Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Магнитогорский государственный технический университет   
им Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт

**КУРСОВОЙ ПРОЕКТ**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

по ПМ.09. Проектирование, разработка оптимизация веб-приложений

МДК.09.01. Проектирование и разработка веб-приложений

на тему: Разработка веб-приложений ”Умный дом”

Исполнитель: Аверин А.А. студент 4 курса, группа ИспВ-20-1

Руководитель: Зорина И.Г., преподаватель МпК

Работа допущена к защите «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Магнитогорск, 2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет   
им. Г. И. Носова»

Многопрофильный колледж

**КУРСОВОЙ ПРОЕКТ**

Тема Разработка автоматизированной информационной системы сайта знакомств

Студент Аверин Артём Александрович

Задание

Исходные данные:

по заданной предметной области:

1. выполнить моделирование информационной системы;
2. спроектировать и разработать базу данных;
3. разработать серверную и клиентскую части приложения;
4. разграничить права доступа;
5. выполнить тестирование информационной системы.

Состав и содержание проекта:

1. построение UML-диаграмм: вариантов использования, последовательности, состояния, деятельности, классов; потоков данных;
2. логическое проектирование, описание концептуальной модели данных, разработка базы данных в выбранной СУБД;
3. разработка интерфейса приложения, логики работы приложения: оформление заявок на услуги, вывод отчетной документации и т.д.;
4. администрирование приложения, разработка панели администратора;
5. разработка комплекта тестирования информационной системы.

Срок сдачи: « » 20 г.

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /И.Г. Зорина

« » 2023 г.

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / А.А.Аверин

« » 2023г.

Магнитогорск, 2023

Содержание

[ВВЕДЕНИЕ 5](#_Toc154575093)

[1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc154575094)

[1.1 Анализ предметной области 7](#_Toc154575095)

[1.2 Анализ и оценка существующих программных продуктов 7](#_Toc154575096)

[1.3 Моделирование программного продукта 8](#_Toc154575097)

[1.4 Логическое и физическое проектирование базы данных 12](#_Toc154575098)

[1.5 Выбор и обоснование средств разработки 14](#_Toc154575099)

[2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc154575100)

[2.1 Создание базы данных 16](#_Toc154575101)

[2.2 Логическая и физическая структура сайта 18](#_Toc154575102)

[2.3 Разработка интерфейса пользователя 19](#_Toc154575103)

[2.4 Разграничение прав доступа 22](#_Toc154575104)

[2.5 Руководство пользователя 23](#_Toc154575105)

[2.6 Разработка запросов 25](#_Toc154575106)

[2.7 Тестирование информационной системы 30](#_Toc154575107)

[ЗАКЛЮЧЕНИЕ 32](#_Toc154575108)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧИКОВ 33](#_Toc154575109)

[ПРИЛОЖЕНИЕ А 35](#_Toc154575110)

# ВВЕДЕНИЕ

В условиях развития современного общества информационные технологии глубоко проникают в жизнь людей. Всё больше процессов автоматизируются, облегчаются. уменьшения степени этого участия или трудоёмкости выполняемых операций.

Сегодняшний мир находится на пороге четвертой промышленной революции, которая будет характеризоваться внедрением и интеграцией технологий искусственного интеллекта, Интернета вещей и киберфизических систем в повседневную жизнь. Одной из главных сфер применения этих технологий является создание умных домов, способных автоматизировать повседневные процессы, экономить ресурсы и обеспечивать комфорт и безопасность.

В современном мире технологии умного дома становятся все более популярными и востребованными. Они предоставляют возможность автоматизации и оптимизации повседневных бытовых процессов, экономии времени и ресурсов, а также обеспечения комфорта и безопасности жилища. Веб-сайт умный дом является одним из ключевых элементов системы умного дома, предоставляя пользователю доступ к управлению устройствами и функциями умного дома из любой точки мира, где есть интернет-соединение.  
Настоящий проект направлен на разработку веб-сайта умного дома, который будет предоставлять пользователям широкий спектр возможностей для управления устройствами, контроля над домом и его безопасностью, а также для мониторинга и анализа данных о потреблении ресурсов.

Умный дом - это система автоматизации бытовых устройств, таких как освещение, отопление, вентиляция, кондиционирование, безопасность, мультимедиа и другие, которые работают в соответствии с заданными пользователем параметрами. Веб-сайт “Умный дом”, в свою очередь, является инструментом для управления этой системой. Он предоставляет пользователю возможность контролировать работу устройств, отслеживать их состояние, устанавливать различные сценарии и автоматизировать процессы.

Цель данного проекта - разработка веб-сайта, который будет являться удобным, функциональным и безопасным инструментом для управления системой умного дома. Для достижения этой цели необходимо решить ряд задач, связанных с разработкой дизайна, структуры, функционала и безопасности сайта, а также интеграции с различными устройствами и платформами.

Исходя из поставленной цели можно выделить следующие задачи:

* выполнить анализ предметной области;
* провести анализ и оценку существующих программных продуктов;
* выполнить постановку задачи;
* выполнить моделирование программного продукта;
* выбрать средства разработки;
* разработать базу данных;
* разработать физическую и логическую структуру сайта;
* выполнить проектирование меню и пользовательского интерфейса;
* разработать клиентскую часть приложения;
* сделать описание запросов при разработке приложения;
* спроектировать отчеты;
* выполнить администрирование программного продукта;
* разработать руководство пользователя;
* разработать руководство программисту;
* учесть меры безопасности;
* выполнить управление проектом.

# 1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **1.1 Анализ предметной области**

История умного дома началась еще в XIX столетии при создании первых систем жизнеобеспечения. Все существующие на сегодняшний день технологии основываются на разработках, которые были созданы на рубеже XX столетия. Эти технологии медленно видоизменялись по мере продвижения вперед технического прогресса. Оснащение дома необходимым техническим оборудованием привело к современному виду жилого дома: кухня со встроенным оборудованием, включая газовую или электрическую плиту и холодильник, ванная и туалет с современной сантехникой и т.д. Дальнейшее усовершенствование домашнего оборудования привело к внедрению в обиход телефона, радио и телевидения. Этот процесс протекал одновременно с созданием проектов будущего жилья с принципиальной переменой представления о нем .

Термин «умный дом» возник от английского intelligent building. Понимая под этим рациональное использование агрегата «дом», не затрагивая понятие «машина».

При этом акцентирование производится на интерактивность. Эта интерактивная система состоит в наши дни из нескольких различных сетей: электросеть, телефонная и телевизионная, интернет сети. Уже сейчас производится объединение сетей, создавая оптимальный интерфейс между сетью и конечным устройством и, главное, интерфейс между человеком и машиной в виде графического пользовательского интерфейса, сенсорного экрана или идентификации голоса или жеста.

## **1.2 Анализ и оценка существующих программных продуктов**

Для разработки веб-сайта “Умный дом” необходимо провести анализ и оценку существующих программных продуктов в данной области. Это позволит выявить основные тенденции, определить преимущества и недостатки существующих решений, а также учесть полученные результаты при разработке собственного продукта.

