**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| **ДП.09.02.03.22.181.15.ПЗ** | УТВЕРЖДАЮ  Зам. директора по УР, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_Е.А. Коробкова |

ИНФОРМАЦИОННАЯ СИСТЕМА «ПОЛИКЛИНИКА»

|  |  |  |
| --- | --- | --- |
| Нормоконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (О.Ю. Безносова) |
| Консультант по экономической части: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.А. Белова) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.П. Гордиенко) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (И.А. Паздников) |

Иркутск 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc101985295)

[1 Предпроектное исследование 5](#_Toc101985296)

[1.1 Исследование предметной области 5](#_Toc101985297)

[1.2 Анализ инструментов разработки 6](#_Toc101985298)

[1.3 Обоснование выбора программного продукта 8](#_Toc101985299)

[2 Проектирование 11](#_Toc101985300)

[2.1 Архитектура программного обеспечения 11](#_Toc101985301)

[2.2 Функциональное проектирование 13](#_Toc101985302)

[2.3 Проектирование базы данных 17](#_Toc101985303)

[2.4 Проектирование пользовательского интерфейса 21](#_Toc101985304)

[3 Реализация программного обеспечения 24](#_Toc101985305)

[3.1 Кодирование программного обеспечения 24](#_Toc101985306)

[4 Документирование программного обеспечения 29](#_Toc101985307)

[4.1 Руководство пользователя программного обеспечения 29](#_Toc101985308)

[5 Стоимость разработки и внедрения программного продукта 33](#_Toc101985314)

[5.1 Организационно-экономическое обоснование проекта 33](#_Toc101985315)

[5.2 Расчет затрат на разработку программного продукта 33](#_Toc101985316)

[5.3 Расчет затрат на внедрение программного продукта 36](#_Toc101985317)

[5.4 Основные выводы 37](#_Toc101985318)

[ЗАКЛЮЧЕНИЕ 38](#_Toc101985319)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc101985320)

[Приложение А - Листинг маршрутов 41](#_Toc101985321)

[Приложение Б - Листинг JavaScript 43](#_Toc101985322)

ВВЕДЕНИЕ

Темой дипломного проекта является информационная система «Поликлиника».

Здоровье важнейших фактор жизни человека, его сохранение благоволить долгожительству. Своевременная обследование в поликлинике может спасти жизнь и сохранить здоровье. Поликлиники организуют лечение и поддержание здоровья граждан.

Самое главное в работе поликлиники — это произведение записи клиентов к врачам и своевременное выполнения ими приема. Все необходимые данные удобно хранить в таблицах. Информация в базе данных (далее БД) тоже представляется в виде таблиц, что позволяет организовать историю приемов врачей, место хранения данных клиентов и работников. Качество и надежность результатов вышеназванных действий служит основой эффективной работы поликлиники. Актуальным является сохранять сведения о названных процедурах в специальных модулях программного обеспечения. Данную базу можно применить в любой поликлинике. Для сохранения безопасности данных необходимо организовать грамотное хранение этих данных. Для этого нужна общая база данных, включающая всю необходимую информацию. И программа, которая будет понятна и способна организовать работу с базой данных.

Цель: разработать информационную систему «Поликлиника».

Задачи:

* + Провести предпроектное исследование.
  + Составить техническое задание на разработку программного продукта в соответствии с ГОСТ.
  + Провести проектирование программного продукта.
  + Реализовать программный продукт.
  + Разработать документы для программного продукта.
  + Рассчитать стоимость разработки и внедрения программного продукта.

База данных должна быть спроектирована так, чтобы обеспечить хранение всех необходимых данных, имея при этом максимально упрощенную структуру. Структура базы данных должна обеспечивать целостность и корректность информации.

1. Предпроектное исследование
   1. Исследование предметной области

Предметной областью дипломного проекта является информационная система «Поликлиника» (далее ИС).

Поликлиника — многопрофильное или специализированное лечебно-профилактическое учреждение для оказания амбулаторной медицинской помощи больным на приёме и на дому.

Врач, также доктор — специалист с высшим медицинским образованием, использующий свои навыки, знания и опыт в профилактике и лечении заболеваний, поддержании нормальной жизнедеятельности организма человека.

Прием врача в обязательном порядке включает в себя консультацию и физикальное обследование и по показаниям проведение других обследований, направленных на установления диагноза заболевания и назначение его лечения.

Все обращающиеся в поликлинику клиенты, при прохождении приема у врача обязаны указать свои данные и следовать рекомендациям врача.

Таким образом, в функционирование поликлиник входит:

1. Навигация по сайту.
2. Разграничение уровней доступа.
3. Чтение данных.
4. Хранение и добавление данных.
5. Удаление и изменение данных.
6. Запись на приём.
7. Регистрация.
8. Авторизация.
9. Уведомление о регистрации на e-mail.
10. Уведомление о записи на приём на e-mail.
    1. Анализ инструментов разработки

Для разработки ИС можно использовать следующие программные продукты: SQL, Django, PyCharm, Python, РНР, Ruby, JavaScript, JetBrains PhpStorm, JetBrains WebStorm, Notepad++, Laravel, NodeJS, MySQL, Oracle, PostgreSQL, Microsoft Visual Studio, MySQL Workbench, PhpMyAdmin, Microsoft Visio.

SQL — «Язык структурированных запросов» декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Python — это высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений и другие.

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django.

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования МVС.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

РНР: Hypertext Preprocessor — «Инструменты для создания персональных веб-страниц» -серверный язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений.

Notepad++ — свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки, а также языков описания аппаратуры VНDL и Verilog.

JetBrains PhpStorm — представляет собой интеллектуальный редактор для РНР, НТМL и JavaScript, с возможностями анализа кода на лету, предотвращения ошибок в коде, автоматизированными средствами рефакторинга для РНР и JavaScript.

JetBrains WebStorm — интегрированная среда разработки на JavaScript, CSS & НТМL от компании JetBrains, разработанная на основе платформы IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями

JavaScript — это полноценный динамический язык программирования, который применяется к НТМL документу, и может обеспечить динамическую интерактивность на веб-сайтах.

Laravel — веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC.

NodeJS — программная платформа, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.j s добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на С++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кoдa.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle

PhpMyAdmin — веб-приложение с открытым кодом, написанное на языке РНР и представляющее собой веб-интерфейс для администрирования CYБД MySQL.

PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIХ-подобных платформ, включая AIX, различные ВSD-системы, НР-UХ, IRIX, Linux, macOS, Tru64, QNX, а также для Microsoft Windows.

Oracle Database или Oracle RDBMS — объектно-реляционная система управления базами данных компании Oracle

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

* 1. Обоснование выбора программного продукта

Для разработки ИС было решено использовать PHPStorm, PHPMyAdmin и MySQL.

Для выбора инструментов разработки были произведены сравнения, которые показаны в таблице 1, таблице 2, таблице 3, и таблице 4.

Таблица 1 — Сравнение IDE для разработки программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название IDE | Visual studio code | Notepad++ | JetBrains PhpStorm | JetBrains  WebStorm |
| Стоимость | Свободное ПО | Свободное ПО | Студенческая лицензия | Студенческая лицензия |
| Автосохранение | + | - | + | + |
| Автодополнение | + | - | + | + |
| Интеграция с системами управления версиями (git) | -  (возможно добавить плагин) | -  (возможно добавить плагин) | + | + |
| Поиск по функции в коде | - | - | + | + |
| Встроенные языки программирования | -  (возможно добавить плагин) | -  (возможно добавить плагин) | PHP, HTML, JavaScript, CSS, SQL | HTML, JavaScript, CSS, SQL |

Таким образом, из представленных IDE можно выделить несколько инструментов, способных облегчить и ускорить разработку программного обеспечения, а именно JetBrains PhpStorm, JetBrains WebStorm, так как программный продукт будет реализован с помощью web - технологий. Исходя из того, что для разработки необходим язык программирования PHP, то в качестве IDE был выбран JetBrains PhpStrorm.

Таблица 2 — Сравнение языков программирования для разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Название языка программирования | PHP | Ruby | Python |
| Простота обучаемости | + | - | + |
| Читабельность | + | - | + |
| Динамическая типизация | + | + | + |
| Интеграция баз данных | + | + | - |
| Объектно-ориентированные  возможности | + | + | + |

Вывод, из представленных языков программирования можно выделить РНР, так как именно этот язык более перспективный в разработке web -приложения.

Таблица 3 — Сравнение СУБД для разработки программного продукта.

|  |  |  |  |
| --- | --- | --- | --- |
| Название СУБД | Oracle | MySQL | PostgreSQL |
| Язык программирования | С, С++, Java, Ruby, Objective | Delphi, С, С++, Java, Perl, РНР, Pvthon, Ruby, SQL | pgSQL, РНР, Python, Ruby и др. |
| Скорость обработки | Быстрая | Быстрая | Медленная |
| Операционная система Windows | Поддерживается | Поддерживается | Поддерживается |
| Исходный код | Закрытый | Открытый | Открытый |
| Лицензия | Коммерческая | Свободная | Свободная |

В результате сравнения, из представленных СУБД можно выделить MySQL, так как оно имеет свободную лицензию и подходит для создания web – приложения.

Таблица 4 — Сравнение средств проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| Название средства проектирования | Draw.io | Visio | NClass |
| Поток данных диаграмм | + | + | - |
| Er - диаграмма | + | + | - |
| Проверка связей | + | + | - |
| Удобство использования | + | + | - |
| Лицензия | свободная | коммерческая | свободная |

Следовательно, из представленных средств проектирования, было принято решение использовать Draw.io, так как оно имеет свободную лицензию и большое количество UМL средств.

1. Проектирование

Перед проектированием информационной системы было разработано техническое задание, на основе которого производилось проектирование информационной системы «Поликлиника». Техническое задание приложено к пояснительной записке.

* 1. Архитектура программного обеспечения

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Архитектура включает:

1. выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
2. соединение выбранных элементов структуры и поведения во всё более крупные системы;
3. архитектурный стиль, который направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение.

Информационная система использует клиент-серверную архитектуру, что говорит о том, что все действия на клиентской части пользователя над данными обрабатываются сервером на серверной части.

Клиент-серверная архитектура позволяет значительно снизить нагрузку у пользователя, что позволяет запускать приложение практически на любом устройстве.

На рисунке 1 демонстрируется архитектура веб-приложений, которая соответствует разрабатываемой информационной системе.

