Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

КАФЕДРА БИЗНЕС-ИНФОРМАТИКИ И МОДЕЛИРОВАНИЯ БИЗНЕС-ПРОЦЕССОВ

**КУРСОВАЯ РАБОТА**

Создание web-приложения с применением современных web-технологий по тематике «Портфолио».

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Никитин

подпись, дата

Студент УБ23-08Б, 432319201 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Ганькулич

подпись, дата

Красноярск 2024

**Содержание**

[Введение 3](#_Toc184307266)

[1. Теоретическая часть 4](#_Toc184307267)

[1.1. HTML – язык разметки гипертекста 4](#_Toc184307268)

[1.2. CSS – каскадные таблицы стилей 6](#_Toc184307269)

[1.3. JavaScript – язык интерактивности 7](#_Toc184307270)

[1.4. PHP – серверный язык обработки данных 8](#_Toc184307271)

[1.5. MySQL – система управление СУБД 10](#_Toc184307272)

[1.6. phpMyAdmin и OpenServer – инструменты для СУБД 11](#_Toc184307273)

[2. Практическая часть 12](#_Toc184307274)

[2.1. Базовые требовании для конфигурации и базы данных 12](#_Toc184307275)

[2.2. Подключение к базе данных 14](#_Toc184307276)

[2.3. Шапка и футер для главной страницы 15](#_Toc184307277)

[2.4. Главная страница 18](#_Toc184307278)

[2.5. Авторизация в личный аккаунт 24](#_Toc184307279)

[2.6. Регистрация нового аккаунта 28](#_Toc184307280)

[2.7. Личный кабинет 31](#_Toc184307281)

[2.8. Управление профилями 34](#_Toc184307282)

[2.9. Просмотр проектов 38](#_Toc184307283)

[2.10. Вкладка контактов – «Обо мне» 41](#_Toc184307284)

[2.11. Вкладка выхода из аккаунта 43](#_Toc184307285)

[Заключение 44](#_Toc184307286)

[Список использованных источников 45](#_Toc184307287)

# **Введение**

Цель данной проекта – разработка веб-приложения, представляющего собой удобную и интуитивно понятную платформу для демонстрации портфолио. Главная идея – создание системы, обеспечивающей легкий доступ к разнообразным работам и способствующей эффективному самопредставлению автора портфолио. В отличие от статичных сайтов или PDF-документов, данная платформа будет фокусироваться на персонализированном опыте пользователя, адаптируясь под его проекты.

Актуальность проекта обусловлена растущим спросом на эффективные инструменты самопрезентации в профессиональной среде. Современный рынок труда требует от специалистов наличия качественного онлайн-портфолио. Наше приложение призвано удовлетворить эту потребность, предлагая удобный, современный и функциональный инструмент для демонстрации навыков и достижений отдельного пользователя.

В процессе разработки будут использованы современные веб-технологии: HTML5, CSS, JavaScript, PHP и MySQL. Это гарантирует высокую производительность, масштабируемость и безопасность приложения.

Ожидаемый результат – полноценное веб-приложение, позволяющее пользователям эффективно презентовать своё портфолио. Приложение будет отличаться интуитивным интерфейсом, удобной навигацией и возможностью персонализации отображения работ.

В данном отчете подробно описаны этапы разработки, использованные технологии и инструменты, а также представлены результаты и перспективы развития проекта.

1. **Теоретическая часть**
   1. **HTML – язык разметки гипертекста**

HTML (HyperText Markup Language) – это фундаментальный язык разметки гипертекста, лежащий в основе всех веб-страниц. Он определяет структуру и содержание веб-документа, используя теги – специальные инструкции, заключенные в угловые скобки < >. Эти теги указывают браузеру, как отображать текст, изображения, видео и другие элементы на странице [4].

Основные элементы HTML:

**<**html**>:** Корневой элемент, содержащий весь код страницы.

**<**head**>:** содержит метаданные страницы, такие как заголовок (<title>), описание (<meta name="description">), ссылки на внешние стили (<link rel="stylesheet">) и скрипты (<script>). Эти данные не отображаются непосредственно на странице, но важны для поисковых систем и функциональности сайта.

**<**body**>:** содержит видимый контент страницы: текст, изображения, ссылки, видео и другие элементы.

Основные теги: HTML использует множество тегов для форматирования текста и организации контента. К наиболее распространенным относятся: <h1>, <p>, <br>, <ul>, <a href= «URL»> текст ссылки</a>, <table>, <tr>, <div>.

Теги часто имеют атрибуты, которые модифицируют их поведение. Например, атрибут href в теге <a> указывает URL-адрес ссылки, а атрибут src в теге <img> указывает путь к изображению. Атрибут alt в теге <img> предоставляет альтернативный текст для изображений, важный для доступности сайта и SEO.

Современный HTML уделяет большое внимание семантической разметке – использованию тегов, которые описывают не только внешний вид, но и смысл контента. Например, <article> обозначает отдельную статью, <aside> – боковую панель, <nav> – навигационное меню, <header> и <footer> – соответственно, заголовок и подвал страницы. Семантическая разметка улучшает доступность сайта для людей с ограниченными возможностями и повышает его рейтинг в поисковых системах.

HTML является основой веб-разработки, обеспечивая структуру и содержание веб-страниц. Понимание принципов HTML, его тегов и атрибутов необходимо для создания любых веб-сайтов и веб-приложений. Изучение семантической разметки и возможностей HTML5 позволяет создавать более качественные, доступные и SEO-оптимизированные веб-ресурсы.

HTML – это фундаментальный язык разметки веб-страниц, отличающийся простотой освоения и универсальностью, обеспечивающей поддержку всеми браузерами. Его структура на основе тегов делает создание, поддержку и изменение веб-страниц относительно легким процессом. Однако, HTML сам по себе имеет ограниченные возможности стилизации и не способен отображать динамический контент. Для создания современных, интерактивных и привлекательных веб-сайтов HTML необходимо дополнять другими языками, такими как CSS (для стилизации) и JavaScript (для динамики), что может усложнить разработку для новичков. В итоге, HTML является необходимой, но не достаточной основой для большинства веб-проектов, требуя интеграции с другими технологиями для достижения полноценной функциональности.

* 1. **CSS – каскадные таблицы стилей**

CSS (каскадные таблицы стилей) – это язык, определяющий визуальное оформление веб-страниц: шрифты, цвета, расположение элементов и многое другое. Его ключевое преимущество – разделение структуры (HTML) и стиля, что делает код чище, понятнее и проще в поддержке. CSS обеспечивает гибкость и масштабируемость, позволяя легко адаптировать дизайн под различные устройства и быстро изменять внешний вид сайта. Возможности CSS простираются и на добавление анимаций и трансформаций, значительно повышая привлекательность веб-страниц [4].

Однако, при работе с CSS необходимо учитывать возможные проблемы совместимости между различными браузерами, что может потребовать дополнительной работы по обеспечению кросс браузерной совместимости. Создание сложных анимационных эффектов требует глубокого понимания CSS и может быть достаточно трудоемким. Наконец, избыточное использование CSS-стилей и больших изображений может негативно сказаться на скорости загрузки страницы, что ухудшит пользовательский опыт. Поэтому, эффективное использование CSS предполагает баланс между желаемой визуальной привлекательностью и производительностью сайта.

Важно отметить также, что современные CSS-фреймворки (такие как Bootstrap или Tailwind CSS) существенно упрощают процесс разработки и помогают избежать многих из перечисленных недостатков, предоставляя готовые стили и компоненты, ускоряющие разработку и улучшающие кросс браузерную совместимость.

* 1. **JavaScript – язык интерактивности**

JavaScript – это мощный язык программирования, используемый для придания интерактивности веб-страницам. Он позволяет создавать динамические элементы, обрабатывать действия пользователя (например, клики, ввод данных), и реализовывать асинхронную загрузку данных с сервера, обновляя содержимое страницы без полной перезагрузки. Эти возможности открывают широкие перспективы для создания сложных и интерактивных веб-приложений [5].

Однако, JavaScript имеет и свои ограничения. Неправильно написанный код может быть уязвим к различным атакам, от инъекций кода до вредоносных сценариев, что требует тщательной проверки и безопасности. Кроме того, различные браузеры могут интерпретировать JavaScript-код по-разному, требуя адаптации кода под разные среды. Наконец, чрезмерное использование JavaScript может привести к увеличению времени загрузки страницы, что негативно скажется на опыте пользователя. Поэтому, эффективное применение JavaScript подразумевает баланс между функциональностью и производительностью.

Важно понимать, что современные фреймворки и библиотеки (React, Angular, Vue.js) значительно упрощают разработку и минимизируют риски. Они предоставляют готовые компоненты, структуры, и инструменты для эффективного управления состоянием и взаимодействия с пользователем, автоматизируя многие рутинные задачи. Это позволяет разработчикам сосредоточиться на логике приложения, а не на технических аспектах JavaScript. Также, использование модулей и организации кода в файлы с использованием ES6 (и выше) методов минимизирует глобальную загрязненость и улучшает понимание и поддержку кода. Таким образом, JavaScript, в сочетании с современными инструментами, становится мощным средством для создания сложных и эффективных веб-приложений.

* 1. **PHP – серверный язык обработки данных**

PHP (Hypertext Preprocessor) – это широко распространенный серверный язык программирования, используемый для создания динамических веб-страниц и веб-приложений. Его ключевое преимущество – способность генерировать контент на сервере, взаимодействуя с базами данных и обрабатывая пользовательские запросы, что позволяет создавать сложные и функциональные веб-ресурсы, от простых сайтов до крупных интернет-магазинов. PHP просто интегрируется с популярными СУБД (MySQL, PostgreSQL и др.), позволяя эффективно управлять данными и создавать динамическое содержимое. Его гибкость и обширная экосистема библиотек и фреймворков открывают возможности для решения разнообразных задач веб-разработки [7].

Однако, использование PHP требует внимательного отношения к безопасности. Неправильное обращение с данными, например, незащищенные от SQL-инъекций запросы к базе данных, может привести к серьёзным уязвимостям [6]. Кроме того, не оптимизированный код может снизить производительность приложения, особенно при обработке больших объемов данных или при большом количестве запросов. Также, по мере роста сложности проекта, структура кода может становиться сложной в поддержке и масштабировании, требуя применения принципов объектно-ориентированного программирования и использования более структурированных подходов к архитектуре.

Для решения проблем масштабируемости и производительности, широко используются различные фреймворки (Laravel, Symfony, CodeIgniter), которые предоставляют готовые структуры, инструменты и паттерны проектирования, значительно упрощающие разработку и поддержку больших и сложных веб-приложений. Эти фреймворки позволяют соблюдать лучшие практики программирования, улучшая производительность и безопасность приложения.

В последние годы, PHP активно развивается, появляются новые версии с улучшенной производительностью и поддержкой современных парадигм программирования. Кроме того, современные инструменты для тестирования и отладки помогают предотвращать ошибки и повышать качество кода.

В заключение, можно сказать, что PHP остается актуальным и популярным языком для веб-разработки, однако его эффективное использование требует знания лучших практик программирования и применения современных инструментов и фреймворков для обеспечения безопасности, масштабируемости и производительности разрабатываемых приложений.

* 1. **MySQL – система управление СУБД**

MySQL — это популярная система управления базами данных (СУБД), широко используемая для хранения и обработки структурированных данных в веб-приложениях. Она обеспечивает высокую производительность при работе с большими объемами информации, поддерживает различные типы данных, и обладает гибкой масштабируемостью [3]. Это делает MySQL хорошим выбором для многих проектов, начиная от небольших сайтов и заканчивая крупными интернет-магазинами и корпоративными системами.

Однако, MySQL имеет определенные ограничения. Некоторые критики отмечают, что возможности работы с хранимыми процедурами и триггерами в MySQL могут быть ограниченными по сравнению с другими СУБД. Это может влиять на сложность реализации сложных бизнес-логик и оптимизации запросов. Также, в отдельных случаях, при работе с крайне большими объемами данных возможна недостаточная эффективность, требующая тщательной настройки и оптимизации базы данных, а также возможных переходов к более масштабируемым решениям.

Масштабирование MySQL — важный момент. Хотя система хорошо масштабируется вертикально, увеличивая мощность сервера, горизонтальное масштабирование (добавление дополнительных серверов) требует более глубокой настройки и понимания принципов кластеризации. Неправильно настроенный кластер может привести к снижению производительности. Поэтому, для больших и сложных проектов, настройка и подбор оптимальной архитектуры базы данных является критическим фактором.

Выбирая MySQL, важно учитывать особенности проекта и требования к обработке данных. Для небольших проектов с ограниченным объемом данных MySQL предоставляет мощные возможности, но для масштабных систем необходимо продумать архитектуру и применять дополнительные инструменты для оптимизации производительности.

* 1. **phpMyAdmin и OpenServer – инструменты для СУБД**

phpMyAdmin — это веб-интерфейс для управления системами управления базами данных (СУБД), в частности, MySQL [3]. Он позволяет администраторам и разработчикам взаимодействовать с базами данных без необходимости использования командной строки, предоставляя удобный графический интерфейс для выполнения различных операций: создания и удаления баз данных, таблиц, добавления, редактирования и удаления данных, выполнения SQL-запросов и многое другое. phpMyAdmin [2] существенно упрощает администрирование MySQL, особенно для пользователей, не знакомых с командной строкой.

