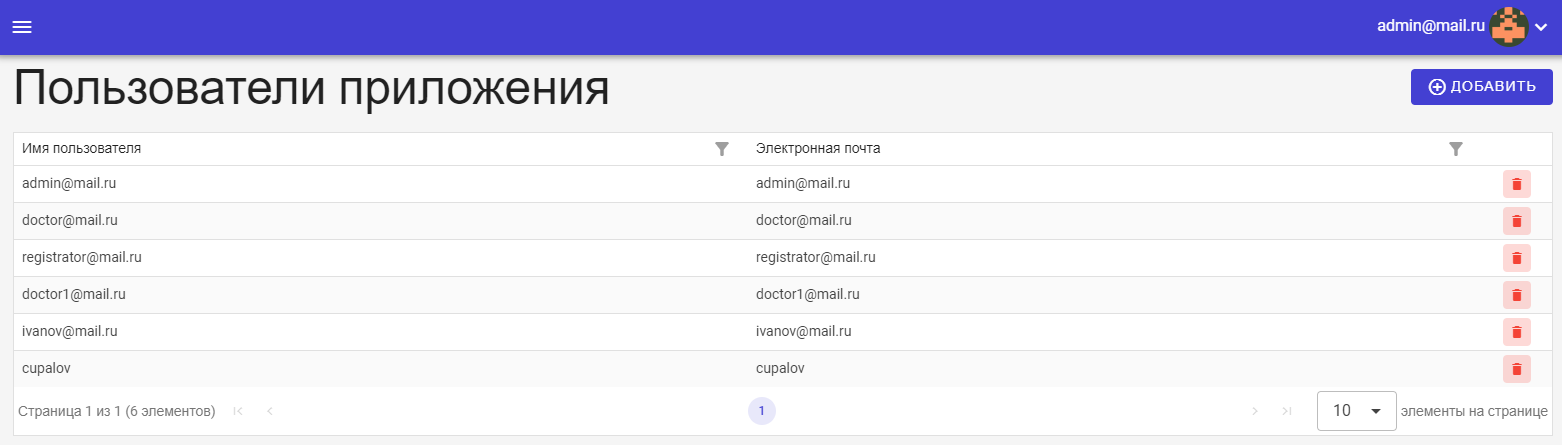


Рисунок 63 – Страница «Пользователи приложения»

Добавление нового пользователя осуществляется путем нажатия на кнопку «Добавить», которая отобразит модальное окно для ввода данных.

При некорректном вводе данных после нажатия на кнопку «Сохранить» отобразится текст с ошибками.

Редактирование происходит путем двойного нажатия по строке, после которого отобразится модальное окно для редактирования ранее вводимых данных.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

Просмотр всех ролей системы осуществляется на странице «Роли приложения» (рисунок 64), переход на данную страницу может быть выполнен как через боковую панель навигации, так и через меню расположенном в заголовке.

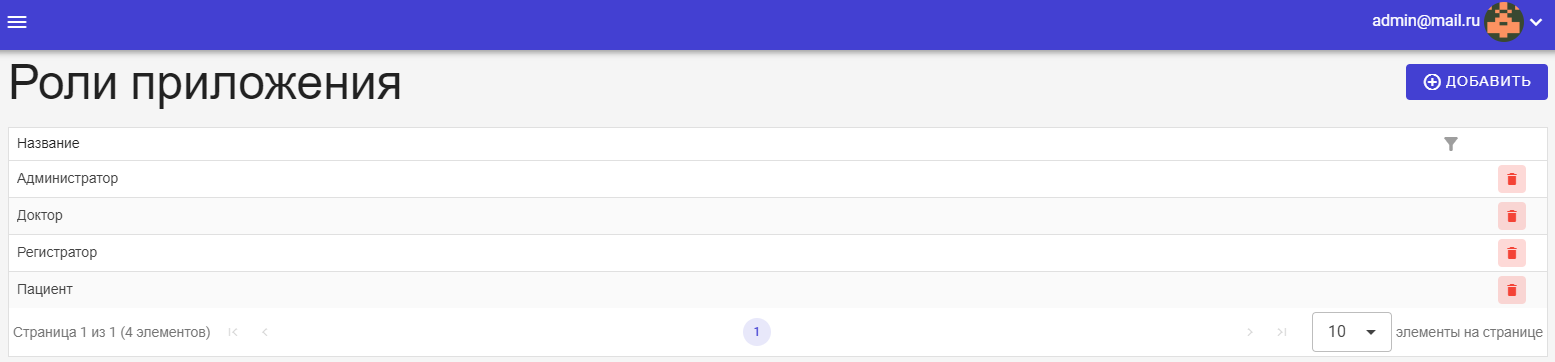


Рисунок 64 – Страница «Роли приложения»

Добавление новой роли осуществляется путем нажатия на кнопку «Добавить», которая отобразит модальное окно для ввода данных.

При некорректном вводе данных после нажатия на кнопку «Сохранить» отобразится текст с ошибками.

Удаление происходит путем нажатия по кнопке удалить у соответствующей строки и осуществить подтверждение.

На большинстве страниц связанных с выводом данных предусмотрена фильтрация (рисунок 65), которая позволяет выводить данные по выбранным параметрам, а также экспорт для дальнейшей работы с ними (рисунок 66).

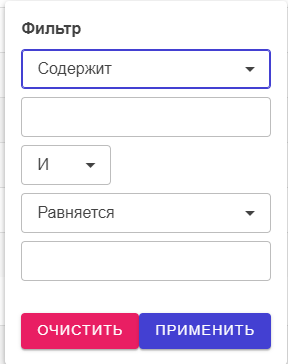


Рисунок 65 – Фильтрация

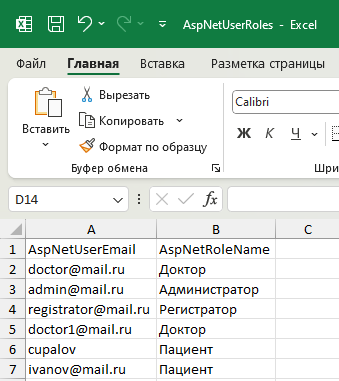


Рисунок 66 – Экспорт

# **2.4.2 Разработка программных модулей решения**

Программный модуль – это независимая часть программного обеспечения, которая имеет определенную функциональность и может быть использована отдельно или в составе других программ.