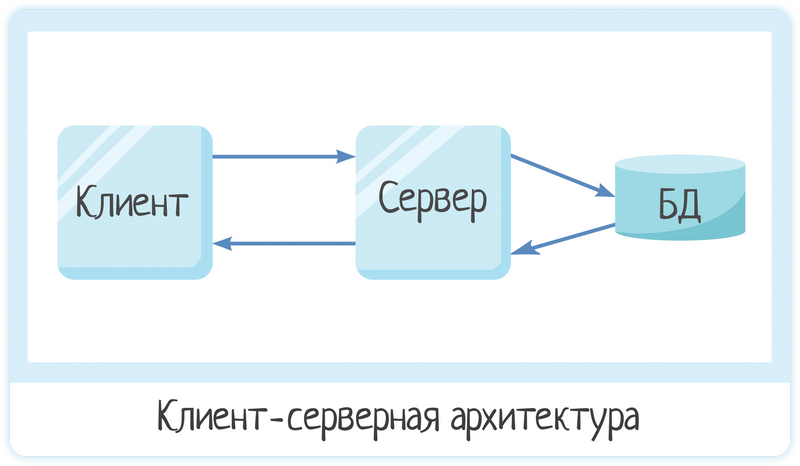


Рисунок 1 – Клиент-серверная архитектура веб-приложения

Браузер клиента отправляет запрос на подключение к странице веб-сервер. Веб-сервер отправляет запрос в веб-приложение, которое запрашивает обработку PHP. PHP может запросить подключение к MySQL, на что Apache отправляет запрос к MySQL, который по запросу берет данные из БД и отправляет обратно на веб-сервер, после чего PHP продолжает обработки. Когда обработка закончит, Apache генерирует необходимую URL и отправляет ответ клиенту, где браузер выполняет скрипты.

Фреймворк Laravel использует архитектурную модель MVC (Model View Controller) представленную на рисунке 1.1.

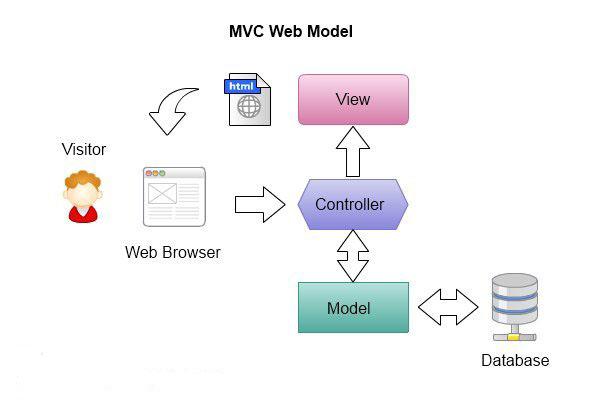


Рисунок 1.1 – Архитектурная модель MVC

* 1. Функциональное проектирование

Функциональное проектирование – это подход к проектированию, который нацелен, прежде всего, на создание эффективно работающего объекта. Функциональное проектирование позволяет раскрыть все функции системы.

Выполнение требуемой функции – главная цель и основа разработки объекта. Во внимание принимаются, прежде всего, функциональные показатели качества и показатели надёжности.

Контекстная диаграмма – это модель, представляющая систему как набор иерархических действий, в которой каждое действие преобразует некоторый объект или набор объектов.

На контекстной диаграмме на рисунке 2 показаны входные данные, управление механизм, выходные данные и функция.



Рисунок 2 – Контекстная диаграмма модуля

Для демонстрации деталей в процессе необходимо спроектировать диаграмму декомпозиции.

Диаграмма декомпозиции – это разбиение функции, т.е. работа информационной системы «Поликлиника».

На рисунке 3 показана диаграмма декомпозиции, которая расписывает работу информационной система из контекстной диаграммы. На этой диаграмме показаны 5 функций — это «Запись на прием врача», «Авторизация», «Прием врача», «Администрирование», «Личный кабинет». А также на этой диаграмме присутствуют входные данные, выходные данные, управления и механизмы.

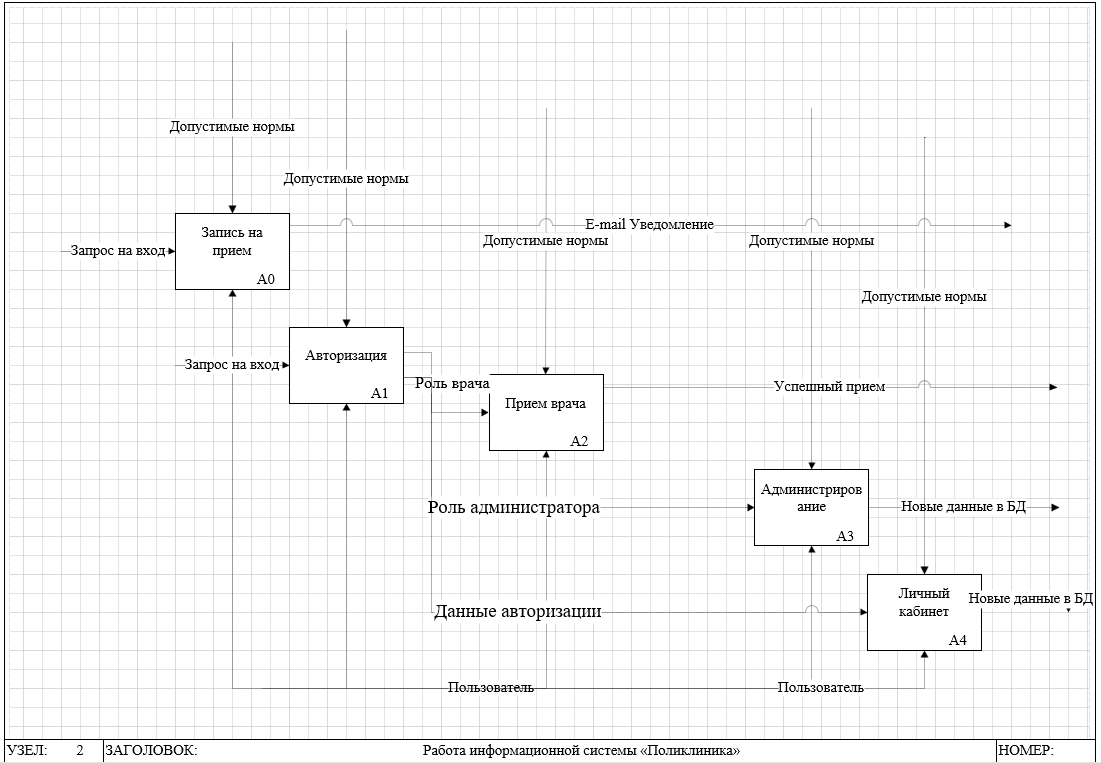


Рисунок 3 – Диаграмма декомпозиций ИС

На рисунке 4 показана диаграмма декомпозиции, которая расписывает работу раздела администрирование их диаграммы декомпозиции первого уровня. На этой диаграмме показаны 4 функций — это «Манипулирование пользователями», «Манипулирование работниками», «Манипулирование данными кабинетов», «Манипулирование данными должностями». А также на этой диаграмме присутствуют входные данные: «Роль администратора», выходные данные: «Данные работников, пользователей, кабинетов и должностей», управления и механизмы: «Пользователи» и «Допустимые нормы» соответственно.

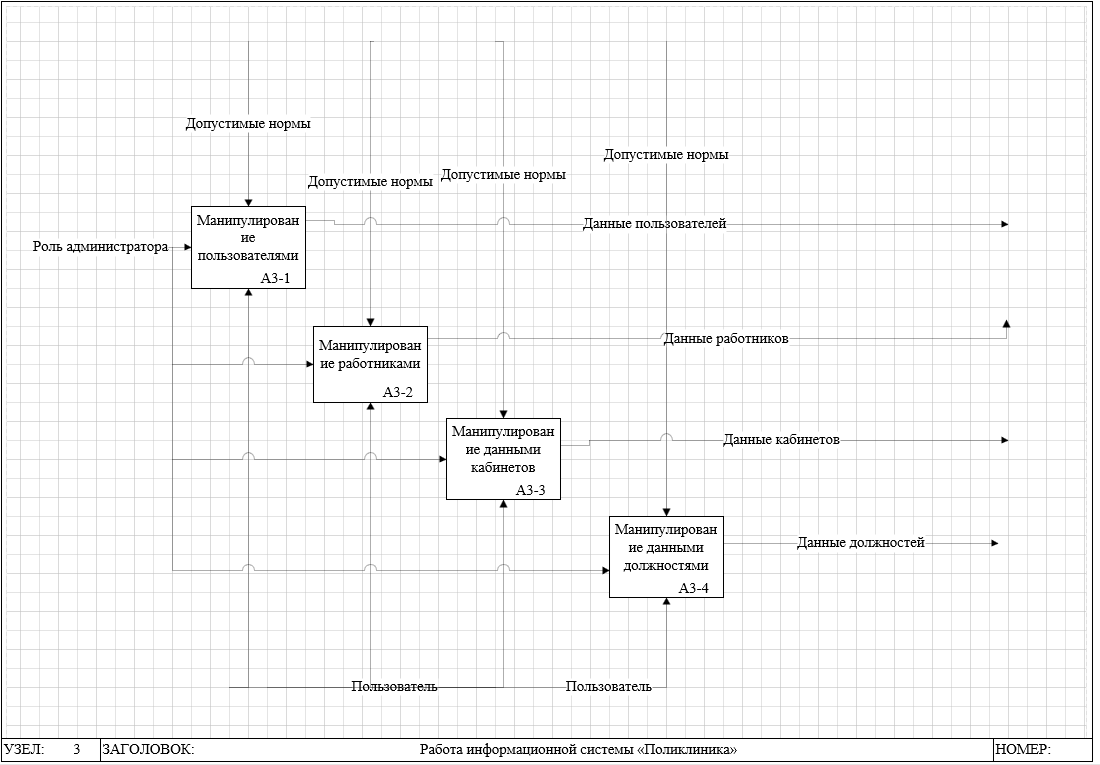


Рисунок 4 – Диаграмма декомпозиций администрирования

На рисунке 5 показана диаграмма IDEF3, которая расписывает работу информационной системы из рисунка 2.



Рисунок 5 – IDEF3 диаграмма

На рисунке 6 показана диаграмма потоков данных, которая описывает передачу данных между разными функция и объектами информационной системы.