OpenServer — это бесплатный локальный веб-сервер для Windows, позволяющий запускать и тестировать веб-приложения на локальной машине. Он включает в себя Apache, MySQL, PHP и другие необходимые компоненты, обеспечивая полную среду разработки [1].

Совместная работа phpMyAdmin и OpenServer позволяет разработчикам эффективно создавать и тестировать веб-приложения, использующие MySQL в качестве СУБД. После установки OpenServer, MySQL уже включен и настроен, а доступ к phpMyAdmin осуществляется через веб-интерфейс, обычно по адресу http://localhost/phpmyadmin [2]. Это позволяет создавать базы данных, наполнять их данными и тестировать взаимодействие веб-приложения с базой данных в реальном времени, не требуя дополнительной настройки.

В целом, комбинация OpenServer [1] и phpMyAdmin [2] представляет собой удобную и эффективную среду разработки для веб-приложений, использующих MySQL, позволяя разработчикам сосредоточиться на коде, а не на настройке сервера. Это значительно упрощает процесс разработки и тестирования, сокращая время на решение сопутствующих задач.

1. **Практическая часть**
   1. **Базовые требовании для конфигурации и базы данных**

Перед тем, как приступить к разработке web-приложения, необходимо разобраться с тем, что от нас требуется. К данной работе предъявлено 8 требований:

1. Должна быть предусмотрена авторизация и регистрация пользователей.
2. Зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги.
3. Администраторы имеют право редактировать пользователей и добавлять услуги.
4. Неавторизированные пользователи имеют право просматривать список услуг.
5. Сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении.
6. Общий функционал может быть продуман самостоятельно.
7. Важно продемонстрировать не большой объем функционала, а минимальные требования к проекту. Проект может максимально простым, но при этом содержать все необходимые функции.
8. Отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

Для начала работы необходимо загрузить OpenServer версии 6.0 с официального сайта и настроить его: активировать версии PHP 8.1 и MySQL 8.0 [3]. Учитывая обновлённый интерфейс версии 6.0, для удобства работы рекомендуется установить phpMyAdmin в папку localhost и внести соответствующие изменения в файл project.init (папка osp) согласно рисунку 2.1.1. После этого phpMyAdmin появится в меню сервисов OpenServer [1]. Для работы с проектами достаточно указать используемую версию PHP, путь к папке с проектом (public\_dir как {base\_dir}) и IP-адрес, как показано на рисунке.

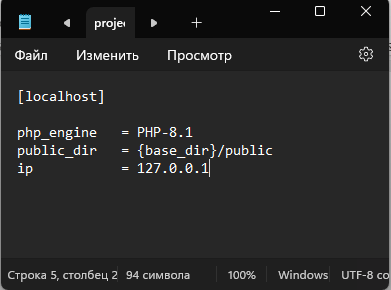


Рисунок 2.1.1 – Настройка OpenServer для работы с phpMyAdmin

Далее мы перейдем к созданию базы данных с использованием phpMyAdmin. Для нашей работы нам потребуется база данных "user\_managment" с таблицей "users", состоящей из 4 столбцов. Следует отметить, что данное приложение будет представлять собой площадку для продажи автомобилей, и список автомобилей не планируется изменять - он будет фиксированным.

Далее мы перейдем к созданию таблицы "users". В этой таблице нам нужно будет создать 7 столбцов: id, username, password, email, role, created\_at, birth\_date. На рисунке 2.1.2 показана данная таблица со всеми необходимыми параметрами [8].

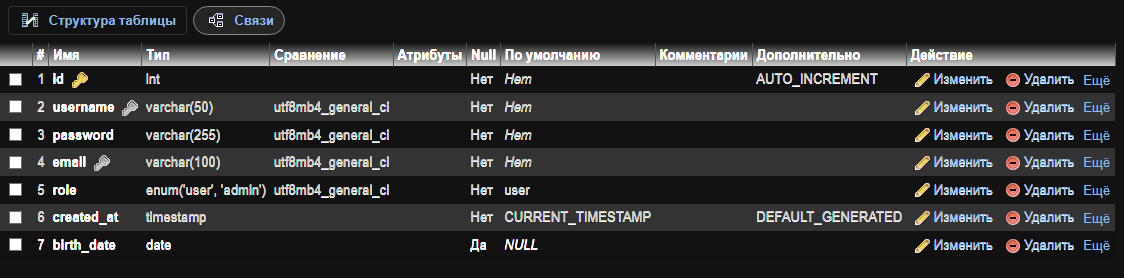


Рисунок 2.1.2 – Таблица «users» из базы данных "user\_managment"

* 1. **Подключение к базе данных**

Для успешной работы с базой данных «user\_management» необходимо её подключить, чтобы затем этот файл внедрять в header.php и другие файлы.

Для подключения к базе данных, используется объектно-ориентированный подход с классом PDO. Этот подход обеспечивает надежное и эффективное взаимодействие с СУБД. Код настроен на подключение к локальному серверу MySQL 8.0, к базе данных user\_management с использованием учетной записи пользователя root без пароля [8].

Для обработки возможных ошибок при подключении используется блок try...catch, в случае возникновения исключения PDOException детальное сообщение об ошибке будет выведено пользователю. Это позволяет быстро выявлять и исправлять проблемы (Рисунок 2.2.1).

Таким образом, данный код обеспечивает надежное и безопасное подключение к базе данных, необходимое для дальнейшего функционирования приложения. Обработка исключений позволяет предотвратить сбои приложения и предоставить пользователю информативные сообщения об ошибках. Однако, необходимо учитывать риски использованных параметров подключения и изменить их в рабочей среде на более безопасные.

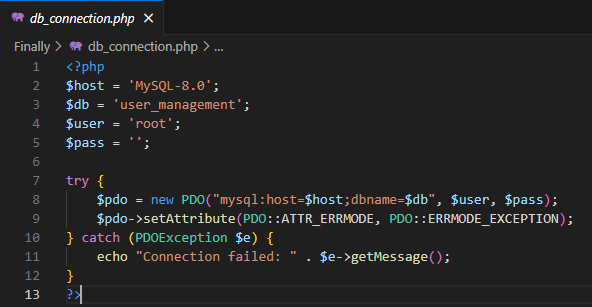


Рисунок 2.2.1 – Реализация подключения к MySQL

* 1. **Шапка и футер для главной страницы**

В коде реализована шапка сайта (header.php), включающая в себя навигационное меню и блок авторизации/выхода. Шапка использует сессии (session\_start()) для определения авторизованного пользователя и его роли из базы данных (db\_connection.php). В зависимости от статуса пользователя (авторизован/неавторизован) и его роли («admin» или обычный пользователь) отображаются различные элементы навигации: ссылки на главную страницу, страницу с проектами, профиль пользователя (для авторизованных), а также админ-панель (только для администратора). Для неавторизованных пользователей отображается кнопка «Авторизация/Регистрация», а для авторизованных — кнопка «Выход». Навигационное меню реализовано двумя способами: стандартное горизонтальное меню и выпадающее меню (с использованием JavaScript. Шапка динамически подстраивается под роль пользователя, обеспечивая персонализированный опыт. Стилевое оформление осуществляется с помощью подключенных CSS-файлов (main.css для основной страницы и project.css для раздела – «Проекты»). Заголовок страницы динамически устанавливается с помощью переменной $title. Ознакомиться с кодом можно на рисунках 2.3.1 и 2.3.2.

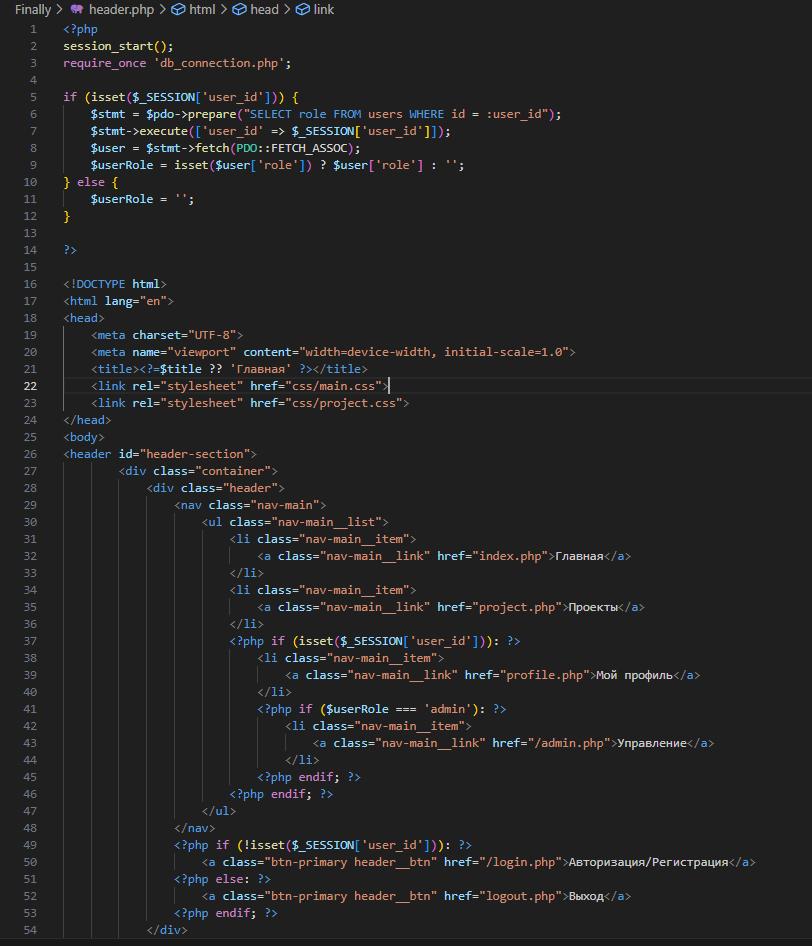


Рисунок 2.3.1 – Реализация кода в header.php часть 1



Рисунок 2.3.2 – Реализация кода в header.php часть 2

Футер (footer.php) сайта содержит стандартную информацию об авторских правах и ссылку на страницу «Обо мне». Футер оформлен с помощью CSS (код CSS не предоставлен, но предполагается, что стили определяют внешний вид). В нём присутствует текст авторских прав, содержащий имя разработчика («TheBelskx») и текущий год, полученный с помощью PHP-функции date('Y'). Также в футере есть ссылка contact.php на страницу «Обо мне», обеспечивающая дополнительную информацию о разработчике. Все элементы размещены внутри контейнера .container, что позволяет контролировать макет и выравнивание. В целом, футер выполняет роль нижней части страницы, предоставляя необходимую информацию об авторских правах и ссылки на полезные ресурсы (Рисунок 2.3.3).

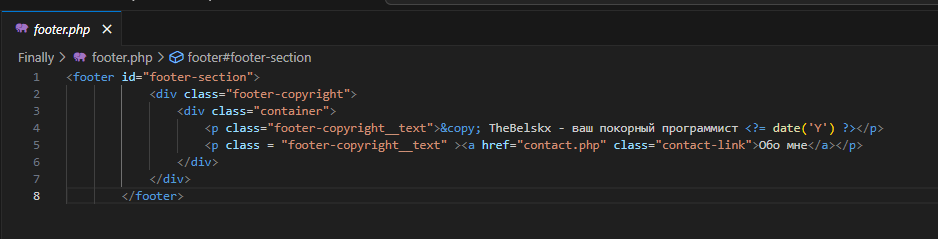


Рисунок 2.3.3 – Реализация footer.php

Таким образом были реализованы успешно два файлы: header.php и footer.php, который в дальнейшем можно использовать на всех страничках, тем самым придерживаясь общей стилистике сайта.

* 1. **Главная страница**

Главная страница (index.php) представляет собой статическую страницу, использующую шапку (header.php) и футер (footer.php), описанные ранее [7]. Основное содержимое страницы размещено внутри тега <main>. Центральным элементом является баннер (banner-section), содержащий заголовок, краткое описание деятельности разработчика (TheBelskx), специализирующегося на C++ и Unreal Engine, и призыв к действию – кнопки для перехода к странице с проектами и просмотра видео. Описание акцентирует внимание на объёме выполненных проектов (более 5 игр и проектов на C++), подчеркивая опыт разработчика.

Баннер включает в себя заголовок (banner\_\_header), описательный текст (banner\_\_text), две кнопки (btn-primary, btn-secondary) для перехода к проектам и видео соответственно, и блок с выделенными ключевыми показателями (количество игровых проектов и проектов на C++). Использование элементов с классами banner\_\_header, banner\_\_text, banner\_\_link, banner\_\_btn, banner-features\_\_text, banner-features\_\_text\_strong предполагает применение CSS для стилизации и верстки баннера. Наличие элемента banner-img указывает на использование фоновой графики или изображения в баннере.

Вся страница структурирована с использованием контейнеров (container), обеспечивающих адаптивность и правильное отображение контента на разных разрешениях экранов. Подключение header.php и footer.php происходит с помощью require\_once, что обеспечивает инкапсуляцию и повторное использование кода. В целом, главная страница выполняет роль визитной карточки разработчика, эффективно представляя его навыки и опыт, и направляя пользователя к дополнительной информации о проектах и контактах. Дизайн и структуру страницы контролирует CSS, код которого включен в header.php (main.css), а реализация навигации и авторизации предоставлена в header.php. Детально код продемонстрирован на рисунке 2.3.1.