На сегодняшний день на рынке представлено множество программных продуктов для управления системами умного дома. Некоторые из них являются универсальными и могут работать с различными устройствами, другие же специализированы на определенных брендах или технологиях.

Одним из наиболее популярных универсальных решений является платформа Amazon Alexa. Она позволяет управлять устройствами различных производителей через голосовые команды и интегрируется с множеством сервисов, таких как Spotify, Netflix, YouTube и другими. Однако, недостатком этой платформы является то, что она работает только на устройствах Amazon и не имеет веб-интерфейса.

Другим популярным решением является система Google Home, которая также поддерживает управление устройствами различных производителей. Она имеет удобный веб-интерфейс и мобильное приложение, однако, как и Amazon Alexa, работает только с устройствами Google.

Также стоит отметить такие решения, как Apple HomeKit, Xiaomi Mi Home, Samsung SmartThings и другие. Все они имеют свои преимущества и недостатки, но в целом, их функционал примерно одинаков.

## **1.3 Моделирование программного продукта**

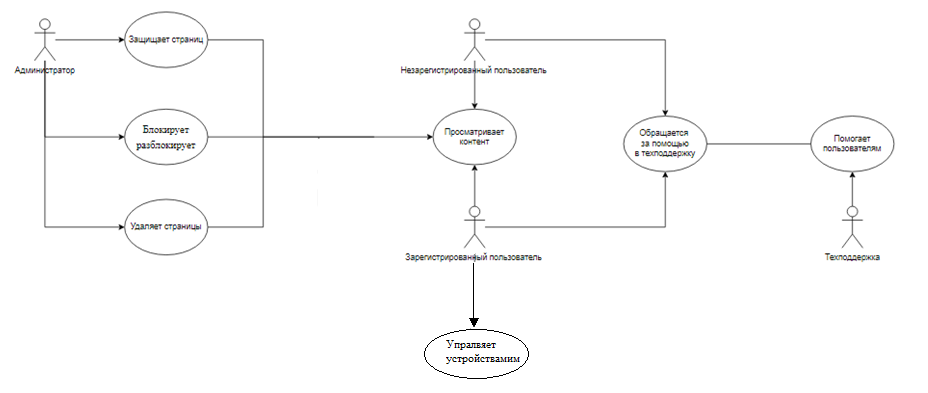
Моделирование — это представление объекта моделью для получения информации о нём путём проведения экспериментов с его моделью.

Под термином моделирование обычно понимают процесс создания точного описания системы, метод познания, состоящий в создании и исследовании моделей.

Моделирование облегчает изучение объекта с целью его создания, дальнейшего преобразования и развития. Оно используется для исследования существующей системы, когда реальный эксперимент проводить нецелесообразно из-за значительных финансовых и трудовых затрат, а также при необходимости проведения анализа проектируемой системы, то есть которая ещё физически не существует в данной организации.

Диаграмма прецедентов предоставляет визуальное представление функциональности системы и взаимодействия с актерами (рисунок 1). Она помогает команде разработчиков и заинтересованным сторонам лучше понять требования и задачи, которые должны быть реализованы в системе. Она также может служить основой для определения тестовых сценариев и функционального тестирования системы.

Важно отметить, что диаграмма прецедентов не описывает внутреннюю структуру системы или детали реализации. Она фокусируется исключительно на функциональных возможностях и взаимодействии с актерами.

Рисунок 1 – Диаграмма прецедентов

IDEF1 (рисунок 2) — это методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи. Одна из основных ценностей и причин стремительного развития информационных технологий — это высочайший темп наращивания человечеством информации. Хранимая и обрабатываемая, она даёт возможность прогресса, и потребность в том, чтобы наращивать возможности по обработке.

Стандарт IDEF1 был разработан как инструмент для анализа и изучения взаимосвязей между информационными потоками в рамках коммерческой деятельности предприятия. Целью подобного исследования является дополнение и структуризация существующей информации и обеспечение качественного менеджмента информационными потоками.

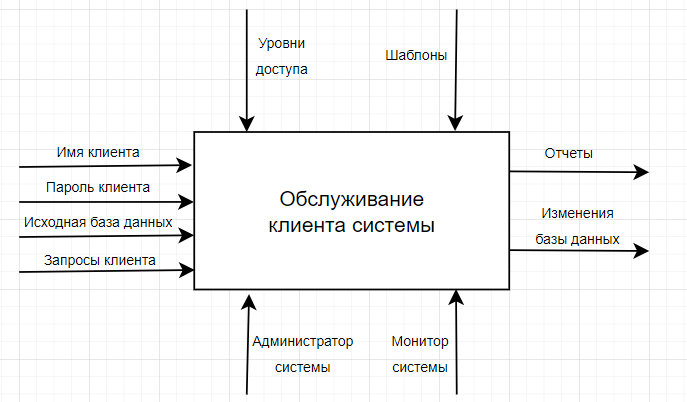


Рисунок 2 – Диаграмма IDEF1

Диаграмма последовательности (рисунок 3) — диаграмма взаимодействия, в которой основной акцент сделан на упорядочении сообщений во времени. Диаграмма последовательности — это способ описания поведения системы "на примерах". Диаграмма последовательности — диаграмма, предназначенная для представления взаимодействия между элементами модели программной системы в терминологии линий жизни и сообщений между ними.

Графически диаграмма последовательности имеет два измерения. Одно — слева направо в виде вертикальных линий, каждая из которых соответствует линии жизни отдельного участника взаимодействия. Второе — вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. Реализация взаимодействия моделируется посредством сообщений, которые передаются между различными линиями жизни. Сообщения изображаются в виде стрелок различной формы и образуют некоторый порядок относительно времени своей передачи.

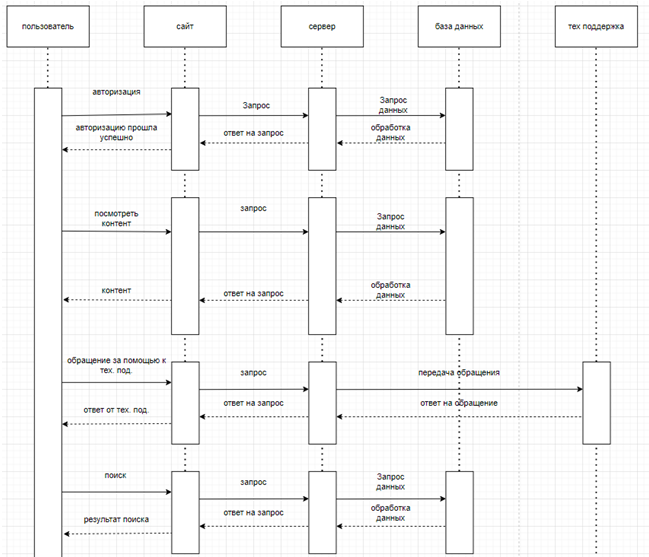
Рисунок 3 – Диаграмма последовательности

Диаграмма деятельности (рисунок 4) используется для моделирования процесса выполнения операций в языке UML. Графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления деятельности и действий, а также в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некой операции, а переход в следующее состояние происходит только после завершения выполнения этой операции. Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или деятельности, а дугами – переходы от одного состояния действия к другому.