Модуль представляет собой логически законченный блок кода или набор функций, который выполняет определенные задачи. Они обеспечивают удобство разработки, сопровождения и расширения программного продукта.

В автоматизированном приложении были реализованы следующие функции:

1. Авторизация (рисунок 67).
2. Получение данных докторов (рисунок 68).
3. Создание доктора (рисунок 69).
4. Обновление доктора (рисунок 70).
5. Удаление доктора (рисунок 71).
6. Отправка сообщения на почту (рисунок 72).
7. Экспорт CSV-файла (рисунок 73).

На рисунке 67 представлен код, отвечающий за авторизацию в систему. После передачи имени пользователя, пароля и ссылки на страницу, происходит проверка того на пустоту в результате появиться либо ошибка, либо успешный переход на страницу.



Рисунок 67 – Авторизация

На рисунок 68 представлен код для получения данных об докторах из базы данных. Изначально происходит получение всех объектов «Doctor» из базы данных в качестве исходного набора данных. После чего происходит связывание трех сущностей: пол, специализация и пользователь. Затем смотрится, если запрос (query) был не пустой, то его свойства используются для фильтрации и сортировки результатов, что позволяет динамически запрашивать нужные данные. После чего происходит заполнение данными и возврат в виде списка, который будет представлять собой запрос к базе данных и выполняться при итерации или вызове метода «ToList()».

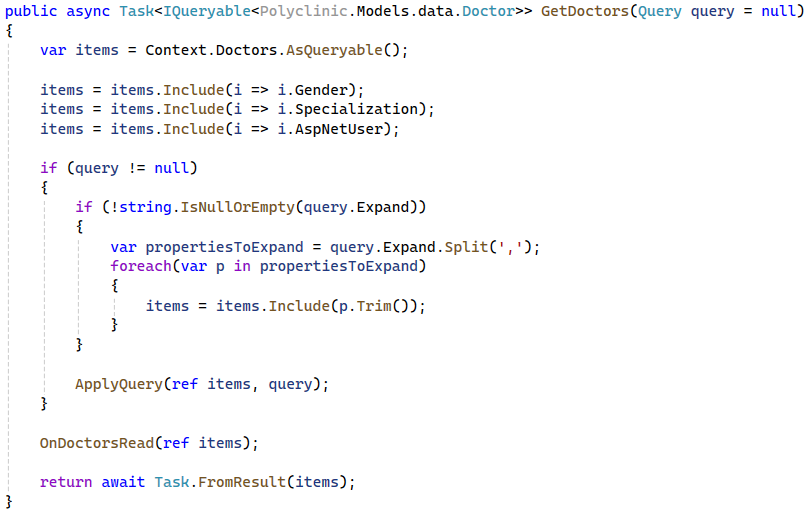


Рисунок 68 – Получение данных докторов

На рисунок 69 представлен код позволяющий создавать докторов. Метод «OnDoctorCreated» позволяет разработчику вносить какие-то правки перед сохранением в базу данных. Изначально происходит проверка на наличие такой-же записи в базе данных, если запись найдена – появиться исключение. После чего вызывается метод сохранения и возвращение созданного объекта.

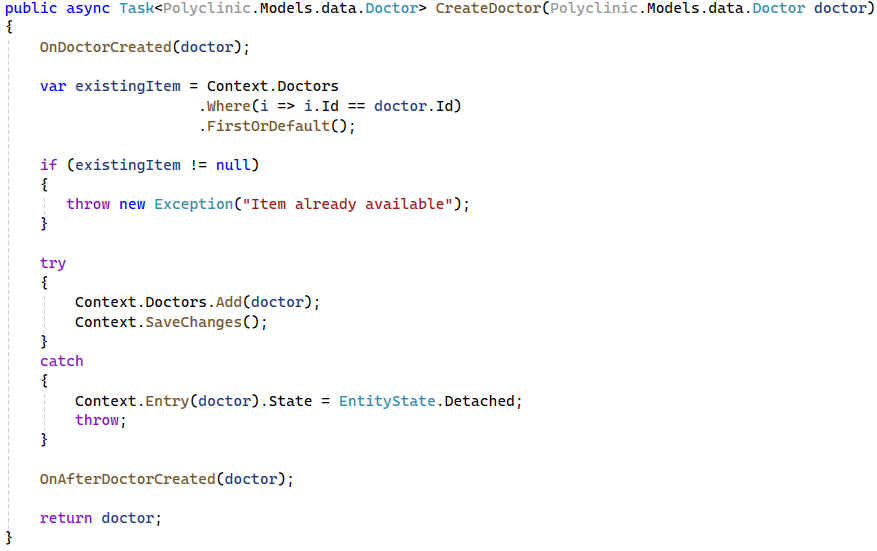


Рисунок 69 – Создание доктора

На рисунке 70 представлен код для обновления данных доктора. Перед обновлением объекта вызывается событие «OnDoctorUpdater», которое позволяет разработчикам вносить правки перед обновлением. Затем метод получает исходный объект по идентификатору, если такого объекта нету – то выйдет исключение. После получения исходного объекта устанавливаются новые данные и сохраняются в базу данных с последующим возвращением обновленного объекта.