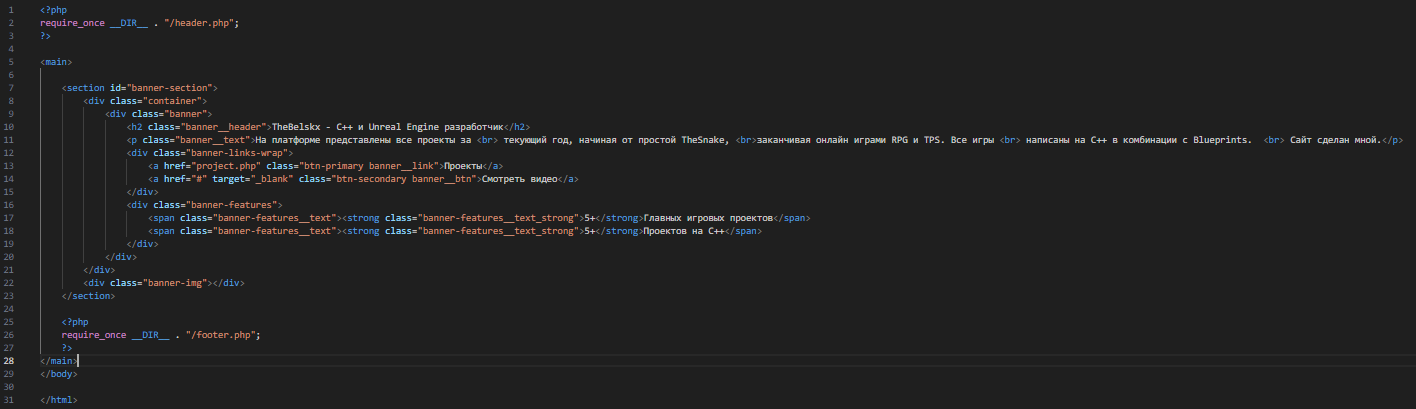
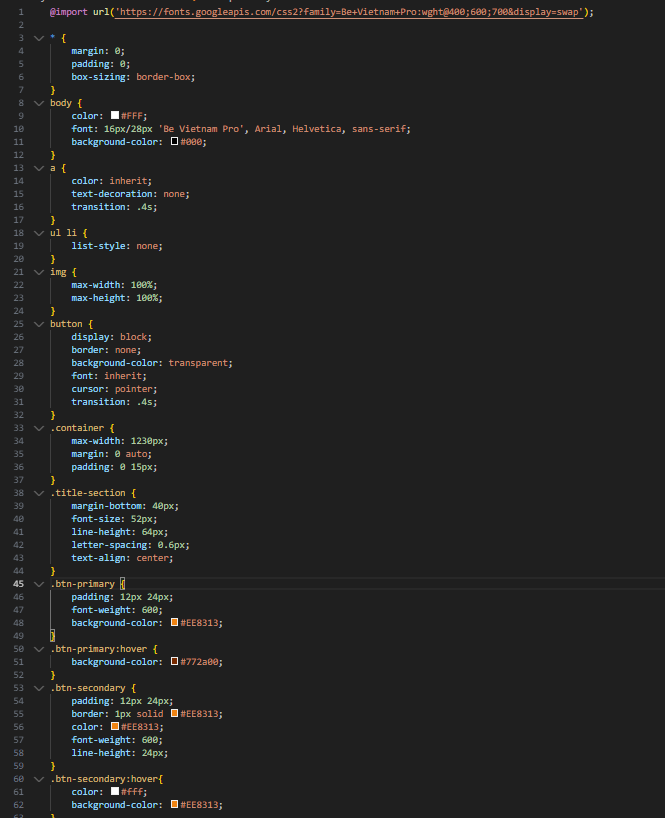


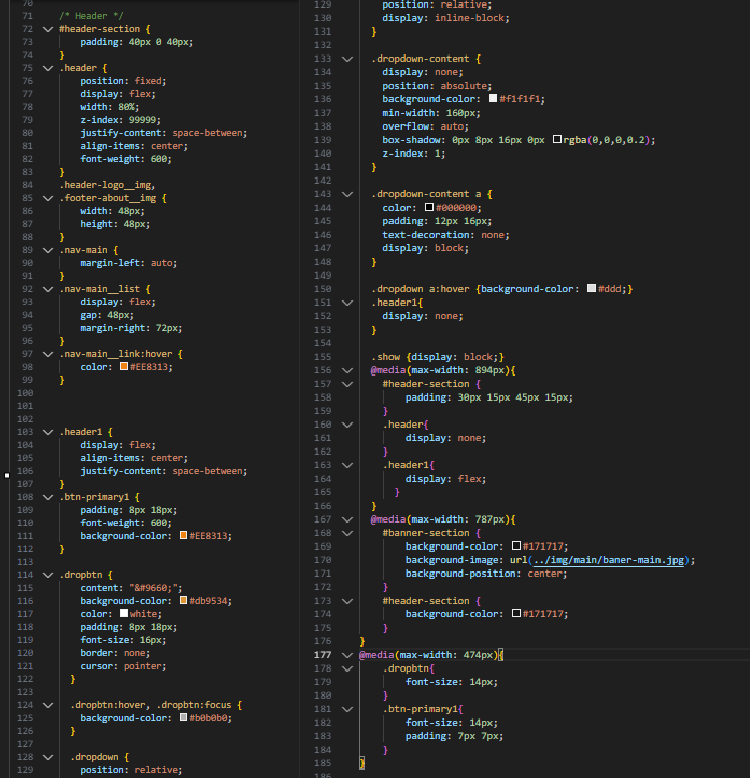
Рисунок 2.4.1 – Реализация главной страницы

Для обеспечения единого стиля сайта, была применена модульная архитектура, с использованием отдельных файлов для шапки (header.php) и подвала (footer.php). Такой подход позволил обеспечить визуальное единство всех страниц сайта за счет наследования этих файлов. Вся стилистика сайта реализована в файле main.css, более детальное описание которого представлено далее.

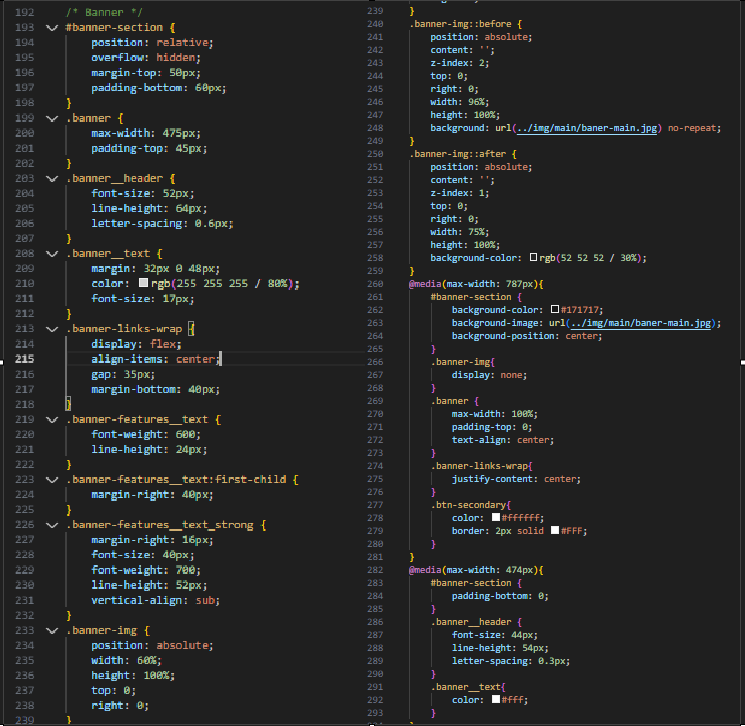
Листинг CSS составляющей main.css:



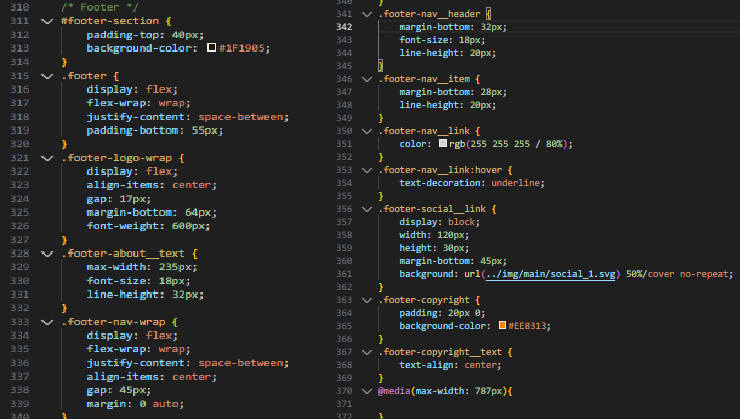
Файл main.css устанавливает единый визуальный стиль сайта, используя шрифт Be Vietnam Pro и сбрасывая стили по умолчанию для предсказуемого отображения и избежания конфликтов. Он задает базовые стили для текста, ссылок, изображений и листов, а также определяет стили для контейнера страницы (.container) и специфические стили для заголовков (.title-section) и кнопок (.btn-primary, .btn-secondary), используя плавную анимацию (transition) и адаптивную верстку (max-width, border-box) [4].



Данный CSS-код отвечает за стилизацию секции заголовка (header-section). Он создает фиксированное верхнее меню (header), обеспечивает плавную смену цвета при наведении (transition), и реализует выпадающее подменю (dropdown) для адаптации на разных устройствах. Верхнее меню имеет фиксированное положение, чтобы оно всегда было видимо. Навигационное меню (nav-main) выполнено в форме список со ссылками. Реализована адаптивность для разных ширин экранов с помощью @media, меняющей отображение и расположение элементов меню и кнопок в зависимости от размера экрана. Стилизация выполнена в соответствии с общим стилистическим решением сайта. В небольших устройствах меню преобразуется в выпадающее. В целом, это обеспечивает единый визуальный стиль и удобство пользователя на различных устройствах.



Данный фрагмент CSS-кода описывает стили для баннерного раздела (#banner-section) сайта. Он создает эффект наложения текстового блока (.banner) на фоновое изображение, используя абсолютное позиционирование (position: absolute) для обрамления изображения (.banner-img) и наложения полупрозрачного фона. Текст баннера (.banner\_\_header, .banner\_\_text) стилизован с использованием большого размера шрифта и отступов, а кнопки (.banner-links-wrap) выровнены. Блок с ключевыми показателями (.banner-features) выделен жирным шрифтом и размерами для подчеркивания важности информации. Адаптивная верстка (@media) изменяет размещение элементов на малых экранах: в этом случае изображение скрывается, текстовый блок растягивается на всю ширину, а кнопки центрируются [4].



Этот CSS-код описывает стили для футера (#footer-section) сайта. Футер имеет темный фон (#1F1905) и содержит разделы, размещенные с помощью flexbox для адаптивности. В нем представлено навигационное меню (footer-nav-wrap) с подзаголовками (.footer-nav\_\_header) и пунктами (.footer-nav\_\_item), а также блок соцсетей (footer-social\_\_link). Копирайты (footer-copyright) выделены отдельным блоком с фоновым цветом (#EE8313).

Разработана главная страница сайта, предоставляющая пользователям доступ к функциям регистрации, авторизации, просмотра проектов и информации о разработчике. Авторизованные пользователи могут просматривать свой профиль, а администраторы — управлять учетными записями.

Стилистически сайт выглядит теперь очень достойно и радует глас. Более детально рассмотреть можно на рисунке 2.4.2. Можно сказать, что первостепенные задачи успешно выполнены.