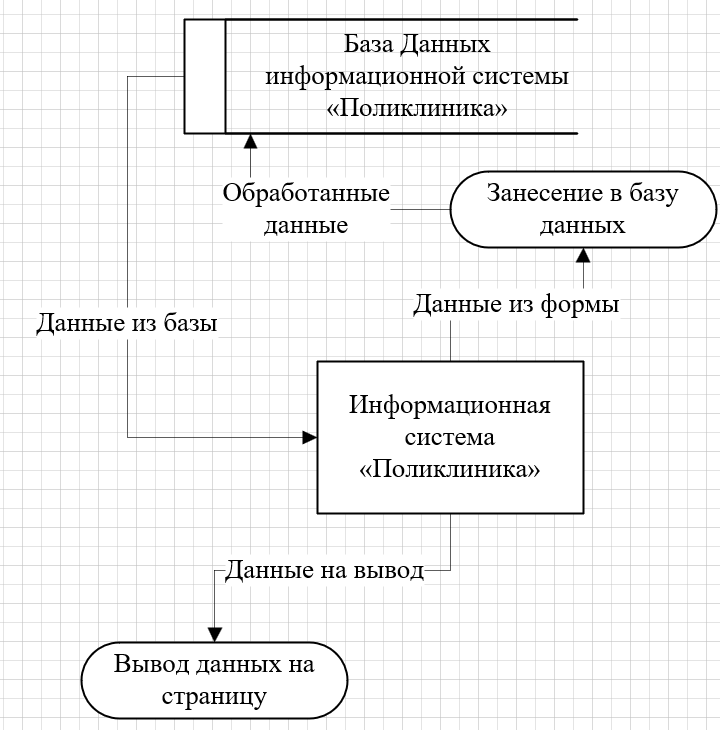


Рисунок 6 – Диаграмма потоков данных

Также на этапе проектирования была разработана диаграмма прецендентов, также называемая диаграмма вариантов использования. На рисунке 7 показана диаграмма прецендентов, в которой описывается взаимодействие различных пользователей с различными функциями информационной системы.

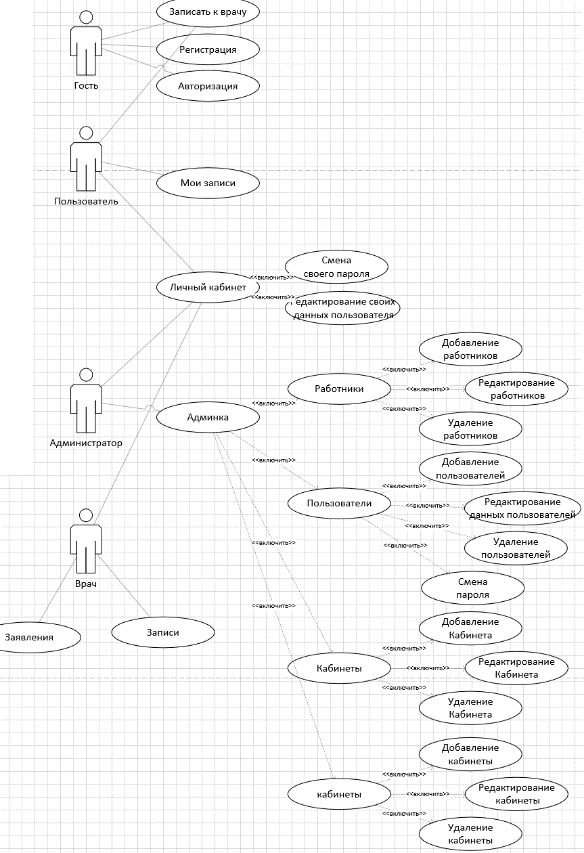


Рисунок 7 – Диаграмма вариантов использования

Подробное функциональное проектирование позволяет глубже исследовать проектную область и цель проекта, что может позволить сэкономить время на разработке информационной системы, а также позволит избежать перенаписание кода.

* 1. Проектирование базы данных

Проектирование информационной системы происходит при помощи CASE средств, которые позволяют быстро создавать схемы и реляционные модели программ.

Для лучшего изучения предметной области и проектирования базы данных. Были построены 3 схемы модели базы данных: концептуальная, логическая и физическая схемы.

Концептуальная модель — это отражение предметной области, для которой разрабатывается база данных. На рисунке 8 представлена концептуальная модель базы данных. Для построения модели были выявлены 8 сущностей: учетные записи, клиенты, записи, заявки, кабинеты, должности, работники, сеансы.

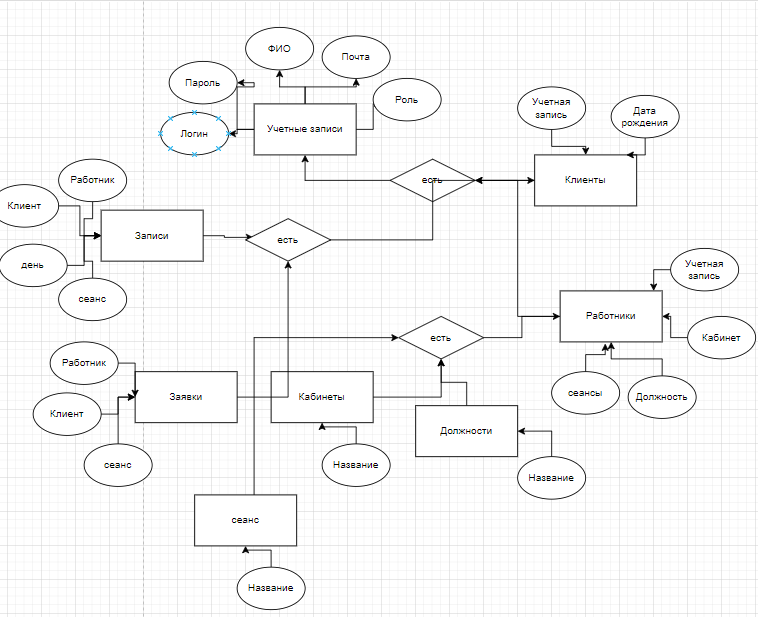


Рисунок 8 – Концептуальная схема базы данных

Модель сущность-связь или схема баз данных – модель данных, позволяющая описывать концептуальные схемы предметной области.

Логическая (даталогическая) модель представляет собой модель базы данных, которая не привязана к конкретной СУБД. В ней выделяют основные объекты БД и определяют связи между этими объектами. Полученная логическая модель представлена на рисунке 9.

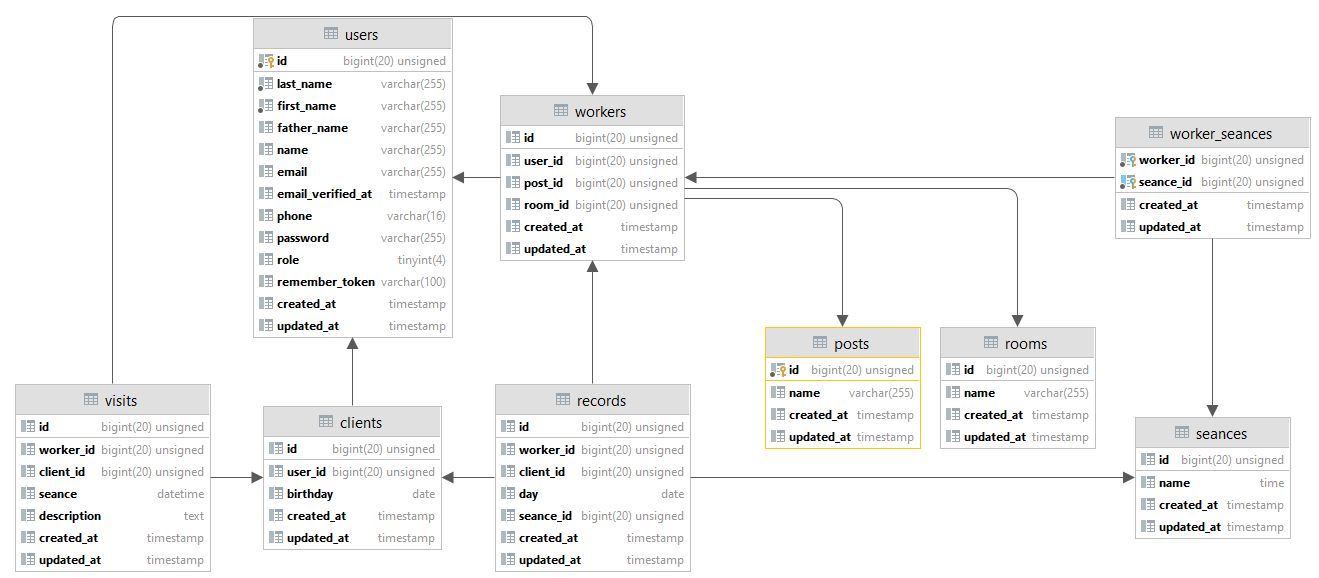


Рисунок 9 – Логическая схема базы данных

Физическая модель базы данных — это модель данных, которая определяет, каким образом представляются данные, и содержит все детали, необходимые СУБД для создания базы данных. На рисунке 10 изображена физическая модель базы данных информационной системы.

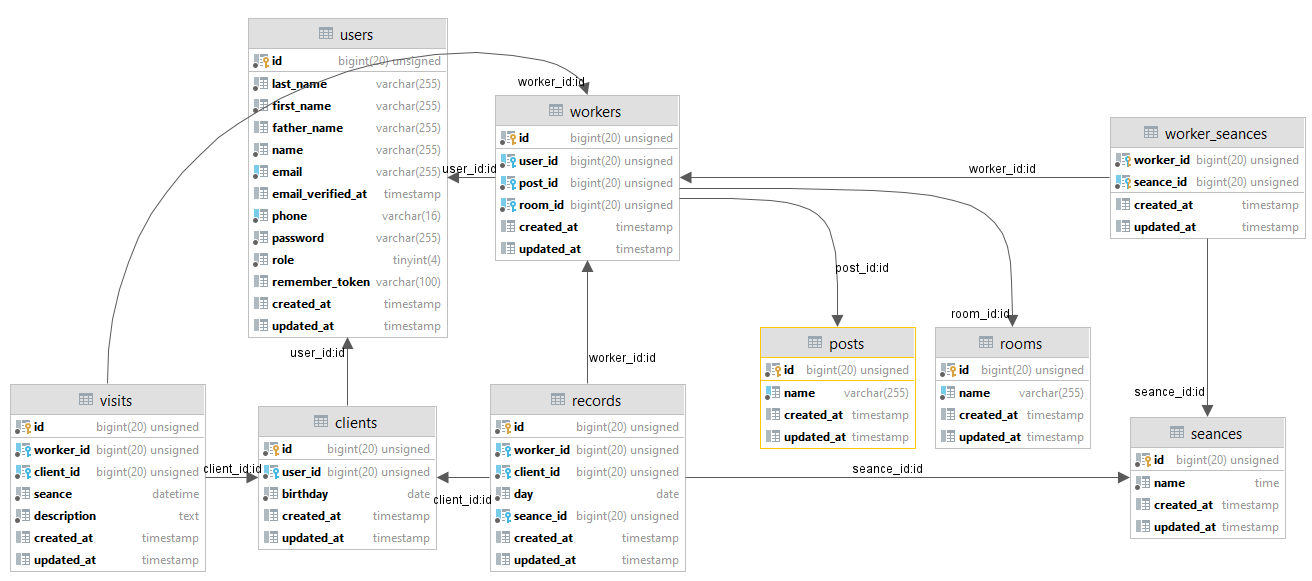


Рисунок 10 – Физическая схема базы данных

База данных состоит из 9 связанных таблицы. Для создания связей между таблицами используются внешние ключи.