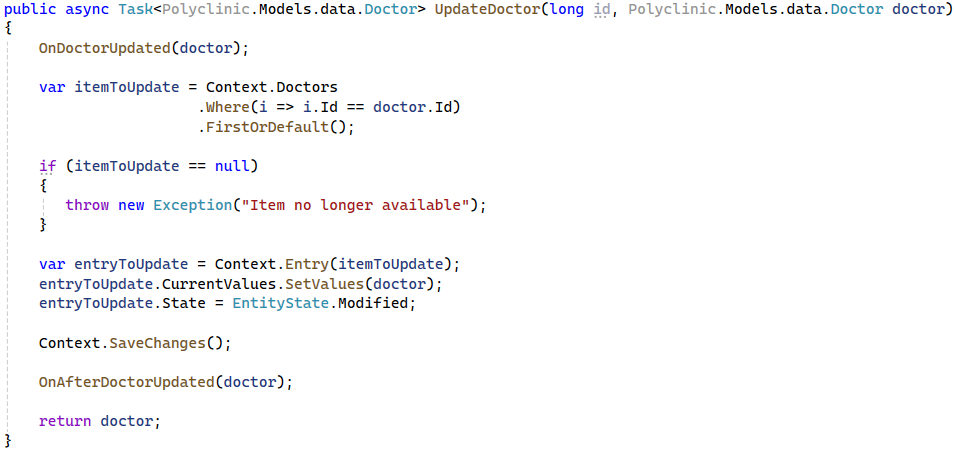


Рисунок 70 – Обновление доктора

На рисунке 71 представлен код для удаления доктора. Перед началом удаления происходит получение всех связанных сущностей для дальнейшего удаления связанных записей с данным доктором. Затем метод получает нужную запись по идентификатору и проверяет на наличие, если данного объекта нет – то выдается исключение. Затем вызывается событие «OnDoctorDelete», которое позволяет выполнять различные модификации разработчику перед удалением. После чего происходит удаление и сохранение изменений в базе данных.



Рисунок 71 – Удаление доктора

На рисунке 72 представлен код отправки сообщения на почту. Данный код изначально читает файл json, который содержит информацию (данные почты, с которого будут осуществляться отправки сообщений, заголовок сообщения, описание и так далее), затем осуществляется поиск почты пациента для отправки сообщения ему на почту. Если пациент не указывал адрес электронной почты, то регистратору отобразиться об этом сообщение.



Рисунок 72 – Отправка сообщения на почту

На рисунке 73 представлен код для формирования экспорта в CSV-файл. Сначала метод получается свойство элемента «query» с помощью «GetProperties» для определения имен столбцов для файла CSV. Затем создается объект «StringBuilder», который будет использоваться для постепенного создания содержимого файла. Далее в цикле перебирается каждый элемент и создается для каждой записи своя строка, а после чего происходит разбитие по столбцам при помощи разделителя по запятой. Затем происходит преобразование в массив байтов кодировки UTF-8, который формирует файл.

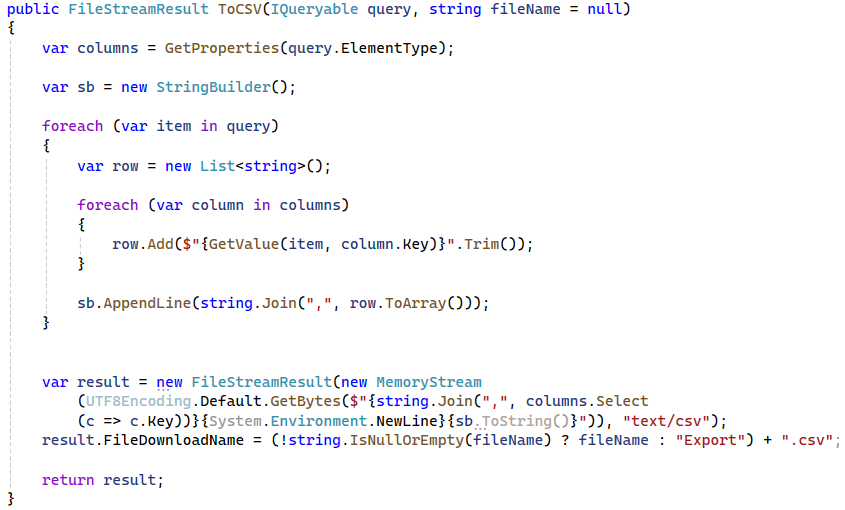


Рисунок 73 – Экспорт CSV-файла

# **2.5 Тестирование разработанного решения**

Для тестирования программы, использовались следующие методы:

Модульное тестирование – это метод тестирования программного обеспечения, при котором отдельные модули или компоненты программы тестируются независимо друг от друга. Целью модульного тестирования является проверка корректности работы отдельных частей программы, исключение ошибок внутри модулей и обеспечение их надежности и стабильности.

Тестирование «чёрным ящиком» – при таком подходе к тестированию тестируются функциональные возможности программы без знания ее внутренней структуры. Тестировщик работает с интерфейсом программы, вводя данные и проверяя вывод, чтобы убедиться, что программа работает правильно.

Тестирование «белым ящиком» – при таком подходе к тестированию тестировщик имеет доступ к исходному коду программы и анализирует его для выявления ошибок и проверки корректности работы программы.

Тестирование авторизации, проверка на корректность ввода данных для входа. При осуществлении входа в систему, когда поля для ввода имени пользователя и пароля являются пустыми, отображается текст с ошибкой (рисунок 74), если поля заполнены, то текст не отображается (рисунок 75).