Диаграммы деятельности – частный случай диаграмм состояний. Они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции определенного класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

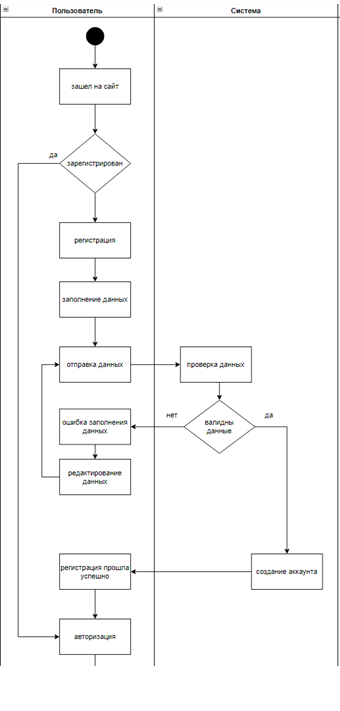


Рисунок 4 – Диаграмма деятельности

## **1.4 Логическое и физическое проектирование базы данных**

База данных — это программа, которая позволяет хранить и обрабатывать информацию в структурированном виде. База данных — это отдельная независимая программа, которая не входит в состав языка программирования. В базе данных можно сохранять любую информацию, чтобы позже получать к ней доступ.

Проектирование базы данных заключается в многоступенчатом описании будущей базы данных с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация ее структуры.

Логическое проектирование — это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой системы управления базами данных и других физических аспектов реализации.

Второй этап проектирования базы данных называется логическим проектированием базы данных. Его цель состоит в создании логической модели данных для исследуемой части предприятия. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в целевой системе управления базами данных (например, реляционная модель).

Если концептуальная модель данных не зависит от любых физических аспектов реализации, то логическая модель данных создается на основе выбранной модели организации данных целевой системы управления базами данных. Иначе говоря, на этом этапе уже должно быть известно, какая система управления базами данных будет использоваться в качестве целевой — реляционная, сетевая, иерархическая или объектно-ориентированная. Однако на этом этапе игнорируются все остальные характеристики выбранной системы управления базами данных, например, любые особенности физической организации ее структур хранения данных и построения индексов.

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей.

Созданная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. Логическая модель данных играет также важную роль на этапе эксплуатации и сопровождения уже готовой системы. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе.

Физическое проектирование базы данных — это процесс создания описания конкретной реализации базы данных, размещаемой во вторичной памяти. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

На основе ситуации описания проекта проектируется модель базы данных и составляется схема логической модели. Концептуальная модель

## **1.5 Выбор и обоснование средств разработки**

В процессе создания веб–приложения были выбранные различные средства для разработки.

Выбранные средства для разработки:

* MySQL;
* Laravel.

Laravel является одним из самых популярных в мире PHP–фреймворков, предназначенный для разработки веб–приложений. Имя при себе ряд плюсов, стоит выделить некоторые их низ:

* Laravel – полнофункциональный фреймворк. Он содержит все программные подсистемы необходимые для разработки веб–приложения. После его установки нет необходимости доустанавливать ещё какие–либо программы или приложения;
* у Laravel низкий порог вхождения. Для программирования веб–приложений хватает базовых знаний PHP и основ веб–разработки. Он не требует сложного конфигурирования и готов к работе сразу после установки. Отдельные модули, составляющие код, не требуется явно связывать друг с другом, достаточно «разложить» их по нужным папкам и веб–приложение будет прекрасно работать;
* Laravel содержит множество дополнительных библиотек, расширяющих функциональность фреймворка, и программ, помогающих в работе.

Таким образом, фреймворк Laravel представляет из себя качественный продукт, которые облегчит разработку веб–приложения, поможет сжать код и обеспечить разработку всего требуемого функционала для веб–приложения.

MySQL – это свободно распространяемая система управления базами данных. MySQL имеет клиент–серверную архитектуру: к серверу MySQL могут обращаться различные клиентские приложения, в том числе с удаленных компьютеров.

MySQL имеет ряд плюсов, за счет которых она была выбрана для разработки:

* MySQL представляет из себя открытый код. Любой желающий может бесплатно скачать программу и при необходимости доработать ее;
* MySQL – кроссплатформенная система. Ее можно использовать практически во всех современных операционных системах;
* MySQL имеет множество программных интерфейсов, благодаря которым к базе данных MySQL могут подключаться приложения, созданные с помощью различных языков, таких как: C/C++, PHP Python и так далее;
* MySQL имеет отличные технические характеристики: многопоточность, многопользовательский доступ, быстродействие, масштабируемость
* MySQL имеет развитую систему обеспечения безопасности и разграничения доступа на основе системы привилегий

MySQL представляет из себя прекрасное средство для разработки веб–приложения, которая содержит сбалансированный набор функций, возможностей и преимущества. MySQL идеально подходит для быстродействия при взаимодействии данных, он понятен и прост в управлении системе управления базами данных.

Bootstrap – это бесплатный CSS–фреймворк с открытым исходным кодом, предназначенный для быстрой верстки адаптивных интерфейсов и веб–приложений с ориентацией на мобильные устройства.

На данный момент Bootstrap является одним из самых распространённых фреймворков для проектирования веб–приложений. Он пользуется популярностью не только у разработчиков сайтов, но и у заказчиков.

# РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

**2.1** **Создание базы данных**

Создание базы данных - это процесс разработки и организации структурированного хранилища данных, которое позволяет эффективно хранить, управлять и извлекать информацию. Это включает в себя определение схемы базы данных, создание таблиц и связей между ними, а также установку правил целостности данных.

Создание таблицы «Пользователь» Характеристики таблицы представлены в таблице1

public function up(): void

    {

        Schema::create('users', function (Blueprint $table) {

            $table->id();

            $table->string('firstname',255);

            $table->string('lastname',255);

            $table->string('patronymic',255);

            $table->enum('gender', ['М', 'Ж']);

            $table->string('image',255)->default('images/static/default.jfif');

            $table->string('role',255)->default('user');

            $table->string('email')->unique();

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('password');

            $table->rememberToken();

            $table->timestamps();

        });

    }

Таблица 1 Характеристики полей таблицы «users»

|  |  |  |
| --- | --- | --- |
| id | int | идентификатор пользователя (первичный ключ) |
| name\_first | varchar(255) | имя пользователя |
| name\_last | varchar(255) | фамилия пользователя |
| patronymic | varchar(255) |  |
| password | char(32) | пароль пользователя |
| role | varchar(255) |  |
| gender | Enum['М', 'Ж'] | пол |
| image | varchar(255) |  |
| email | varchar(255) |  |
| email\_verified\_at | timestamp |  |
| password | varchar(255) | пароль |
| remember\_token | varchar(100) | токен |
| created\_at | timestamp | дата создания |
| updated\_at | timestamp | дата обновления |

Создание таблицы «Категория» Характеристики таблицы представлены в таблице2

public function up(): void

    {

        Schema::create('categories', function (Blueprint $table) {

            $table->id();

            $table->string('name',255);

            $table->longText('description',255);

            $table->string('image',255)->default('images/static/device.jpg');

            $table->timestamps();

        });

    }

Таблица 2 Характеристики полей таблицы «categories»

|  |  |  |
| --- | --- | --- |
| id | int | идентификатор пользователя (первичный ключ) |
| name | varchar(255) | название |
| description | varchar(255) | описание |
| image | varchar(255) | изображение |
| created\_at | timestamp | дата создания |
| updated\_at | timestamp | дата обновления |

**2.2 Логическая и физическая структура сайта**

Структура сайта — это логическое построение всех страниц сайта, категорий и подкатегорий. Это логическая схема, в соответствии с которой все страницы и разделы сайта расположены относительно друг друга и принцип, по которому они друг с другом взаимосвязаны.