Были выявлены следующие сущности и их атрибуты представленные в таблицах от 5 до 13.

Таблица 5 — Атрибуты сущности visits

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор приема |
| worker\_id | bigint (20) unsigned | Идентификатор работника |
| client\_id | bigint (20) unsigned | Идентификатор клиента |
| seance | datetime | Дата и время сеанса |
| description | text | Описание приема |

Таблица 6 — Атрибуты сущности workers

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор работника |
| user\_id | bigint (20) unsigned | Идентификатор учетной записи |
| post\_id | bigint (20) unsigned | Идентификатор должности |
| room\_id | bigint (20) unsigned | Идентификатор кабинета |

Таблица 7 — Атрибуты сущности posts

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор должности |
| name | varchar (255) | Название должности |

Таблица 8 — Атрибуты сущности rooms

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор кабинета |
| name | varchar (255) | Название кабинета |

Таблица 9 — Атрибуты сущности users

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор учетной записи |
| name | varchar (255) | Логин учетной записи |
| password | double | Пароль учетной записи |
| first\_name | varchar (255) | Имя пользователя |
| last\_name | varchar (255) | Фамилия пользователя |
| father\_name | varchar (255) | Отчество пользователя |
| email | varchar (255) | Почта учетной записи |
| phone | varchar (16) | Номер телефона пользователя |
| role | tinyint(4) | Роль учетной записи |

Таблица 10 — Атрибуты сущности clients

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор клиента |
| user\_id | bigint (20) unsigned | Идентификатор учетной записи |
| birthday | date | Дата рождения |

Таблица 11 — Атрибуты сущности records

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор записи |
| worker\_id | bigint (20) unsigned | Идентификатор работника |
| client\_id | bigint (20) unsigned | Идентификатор клиента |
| seance\_id | bigint (20) unsigned | Идентификатор сеанса |
| day | date | День записи |

Таблица 12 — Атрибуты сущности seances

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| id | bigint (20) unsigned | Идентификатор сеанса |
| name | time | Время проведения сеанса |

Таблица 13 — Атрибуты сущности worker\_seances

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание атрибута |
| seance\_id | bigint (20) unsigned | Идентификатор сеанса |
| worker\_id | bigint (20) unsigned | Идентификатор работника |

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Третья нормальная форма, которую придерживается база данных, содержит в себе следующее требование:

– Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

* 1. Проектирование пользовательского интерфейса

Интерфейс информационной системы – это одна из важнейших частей проектирования программного обеспечения.

Интерфейсом информационной системы «Поликлиника» является меню навигации, содержащее разделы информационной системы, панели управления и рабочей области.

На рисунке 11 показан прототип страницы авторизации, который контролирует вход только зарегистрированных пользователей.

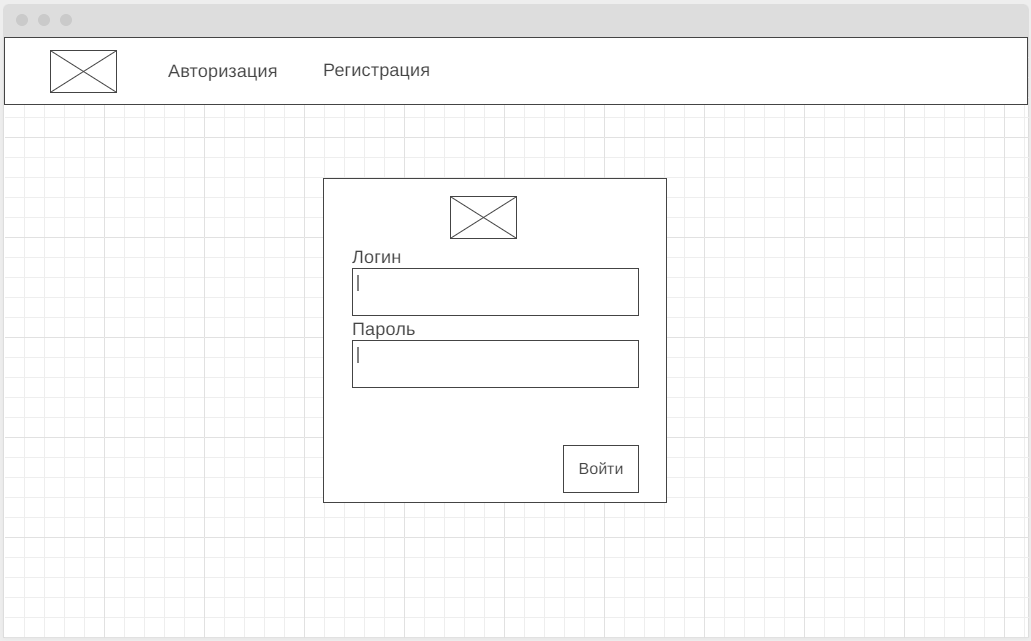


Рисунок 11 – Прототип страницы авторизации

На рисунке 12 демонстрируется прототип страницы регистрация, которая позволяет клиентам поликлиники регистрироваться в информационной системе.

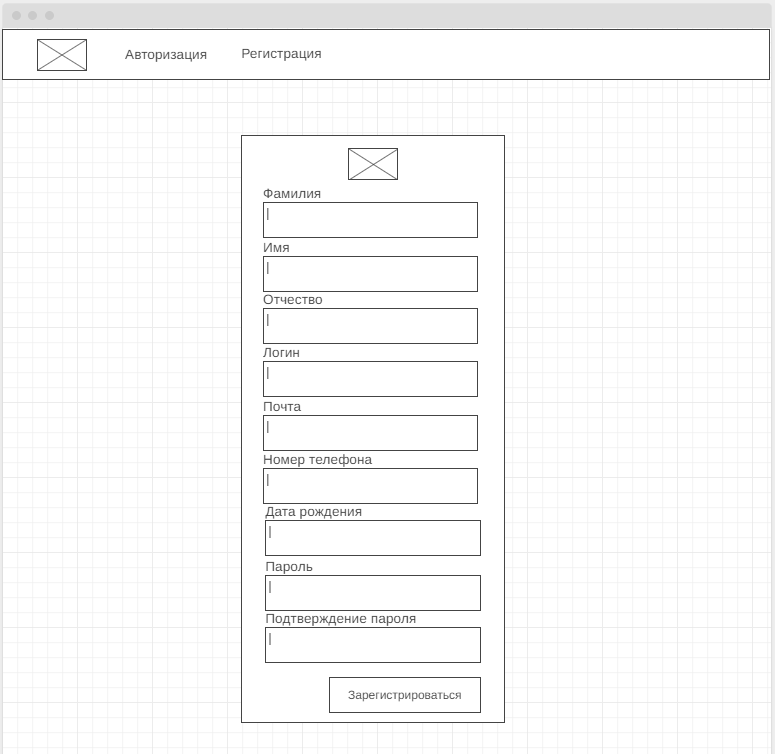


Рисунок 12 – Прототип страницы регистрация

На рисунке 13 демонстрируется прототип страницы записи на прием. На рабочем пространстве размещена форма для заполнения клиентам. Также она является главной страницей для неавторизированного пользователя.

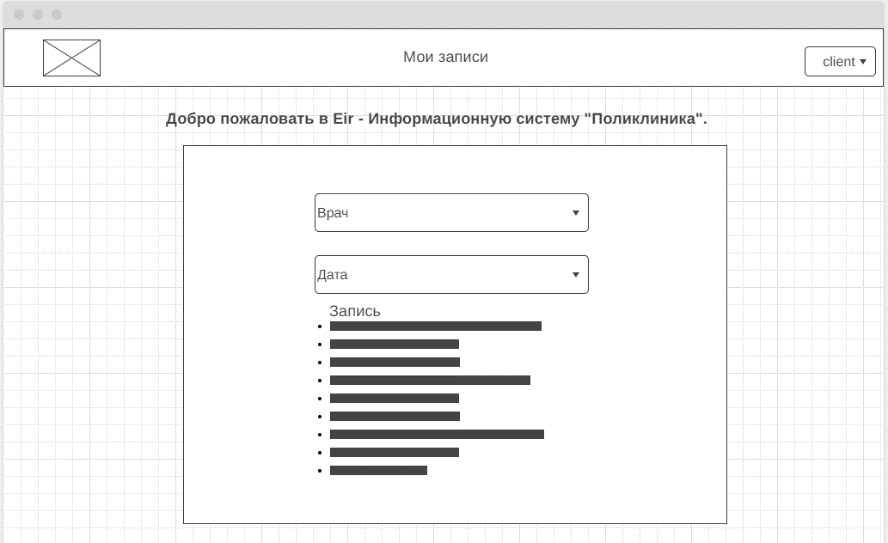


Рисунок 13 – Прототип страницы записи на прием

На рисунке 14 демонстрируется прототип страницы приема врача. На рабочем пространстве размещена форма для заполнения врачом. В ней выбирается клиент или добавляется новый, после чего описывается процесс проведения приема.