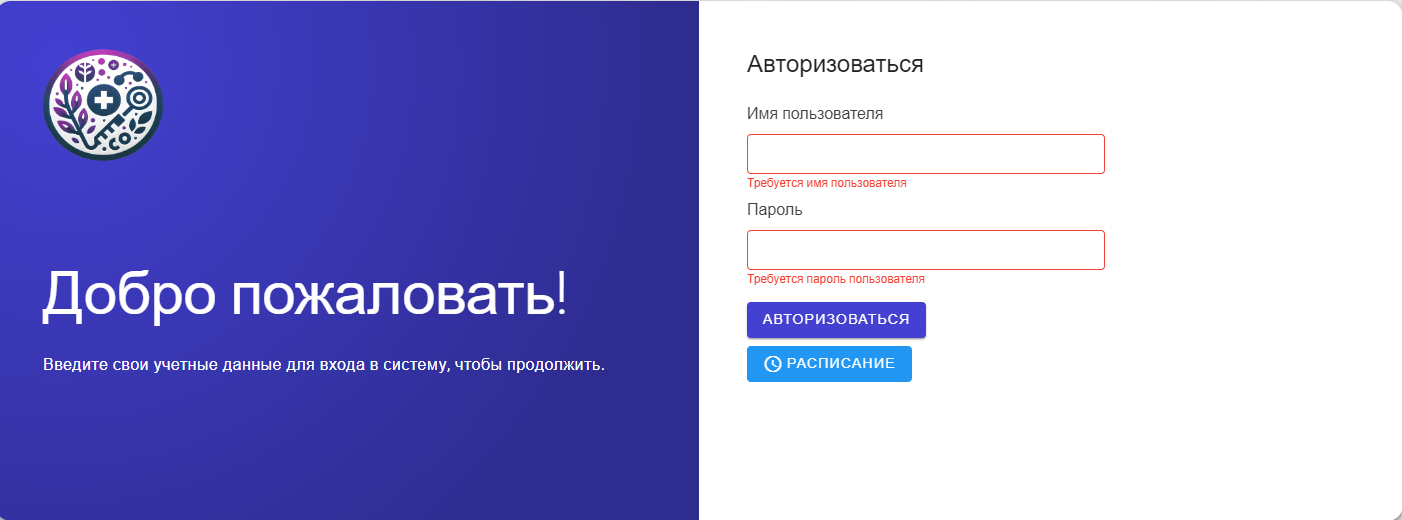


Рисунок 74 – Поля являются пустыми

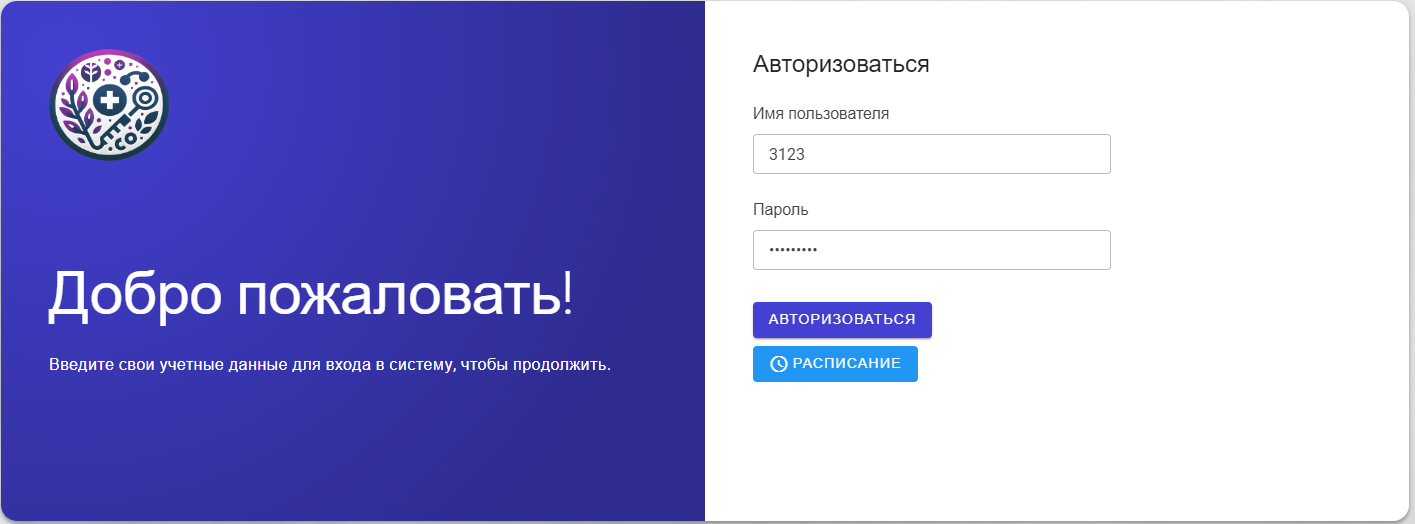


Рисунок 75 – Поля не являются пустыми

Тестирование на наличие учетной записи в системе для входа. Если поля для ввода имени пользователя или пароля не совпадают с учетной записью в базе данных, отображается ошибка (рисунок 76), иначе осуществляется вход в систему (рисунок 77).

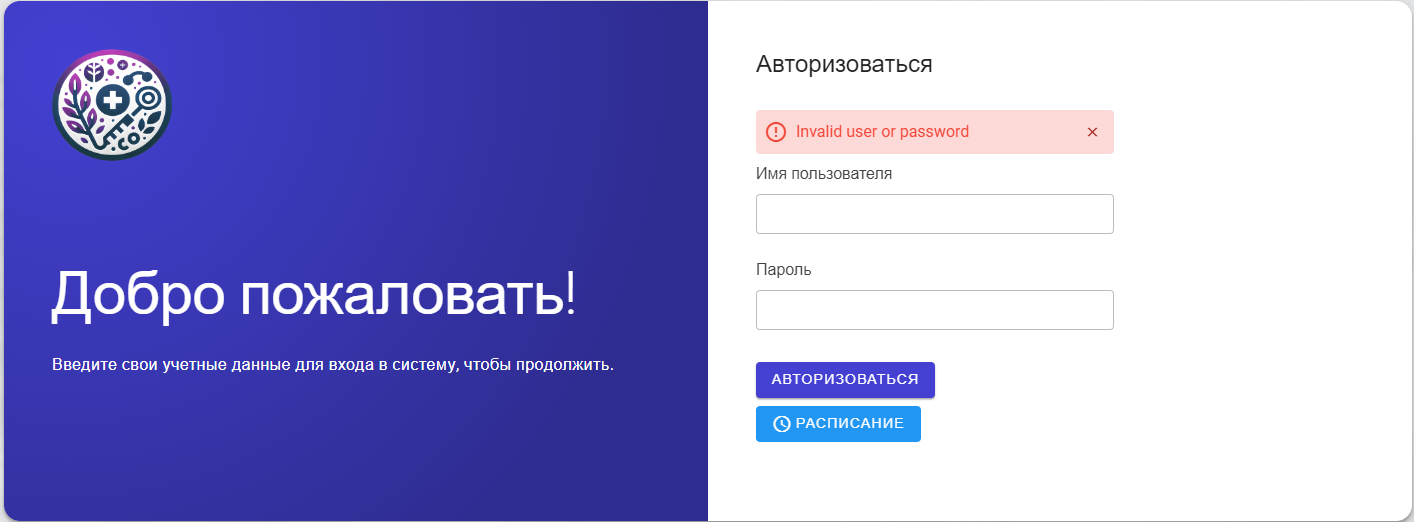


Рисунок 76 – Пользователя с введенными данными не существует

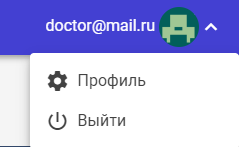


Рисунок 77 – Успешная авторизация

Тестирования появления подтверждения перед удалением записи (рисунок 78). Окно подтверждения должно появляться перед каждым удалением записей из любой таблицы. Если осуществить нажатие по кнопке «Да», то удаление произойдет, иначе, если нажать на кнопке «Нет» или просто закрыть окно, то удаления не произойдет.