От того, насколько проста и понятна структура сайта, зависит то, как быстро посетитель найдет нужную ему информацию.

Основные модели логической структуры:

* Строгая линейная структура ресурса. Используется достаточно редко, поскольку не обеспечивает предоставление необходимого удобства – пользователи могут переходить только от фактической страницы к следующей. Очевидная простота в данном случае противопоставлена свободе выбора, что не каждому человеку может понравиться.
* Логическая структура ресурса, так называемая «решетка». Наиболее часто встречается при создании интернет-магазинов и представляет собой линейную структуру, функционирующую в двух направлениях, подразумевающих присутствие вертикальной и горизонтальной связи между страницами.
* Логическая структура сайта, так называемое «дерево». Наиболее распространенная модель, выстраиваемая в определенной иерархической последовательности. Посетители получают полную свободу управления глубиной анализа ресурса и могут заходить на страницы любого уровня.

Логическая структура, так называемая «паутина». Ее может быть связано с желанием обеспечить выразительность предоставления информации или обычным стечением обстоятельств. Определенная страница может ссылаться на другой источник информации, что и обеспечивает максимальную полноту получения данных.

Физическая структура сайта описывает размещение файлов и папок на диске компьютера. Правильная организация файлов и папок поможет в дальнейшей работе с сайтом.

Физическая и логическая структуры не всегда совпадают, потому что обычно физическая структура сайта разрабатывается, основываясь на удобстве размещения файлов. Но более или менее точное сохранение логических разделов в физической структуре сайта позволяет упростить работу по изменению обновлению информационных материалов (рисунок 5).

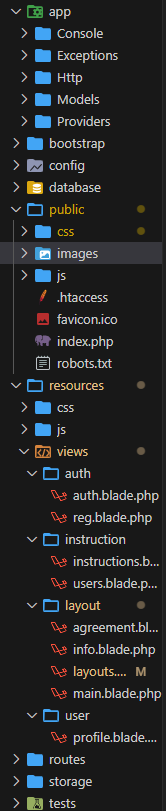


Рисунок 5 – Физическая структура

**2.3 Разработка интерфейса пользователя**

Пользовательский интерфейс — это набор программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае понимают регламентированный обмен информацией между человеком и компьютером, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи. Каждый диалог состоит из отдельных процессов ввода/вывода, которые физически обеспечивают связь пользователя и компьютера. Обмен информацией осуществляется передачей сообщения.

Интерфейс — это место соприкосновения двух функциональных объектов. Если говорить более узко, то интерфейс — это «мост», посредник между человеком, программами и машинами, иными системами. Это инструменты взаимодействия, с помощью которых одна система контактирует и строит диалог с другой.

Качество и удобство интерфейса пользователя играют важнейшую роль в определении уровня удовлетворенности пользователей. Что же делает хороший интерфейс?

Во-первых, он должен быть интуитивно понятным. Пользователь должен легко понимать, как использовать интерфейс и выполнять нужные действия без особых усилий. Отображение информации и доступ к функционалу должны быть логичными и простыми.

Во-вторых, интерфейс должен быть информативным. Вся необходимая информация должна быть предоставлена пользователю в понятном и доступном виде. Важные элементы интерфейса, такие как кнопки, ссылки и подсказки, должны быть ясно видимыми и четко описанными.

В-третьих, интерфейс пользователя должен быть эстетически приятным и привлекательным. Дизайн должен соответствовать общему стилю и образу компании или бренда. Цветовая гамма, шрифты и графические элементы должны гармонично сочетаться и быть привлекательными для глаз пользователя.

В-четвертых, интерфейс должен быть адаптивным и отзывчивым. Он должен быть способен адаптироваться к различным устройствам и экранам, чтобы обеспечить комфортное использование на мобильных телефонах, планшетах и компьютерах. Кроме того, интерфейс должен реагировать на действия пользователя быстро и без задержек.

В-пятых, интерфейс пользователя должен обеспечивать хорошую функциональность и навигацию. Пользователь должен легко находить нужные функции и возможности, а также иметь возможность легко возвращаться к предыдущим шагам. Навигационное меню и разделение информации на логические блоки помогают пользователям легко перемещаться по интерфейсу и быстро находить нужное.

Хороший интерфейс пользователя должен постоянно обновляться и улучшаться на основе обратной связи от пользователей. Слушание и анализ отзывов пользователей позволяет выявлять недостатки интерфейса и вносить изменения, чтобы сделать его еще более удобным и функциональным.

Интерфейс пользователя является важным элементом любой системы, который влияет на ее успешность и эффективность. Создание хорошего интерфейса требует тщательного анализа требований и потребностей пользователей, а также применения лучших практик в области дизайна и удобства использования.

Разработка интерфейса пользователя является одной из важных задач в сфере создания программного обеспечения. UI представляет собой то, через что пользователи взаимодействуют с приложением или веб-сайтом, поэтому его дизайн и удобство использования имеют большое значение

Перед началом разработки интерфейса пользователя необходимо провести исследование и анализ целевой аудитории. Это поможет понять, какие функции необходимо предоставить пользователям, какие проблемы они могут испытывать и как сделать интерфейс максимально интуитивным и удобным в использовании.