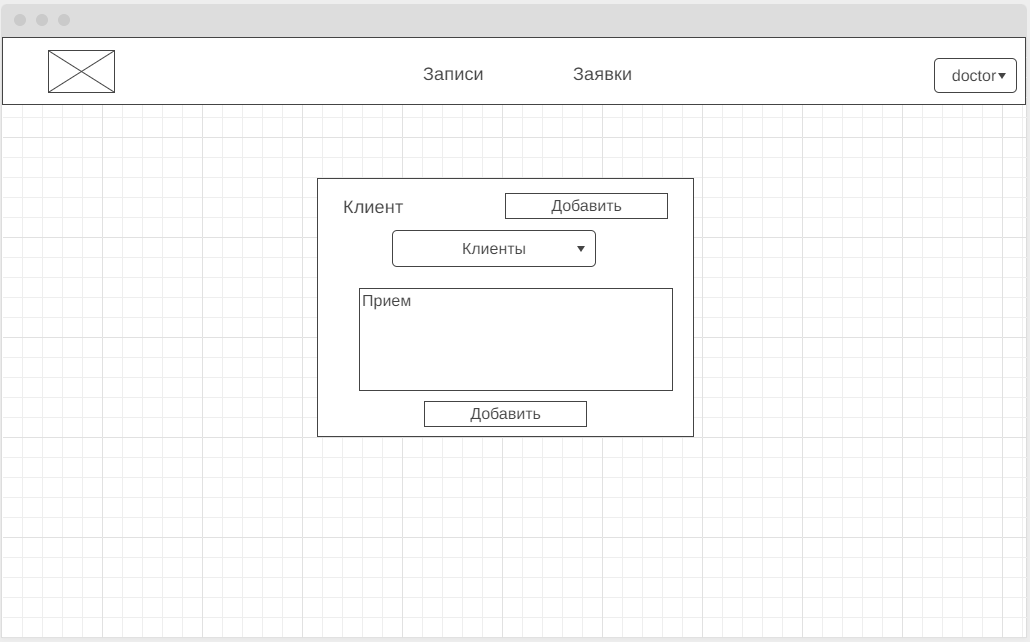


Рисунок 14 – Прототип страницы приема врача

Проектирование интерфейса пользователя позволяет определиться с расположение объектов и данных на рабочем пространстве. Также позволяет определиться с количеством страниц.

1. Реализация программного обеспечения
   1. Кодирование программного обеспечения

Суть реализации программного обеспечения заключается в кодировании – процесс написания программного кода. В связи с тем, что разрабатываемая информационная система основана на веб-технологиях, в ходе разработки было принято решение использовать:

* фреймворк TailwindCSS - для клиентской части;
* скриптовый язык JavaScript – для клиентской части;
* фреймворк Laravel - для серверной части.

Благодаря фреймворку TailwindCSS информационная система «Поликлиника» имеет простой, но стильный внешний вид, который продемонстрирован на рисунке 15

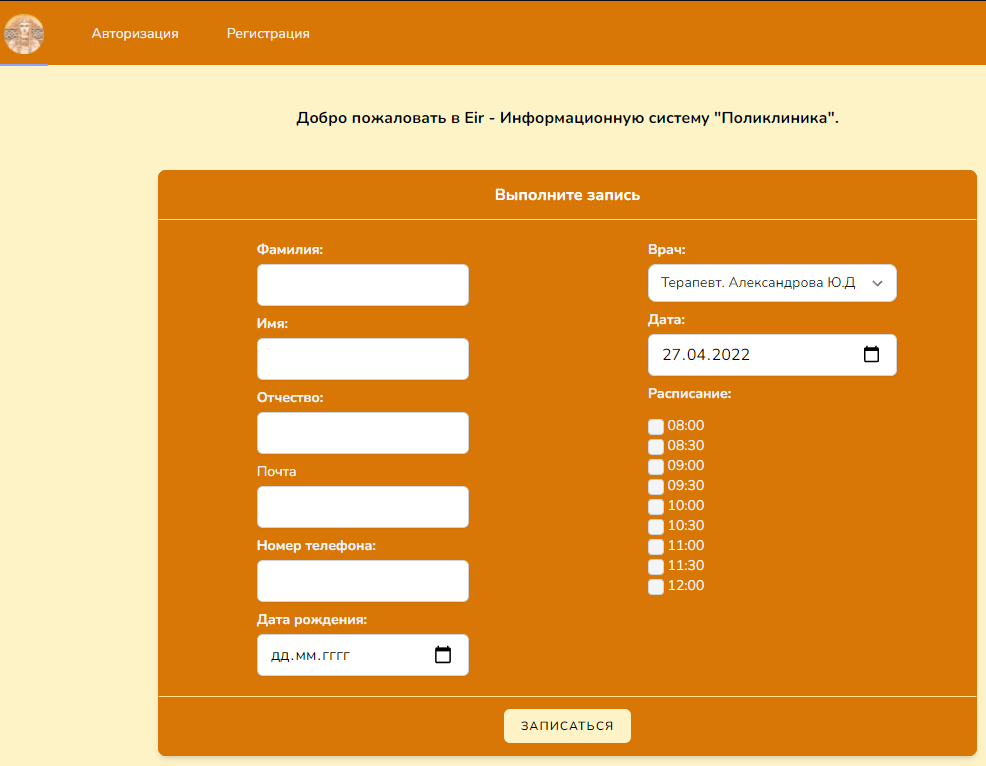


Рисунок 15 – Главная страница не авторизированного пользователя

Скриптовый язык JavaScript позволил организовать динамическое получение данных, что позволило организовать обновление некоторых данных страниц. Фрагмент кода представлен в Приложении Б. Листинг JavaScript.

Фреймворк Laravel, в данной разработке, позволяет осуществить защиту передачи данных и облегчить работу с БД. Принцип работы, которой необходимо создавать файл контроллера, модель и представления.

Контроллеры отвечают за обработку запросов пользователя и вызов соответствующих представлений. Контроллер представляет собой класс, который содержит методы обработки функций. Все контроллеры информационной системы находятся в папке /app/http/Controllers. На рисунке 16 представлен код добавления нового кабинета в базу данных из контроллера RoomController.

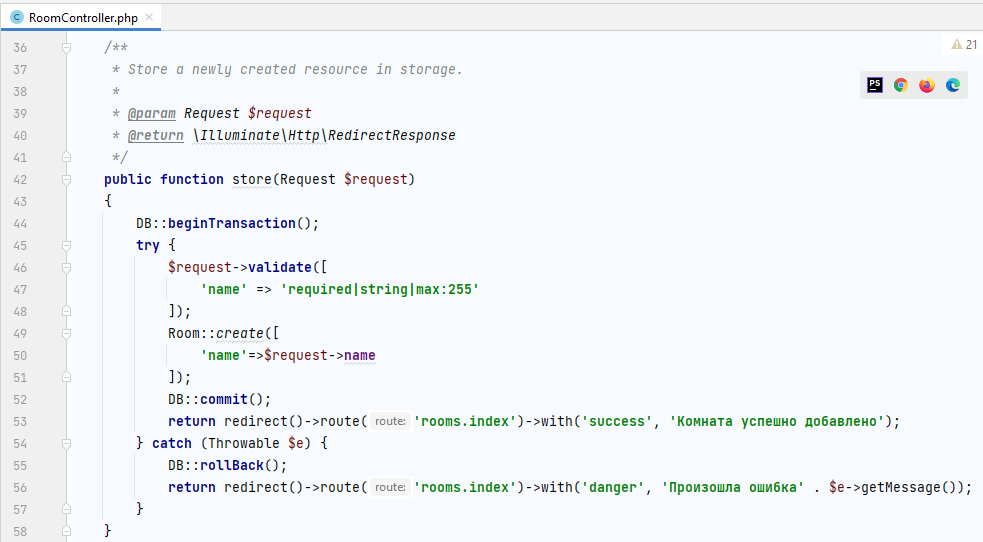


Рисунок 16 – Фрагмент контроллера RoomController

Представления отвечают за показ результатов запроса пользователю. Каждое представление – это отдельная страница, содержащая html и php вставки. Все представления находятся в папке /resource/views. На рисунке 17 изображен код представления изменения приема данных врачом. Представление состоит из панели управления и заполненной формы, в которой врач может изменить свою заключение приема.

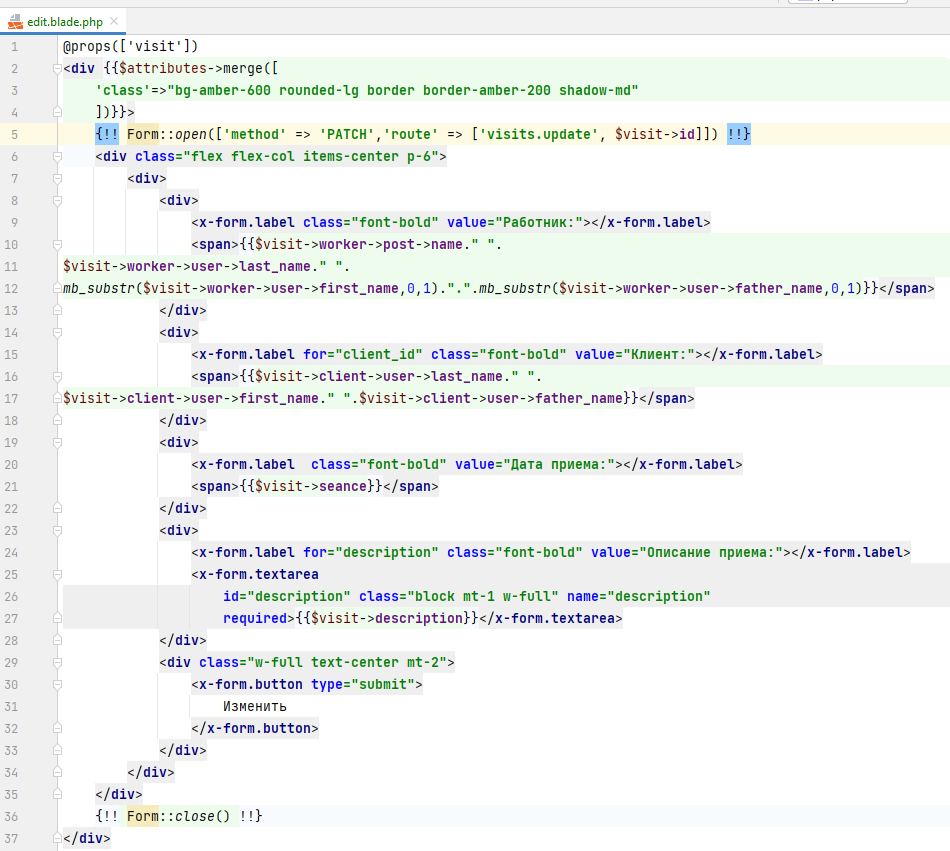


Рисунок 17 – Представление изменения заявления

В работе приложения не малую роль играет система маршрутов, которые связывают файл представления с нужным ему контроллером. Все маршруты проекта располагаются в папке routes. Все маршруты проекта можно просмотреть в Приложении А Листинг маршрутов.