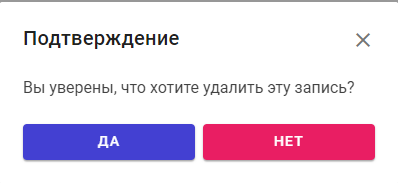


Рисунок 78 – Подтверждение перед удалением

Тестирования ввода обязательных полей. Если пользователь хочет сохранить данные при этом не заполнил обязательные поля, отобразиться текст с ошибкой (рисунок 79). Если пользователь внесет данные в поля, то данный текст не отобразиться (рисунок 80).

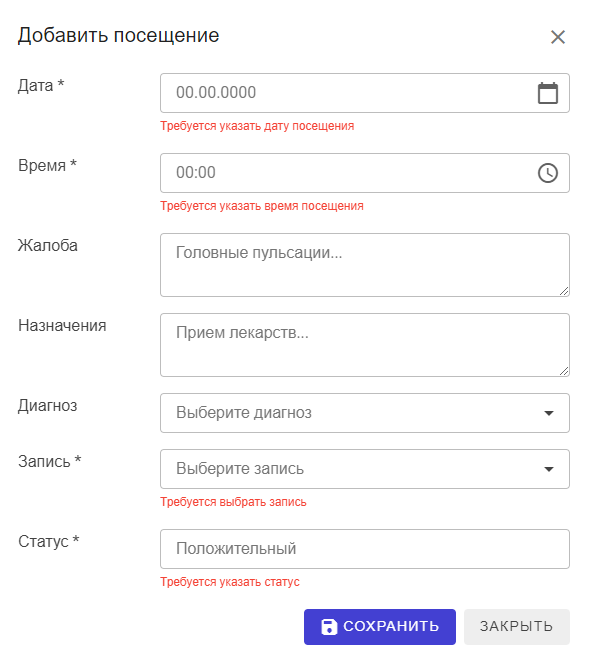


Рисунок 79 – Отсутствие данных в обязательных полях

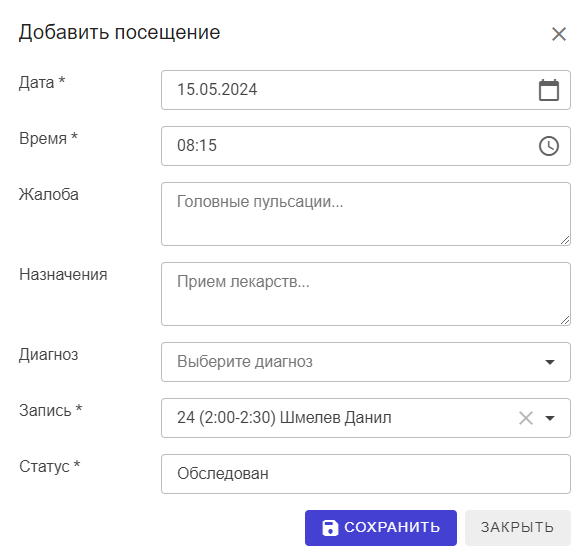


Рисунок 80 – Наличие данных в обязательных полях

Тестирование появление ошибки при сохранении или редактирование данных с неверным форматом или с уже имеющимися аналогичными данными в базе данных. Если пользователь хочет создать роль, которая уже имеется в системе, отобразиться ошибка (рисунок 81).

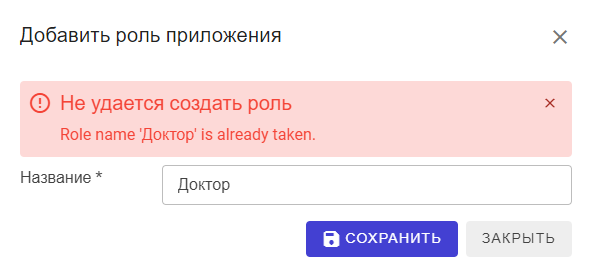


Рисунок 81 – Создание роли, которая уже имеется

Проверка на отображения сообщения об отсутствие электронной почти при осуществлении записи пациента. Если при сохранении во время создания или редактирования записи у пользователя не была указана почта, то отобразиться сообщение (рисунок 82).

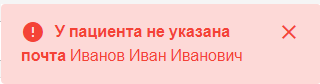


Рисунок 82 – Отсутствие электронной почти у пациента

Проверка открытия страницы, которая не отображается пользователю. Если пользователь пытается открыть страницу будучи неавторизованным, либо не имеющим к ней доступ, то ему отобразится страница с информацией об том, что он неавторизованный (рисунок 83).