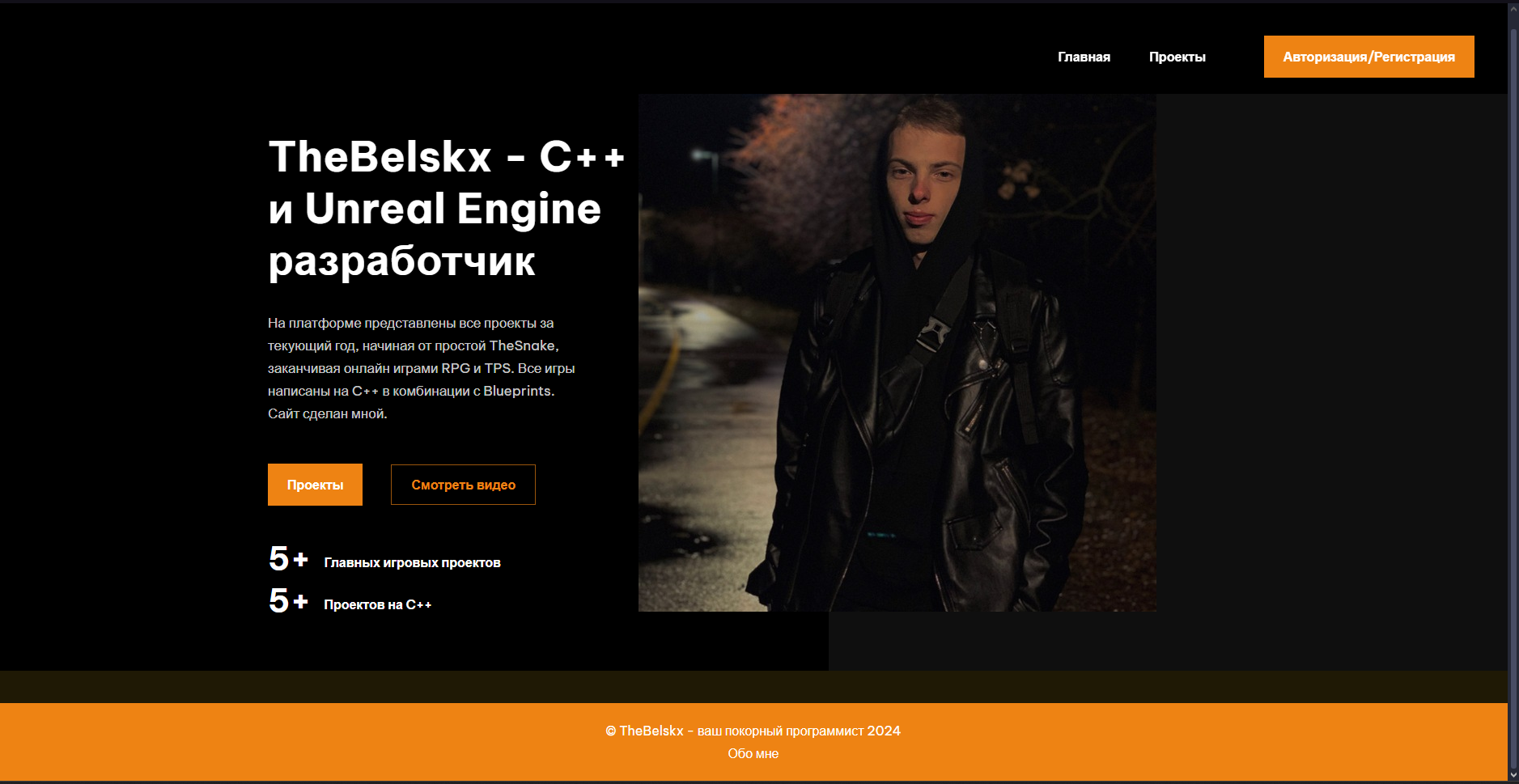


Рисунок 2.4.2 – Общий вид «Главной» страницы без авторизации

* 1. **Авторизация в личный аккаунт**

Форма авторизации является обязательным компонентом нашего проекта в связи с требованиями к работе, предполагающими наличие системы авторизации и аутентификации пользователей. Для этого создадим php файл под названием login.php, в котором будет размещена форма авторизации.

Фрагмент кода на рисунке 2.5.1 представляет собой страницу авторизации, реализованную с использованием PHP и HTML. PHP-часть обрабатывает отправку формы авторизации. Сначала инициализируется сессия и подключается файл db\_connection.php, содержащий параметры подключения к базе данных. Затем проверяется, был ли отправлен POST-запрос. Если да, извлекаются имя пользователя и пароль из массива $\_POST. С помощью подготовленного запроса SELECT проверяется наличие пользователя в базе данных. Если пользователь найден и пароль верифицируется с помощью password\_verify, его ID и роль записываются в сессию, и пользователь перенаправляется на главную страницу (index.php). В случае неверного имени пользователя или пароля, сообщение об ошибке записывается в сессию [6].

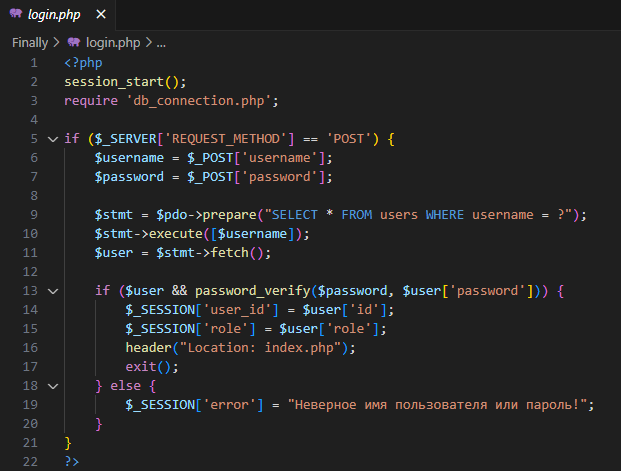
****

Рисунок 2.5.1 – Реализация обработки отправки формы

HTML-часть представляет форму авторизации. Она содержит поля для ввода имени пользователя и пароля, кнопку «Войти» и ссылку на страницу регистрации. Если в сессии присутствует сообщение об ошибке, оно отображается на странице после сброса сессии. Стилевое оформление формы осуществляется с помощью подключенного CSS-файла login.css. В целом, код представляет простую, но функциональную страницу авторизации с обработкой ошибок и перенаправлением. Используется стандартный механизм PHP-сессий для хранения информации о авторизованном пользователе. Обработка паролей осуществляется с помощью функции password\_verify, обеспечивающей безопасность хранения паролей в базе данных.

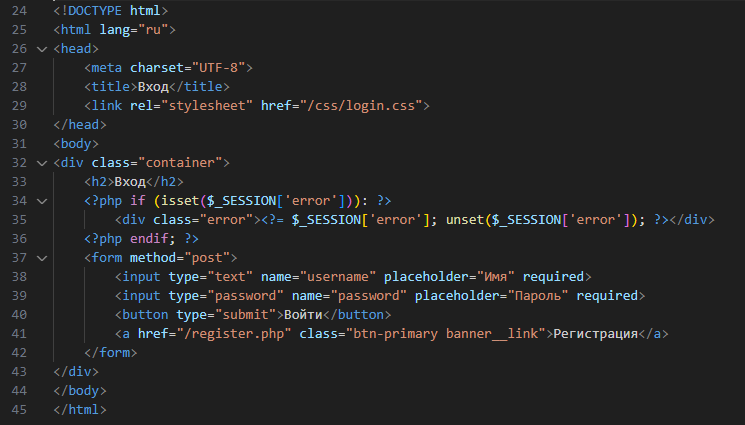


Рисунок 2.5.2 – Реализация формы регистрации

Как уже говорилось, мы на странице подключаем CSS-файл login.css, который отвечает за стилизацию страницы авторизации. Стили body центрируют контент по вертикали и горизонтали на странице, устанавливают темный фоновый цвет (#000) и удаляют стандартные отступы. Контейнер (.container) имеет оранжевый фон (#EE8313), закругленные углы, тень и центрированный текст. Заголовки (h2) выполнены серым цветом. Блок ошибок (.error) выделен розовым фоном и красным текстом. Поля ввода (input) имеют стандартные стили: стандартные цвета и закругленные углы. Кнопка “Войти” (button) имеет темно-оранжевый фон, белый текст и изменение цвета при наведении. Ссылка на регистрацию (.btn-primary) стилизована аналогично кнопке. Медиа-запрос (@media) адаптирует стили для малых экранов, уменьшая отступы и ширину полей ввода (Рисунок 2.5.3).

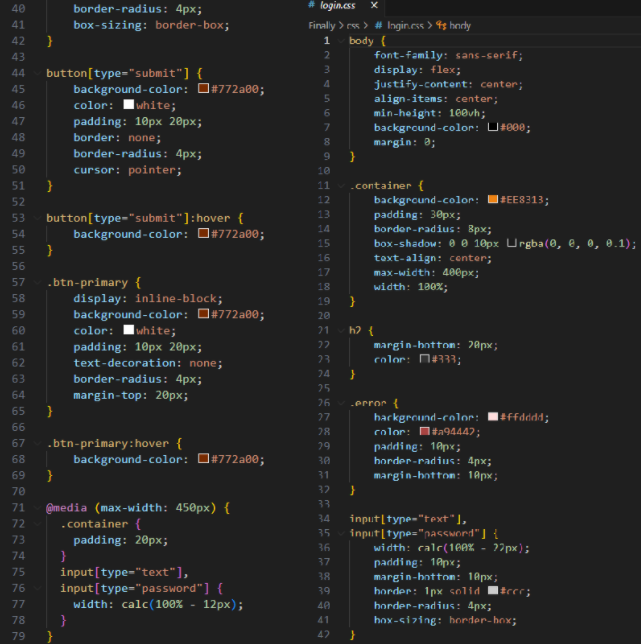
****

Рисунок 2.5.3 – Реализация стилистики страницы регистрации

Разработанный код представляет собой функциональный и стилистически согласованный интерфейс для авторизации и навигации по веб-сайту. Можно сказать, что успешно выполнена задача «авторизации пользователей». Страница выполнена в примерно таком же стили, как и другие страницы, поэтому в будущем нет смысла показывать все css файлы, так стилистика одинаковая. Ознакомиться с внешним представлением страницы можно на рисунке 2.5.4.

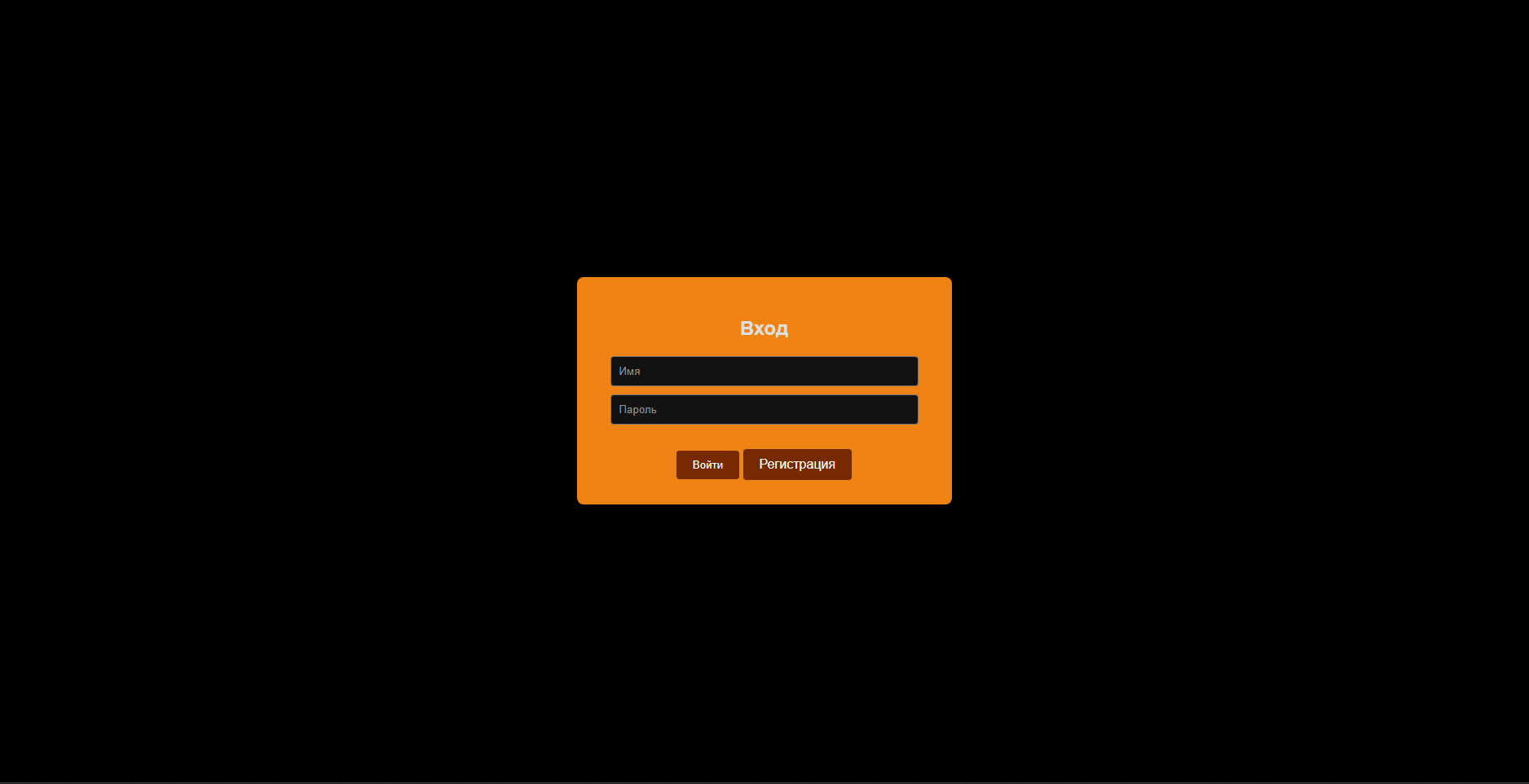


Рисунок 2.5.4 – Общий вид страницы «Авторизация»

* 1. **Регистрация нового аккаунта**

Как уже говорилось ранее про форму авторизации, необходимо уточнить, что форма регистрации также является обязательным компонентом нашего проекта в связи с требованиями к работе, предполагающими наличие системы авторизации и аутентификации пользователей. Для этого создадим php файл под названием register.php, в котором будет размещена форма регистрации. Рассмотрим также код в формате разбивки на PHP и HTML блоки.