Информационная система содержит несколько крупных функций. Одна из самых крупных функций — это функция записи клиента на прием (фрагмент кода на рисунке 18).

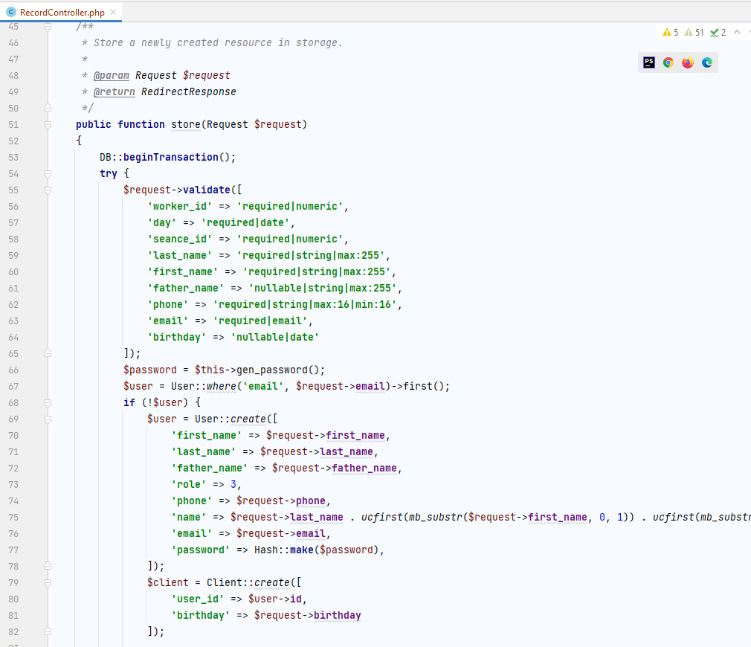


Рисунок 18 – Фрагмент функции записи на прием

Эта функция позволяет зарегистрированному и не авторизованному пользователю записаться на прием к врачу.

На страницах «Кабинеты», «Работники» и «Должности» имеются функции вставки, обновления и удаления данных. Эти функции описаны в соответствующих контроллерах страниц.

Функция вставки данных в базу страницы «Должности» демонстрируется на рисунке 19.

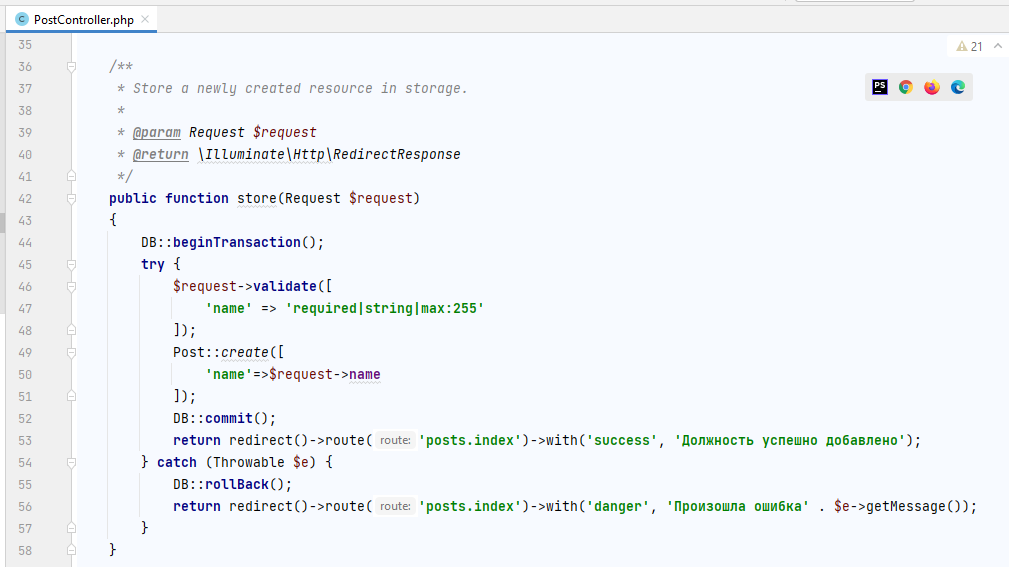


Рисунок 19– Функция store контроллера PostController

Данная функция – это транзакция, которая проверяет корректность заполнения формы страницы и вставляет в базу данных, прошедшие проверку данные, если же в форме были ошибки, то произойдет откат и выведется соответствующее сообщение. Сборка и отправка запросов происходит с помощью конструктора запросов фреймворка Laravel. Он облегчает работу с базой данных и позволяет избежать ошибок в написании запросов.

Функция обновления данных базы страницы «Работники» демонстрируется на рисунке 20.



Рисунок 20 – Фрагмент функции update контроллера WorkerController

Данная функция аналогична функции вставки. Она также проверяет корректность заполнения формы и при совершении ошибки происходит откат данных и выводится соответствующее сообщение. Функция комбинирует запросы insert и update для MySQL сервера.

1. Документирование программного обеспечения
   1. Руководство пользователя программного обеспечения
      1. Описание системы

Информационная система содержит разделы с разным уровнем доступа. Информационной системой могут пользоваться частично не авторизированные и зарегистрированные пользователи на уровне своих ролей. Для первоначального запуска существует стандартная учетная запись, которую в последующем рекомендуется удалить. Работа в разделе зависит от уровня доступа учетной записи авторизированного пользователя. Подробнее смотрите в «Страница входа».

* + 1. Страницы входа и учетные записи

Страница входа позволяет войти в одну из учетных записей информационной системы. Страница представляет собой форму, которая демонстрируется на рисунке 21.

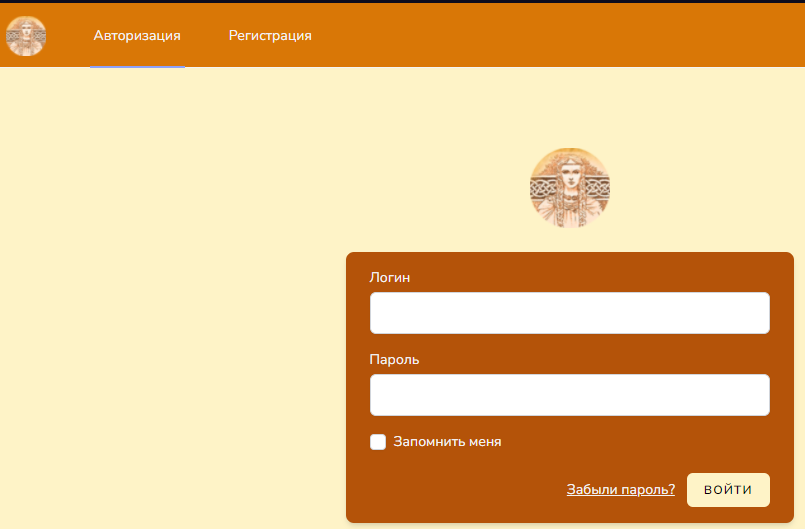


Рисунок 21 – Страница входа

В информационной системе существует система ролей. Существует всего 4 роли: неавторизированный пользователь, клиент, врач и администратор. Изначально в информационной системе есть учетная запись для каждой из ролей. Рекомендуется их удалить при полноценном использовании системы. Администраторская учетная запись admin с паролем admin, имеет право на заполнение данных о поликлинике. Учетная запись врача АлександроваЮД c паролем doctor, позволяет ознакомиться с интерфейсом страниц врачей. Учетная запись СанёкЮО с паролем client, позволяет ознакомиться с интерфейсом страниц клиентов.

* + 1. Раздел «Пользователи»

Раздел «Пользователи» позволяет администратору изменять данные учетных записей информационной системы. На странице 22 представлена главная страница раздела «Пользователи».

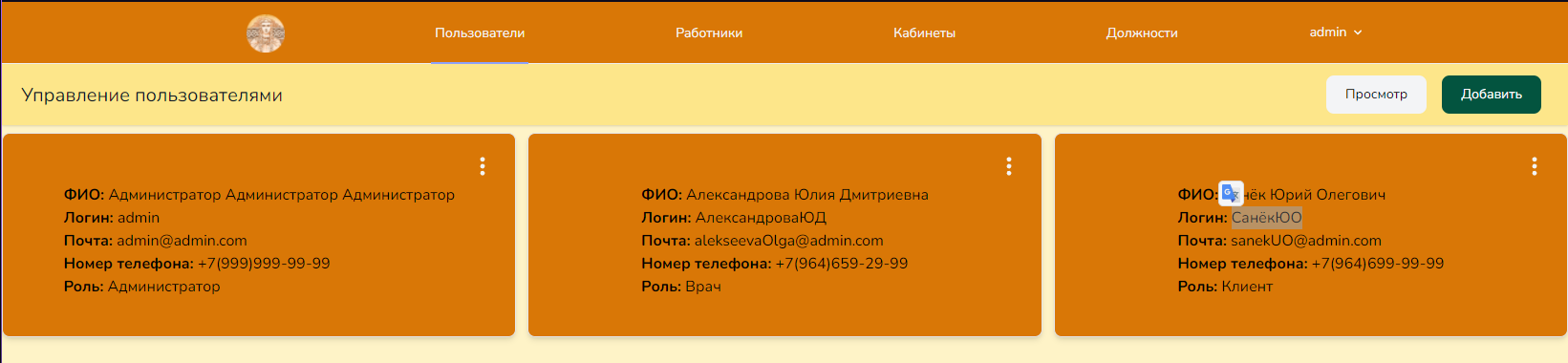


Рисунок 22 – Главная страница раздела «Пользователи»

Это страница позволяет отредактировать, удалить и добавить учетные записи. На странице 23 представлена форма добавления новой учетной записи.

