|  |  |  |
| --- | --- | --- |
|  | A picture containing text  Description automatically generated |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: Разработка серверных частей интернет-ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть веб-приложения «Автосалон»

Студент: Фомичев Роман Алексеевич

Группа: ИКБО-20-21

Работа представлена к защите \_\_\_\_\_\_\_\_\_\_\_\_(дата) \_\_\_\_\_\_\_\_\_\_ /Фомичев Р.А./

(подпись и ф.и.о. студента)

Руководитель: старший преподаватель Синицын Анатолий Васильевич

Работа допущена к защите \_\_\_\_\_\_\_\_\_\_\_\_(дата) \_\_\_\_\_\_\_\_\_\_ /Синицын А.В./

(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

2023 г.

**РЕФЕРАТ**

Отчёт 29 с., 13 рис., 1 табл., 21 источн.

ЧИСТАЯ АРХИТЕКТУРА, MVC, ИНТЕРНЕТ-МАГАЗИН

Объектом разработки является автосалон.

Цель работы – разработка архитектуры и серверной части приложения автосалон.

В ходе работы был проведён анализ предметной области и обзор сайтов с аналогичной тематикой. Методом сравнительного анализа определены необходимые сущности и возможности, которые должны быть включены в приложение.

Рассмотрен процесс разработки архитектуры, выбор необходимых технологий для разработки и процесс создания приложения.

Результатом является приложение, отвечающее стандартам RESTful, позиционирующий готовое API для интернет-магазина детских товаров.

Report 29 p., 13 fig., 1 table, 21 sources.

CLEAR ARCHITECTURE, MVC, ONLINE STORE

The object of development is an autoshowroom.

The purpose of the work is to develop the architecture and server part of the authoshowroom.

In the course of the work, an analysis of the subject area and a review of sites with similar topics were carried out. By the method of comparative analysis, the necessary entities and capabilities that should be included in the application are determined.

The process of architecture development, the choice of necessary technologies for development and the process of creating an application are considered.

The result is an application that meets RESTful standards, positioning a ready-made API for an online store of children's toys.

**Аннотация**

В курсовой работе описывалось создание интернет-ресурса, на тему «Автосалон». Работа содержит анализ предметной области разрабатываемого интернет-ресурса, создание веб-страниц интернет-ресурса с использованием технологий php.

В введении обосновывается актуальность выбранной темы, определяется цель работы и задачи, подлежащие решению для её достижения, описываются объект и предмет исследования используемые методы и информационная база исследования, а также кратко характеризуется структура КР по разделам.

В основной части содержится материал, необходимый для достижения цели КР. Основная часть включает в себя общие сведения (в частности, наименование интернет-ресурса, перечисление прикладного программного обеспечения, необходимого для разработки и функционирования интернет- ресурса, а также названия языков и технологий, с помощью которых реализован интернет-ресурс), описание функционального назначения интернет-ресурса и его логической структуры, описание разработки и функций программного приложения. В заключении последовательно излагаются теоретические выводы, которые были сформулированы в результате выполнения данной курсовой работы.

Курсовая работа на 29 листах, содержит 13 рисунков, 21 использованный источник.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc152784242)

[1 ОБЩИЕ СВЕДЕНИЯ 8](#_Toc152784243)

[1.1 Обозначение и наименование интернет-ресурса 8](#_Toc152784244)

[1.2 Функциональное назначение 8](#_Toc152784245)

[2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc152784246)

[3 ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ 10](#_Toc152784247)

[3.1 Выбор архитектуры 10](#_Toc152784248)

[3.2 Выбор основных технологий 12](#_Toc152784249)

[3.3 Прикладное программное обеспечение, необходимое для разработки и функционирования приложения 14](#_Toc152784250)

[4 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА 15](#_Toc152784251)

[5 РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА 18](#_Toc152784252)

[5.1 Разработка слоя моделей 18](#_Toc152784253)

[5.2 Разработка слоя контроллеров 18](#_Toc152784254)

[5.3 Разработка слоя представлений 18](#_Toc152784255)

[6 ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ 20](#_Toc152784256)

[ЗАКЛЮЧЕНИЕ 25](#_Toc152784257)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc152784258)

[ПРИЛОЖЕНИЕ А 28](#_Toc152784259)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

БД – база данных;

СУБД – система управления базами данных;

URL – унифицированный указатель ресурса;

http – протокол передачи данных.

ВВЕДЕНИЕ

Сегодня информационные технологии играют важную роль во многих сферах, таких как: государство, общество и бизнес. Без них трудно представить организацию любого процесса, ведь современные технологии значительно упрощают, систематизируют и оптимизируют работу. Разработка современных приложений является сложной и многогранной задачей, требующей от разработчика глубокого понимания архитектуры, технологий и инструментов.

Целью данной курсовой работы является разработка серверной части веб-приложения «Автосалон» с использованием микросервисов, архитектурных паттернов проектирования MVC и современных технологий.