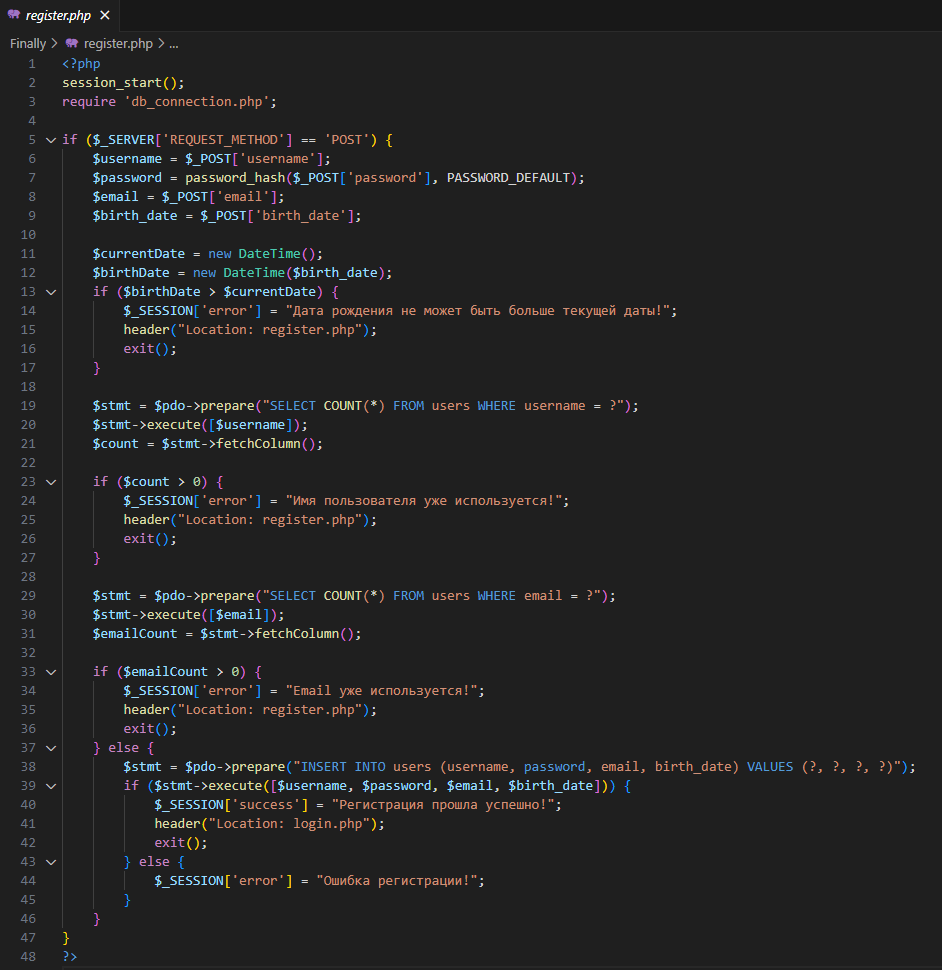


Рисунок 2.6.1 – Реализация PHP кода на странице регистрации

PHP-часть начала инициализирует сессию и подключает файл db\_connection.php. Затем проверяется, был ли отправлен POST-запрос с данными регистрации. Если да, из массива $\_POST извлекаются имя пользователя, email, дата рождения и пароль. Пароль хэшируется с помощью password\_hash перед хранением в базе данных. Перед добавлением пользователя в базу данных проводится проверка на уникальность имени пользователя и email с помощью SQL-запросов. Если имя пользователя или email уже используются, соответствующее сообщение об ошибке выводится на странице. Также проверяется, что дата рождения не превышает текущую дату. В случае успешной регистрации, сообщение об успехе записывается в сессию, и пользователь перенаправляется на страницу авторизации. В случае ошибки, соответствующее сообщение записывается в сессию.

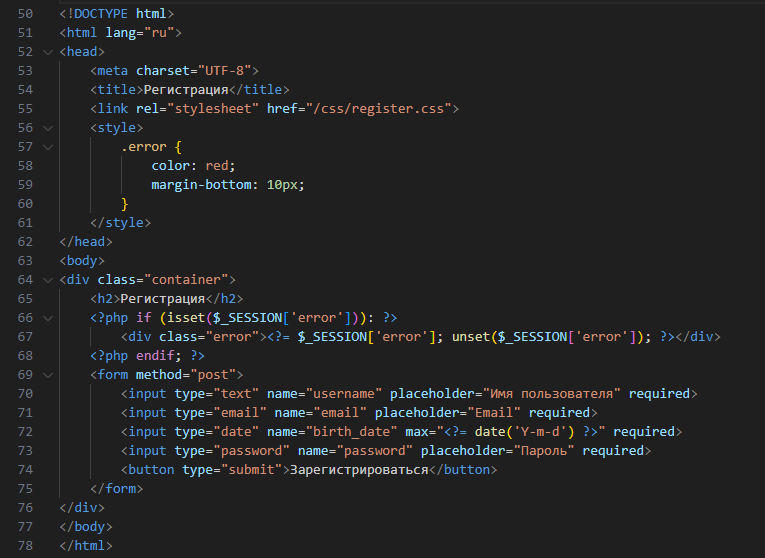


Рисунок 2.6.2 – Реализация формы регистрации

HTML-часть состоит из формы регистрации, содержащей поля для ввода имени пользователя, email, даты рождения и пароля. Поля email и password используются для ввода соответствующих данных пользователя, а поле birth\_date представляет собой элемент календаря для выбора даты рождения пользователя. Все поля обозначены как обязательные (required). В случае ошибки, сообщение об ошибке отображается под формой. После успешной регистрации, пользователь перенаправляется на страницу авторизации. Стилевое оформление страницы осуществляется с помощью подключенного CSS-файла register.css, дополнительно стили для блока ошибок (.error) определены внутри <style> тегов. В целом, код обеспечивает функциональную и удобную форму регистрации с проверкой уникальности данных и обработкой ошибок. Ознакомиться с кодом можно на рисунке 2.6.2.

Рассматривать файлы стилей будет излишним, так как он не будет особо отличаться от основных css файлов, так как всё выполняется в одном стиле. Можно сказать, мы успешно справились с первоочередной задачей. Посмотреть, как выглядит страница можно на рисунке 2.6.3.

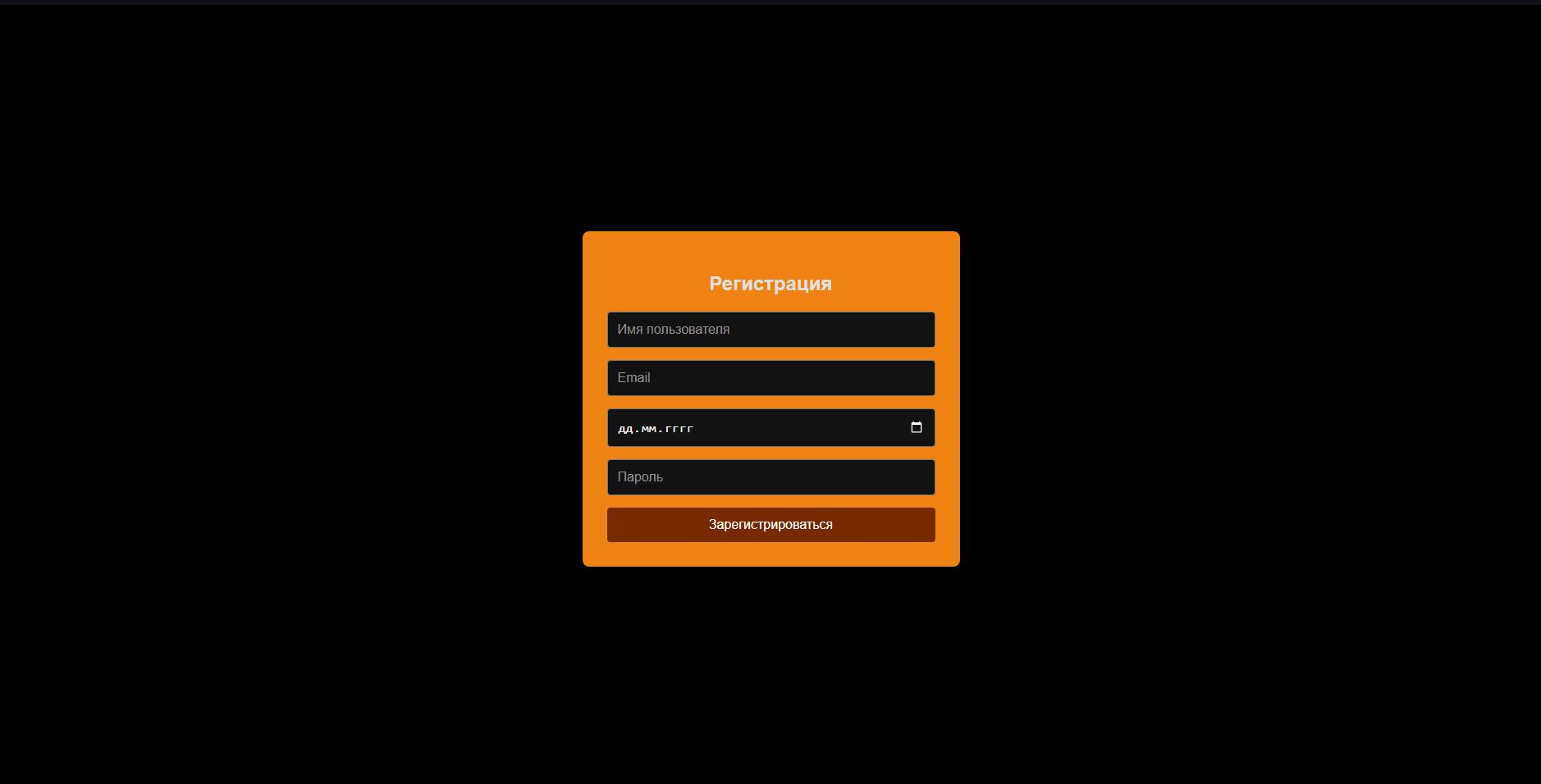


Рисунок 2.6.3 – Общий вид страницы «Регистрация»

* 1. **Личный кабинет**

Ранее мы уже разработали форму регистрации и авторизации, после этого у пользователя в шапке странице появляется вкладка «Мой профиль» (Рисунок 2.7.1).

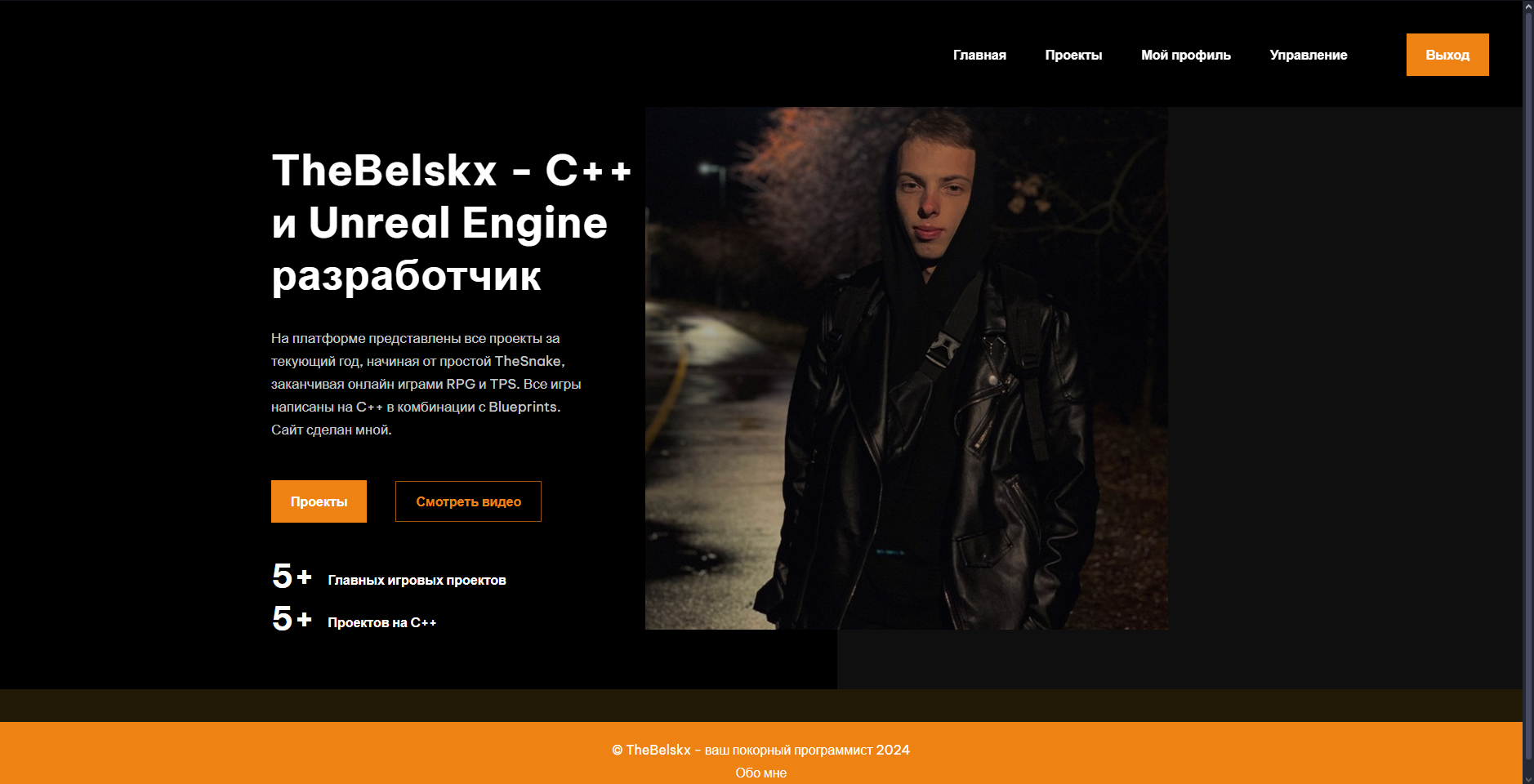
****

Рисунок 2.7.1 – Общий вид страницы авторизованного пользователя

Согласно требованиям проекта, необходимо обеспечить пользователям возможность редактировать собственные данные. Поэтому мы реализуем отображение информации о пользователе в личном кабинете и добавим функцию изменения пароля. Изменение логина и адреса электронной почты считается непрактичным.