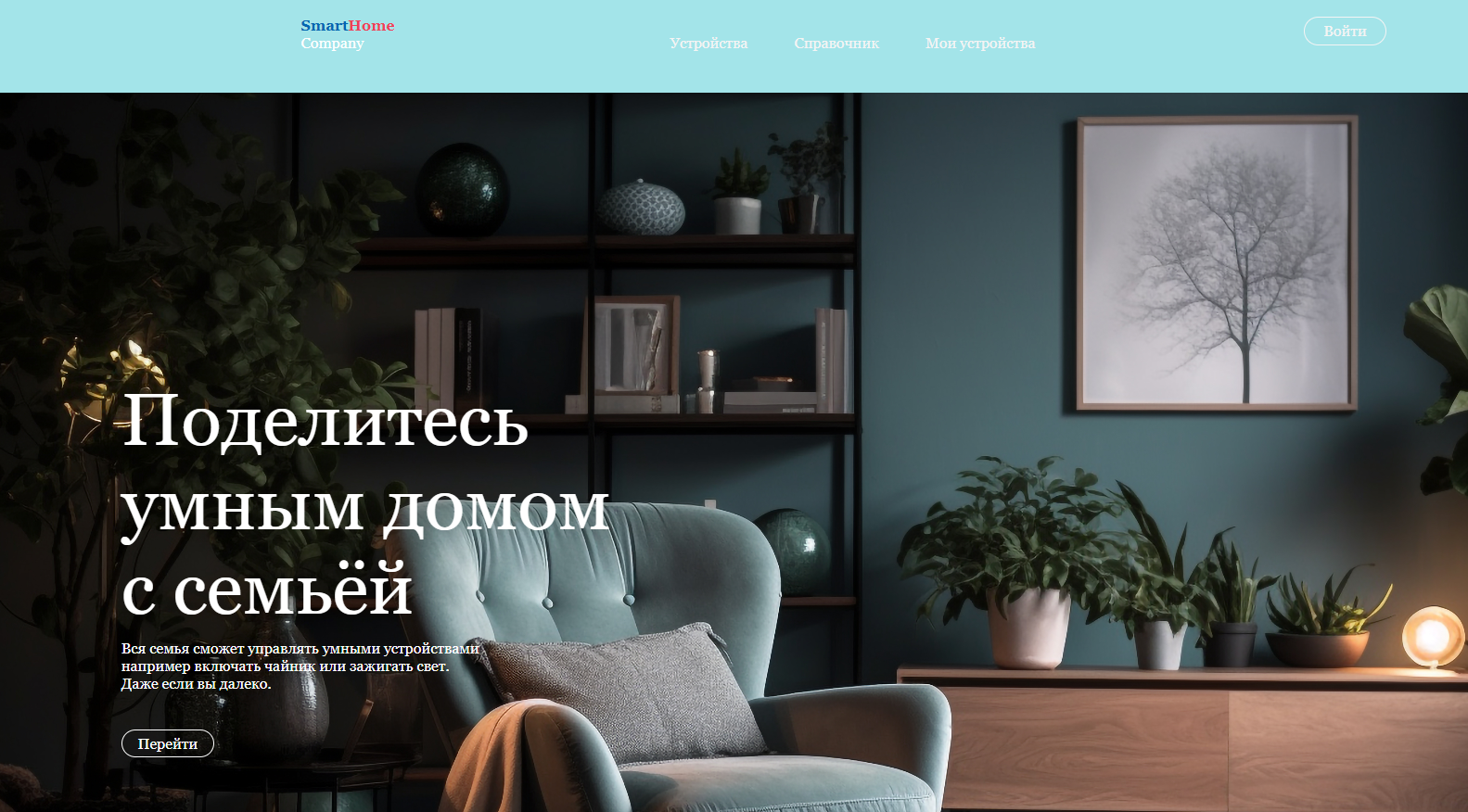
Далее начинается проектирование интерфейса. Оно включает разработку визуальной составляющей, такой как цветовая гамма, типографика, иконки и графика. Учитывается анимация и стилизация элементов интерфейса, которые помогают пользователям взаимодействовать с приложением.

Разработка интерфейса - это не только визуальные аспекты. Важным является также разработка архитектуры интерфейса, которая определяет, какие элементы будут включены в интерфейс и как они будут располагаться на странице или экране. Это включает в себя размещение кнопок, форм, меню и других элементов, которые пользователи будут использовать при взаимодействии с приложением.

Одной из важных задач при разработке интерфейса пользователя является тестирование и улучшение. В процессе разработки необходимо проводить тестирование с реальными пользователями, чтобы выявить проблемы и недостатки интерфейса. На основе обратной связи пользователей можно вносить изменения и улучшать интерфейс, чтобы сделать его более удобным и интуитивным.

В современных технологиях разработки пользовательских интерфейсов широко используется методика адаптивного дизайна. Это позволяет создавать интерфейсы, которые оптимально отображаются на различных устройствах и экранах, таких как компьютеры, смартфоны и планшеты.

Разработка интерфейса пользователя является неотъемлемой частью процесса создания программного обеспечения. Она включает в себя не только визуальные аспекты, но и архитектуру и функциональность. Качественный и удобный интерфейс помогает пользователям эффективно взаимодействовать с приложением и достигать своих целей(Рис.6).

Рисунок 6 – Интерфейс на примере курсового сайта

**2.4** **Разграничение прав доступа**

Права доступа определяют набор действий (например, чтение, запись, выполнение), разрешённых для выполнения субъектам (например, пользователям системы) над объектами данных. Для этого требуется некая система для предоставления субъектам различных прав доступа к объектам. Это система разграничения доступа субъектов к объектам, которая рассматривается в качестве главного средства защиты от несанкционированного доступа к информации или порчи самой системы.

Функции системы разграничения доступа:

* реализация правил разграничения доступа субъектов и их процессов к данным;
* реализация правил разграничения доступа субъектов и их процессов к устройствам создания твёрдых копий;
* изоляция программ процесса, выполняемого в интересах субъекта, от других субъектов.

Права доступа – совокупность правил, регламентирующих порядок и условия доступа субъекта к объектам информационной системы (информации, с носителем, процессам и другим ресурсам) установленных правовыми документами или собственником, владельцем информации.

В таблице 3 представлено разграничение прав доступа данного проекта.

Таблица 3 – Разграничение прав доступа

|  |  |  |  |
| --- | --- | --- | --- |
| Пользователи | Email | Пароль | Права |
| Admin | admin@mail.ru | admin123 | Редактирование контента на сайте |
| User | user@mail.ru | user123 | Использование функционала для управления умным домом |
| Guest |  |  | Просмотр главной страницы |

**2.5**  **Руководство пользователя**

"Инструкция по использованию программы - это краткое руководство, предназначенное для пользователей программы, чтобы они могли корректно и эффективно использовать все ее функции и возможности.

В инструкции пользователю должна содержаться всесторонняя и пошаговая информация, которая поможет пользователю понять основную функциональность программы и научиться работать с ней без необходимости обращаться к дополнительным материалам или спецификациям. Вся необходимая информация для этого извлекается из полной документации, но представляется в удобной форме пользователю.

Первая часть инструкции - описательная. В ней должны присутствовать следующие пункты:

* Наименование программы: указывается название программы, чтобы пользователь мог легко ее идентифицировать.
* Краткое описание программы: дается краткое описание основных функций программы и ее назначения.
* Перечень выполняемых функций: перечисляются все функции, которые может выполнять программа, чтобы пользователь знал, какие возможности у него есть.
* Характеристика метода решения задачи: предоставляется краткое описание метода (или методов), которые использовались для решения поставленной задачи, а также указываются их достоинства и недостатки.
* Библиографическая ссылка: указывается полная библиографическая ссылка на описание метода, чтобы пользователь мог ознакомиться с ним подробнее, если потребуется.
* Входные и выходные данные: описываются данные, которые программа принимает на входе и данные, которые она выдает на выходе.
* Структура базы данных: если программа использует базу данных, то описывается структура базы данных и все ее таблицы.