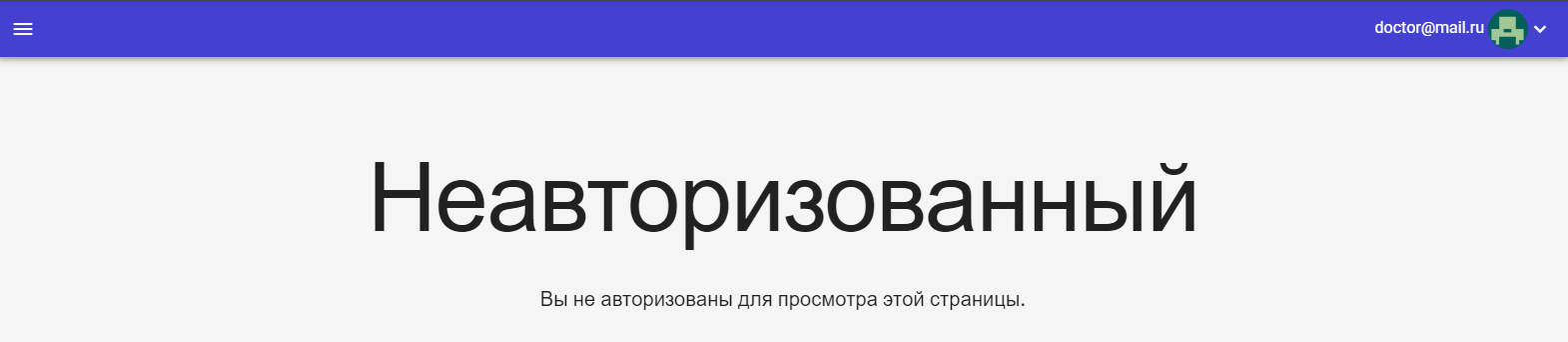


Рисунок 83 – Открытие страниц недоступных пользователю

# **3 Организационно-экономическая часть**

## **3.1 Расчет сметы затрат на разработку программы**

Для расчета сметы затрат составлен проект выполнения работ по созданию программы. Он представляет собой перечень мероприятий, которые необходимо выполнить, чтобы разработать и внедрить программу «Автоматизации учёта данных регистратуры поликлиники». Работы перечислены в требуемой последовательности с установленной продолжительностью каждого этапа, данные в таблице 19.

Таблица 19 – Проект выполнения работ по созданию программы.

|  |  |
| --- | --- |
| **Наименование этапов** | **Продолжительность, дни** |
| Получение задания на разработку программы | 1 |
| Сбор и изучение требований заказчика, анализ предметной области | 2 |
| Проектирование концептуальной модели | 2 |
| Проектирование графического интерфейса программного приложения | 4 |
| Разработка функциональных возможностей программы | 6 |
| Тестирование программы | 2 |
| Отладка программного продукта и встреча с заказчиком | 2 |
| Эксплуатация, сдача проекта заказчику | 1 |

Общие затраты времени на разработку программы определены как сумма продолжительности работ и составляют 20 дней.

В составе материальных затрат по разработке программы отражена стоимость:

* приобретаемых материалов, которые являются необходимым компонентом при проведении работ;
* покупной энергии, расходуемой на производственные и хозяйственные нужды.

Рассчитываем затраты на эксплуатационные материалы, исходные данные представлены в таблице 20, в ней же и результаты расчетов.

Таблица 20 – Затраты на эксплуатационные материалы.

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование материала** | **Количество** | **Цена, руб.** | **Сумма, руб.** |
| Диск (шт.) | 1 | 25 | 25 |
| Интернет (дней) | 20 | 23 | 460 |
| Бумага (шт.) | 20 | 0,6 | 12 |
| Ручка (шт.) | 1 | 5 | 5 |
| Итого |  |  | См = 502 руб. |

См – сумма затрат на эксплуатационные материалы.

Расчёт стоимости электроэнергии.

ЭВМ будет использоваться 20 дней по 8 часов в день, то есть 160 часов.

Исходные данные:

* потребляемая мощность – 0,30 кВт/ч;
* время работы на ЭВМ – 160 ч;
* тариф по электроэнергии – 3.9 руб. /кВт

Рассчитываем стоимость электроэнергии (Сэл.) по формуле:

Сэл. = P × tраб. × Ц, руб.,

где P – потребляемая мощность, кВт/ч;

tраб. – время работы на ЭВМ, ч;

Ц – цена за 1 кВт/ч, руб.

Сэл. = 0.30 × 160 × 3.9 = 187,2 руб.

Рассчитываем сумму материальных затрат (См.з.):

См.з. = См. + Сэл., руб.,

См.з. = 502 + 187,2 = 689,2 руб.

Для оплаты труда программиста чаше всего используется повременная зарплата.

Затраты на оплату труда включают:

* заработную плату программиста;
* оплату отпуска программиста за отработанное время.

Расчет заработной платы программиста.

Исходные данные:

* время работы над программой, 20 дн.;
* должностной оклад за месяц, 20000 руб.;
* количество рабочих дней за месяц, 22 дн.;
* процент премии, 50 %;
* районный коэффициент, 1,15.

Рассчитываем заработную плату по тарифу:

где Зпл.по тар.- заработная плата по тарифу, руб;

T – время работы над программой, дн.;

ДО – должностной оклад за месяц, руб.;

tм-ц – количество дней за месяц, дн.

Рассчитываем сумму премии:

где ∑прем. – сумма премии, руб.;

%Пр. – процент премии.

Рассчитываем сумму зарплаты за отработанное время:

. руб.

Рассчитываем сумму зарплаты с районным коэффициентом:

,

где с районным коэффициентом, руб.,

РК – районный коэффициент, 1,15

Для расчета очередного отпуска определяем дневную зарплату (Дн.з.пл.):

Исходные данные:

* дневная зарплата, 1568,18 руб.;
* средняя продолжительность отпуска, 20 рабочих дн.;
* годовой фонд рабочего времени, 247 дн.;
* время работы над программой, 20 дн.

Рассчитываем сумму оплаты за отпуск:

где ∑оп.от. – сумма оплаты отпуска, руб.;

tот. – продолжительность отпусков, дн.;

Фр.в. – годовой фонд рабочего времени, дн.;

Т – время работы над программой, дн.;

Дн.з.пл. – дневная заработная плата, руб.;

Рассчитываем сумму затрат на оплату труда (Зот.):

, руб.

Для разработки программы «Автоматизации учёта данных регистратуры поликлиники» использовалась вычислительная техника в виде персонального компьютера, в который входит монитор и системный блок. Вычислительная техника входит в состав основных производственных фондов.

Основные производственные фонды – это совокупность средств труда, функционирующих в сфере материального производства в неизменной натуральной форме в течение длительного времени и переносящих свою стоимость на вновь созданный продукт по частям, по мере изнашивания.

Амортизационные отчисления – это постепенное погашение стоимости основных фондов предприятия (зданий, оборудования, подвижного состава и т.п.), изнашивающихся в процессе работы и от времени.

Рассчитываем амортизационные отчисления за год, исходные данные и результаты расчётов в таблице 21.

Таблица 21 - Амортизационные отчисления за год

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование основных производственных фондов** | **Стоимость ОПФ, руб.** | **Норма амортизации, %** | **Сумма амортизации, руб.** |
| Системный блок | 100000 | 17 | 17000 |
| Монитор | 32000 | 17 | 5440 |
| Итого |  |  | Ar = 22440 |

Ar - годовая сумма амортизационных отчислений.