Листинг PHP и HTML кода создания личного кабинета:

<?php

session\_start();

require 'db\_connection.php';

if (!isset($\_SESSION['user\_id'])) {

    header("Location: login.php");

    exit();

}

$user\_id = $\_SESSION['user\_id'];

$stmt = $pdo->prepare("SELECT \* FROM users WHERE id = ?");

$stmt->execute([$user\_id]);

$user = $stmt->fetch();

$successMessage = '';

$errorMessage = '';

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['old\_password']) && isset($\_POST['new\_password'])) {

    $oldPassword = $\_POST['old\_password'];

    $newPassword = $\_POST['new\_password'];

    if (password\_verify($oldPassword, $user['password'])) {

        $hashedNewPassword = password\_hash($newPassword, PASSWORD\_DEFAULT);

        $stmt = $pdo->prepare("UPDATE users SET password = ? WHERE id = ?");

        $stmt->execute([$hashedNewPassword, $user\_id]);

        $successMessage = "Пароль успешно изменен!";

    } else {

        $errorMessage = "Неверный старый пароль!";

    }

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Мой профиль</title>

    <link rel="stylesheet" href="css/profile.css">

</head>

<body>

<div class="container">

    <h2>Профиль пользователя</h2>

    <?php if ($successMessage): ?>

        <div class="success-message"><?php echo $successMessage; ?></div>

    <?php endif; ?>

    <?php if ($errorMessage): ?>

        <div class="error-message"><?php echo $errorMessage; ?></div>

    <?php endif; ?>

    <a href="index.php" class="button">На главную</a> <br><br>

    <p>Имя: <?= htmlspecialchars($user['username']); ?></p>

    <p>Email: <?= htmlspecialchars($user['email']); ?></p>

    <p>Дата рождения: <?= htmlspecialchars($user['birth\_date']); ?></p>

    <p>Время регистрации: <?= htmlspecialchars($user['created\_at']); ?></p>

    <h3>Изменить пароль</h3>

    <form method="post">

        <label for="old\_password">Старый пароль:</label>

        <input type="password" id="old\_password" name="old\_password" required><br>

        <label for="new\_password">Новый пароль:</label>

        <input type="password" id="new\_password" name="new\_password" required><br>

        <button type="submit">Изменить пароль</button>

    </form>

</div>

</body>

</html>

В PHP-части кода проверяет авторизацию пользователя и извлекает его данные из базы данных. При отправке формы изменения пароля, проверяется верность старого пароля с помощью password\_verify, и при успешной проверке новый пароль хэшируется и записывается в базу данных. Выводится сообщение об успехе или ошибке.

В HTML-части кода отображает информацию о пользователе (имя, email, дата рождения, дата регистрации), форму для изменения пароля и ссылку на главную страницу. В зависимости от результата изменения пароля отображаются соответствующие сообщения. Рассмотреть внешний вид страницы можно на рисунке 2.7.2.

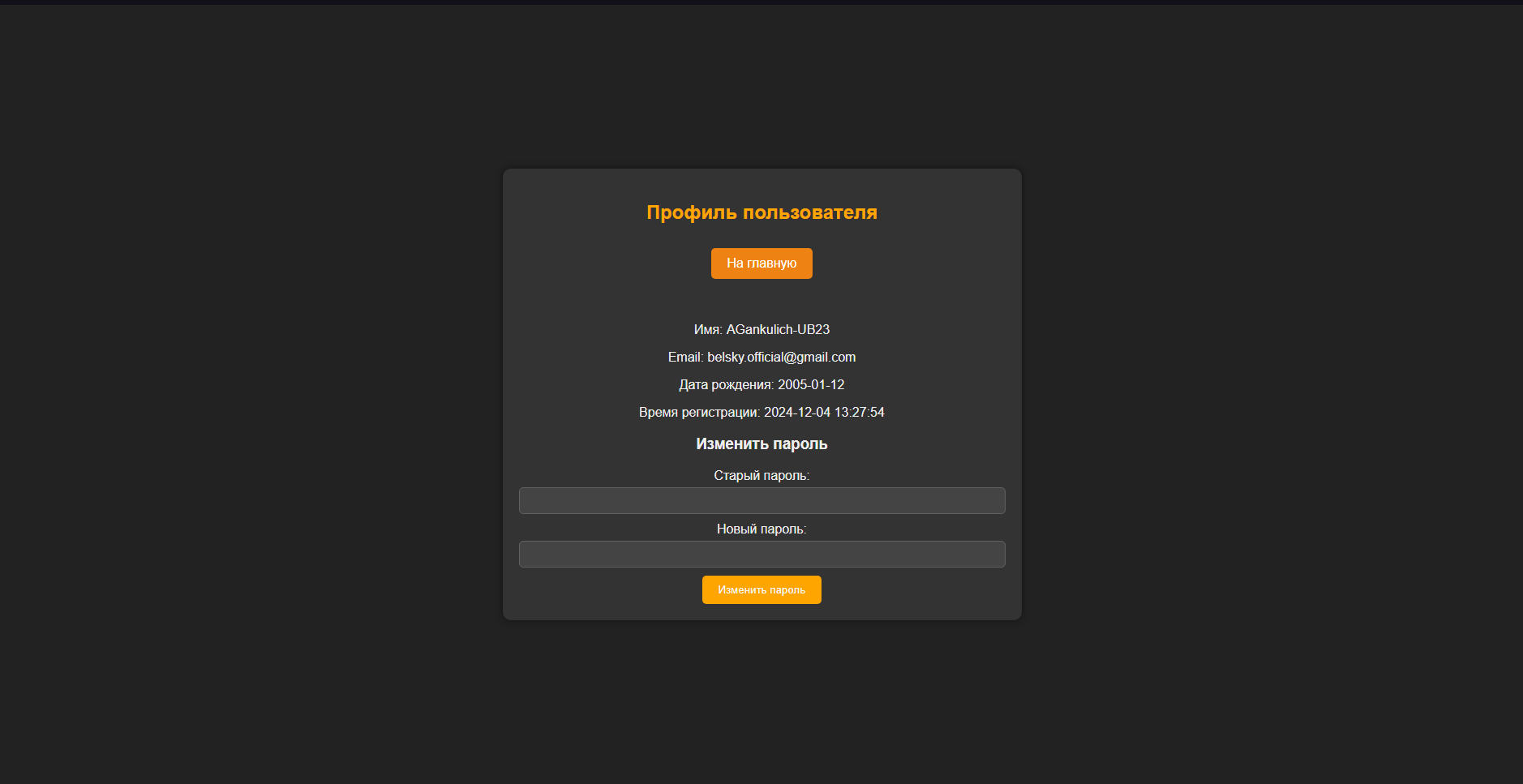


Рисунок 2.7.2 – Общий вид страницы «Мой профиль»

* 1. **Управление профилями**

Для предоставления административных прав необходимо изменить роль пользователя AGankulich-UB23 в базе данных через phpMyAdmin [8]. Это позволит управлять профилями пользователей (Рисунок 2.8.1).

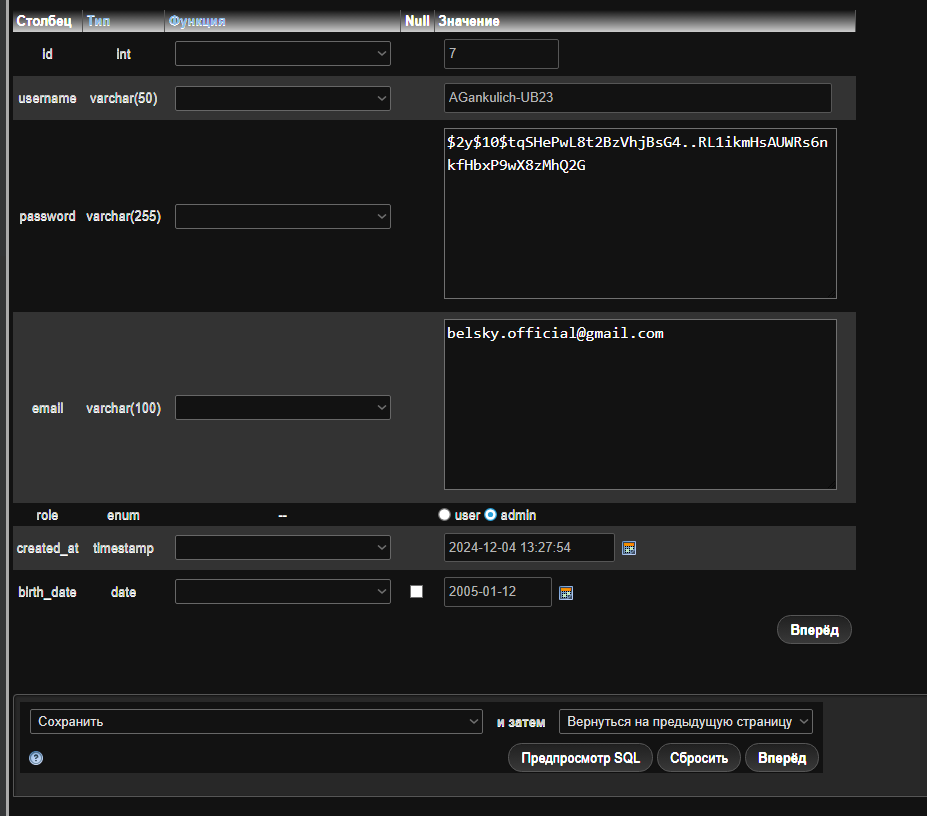


Рисунок 2.8.1 – Установка прав администратора для AGankulich-UB23

Выполним реализацию базовых возможностей администратора: изменять роль, удалять пользователя, чтобы не вмешиваться в личные данные пользователей и не нарушать тем самым этические нормы.

Листинг PHP и HTML кода для управления пользователями.

<?php

session\_start();

require 'db\_connection.php';

if (!isset($\_SESSION['user\_id']) || $\_SESSION['role'] !== 'admin') {

    header("Location: index.php");

    exit();

}

$currentUserId = $\_SESSION['user\_id'];

$stmt = $pdo->query("SELECT \* FROM users");

$users = $stmt->fetchAll();

if (isset($\_POST['delete'])) {

    $userIdToDelete = $\_POST['user\_id'];

    if ($userIdToDelete != $currentUserId) {

        $deleteStmt = $pdo->prepare("DELETE FROM users WHERE id = ?");

        $deleteStmt->execute([$userIdToDelete]);

    } else {

        echo "<script>alert('Вы не можете удалить своего собственного аккаунт.');</script>";

    }

    header("Location: admin.php");

    exit();

}

if (isset($\_POST['edit'])) {

    $userIdToEdit = $\_POST['user\_id'];

    $newRole = $\_POST['role'];

    if ($userIdToEdit != $currentUserId) {

        $updateStmt = $pdo->prepare("UPDATE users SET role = ? WHERE id = ?");

        $updateStmt->execute([$newRole, $userIdToEdit]);

    } else {

        echo "<script>alert('Вы не можете изменить свою собственную роль.');</script>";

    }

    header("Location: admin.php");

    exit();

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Управление пользователями</title>

    <link rel="stylesheet" href="css/admin.css">

</head>

<body>

    <div class="container">

        <a href="/index.php" class="btn-primary banner\_\_link">Главная</a><br>

        <h2 style="text-align: center;">Управление пользователями</h2>

        <table class="user-table">

            <thead>

                <tr>

                    <th>ID</th>

                    <th>Имя</th>

                    <th>Email</th>

                    <th>Роль</th>

                    <th>Действия</th>

                </tr>

            </thead>

            <tbody>

                <?php foreach ($users as $user): ?>

                    <tr>

                        <td><?= htmlspecialchars($user['id']); ?></td>

                        <td><?= htmlspecialchars($user['username']); ?></td>

                        <td><?= htmlspecialchars($user['email']); ?></td>

                        <td><?= htmlspecialchars($user['role']); ?></td>

                        <td>

                            <form action="" method="post" style="display:inline;">

                                <input type="hidden" name="user\_id" value="<?= htmlspecialchars($user['id']); ?>">

                                <select name="role">

                                    <option value="user" <?= $user['role'] === 'user' ? 'selected' : ''; ?>>Пользователь</option>

                                    <option value="admin" <?= $user['role'] === 'admin' ? 'selected' : ''; ?>>Администратор</option>

                                </select>

                                <button type="submit" name="edit">Изменить роль</button>

                            </form>

                            <form action="" method="post" style="display:inline;">

                                <input type="hidden" name="user\_id" value="<?= htmlspecialchars($user['id']); ?>">

                                <button type="submit" name="delete" onclick="return confirm('Вы уверены, что хотите удалить этого пользователя?');">Удалить</button>

                            </form>

                        </td>

                    </tr>

                <?php endforeach; ?>

            </tbody>

        </table>

    </div>

</body>

</html>

Код реализует административную панель для управления пользователями. PHP-часть проверяет авторизацию администратора. Затем извлекаются все пользователи из базы данных. При отправке формы (изменение роли или удаление пользователя), проверяется, что администратор не пытается изменить или удалить свой собственный аккаунт. В случае успешного изменения или удаления, происходит перенаправление на ту же страницу. Используются подготовленные запросы для изменения данных в базе данных.