Вторая часть инструкции должна содержать описание процесса работы с программой. В ней должны присутствовать следующие пункты:

* Запуск программы: описывается, как запустить программу, чтобы пользователь мог начать работу с ней.
* Режимы работы программы: описываются все режимы работы программы, чтобы пользователь мог выбрать нужный.
* Печати и диагностические сообщения: описываются все печати и сообщения, которые могут появляться в процессе работы программы, чтобы пользователь мог корректно на них реагировать.
* Подготовка и ввод данных: описывается, как подготовить и ввести начальные данные в программу.
* Реакция на запросы программы: описывается, как пользователь должен реагировать на запросы программы, чтобы получить желаемый результат.
* Работа в исключительных ситуациях: описывается, как пользователь должен действовать в случае возникновения исключительных ситуаций или ошибок.
* Восстановление работы программы: описывается, как восстановить работу программы в случае ее аварийного завершения.
* Получение требуемого результата: объясняется, как получить конечный результат работы с программой.
* Правильное завершение работы с программой: описывается, как корректно закончить работу с программой.
* Другие сведения: предоставляются дополнительные сведения, которые могут быть полезными пользователю во время работы с программой.

Инструкция пользователю представляет собой независимый документ, содержащий всю необходимую и понятную информацию для пользователя программы, чтобы он мог успешно и эффективно ею пользоваться.

Руководство для обычного пользователя

Перед использование веб сайта пользователь может осмотреть основную информацию о сайт и его пользовательское соглашение перед использованием. Эту информация можно найти в подвале сайта (Рисунок 7).

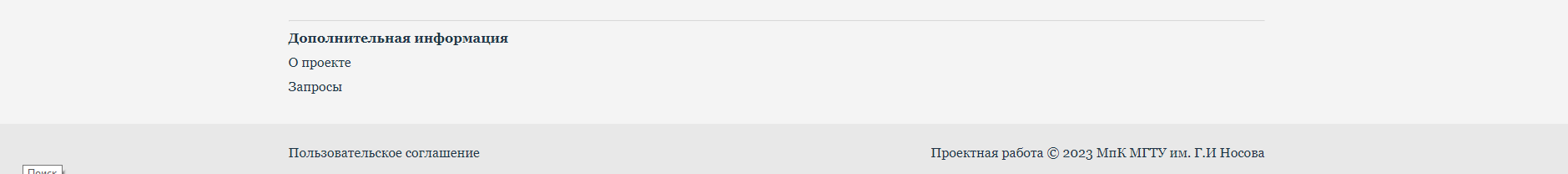


Рисунок 7 – Подвал сайта

После просмотра интересующего вас информации и пользовательского соглашения (Рисунок 8).



Рисунок 8 – Страница с пользовательским соглашением

У пользователем возникает некоторое количество ограничений. Например, неавторизованный пользователь не имеет возможности отправить заявку на создание свое странички продукта или оставить свой отзыв на сайте. Также у него отсутствует доступ к личному кабинету, поскольку учетной записи просто не существует. Чтобы получить возможность зарегистрироваться или войти в систему, необходимо нажать на соответствующую кнопку "Войти" в меню сайта (Рисунок 9).



Рисунок 9 – Меню сайта

После нажатия соответствующей кнопки появляется страничка с авторизацией (Рисунок 10)

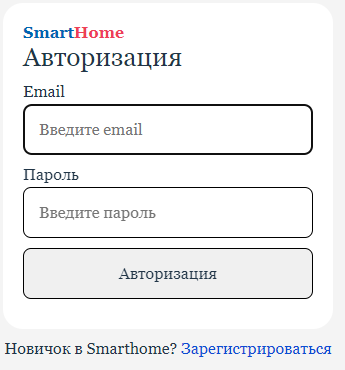


Рисунок 10 – Страница с авторизацией

На этой же странице можно перейди на страничку регистрации (Рисунок 11) при нажатии на ссылку с текстом “Зарегистрироваться”

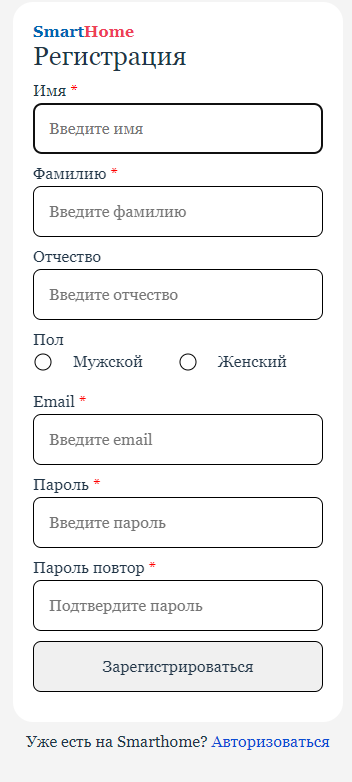


Рисунок 11 – Страница с авторизацией

После того, как пользователь зарегистрировался, он может зайти в личный кабинет (Рисунок 12)

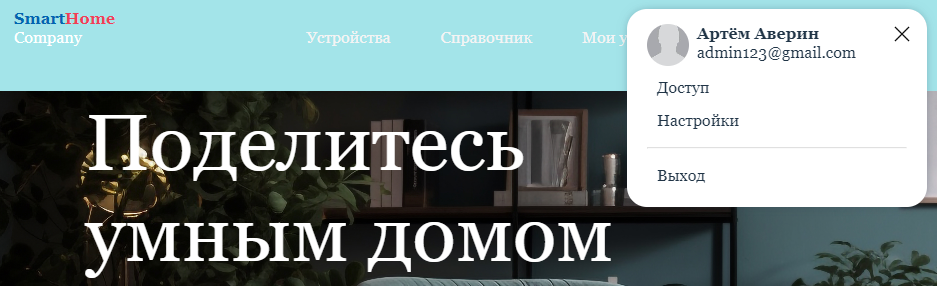


Рисунок 12 – Личный кабинет

Функционал во вкладке “личном кабинете” представляет собой набор возможностей, предоставляемых пользователю для удобного управления своим личными данными. Такой функционал дает пользователю доступ к различным инструментам, позволяющим контролировать свои данные и обновлять персональную информацию на странице профиля пользователя (Рис 13).

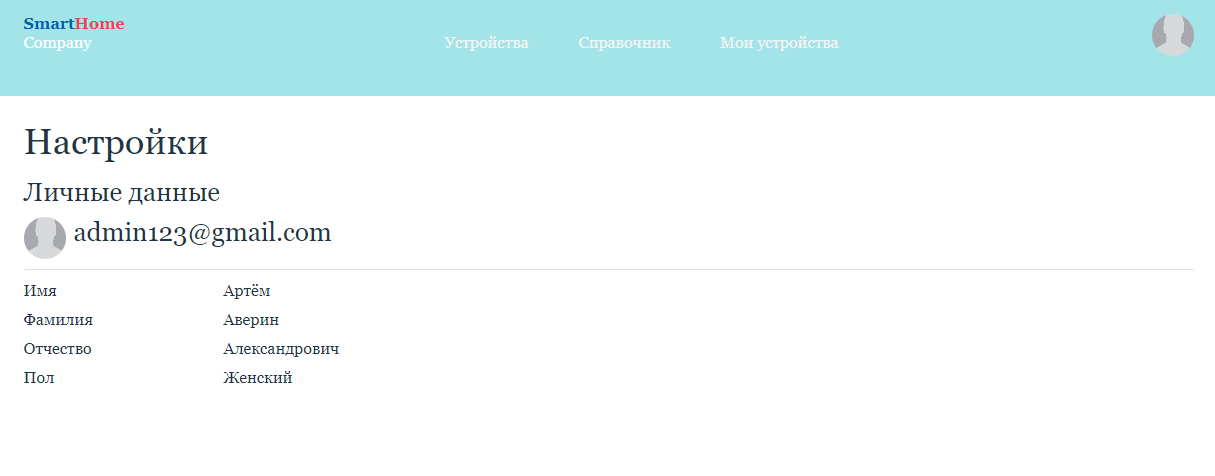


Рисунок 13 – Страница с профиля пользователя

## **2.6 Разработка запросов**

Разработка запросов с использованием Laravel для базы данных MySQL позволяет взаимодействовать с базой данных, извлекать информацию, вносить изменения, а также выполнять другие операции.