Годовая сумма амортизационных отчислений (Ar) рассчитывается по формуле:

где Cn – стоимость первоначальная, руб;

Ha – норма амортизации, %

Рассчитываем годовую сумма амортизационных отчислений системного блока, Aг с.б.:

Рассчитываем годовую сумму амортизационных отчислений монитора, Aг м.:

Рассчитываем сумму амортизационных отчислений за время разработки программы:

где Ar – годовая сумма амортизационных отчислений, руб.;

n – количество дней на разработку программы;

365 – количество дней в году.

Себестоимость — стоимостная оценка текущих затрат на производство и реализацию продукции.

Себестоимость продукции формируется из следующих элементов

* материальные затраты;
* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных фондов;
* накладные расходы.

В элементе «отчисления на социальные нужды» отражаются обязательные отчисления органам государственного и негосударственного социального страхования, Пенсионного фонда, Государственного фонда занятости и медицинского страхования, в процентах к оплате труда работников.

Общий процент отчислений в фонды остается прежним по сравнению с предыдущим 2023 годом – 30%, из них:

* 22% в пенсионный фонд;
* 2,9% в фонд социального страхования;
* 5,1% в фонд медицинского страхования.

Накладные расходы — это налоги, сборы, отчисления в специальные фонды, платежи по кредитам, затраты на командировки, оплата услуг связи и другие.

Рассчитываем себестоимость разрабатываемой программы по формуле:

руб.,

где С/С – себестоимость, руб.;

См.з. – материальные затраты, руб.;

З.о.т. – затраты га оплату труда, руб.;

А – амортизационные отчисления, руб.;

От. – отчисления на социальные нужды, руб.;

НР – накладные расходы, руб.

Рассчитываем отчисления на социальные нужды.

Исходные данные:

* затраты на оплату труда -руб.;
* установленный размер отчислений - 36 %.

где %отчисл. – процент отчислений, %

От. – отчисления на социальные нужды, руб.

Рассчитываем накладные расходы (80% от основной заработной платы)

Рассчитываем себестоимость.

Исходные данные:

См.з – материальные затраты, 871,45 руб.;

З.о.т. – затраты на оплату труда, руб.;

А – амортизационные отчисления, руб.;

От. – отчисления на социальные нужды, руб.;

НР – прочие расходы (накладные расходы), руб.

Определяем структуру себестоимости разрабатываемой программы.

Структура – процентное содержание элементов затрат к общей сумме себестоимости.

Таблица 22 – Структура себестоимости

|  |  |  |
| --- | --- | --- |
| **Наименование статей затрат** | **Сумма, руб.** | **Структура, %** |
| Материальные затраты | 871,45 | 1,19 |
| Затраты на оплату труда | 33903,2 | 46,25 |
| Отчисления на социальные нужды | 10170,96 | 13,88 |
| Амортизационные отчисления | 1229,6 | 1,68 |
| Накладные расходы | 27122,56 | 37 |
| Итого: | 73297,77 | 100 |

Вывод: самую большую долю затрат на разработку программы составляет 46,25% затраты на оплату труда, а самую малую долю – 1,19% материальные затраты.

# **3.2 Расчет экономической эффективности внедрения программы**

Экономическая эффективность – это соотношение результатов с затратами.

Если разрабатываемая программа пользуется спросом, то возможна ее реализация. В этом случае эффективность определяется возможной прибылью от реализации разработанной программы.

Для этого необходимо установить цену на разработанную программу.

При установлении цены используют полную сбытовую себестоимость товарной продукции, она включает производственную себестоимость и внепроизводственные расходы.

Внепроизводственные (коммерческие) расходы включают затраты, связанные с реализацией продукции (расфасовка, упаковка, отгрузка, реклама, маркетинговые исследования, комиссионные; хранение, транспортировка), а также различного рода отчисления и платежи.

Полная себестоимость рассчитывается по формуле:

где С/Ск.– полная себестоимость единицы продукции, руб;

С/С – себестоимость единицы продукции производственная, руб;

Вн.р. – внепроизводственные расходы (7-10% от производственной себестоимости), руб.

Исходные данные:

* внепроизводственные расходы – руб. (8 % от производственной себестоимости);
* производственная стоимость – руб.

Оптовую цену предприятия можно определить расчетным путем, используя уровень рентабельности продукции.

Уровень рентабельности продукции устанавливают в зависимости от величины прибыли (20-40% от коммерческой (полной) себестоимости единицы продукции).

Рассчитываем цену разрабатываемой программы по формуле:

где Ц – цена разрабатываемой программы, руб;

С/Ск. – себестоимость коммерческая (полная) разрабатываемой программы, руб.;

Pnp. – рентабельность продукции, %.

Исходные данные:

* рентабельность продукции – 20 %;
* полная себестоимость продукции – руб.

Расчет прибыли от реализации программы.

Прибыль от реализации продукции (работ, услуг) определяется как разница между выручкой от реализации продукции (работ, услуг) в действующих ценах (без налога на добавленную стоимость, акцизного налога) и затратами на ее производство и реализацию, включаемыми в себестоимость продукции (работ, услуг).

Рассчитываем прибыль от реализации разрабатываемой программы:

где Пр. – прибыль от реализации программы, руб.

# **4 Мероприятия по технике безопасности и охране окружающей среды**

# **4.1 Организация безопасности труда на предприятии**

Для обеспечения безопасных условий труда и минимизации рисков возникновения пожаров и других чрезвычайных ситуаций в медицинских учреждениях необходимо соблюдать следующие меры противопожарной безопасности:

1. Проведение обязательного вводного и повторного противопожарного инструктажа для всех работников, включая административный персонал, согласно утвержденным программам МЧС РФ.
2. Проведение первичного противопожарного инструктажа на рабочем месте перед началом трудовой деятельности для каждого нового сотрудника.
3. Ежегодное или полугодовое проведение повторного противопожарного инструктажа для всех работников, особенно на объектах с повышенной взрыво-пожароопасностью.
4. Обучение работников правилам эвакуации и действиям в случае пожара.
5. Проведение тренировок и учений по эвакуации, использованию средств пожаротушения.
6. Проведение регулярных проверок и обслуживания пожарного оборудования.
7. Обеспечение наличия необходимого количества средств индивидуальной защиты (СИЗ) и их использование при необходимости.
8. Проведение аудитов и проверок безопасности труда для выявления потенциальных опасностей и рисков.
9. Обеспечение доступа к информации о правилах и мерах безопасности труда для всех работников.