HTML-часть отображает таблицу со списком пользователей, включая их ID, имя, email и роль. Для каждого пользователя предоставляются кнопки для изменения роли и удаления. Форма изменения роли содержит выпадающий список с вариантами «Пользователь» и «Администратор». Подтверждение удаления пользователя выполняется с помощью javascript функции confirm.

В целом, код предоставляет простой, но функциональный интерфейс для управления пользователями с базовой защитой от несанкционированных действий. Разработав стили, можно считать, что панель администратора создана и готова к использованию (Рисунок 2.8.2).

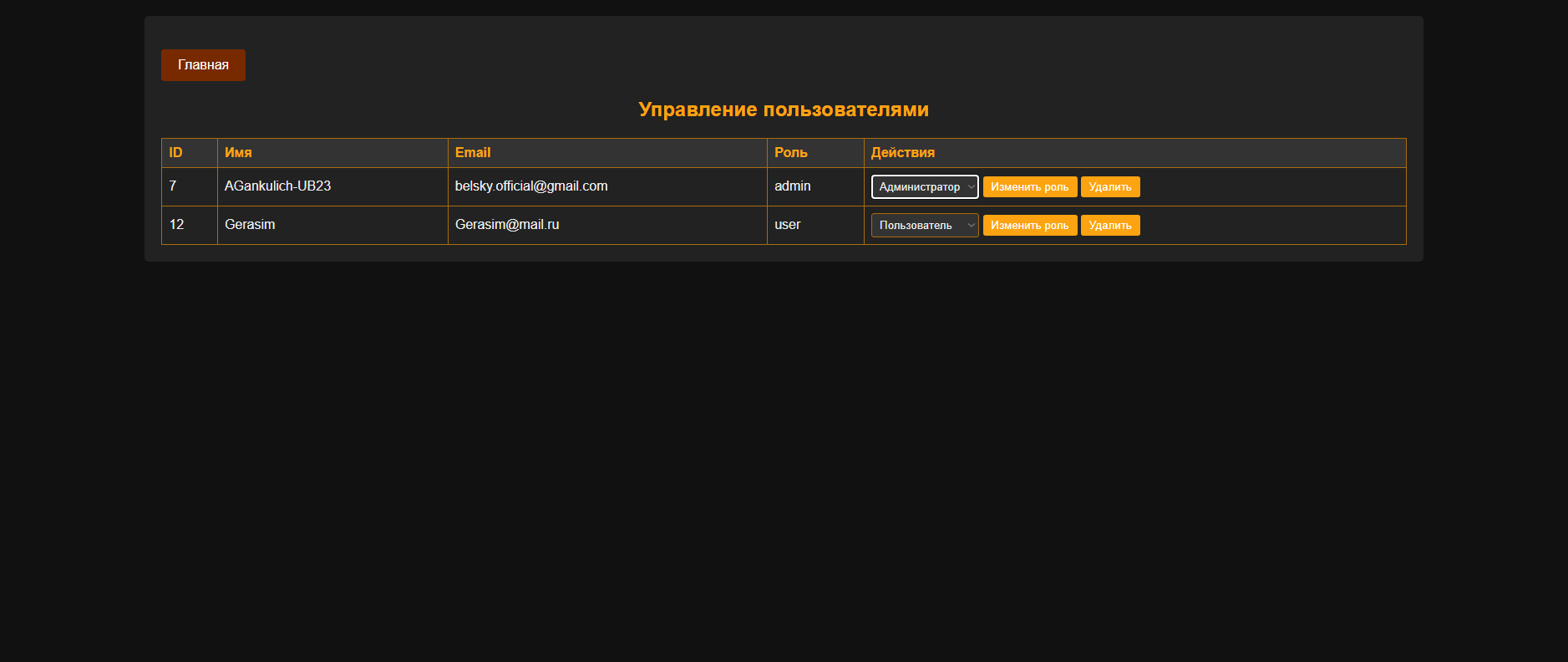


Рисунок 2.8.2 – Общий вид страницы «Управления пользователями»

* 1. **Просмотр проектов**

Для расширения функциональности сайта, необходимо реализовать доступ к проектам только для авторизованных пользователей на странице project.php. Авторизованные пользователи получат доступ к ссылкам на репозитории проектов. Неавторизованные пользователи будут информированы о необходимости авторизации. Это расширение дополняет существующие функции управления пользователями и личный кабинет.

Начальная часть (PHP): вначале устанавливается переменная $title со значением «Проекты». Затем подключается файл header.php, в котором и будет обновляться title с названием переменной соответствующей. Далее, запускается сессия PHP ( session\_start()). Затем проверяется, есть ли ID пользователя в сессии и сам ID не равен null. Результат этой проверки хранится в переменной $isLoggedIn. Эта переменная будет использоваться в дальнейшем для управления доступом к определенному контенту страницы (Рисунок 2.9.1).

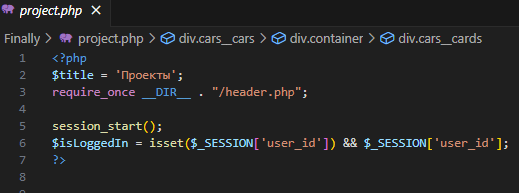


Рисунок 2.9.1 – Первоначальная проверка на авторизацию пользователя

Отображение заголовка и галереи: далее идёт HTML-разметка для секции заголовка cars\_\_title. Этот блок содержит текстовый контент «Все мои проекты за год» Заголовок cars\_\_title и контейнер container — это элементы оформления, вероятно, определенные в CSS. Затем, в cars\_\_cars содержится галерея проектов. В cars\_\_cards создаются карточки для каждого проекта. Обратите внимание, что данные проектов содержатся в массиве $projects, и PHP динамически генерирует разметку для каждой карточки проекта. Массив $projects содержит массив проектов с названием, путем к изображению и ссылкой. Он перебирается с помощью цикла foreach. Более детально можно ознакомиться на рисунке 2.9.2.

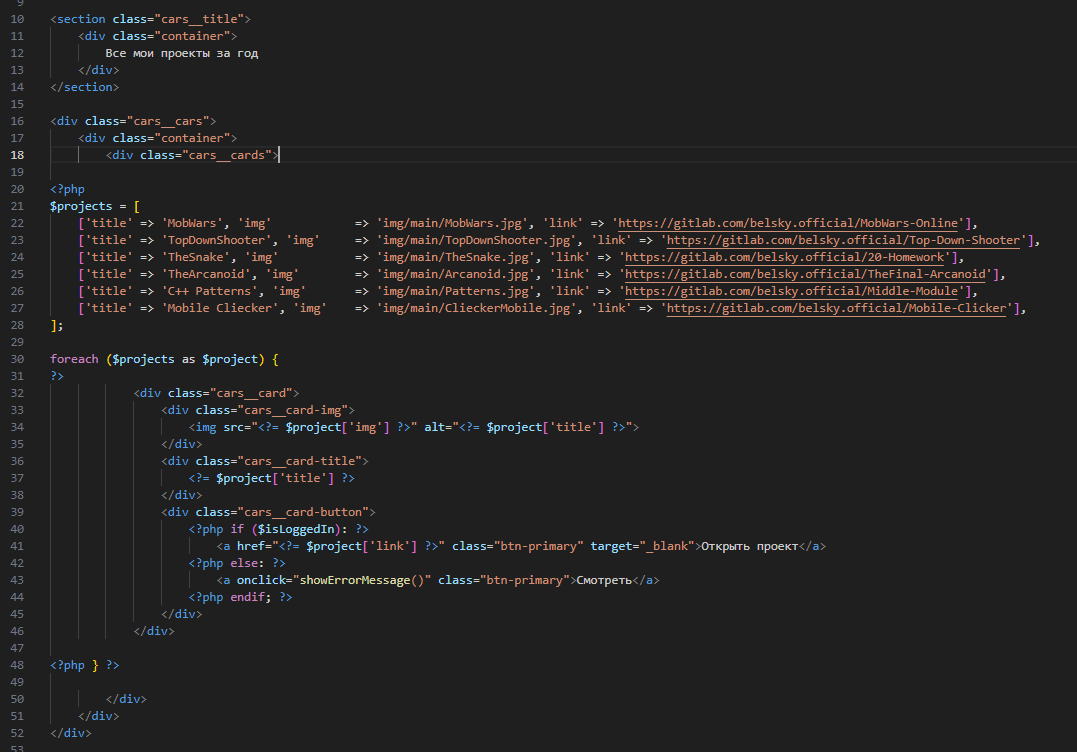


Рисунок 2.9.2 – Перебор проектов через цикл foreach для отображения

Управление доступом и JavaScript: Внутри цикла foreach, в блоке PHP, проверяется значение $isLoggedIn. Если пользователь авторизован, то выводится ссылка на страничку проекта ($project['link']). Если пользователь не авторизован, выводится кнопка “Смотреть”, которая, при нажатии, вызывает функцию showErrorMessage в JavaScript. Функция showErrorMessage выводит диалоговое окно с сообщением об ограничении доступа. В конце файла подключается footer.php, предполагая, что это внешний файл, содержащий подвал сайта. В самом нижнем блоке присутствует showErrorMessage — JavaScript-функция [5], вызывающая alert при неавторизованном доступе. Рассмотреть данный код можно на рисунке 2.9.3.

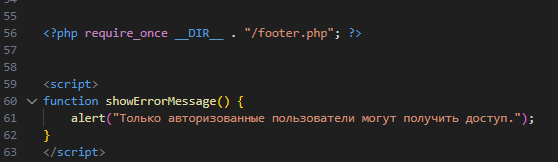


Рисунок 2.9.3 – Реализация сообщения об авторизации

Стилистика сайта также придерживается общей закономерности темного цвета с оранжевыми элементами. Задачи для авторизации и регистрации на данном этапе закончены, есть разграничения функционала между пользователем авторизованным и неавторизованным. Администратор может управлять пользователями прям изнутри платформы. Проект движется к завершающей стадии после данного этапа. Рассмотреть общий вид раздела «Проекты» можно на рисунке 2.9.4.

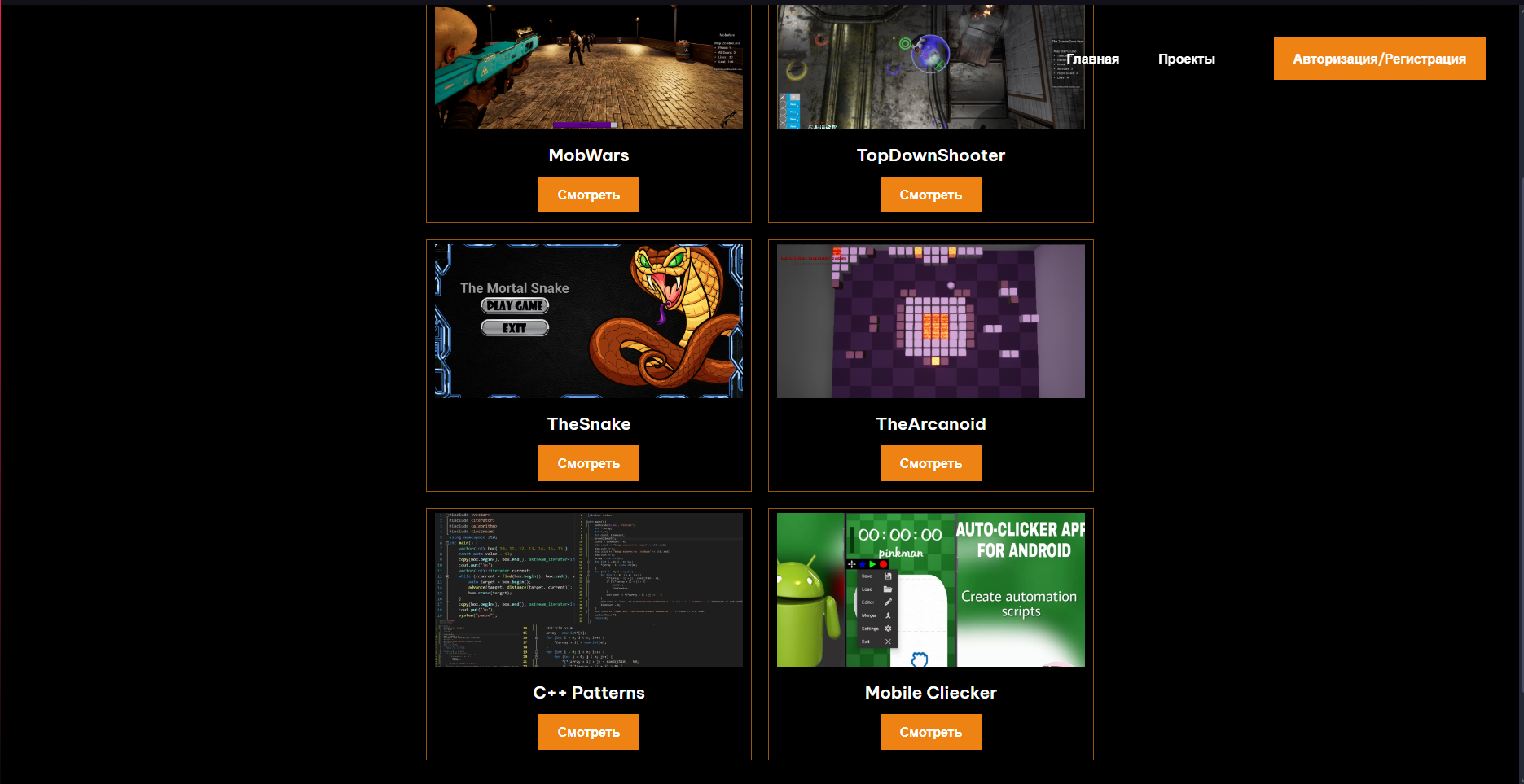


Рисунок 2.9.4 – Общий вид страницы «Проекты»

* 1. **Вкладка контактов – «Обо мне»**

Для повышения целостности и соответствия тематике «Портфолио», необходимо добавить на сайт раздел с информацией об авторе проектов. Доступ к этой информации также будет предоставлен через ссылку «Обо мне» в подвале главной страницы и страницы проектов (Рисунок 2.10.1).