Laravel предоставляет различные функции и расширения для работы с базами данных MySQL. Для начала работы необходимо настроить соединение с базой данных. Это включает указание имени хоста, имени пользователя, пароля и имени базы данных.

После успешного установления соединения можно использовать функции Laravel для создания и выполнения запросов к базе данных. Для этого можно использовать миграции удобная функция которая упрощает работу с базой данных

Примеры запросов могут включать выборку данных из таблицы, добавление новых записей, обновление существующих записей или удаление записей. Например это запрос на авторизацию.

public $user\_id;

    public $user\_role;

    public $user\_name;

    public $user\_image;

    public $user\_email;

    public function authUser(){

        if(Auth::check()){

            $user=Auth::user();

            $this->user\_id=$user->id;

            $this->user\_role=$user->role;

            $this->user\_image='public/'.$user->image;

            $this->user\_name=$user->firstname.' '.$user->lastname;

            $this->user\_email=$user->email;

        }else{

            $this->user\_id=null;

            $this->user\_role='quest';

            $this->user\_image=null;

            $this->user\_name=null;

            $this->user\_email=null;

        }

    }

    public function index(Request $request){

        $this->authUser();

        $user=Auth::user();

        $data=(object)[

            'id'=>$this->user\_id,

            'name'=>$this->user\_name,

            'image'=>$this->user\_image,

            'email'=>$this->user\_email,

            'user'=>$user,

            'role'=>$this->user\_role,

        ];

        return view('user.profile')->with(['data'=>$data]);

    }

    public function create(Request $request){

        $this->authUser();

        $data=(object)[

            'id'=>$this->user\_id,

            'name'=>$this->user\_name,

            'image'=>$this->user\_image,

            'email'=>$this->user\_email,

            'role'=>$this->user\_role,

        ];

        return view('auth.reg')->with(['data'=>$data]);

    }

    public function store(Request $request){

        $validator=Validator::make($request->all(),[

            'firstname'=>['required','alpha'],

            'lastname'=>['required','alpha'],

            'patronymic'=>['nullable'],

            'email'=>['email','unique:users'],

            'password'=>['required','min:6','confirmed']

        ]);

        if($validator->fails()){

            return redirect()->route('create')->withErrors($validator)->with('success','Ошибка регистрации');

        }

        else{

            User::create($validator->validated());

            return redirect()->route('login')->with('success','Регистрации прошла успешно');

        }

    }

    public function login(Request $request){

        $this->authUser();

        $data=(object)[

            'id'=>$this->user\_id,

            'name'=>$this->user\_name,

            'image'=>$this->user\_image,

            'email'=>$this->user\_email,

            'role'=>$this->user\_role,

        ];

        return view('auth.auth')->with(['data'=>$data]);

    }

    public function signup(Request $request){

        if(Auth::attempt($request->only(['email','password']))){

            return redirect()->route('home')->with('success','Вы авторизованы');

        }

        else{

            return redirect()->route('login')->with('success','Ошибка авторизации');

        }

    }

    public function logout(Request $request){

        if(Auth::check()){

            Auth::logout();

        }

        return redirect()->route('home')->with('success','Вы вышли');

    }

}

## **2.7 Тестирование информационной системы**

Тестирование информационной системы является важной и неотъемлемой частью процесса ее разработки и внедрения. Оно выполняет роль проверки работоспособности и соответствия системы заявленным требованиям и ожиданиям пользователей.

Перед началом тестирования информационной системы необходимо разработать план тестирования, который определит последовательность и объем работы, а также ресурсы, необходимые для проведения тестирования. В этом плане должны быть определены тестовые сценарии, критерии успеха и ошибок, а также ответственные за выполнение и анализ результатов.

Одной из основных задач тестирования информационной системы является проверка функциональности системы. Это включает тестирование работы отдельных функций и модулей системы, а также проверку их взаимодействия друг с другом. Для этого используются различные методы и техники тестирования, такие как тестирование черного ящика, тестирование белого ящика, тестирование пограничных значений и т.д.

Однако функциональное тестирование не является единственным аспектом тестирования информационной системы. Также проводится тестирование производительности системы, которое включает проверку ее работы при большой загрузке и в неблагоприятных условиях. Тестирование безопасности информационной системы является также важной составляющей, особенно если система содержит конфиденциальную информацию пользователей.

В процессе тестирования информационной системы необходимо учитывать множество факторов, таких как различные варианты использования системы, поведение системы при ошибочных действиях пользователей, совместимость системы с другими программными и аппаратными средствами, а также ее надежность и устойчивость к сбоям.

Оценка результатов тестирования информационной системы позволяет выявить ошибки и недостатки в ее работе, а также принять меры по их устранению и улучшению системы. Регулярное тестирование позволяет поддерживать систему в рабочем состоянии и обеспечивать ее высокое качество.

Таким образом, тестирование информационной системы играет важную роль в обеспечении ее работоспособности, эффективности, безопасности и соответствия требованиям пользователей. Оно помогает выявить и устранить ошибки и недостатки системы, обеспечивая ее стабильное и надежное функционирование.

Таблица 4 - Тестирование методом «черного ящика»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Назначение теста | Ожидаемый результат | Фактический результат | Итог |
| 1 | Нажать кнопку «Регистрация»» | Появятся формы для регистрации | Появились формы для регистрации | Выполнено |
| 2 | Нажать «Регистрация» после заполнения данных в формы регистрации | Пользователь добавится в базу данных и переход на главную страницу | Новый пользователь добавился в базу данных. и переход на главную страницу | Выполнено |
| 3 | Нажать кнопку «Войти» | Откроется окно входа в систему | Открылось окно входа | Выполнено |
| 4 | Нажать на иконку пользователя | Раскроется список | Раскроется список | Выполнено |
| 5 | Нажать кнопку «Выход» | Выход из системы и переход на главную страницу | Выход из системы и переход на главную страницу | Выполнено |

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта был разработан веб-сайт, который отвечает современным требованиям и стандартам веб-разработки. Проект был основан на анализе потребностей и целей заказчика, а также учете требований пользователей.