Соблюдение этих мер позволяет медицинским учреждениям создать безопасную рабочую среду, предотвратить возникновение пожаров и других чрезвычайных ситуаций, а также обеспечить безопасность пациентов и персонала.

# **4.2 Техника безопасности при работе с вычислительной техникой**

Для создания безопасных условий работы с вычислительной техникой в медицинских учреждениях необходимо соблюдать следующие меры безопасности:

1. Регулярно проводить проверку электрооборудования и розеток на соответствие стандартам безопасности.
2. Использовать стабилизаторы напряжения для защиты компьютерной техники от перепадов напряжения.
3. Размещать компьютеры и другие устройства таким образом, чтобы обеспечить хорошую вентиляцию и предотвратить перегрев.
4. Следить за состоянием кабелей и проводов, избегать их перекручивания и повреждения.
5. Использовать эргономическую мебель и оборудование, чтобы предотвратить возможные травмы и нагрузки на позвоночник.
6. Регулярно делать перерывы для отдыха и гимнастики, чтобы снять напряжение с мышц и глаз.
7. Проводить обучение сотрудников по правильной организации рабочего места и правильным приемам работы с компьютером.

Соблюдение этих мер позволяет медицинским учреждениям создавать безопасные условия труда для сотрудников, работающих с вычислительной техникой, предотвращать возникновение травм и профессиональных заболеваний, а также обеспечить бесперебойную работу компьютерного оборудования.

# **Заключение**

В соответствии с полученным заданием, была достигнута цель дипломного проекта: «Автоматизация учета данных регистратуры поликлиники».

В ходе работы над дипломным проектом были выполнены следующие задачи:

1. Изучены требования заказчика и проведен анализ предметной области — данные пациента и их учет.
2. Построена модель базы данных.
3. Создан пользовательский интерфейс учета данных регистратуры поликлиники.
4. Реализованы функциональные возможности электронной системы.
5. Проведено тестирование и выполнена отладка приложения для обеспечения его корректной работы и в соответствии со всеми требованиями.
6. Оформлена техническая документация в виде пояснительной записки на дипломный проект.

Итогом дипломного проекта стала разработанная система автоматизация учета данных регистратуры поликлиники с помощью языка программирования C#, системы управления базами данных SQLiteStudio и среды разработки Microsoft Visual Studio и Radzen Blazor Studio.

# **Список использованных источников**

1. Database setup | Create a Blazor CRM application with Radzen Blazor Studio. URL: https://www.radzen.com/blazor-studio/documentation/crm/database/ (Дата обращения 20.04.2024).
2. Basic CRUD operations in Blazor using SQLite as the database. URL: https://executecommands.com/crud-in-blazor-using-sqlite-entity-framework/ (Дата обращения 20.04.2024).
3. Первое приложение на Blazor. URL: https://metanit.com/sharp/blazor/1.2.php (Дата обращения 20.04.2024).
4. Radzen Blazor Components. URL: https://blazor.radzen.com/ (Дата обращения 20.04.2024).
5. Дружинина, И.В. Информационные технологии в профессиональной деятельности средних медицинских работников: учебное пособие / И.В. Дружинина. – СПб.: Лань, 2020. – 122с.: ил. – ISBN 978-5-8114-7186-7.
6. Кислицын, Е.В. Современные технологии разработки программного обеспечения: учеб. пособие / Е. В. Кислицын, М. А. Панов. - Екатеринбург: Изд-во УрГЭУ, 2021. - 176 с.: ил. - ISBN 978-5-8111-7114-5.
7. Часовских, В.П. Управление ИТ-разработкой и внедрением: учеб. пособие / В. П. Часовских, М. П. Воронов, В. Г. Лабунец, Е. Н. Стариков. - Екатеринбург: Изд-во УрГЭУ, 2021. - 173 с.: ил. - ISBN 978-5-8458-1812-5.
8. Кэмпбелл, Л. Базы данных. Инжиниринг надежности: научная литература / Л. Кэмпбелл, Ч. Мейджорс – СПб.: Питер, 2020. – 304 с.: ил. – ISBN 978-5-4461-1310-8.
9. ASP.NET Core Blazor. URL: https://learn.microsoft.com/ru-ru/aspnet/core/blazor/?view=aspnetcore-5.0 (Дата обращения 19.04.2024).
10. Как использовать Entity Framework Core в приложении Blazor. URL: <https://www.c-sharpcorner.com/article/how-to-use-entity-framework-core-in-blazor-application/> (Дата обращения: 19.04.2024).
11. Паттерны проектирования в C#. URL: <https://refactoring.guru/ru/design-patterns/csharp> (Дата обращения: 19.04.2024).
12. C# 9.0 in a Nutshell: The Definitive Reference. URL: https://www.amazon.com/C-9-0-Nutshell-Definitive-Reference/dp/1098100964 (Дата обращения 21.04.2024).
13. Руководство по ASP.NET Core 5. URL: https://metanit.com/sharp/aspnet5/ (Дата обращения 20.04.2024).
14. Создание веб-API с помощью ASP.Net Core. URL: https://learn.microsoft.com/ru-ru/aspnet/core/web-api/?view=aspnetcore-5.0 (Дата обращения 22.04.2024).
15. Синтаксис LINQ. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/linq/ (Дата обращения 22.04.2024).
16. Атрибуты. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/advanced-topics/reflection-and-attributes/ (Дата обращения 22.04.2024).
17. Асинхронное программирование с async и await. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/asynchronous-programming/ (Дата обращения 20.04.2024).
18. Делегаты (Руководство по программированию на C#). URl: https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/delegates/ (Дата обращения 19.04.2024).
19. Blazor in Action. URL: https://www.manning.com/books/blazor-in-action (Дата обращения 19.04.2024).
20. Blazor. URL: https://www.c-sharpcorner.com/technologies/blazor (Дата обращения 19.04.2024).