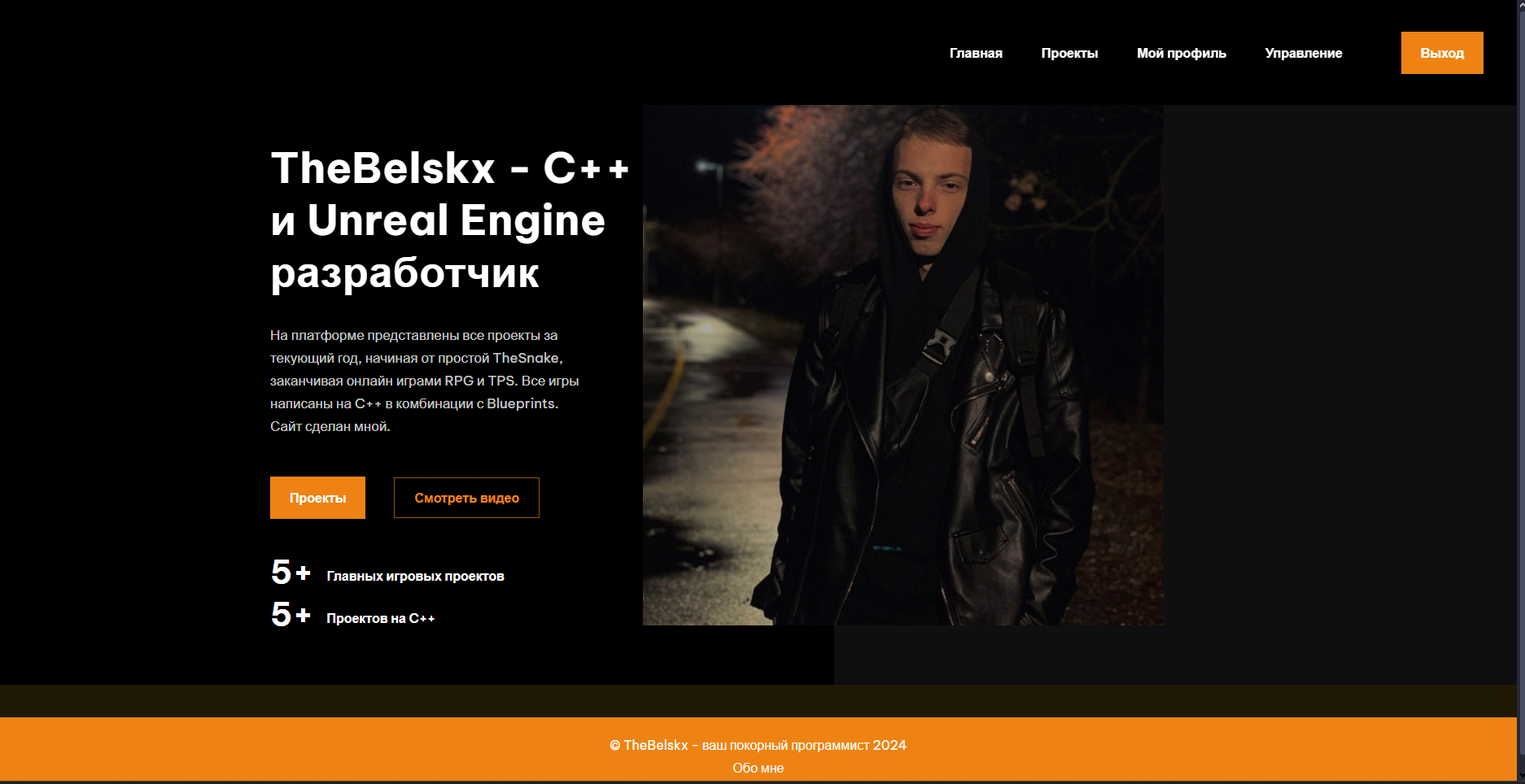


Рисунок 2.10.1 – доступ к «Обо мне» из футера главного меню

В данной вкладке нет сложных методов, функций, необходимо лишь придерживаться общей стилистике и предоставить доступ ко всем способами связи с автором данной работы.

Код представляет собой простую веб-страницу «“Обо мне», отображающую контактную информацию автора. В начале устанавливается кодировка UTF-8. HTML-разметка формирует страницу с заголовком «Обо мне» и блоками контактной информации: телефон, ссылки на профили ВКонтакте и Telegram, и email-адрес. Ссылки открываются в новой вкладке браузера (target="\_blank"). Также присутствует ссылка на главную страницу сайта. Внешний вид страницы определяется подключенным CSS-файлом contac.css, который выполнен в аналогичном стиле прошлых страниц. Весь контент страницы статический и не извлекается из базы.. Рассмотреть код на рисунке 2.10.2.



Рисунок 2.10.2 – Реализация вкладки «Обо мне».

Получилась достаточно красивая вкладка «Обо мне», содержащая всю нужную информацию для связи и соответствующая общей стилистике всего проекта (Рисунок 2.10.3).

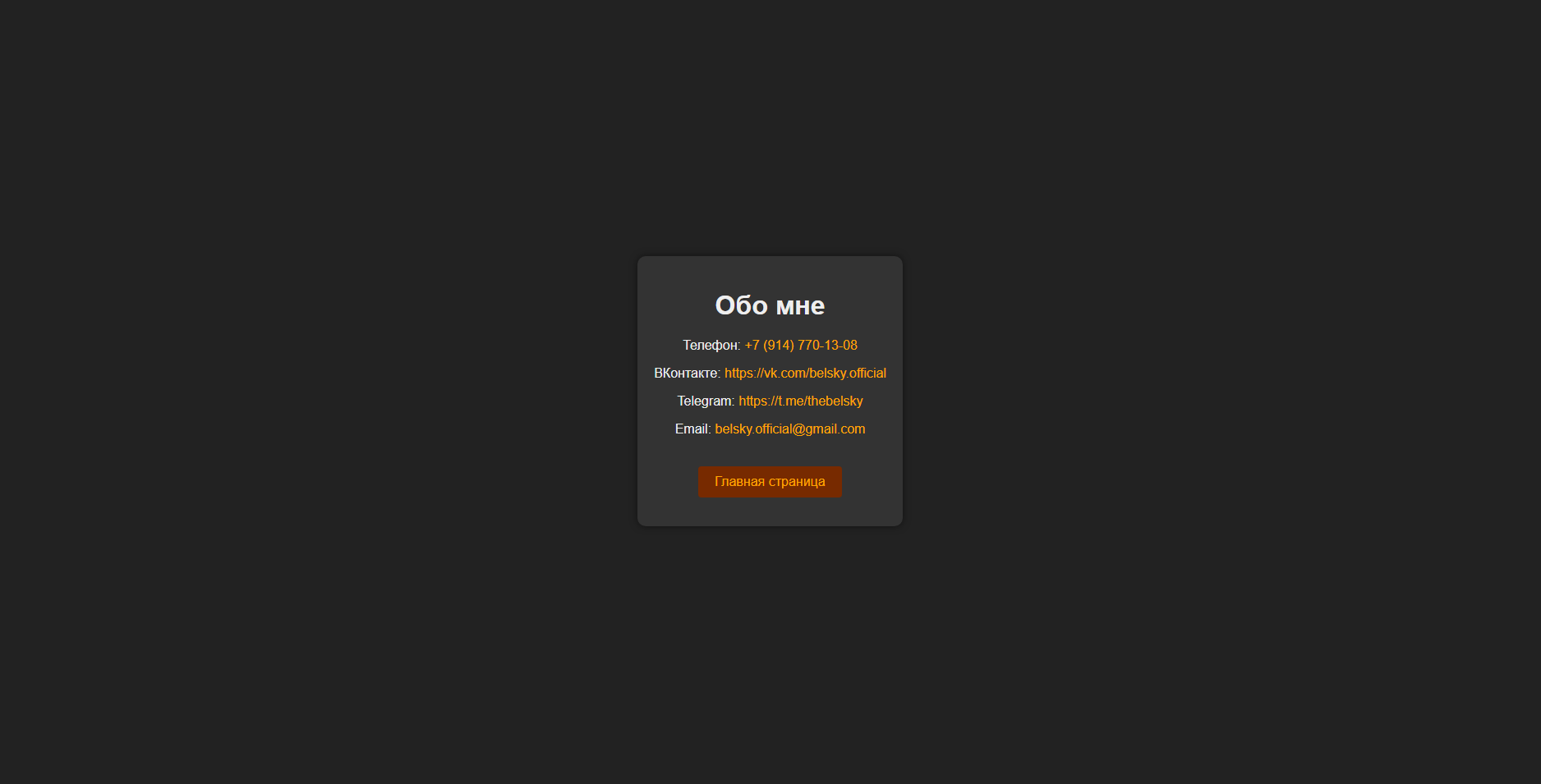


Рисунок 2.10.3 – Общий вид страницы «Обо мне»

* 1. **Вкладка выхода из аккаунта**

Возможность выхода из аккаунта это лишь реализация дополнительного функционала, который никак не будет явно отображаться в виде страницы. Но тем не менее каждый должен иметь доступ ко входу и выхода из своего аккаунта на платформе.

На рисунке 2.11.1 предоставлен PHP-код для выхода пользователя из системы. Сначала он запускает сессию PHP (session\_start()), что необходимо для доступа к данным текущей сессии. Затем он полностью уничтожает текущую сессию (session\_destroy()), удаляя все сохраненные в ней данные пользователя, такие как ID пользователя, роль и другую информацию об авторизации. После уничтожения сессии, код перенаправляет пользователя на главную страницу сайта (index.php) с помощью header("Location: index.php");. Оператор exit(); прекращает дальнейшее выполнение скрипта после перенаправления. Таким образом, этот код обеспечивает безопасный выход пользователя из системы, удаляя все данные сессии и перенаправляя его на главную страницу.

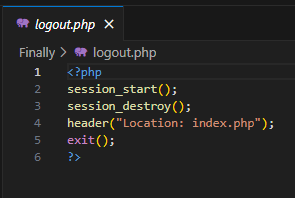


Рисунок 2.11.1 – Реализация системы выхода из аккаунта

# **Заключение**

В рамках проекта было создано веб-приложение из семи страниц, посвященных портфолио автору проекта. Это включает главную страницу index.php, личный кабинет каждого участника – profile.php, страницу управления аккаунтами для администратора – admin.php, страницу для авторизации – login.php, а также страницу для регистрации – register.php и страницу, касаемо связи с автором – contact.php. Помимо этого, были дополнительно реализованы отдельно для удобства файлы – header.php, footer.php, logout.php и db\_connection.php. Для придерживания общей стилистике кода были подключены css файлы – main.css, admin.css, contact.css, profile.css, project.css, register.css, login.css.

Практически каждая страница платформы взаимодействует с базой данных "management\_users" и связанной с ней таблицей "users". На необходимых страницах происходит проверка аутентификации пользователя.

Администратор имеет возможность редактировать данные пользователей, а именно выдавать необходимые роли и удалять их через панель администратора. Доступ к этой панели предоставлен только пользователю с логином «AGankulich-UB23» внутри базы данных через phpMyAdmin и MySQL.

Это веб-приложение было разработано с использованием HTML, CSS, JavaScript, PHP и MySQL, также дополнительно были задействованы такие платформы, как phpMyAdmin, OpenServer. Цель проекта достигнута. В проекте предусмотрена системы авторизации и регистрации пользователей, зарегистрированные пользователи имеет право редактировать свои данные и получают доступ к репозиториям проектов автора портфолио. Администраторы получают права редактирования пользователей и удалять их, а также неавторизованные пользователи ограничены в правах на платформе. Весь проект выдержан в одном общем стилистическом виде, не нарушая общих концепций. Проект закончен.

# **Список использованных источников**

1. Главная страница OpenServer // operserver URL: <https://ospanel.io> (дата обращения 01.12.2024).
2. Главная страница phpMyAdmin // phpMyAdmin URL: <https://www.phpmyadmin.net/downloads/> (дата обращения 01.12.2024).
3. Документация MySQL. // metanit URL: <https://metanit.com/sql/mysql/> (дата обращения: 01.12.2024).
4. Руководство по HTML и CSS // metanit URL: <https://metanit.com/web/html5/> (дата обращения: 03.12.2024).
5. Руководство по JS // metanit URL: <https://metanit.com/web/javascript/> (дата обращения: 03.12.2024).
6. Руководство по PHP // metanit URL: <https://metanit.com/php/tutorial/> (дата обращения: 04.12.2024).
7. PHP, MySQL, JavaScript, CSS и HTML5 / Робин Никсон 6-е издание, 2023. – 832 с. (дата обращения: 04.12.2024).
8. **PHP и MySQL. Разработка веб-приложений** / Д. Колисниченко 6-ое издание, 2017 – 640 с. (дата обращения 04.12.2024).