В процессе разработки были использованы современные технологии и инструменты, такие как HTML, CSS, JavaScript, а также фреймворки и библиотеки, позволяющие создать удобный и функциональный интерфейс. Веб-сайт был разработан с учетом адаптивного дизайна, что позволяет его корректное отображение на различных устройствах, включая планшеты.

Также в процессе разработки был проведен тестирование функциональности веб-сайта. Были выявлены и исправлены возможные ошибки и недочеты, что позволило достичь высокой стабильности сайта.

В результате выполнения данного курсового проекта был создан веб-сайт. Он предоставляет удобный и интуитивно понятный интерфейс, а также функциональность, необходимую для достижения поставленных целей.

Разработка данного веб-сайта позволила применить и углубить знания в области веб-разработки, а также приобрести опыт работы и управлении проектом.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧИКОВ

1. **Агальцов, В.П.** Базы данных. В 2-х кн. Книга 2. Распределенные и удаленные базы данных [Электронный ресурс]: учебник / В.П. Агальцов. – Москва: ИД «ФОРУМ»: ИНФРА-М, 2017. – 271 с. – Режим доступа: <https://new.znanium.com/catalog/document?id=155080> (дата обращения: 04.02.2023).
2. **Ананьева, Т.Н.** Стандартизация, сертификация и управление качеством программного обеспечения [электронное пособие]: учеб. пособие / Т.Н. Ананьева, Н.Г. Новикова, Г.Н. Исаев. – Москва: ИНФРА-М, 2017. – 232 с. – (Высшее образование: Бакалавриат). —Режим доступа: <https://new.znanium.com/catalog/product/792682>  
   (дата обращения: 03.04.2023).
3. **Коваленко, В. В.** Проектирование информационных систем [Электронный ресурс] : учебное пособие / В. В. Коваленко. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — Режим доступа: <https://znanium.com/read?id=329691> (дата обращения: 10.03.2023).
4. **Кроксен-Джон Д.** Оптимизация интернет-магазина: Почему 95% посетителей вашего сайта ничего не покупают и как это исправить [Электронный ресурс] : Справочное пособие / Кроксен-Джон Д., Ван Тондер Й. - Москва :Альпина Паблишер, 2018. - 318 с.: ISBN 978-5-9614-7131-1 - Режим доступа: <https://znanium.com/read?id=333470>  
   (дата обращения: 20.01.2023).
5. Курсы по программированию: [Электронный ресурс] // URL: <https://htmlacademy.ru/> (дата обращения: 28.04.2023).
6. **Лаврентьев, А. Н.** Цифровые технологии в дизайне. История, теория, практика [Электронный ресурс] : учебник и практикум для вузов / А. Н. Лаврентьев [и др.] ; под редакцией А. Н. Лаврентьева. - 2-е изд., испр. и доп. - Москва : Издательство Юрайт, 2019. — 208 с. — (Авторский учебник). — ISBN 978-5-534-07962-3. — Режим доступа: <https://www.urait.ru/bcode/424029> (дата обращения: 06.02.2023).
7. **Лисьев, А.В.** Программное обеспечение компьютерных сетей и web-серверов: учеб. пособие / Г.А. Лисьев, П.Ю. Романов, Ю.И. Аскерко. – Москва: ИНФРА-М, 2018. – 145 с. – (Высшее образование: Бакалавриат) Режим доступа: <https://new.znanium.com/catalog/document?id=328870> (дата обращения: 13.04.2023).
8. **Назаров, С.В.** Архитектура и проектирование программных систем [электронное пособие]: монография / С.В. Назаров. – 2-е изд., перераб. и доп. – Москва: ИНФРА-М, 2017. – 374 с. – (Научная мысль). Режим доступа: <https://new.znanium.com/catalog/product/907016> (дата обращения: 05.03.2023).
9. **Немцова, Т. И.** Компьютерная графика и web-дизайн [Электронный ресурс] : учебное пособие / Т. И. Немцова, Т. В. Казанкова, А. В. Шнякин ; под ред. Л.Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с. + Доп. материалы. - Режим доступа: <https://znanium.com/read?id=329728> (дата обращения: 10.02.2023).
10. Основы языка гипертекстовой разметки HTML и CSS [электронный ресурс]: учебное пособие /Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. // Режим доступа: <http://sernam.ru/book_html.php> (дата обращения: 10.03.2023).
11. Портал по php, MySQL и другим веб-технологиям [Электронный ресурс] ‑ Режим доступа : <http://www.php.su> (дата обращения: 15.04.2023).
12. Ресурсы по программированию на php: [Электронный ресурс] // URL: <http://www.php.net> (дата обращения: 29.04.2023).
13. Ресурсы по программированию: [Электронный ресурс] // URL: <http://www.realcoding.net/> (дата обращения: 01.05.2023).
14. **Шаньгин, В. Ф.** Информационная безопасность компьютерных систем и сетей [Электронный ресурс] : учебное пособие / В. Ф. Шаньгин. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 416 с. — (Среднее профессиональное образование). - Режим доступа: <https://znanium.com/read?id=336332> (дата обращения: 10.04.2023).
15. **Шаньгин, В. Ф.** Комплексная защита информации в корпоративных системах [Электронный ресурс] : учебное пособие / В.Ф. Шаньгин. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 592 с. — (Высшее образование: Бакалавриат). - Режим доступа: <https://znanium.com/read?id=330966> (дата обращения: 11.03.2023).

# ПРИЛОЖЕНИЕ А

Листинг 1 – Регистрация пользователя в базе данных.

public function create(Request $request){

        $this->authUser();

        $data=(object)[

            'id'=>$this->user\_id,

            'name'=>$this->user\_name,

            'image'=>$this->user\_image,

            'email'=>$this->user\_email,

            'role'=>$this->user\_role,

        ];

        return view('auth.reg')->with(['data'=>$data]);

    }

public function store(Request $request){

        $validator=Validator::make($request->all(),[

            'firstname'=>['required','alpha'],

            'lastname'=>['required','alpha'],

            'patronymic'=>['nullable'],

            'email'=>['email','unique:users'],

            'password'=>['required','min:6','confirmed']

        ]);

        if($validator->fails()){

            return redirect()->route('create')->withErrors($validator)->with('success','Ошибка регистрации');

        }

        else{

            User::create($validator->validated());

            return redirect()->route('login')->with('success','Регистрации прошла успешно');

        }

    }

Листинг 2 – Авторизация пользователя в системе.

public function login(Request $request){

        $this->authUser();

        $data=(object)[

            'id'=>$this->user\_id,

            'name'=>$this->user\_name,

            'image'=>$this->user\_image,

            'email'=>$this->user\_email,

            'role'=>$this->user\_role,

        ];

        return view('auth.auth')->with(['data'=>$data]);

    }

    public function signup(Request $request){

        if(Auth::attempt($request->only(['email','password']))){

            return redirect()->route('home')->with('success','Вы авторизованы');

        }

        else{

            return redirect()->route('login')->with('success','Ошибка авторизации');

        }

    }

Листинг 3 – Выход пользователя из системы.

public function logout(Request $request){

        if(Auth::check()){

            Auth::logout();

        }

        return redirect()->route('home')->with('success','Вы вышли');

    }