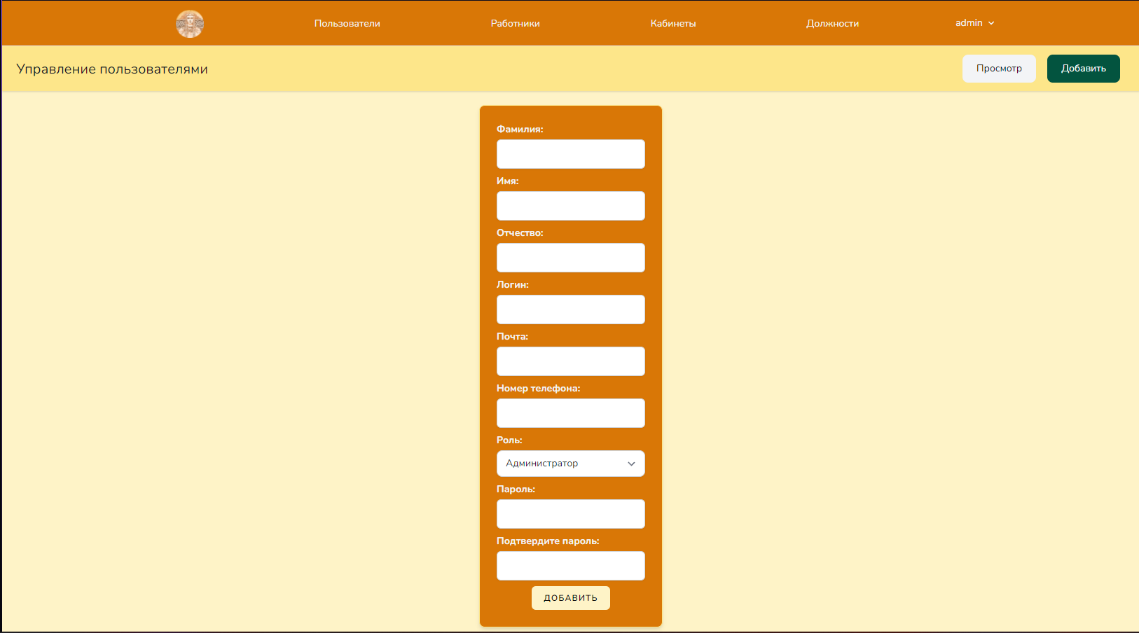


Рисунок 23 – Страница добавления учетной записи

Страница изменения учетной записи похоже на страницу добавления. Форма просто заполнена и её можно отредактировать. Страница представлена на рисунке 24.

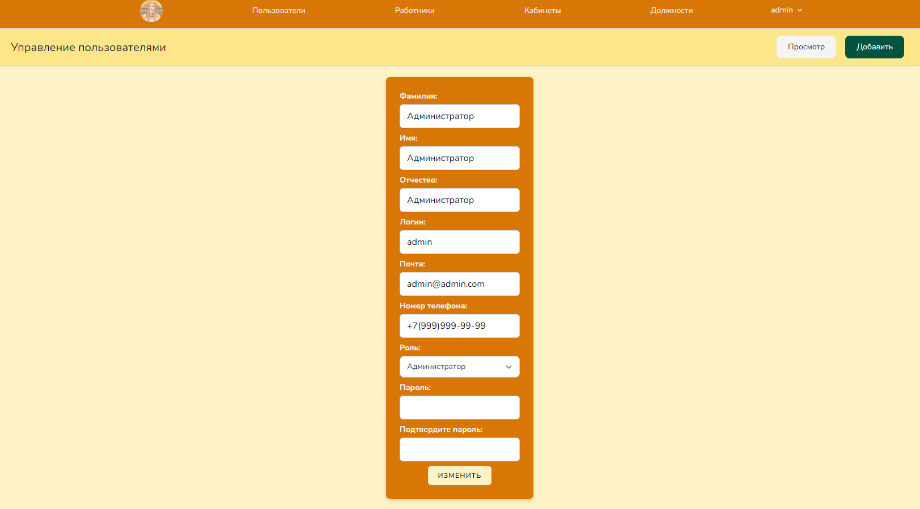


Рисунок 24 – Страница изменения учетной записи

* + 1. Раздел «Работники»

Раздел «Работники» позволяет манипулировать данными работников. В этом разделе имеется возможность изменить личные данные сотрудника, его расписание приема, должности и занимаемый кабинет. Внешний вид продемонстрирован на рисунке 25.

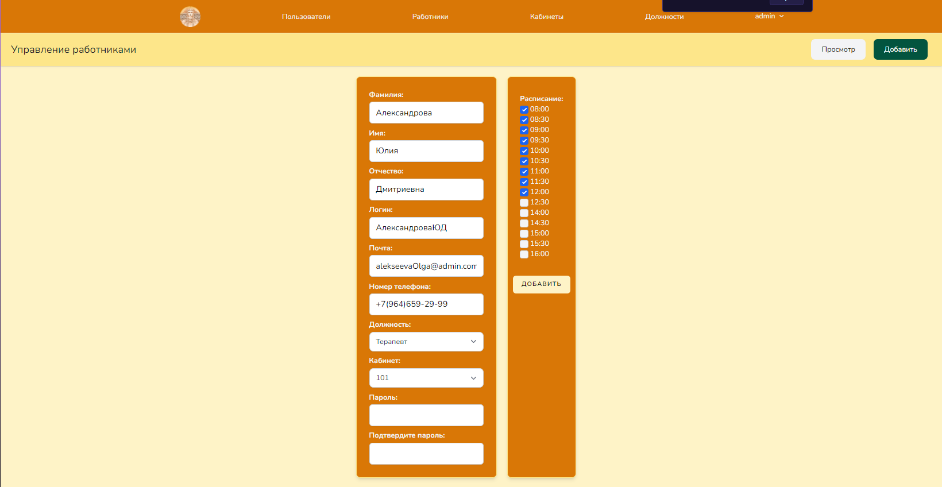


Рисунок 25 – Изменение данных в разделе «Работники»

Раздел позволяет и добавлять новых работников. Форма добавления представлена на рисунке 26.

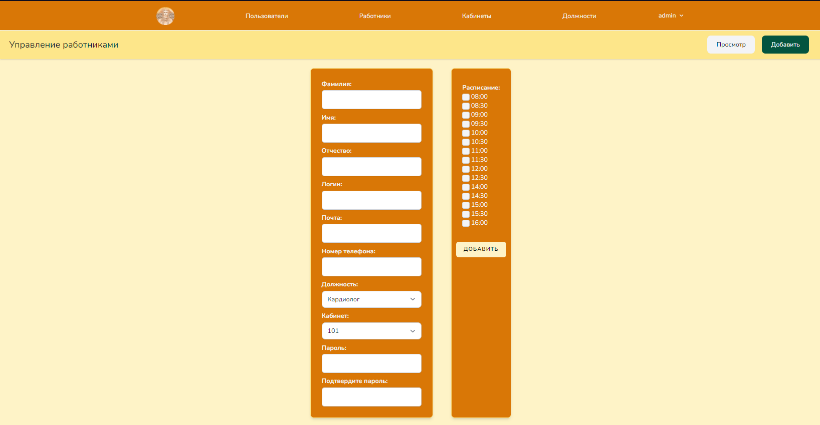


Рисунок 26 – Страница добавления в разделе «Работники»

В информационной системе имеется возможность удалить работника из базы данных, расположение данной функции представлено на рисунке 27.

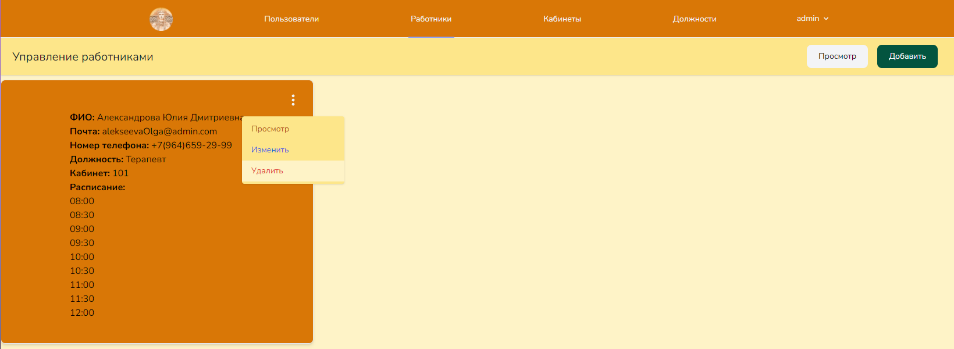


Рисунок 27 – Функция удаления в разделе «Работники»

* + 1. Раздел «Записи»

Раздел «Записи» позволяет принять записанного на прием клиента врачом и увидеть ближайшие записи. На рисунке 28 представлена форма принятия врачом клиента.

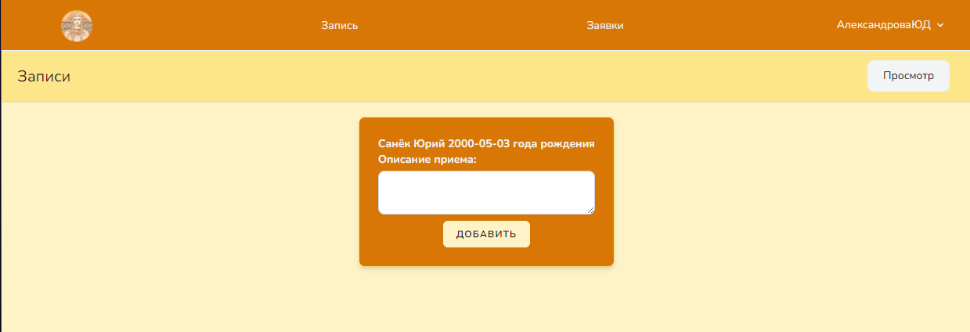


Рисунок 28 – Раздел «Записи»

1. Стоимость разработки и внедрения программного продукта
   1. Организационно-экономическое обоснование проекта

Разрабатываемое приложение будет использоваться для упрощения процесса оказания услуг, накопления клиентской базы предприятия, связанного c автосервисом.

Разработанный программный продукт достаточно распространённый, из-за чего все аналоги находятся в открытом доступе. Стоимость разработки информационной системы составляет от 50 000 рублей, в зависимости от функционала.

* 1. Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку проектного решения (КРПР) осуществляется по формуле 6.1:

(6.1)

КРПР=ЗОТР+ЗЭВМ+ЗСПП+ЗХОН+Е+А,

гдеЗОТР – сумма оплаты труда разработчика/разработчиков ПП;

ЗЭВМ – затраты, связанные с эксплуатацией техники;

ЗСПП – затраты на специальные программные продукты, необходимые для разработки ПП;

ЗХОН – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);

Е – затраты на электроэнергию, руб.;

А – амортизация ПК, руб.

Для подсчета фонда оплаты труда разработчика необходимо определить общее время разработки (таблица 14). Время, затрачиваемое на разработку проектного решения j-м разработчиком, определяется методом экспертных оценок или хронометража. Итоговое значение рассчитывается на основании приведенных исходных данных по формуле 6.2:

(6.2)

где tβ– время β-го этапа разработки проектного решения, дн.

Таблица 14 - Затраты времени на создание программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап создания | | Затраты времени  (в днях) | Затраты времени (в часах) | Машинное время работы над ПП  (в часах) |
| Разработка | Обследование объекта автоматизации | 3 | 24 | 16 |
| Анализ и уточнение требований | 3 | 24 | 8 |
| Разработка технического задания | 4 | 32 | 32 |
| Проектирование структуры | 5 | 40 | 40 |
|  | Программная реализация | 58 | 464 | 450 |
| Тестирование программного продукта | 5 | 40 | 40 |
| Отладка программного продукта | 5 | 40 | 40 |
| Разработка описания | 5 | 40 | 40 |
| ИТОГО | 88 | 704 | 666 |

Рабочий день принимается равным 8 часам.

Сумму оплаты труда разработчика за время работы над программным продуктом рассчитаем исходя из часовой тарифной ставки и фонда фактического времени, затраченного на разработку программного продукта (по формуле 6.3):

(6.3)

Зотр=Ст1\*Фвр,

где Ст1 - часовой тарифной ставки (принимаем равной 20 руб./час);

Фвр – фонд фактического времени, затраченного на разработку программного продукта, час.

Разработка программного продукта проводилась бесплатно, так как связывающий договор не требует оплаты труда.

Затраты, связанные с использованием вычислительной и оргтехники, рассчитываются по формуле 6.4:

,

(6.4)

где ТМРПР – машинное время работы над программным продуктом, час;

kГ– коэффициент готовности ЭВМ,  ;

n – количество единиц техники, равно 1;

СМ-Ч – себестоимость машино-часа, СМ-Ч=6руб.

Затраты на электроэнергию рассчитываются по следующей формуле 6.5:

Е=W×t×T,

(6.5)

где W – мощность, потребляемая ПК, кВт/час;

t – время работы ПК, час;

T – тариф электроэнергии, руб.

Тариф 1,07 рубля за киловатт.

Амортизация ПК рассчитывается по следующей формуле 6.6:

(6.6)

где – первоначальная стоимость ПК, руб;

– процент амортизации в год.

При разработке программного продукта использовались следующие свободно распространяемые специальные продукты: Tailwind, PhpStorm, Laravel.

Затраты на хозяйственно-организационные нужды приводятся в таблице 15 и вычисляются по формуле 6.7:

,

(6.7)

где Цτ – цена τ-го товара, руб.;

Кτ – количество τ-го товара.

Таблица 15 - Затраты на хозяйственно-организационные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол-во (шт.) | Всего (руб.) |
| Диск CD-RWDigitex | 25 | 1 | 25 |
| Бумага | 0,5 | 110 | 55 |
| Чернила для принтера | 700 | 2 | 1400 |
| Итого | | | 1480 |

Результаты выполненных расчетов сводятся в общей таблице (таблица 16).

Таблица 16 – Затраты на разработку

|  |  |  |
| --- | --- | --- |
| Наименование затрат | Условное обозначение | Значение |
| Оплата труда разработчика программного продукта |  | 14080 |
| Затраты, связанные с эксплуатацией техники |  | 3796,2 |
| Затраты на электроэнергию | Е | 623,38 |
| Затраты на амортизацию ПК | А | 1250 |
| Затраты на хозяйственно-операционные нужды |  | 1480 |
| Итого затрат на разработку |  | 21229,58 |

* 1. Расчет затрат на внедрение программного продукта

Информационная система не внедряется в предприятие, поэтому затрат на внедрение нет. Также все используемые программные средства бесплатны и свободно распространяются.

* 1. Основные выводы

Исходя из расчетов стоимости разработки, можно прийти к выводу, что информационная система обойдется предприятию (без учета оплаты труда) в размере 7149,58 рублей. Такие затраты значительно меньше стоимости рассматриваемых аналогов.

# ЗАКЛЮЧЕНИЕ

В ходе дипломного проекта была разработана информационная система «Поликлиника», в которой реализовано хранение информации о клиентах, работниках, выполняемых работах, добавление и редактирование данных.

Информационная система «Поликлиника» позволяет упростить процесс оказания услуг, накапливать клиентскую базы. Это ускорит процесс оказания услуг, что позволит оказывать большее количество услуг, что приведет к росту.

Все поставленные задачи, а именно: навигация по сайту, разграничение уровней доступа, чтение данных, хранение и добавление данных, удаление и изменение данных, запись на приём, регистрация, авторизация, уведомление о регистрации на e-mail, уведомление о записи на приём на e-mail.

Все цели, поставленные для разработки, были достигнуты.

В информационной системе были реализованы следующие функции:

* уведомление на e-mail;
* добавление, редактирование и удаление данных;
* авторизация и регистрация;
* разграничение уровня доступа;
* организация приема врача.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bootstrap [Электронный ресурс] – Режим доступа: http://getbootstrap.ru – Bootstrap Самый популярный в мире front-end фреймворк. (Дата обращения: 01.03.22).
2. CyberForum [Электронный ресурс] – Режим доступа: http://www.cyberforum.ru – (Дата обращения: 03.03.22).
3. BizzApps [Электронный ресурс] – Режим доступа: https://bizzapps.ru/p/draw-io/ – Документация (Дата обращения: 21.04.2022).
4. Hostinger Руководства [Электронный ресурс] – Режим доступа: https://www.hostinger.ru/rukovodstva/web-server-apache/ – Что такое Apache? (Дата обращения: 16.03.22).
5. Laravel [Электронный ресурс] – Режим доступа: https://laravel.ru/docs/v5/queries – Конструктор запросов (Дата обращения: 02.03.22).
6. Wikipedia [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHP – PHP (Дата обращения: 10.03.22).
7. PHP.NET [Электронный ресурс] – Режим доступа: https://www.php.net/ – Документация по PHP (Дата обращения: 18.04.2022).
8. PHPMyAdmin [Электронный ресурс] – Режим доступа: https://php-myadmin.ru/doc/ability.html – Руководство. Возможности PhpMyAdmin (Дата обращения: 02.04.22).
9. Wikipedia [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/PHPStorm – PHPStorm (Дата обращения: 10.03.22).
10. SQL. Полное руководство. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. 2015 – 959 с.
11. Tailwindcss [Электронный ресурс] – Режим доступа: https://tailwindcss.com/docs – Getting started with Tailwind CSS (Дата обращения: 11.04.2022).
12. Life-prog [Электронный ресурс] – Режим доступа: https://life-prog.ru – Документация (Дата обращения: 21.04.2022).
13. Lucidchart [Электронный ресурс] – Режим доступа: https://www.lucidchart.com/pages/ru/visio-что-это – Что такое Visio? (Дата обращения: 18.04.2022).
14. Computerra [Электронный ресурс] – Режим доступа: https://www.computerra.ru/230618/7-besplatnyh-instrumentov-wireframe-dlya-proektirovaniya-interfejsov-i-protsessov/– 7 бесплатных инструментов wireframe (Дата обращения: 11.03.2022).
15. WebForMyself [Электронный ресурс] – Режим доступа: https://webformyself.com/peredacha-znachenij-peremennyx-iz-javascript-v-php-i-naoborot/ – Документация (Дата обращения: 27.04.2022).
16. Habr [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/ruvds/blog/539098/ – 9 самых популярных php-фреймворков (Дата обращения: 28.04.2022).
17. JavaScript [Электронный ресурс] – Режим доступа: https://learn.javascript.ru/manuals-specifications – Справочники и спецификации (Дата обращения: 13.04.2022).

# Приложение А - Листинг маршрутов

Route::get('/', function () {

if (Auth::check()) {

if (Auth::user()->role === 2) {

return redirect()->intended(RouteServiceProvider::HOME\_DOCTOR);}

if (Auth::user()->role === 1) {

return redirect()->intended(RouteServiceProvider::HOME\_ADMIN);}}

$users = User::with('workers.post')->whereHas('workers')->get();

$seances = Seance::with('workers')

->whereHas('workers', function ($q) use ($users) {

return $q->where('id', $users[0]->workers[0]->id);

})->get();

return view('welcome', [

'users' => $users,

'seances' => $seances]);})->name('main');

Route::group(['middleware' => ['auth']], function () {

Route::resource('users', UserController::class);

Route::resource('workers', WorkerController::class);

Route::resource('rooms', RoomController::class);

Route::resource('posts', PostController::class);

Route::resource('lk', LkController::class);

Route::resource('visits', VisitController::class);

Route::get('records/my/{id}', [RecordController::class, 'my'])->name('records.my');

});Route::resource('records', RecordController::class);

Route::post('records/seances', [RecordController::class, 'seances'])-> name ('records.seances');Route::middleware('guest')->group(function () {

Route::get('register', [RegisteredUserController::class, 'create'])->name('register');

Route::post('register', [RegisteredUserController::class, 'store']);

Route::get('login', [AuthenticatedSessionController::class, 'create'])->name('login');

Route::post('login', [AuthenticatedSessionController::class, 'store']);

Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])

->name('password.request');

Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])

->name('password.email');

Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])

->name('password.reset');

Route::post('reset-password', [NewPasswordController::class, 'store'])

->name('password.update');});

Route::middleware('auth')->group(function () {

Route::get('verify-email', [EmailVerificationPromptController::class, '\_\_invoke'])

->name('verification.notice');

Route::get('verify-email/{id}/{hash}', [VerifyEmailController::class, '\_\_invoke'])

->middleware(['signed', 'throttle:6,1'])->name('verification.verify');

Route::post('email/verification-notification', [EmailVerificationNotificationController::class, 'store'])->middleware('throttle:6,1')

->name('verification.send');

Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])

->name('password.confirm');

Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);

Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])

->name('logout');});

# Приложение Б - Листинг JavaScript

document.addEventListener('DOMContentLoaded', function () {

const worker = document.querySelector('#worker')

worker.addEventListener('change', function (e) {

fetch('{{route('records.seances')}}', {

headers: {

"Content-Type": "application/json",

"Accept": "application/json",

"X-Requested-With": "XMLHttpRequest",

"X-CSRF-Token": '{{csrf\_token()}}'},

method: 'POST',

credentials: "same-origin",

body: JSON.stringify({

worker\_id: e.target.value,})

}).then(res => res.json()).then(res => {reload(res.seances)})})})

function reload(arr) { const seances = document.querySelector('.seances')

const checks = seances.querySelectorAll('div')

checks.forEach(function (el) {el.remove()})

arr.forEach(function (seance) {

const checkClone = document.querySelector('.seanceCheck').cloneNode(true)

checkClone.classList.remove('hidden')

const input = checkClone.querySelector('input[type="radio"]')

input.value = seance.id

checkClone.querySelector('span').innerHTML = seance.name.slice(0, 5)

seances.append(checkClone)})}