Для выполнения поставленной цели курсовой работы необходимо выполнить следующие пункты:

* провести анализ предметной области разрабатываемого веб-приложения;
* обосновать выбор технологий разработки веб-приложения;
* разработать архитектуру веб-приложения на основе выбранного паттерна проектирования;
* реализовать слой серверной логики веб-приложения с применением выбранной технологии;
* реализовать слой логики базы данных;
* разработать слой клиентского представления веб-приложения;
* создать презентацию по выполненной курсовой работе;

# ОБЩИЕ СВЕДЕНИЯ

## Обозначение и наименование интернет-ресурса

Темой разработанной серверной части веб-приложения интернет-магазина является «Автосалон». Разработанное приложение получило название – «Black Lightning».

## Функциональное назначение

Разработанное приложение имеет прикладной характер и может послужить серверной частью для настоящего автосалона. Приложение для незарегистрированных пользователей включает в себя такие функции как просмотр главной страницы сайта, информации об автосалоне, способов связи, регистрация аккаунта и авторизация. Для зарегистрированных пользователей имеется следующий функционал: просмотр каталога автомобилей, а также просмотр информации профиля.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Для анализа предметной области было проведено исследование в области аналогичных приложений и были выявлены их преимущества и недостатки, представленные в таблице 1.

Таблица 1 – Сравнение аналогичных приложений

|  |  |  |  |
| --- | --- | --- | --- |
|  | Yellow Bird | Dana Bros | River North Collision Repair |
| Наличие каталога товаров | + | + | + |
| Наличие цен на странице каталога | + | + | - |
| Регистрация аккаунта | + | - | + |
| Возможность просмотра профиля | - | + | - |

На основе информации из таблицы 1 можно выделить несколько основных функций, которые должны присутствовать в приложении, такие как, каталог товаров, возможность создания аккаунта и просмотра информации о нем. Одним из минусов аналогичных приложений является отсутствие каталога с актуальными товарами и ценами, а также возможности зарегистрироваться.

На основе полученных данных были составлены функциональные требования для приложения. Приложение для незарегистрированных пользователей включает в себя такие функции как просмотр главной страницы сайта, информации об автосалоне, способов связи, регистрация аккаунта и авторизация. Для зарегистрированных пользователей имеется следующий функционал: просмотр каталога автомобилей, а также просмотр информации профиля.

# ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ

## Выбор архитектуры

Первым этапом выбора технологий является выбор архитектуры приложения. Рассмотрим несколько наиболее используемых.

Монолитная архитектура [1] – это классический подход, при котором все компоненты приложения (бизнес-логика, пользовательский интерфейс, база данных и т. д.) находятся в одной единственной программе или кодовой базе. Монолит обычно легко развертывается, однако может быть сложным для масштабирования и поддержки при увеличении размера и сложности проекта.

Микросервисная архитектура [2] – это подход, основанный на разделении приложения на составные части, называемые микросервисами. Каждый микросервис является небольшим, автономным модулем, который выполняет определенную функцию приложения. Микросервисы могут взаимодействовать друг с другом посредством API. Эта архитектура облегчает масштабирование, гибкость разработки и обновления отдельных компонентов, но может быть сложной для управления и требовать дополнительных усилий в области координации и управления межсервисными взаимодействиями.

Микросервисный монолит [3] – это гибридный подход, сочетающий черты монолитной и микросервисной архитектур. В этой модели весь функционал приложения разделен на изолированные модули (микросервисы), но эти модули все еще развертываются и работают внутри одной кодовой базы. Это позволяет извлечь выгоды микросервисной архитектуры, не готовя новые инфраструктурные компоненты для каждого сервиса.

В рамках интернет-магазина наиболее подходящей является микросервисная архитектура, так как она позволяет распределить нагрузку на разные сервисы, а в частности для отведения нагрузки от большого количества запросов авторизации на отдельный сервис.

Рассмотрим несколько популярных архитектур для использования внутри микросервисов.

Чистая архитектура (Clean Architecture) [4] – это концепция разработки программного обеспечения, предложенная Робертом Мартином. Она нацелена на создание модульных, гибких и легко поддерживаемых приложений. Архитектура должна:

* быть тестируемой;
* не зависеть от UI;
* не зависеть от БД, внешних фреймворков и библиотек.

Это достигается путем разделения системы на слои, и следования принципам SOLID [4]. Далее рассмотрим принципы SOLID.

Принцип единственной ответственности (Single Responsibility Principle - SRP) означает что каждый класс или модуль должен быть ответственным только за выполнение одной задачи. Это помогает создавать более когерентный код и упрощает его поддержку.

Принцип открытости/закрытости (Open/Closed Principle - OCP) предполагает, что код должен быть открытым для расширения, но закрытым для модификации. Вместо изменения существующего кода, мы должны добавлять новый код для изменения поведения.

Принцип подстановки Барбары Лисков (Liskov Substitution Principle - LSP) гласит что объекты подклассов должны быть способны заменить объекты своих суперклассов без изменения корректности программы. Это помогает поддерживать полиморфизм и избегать неожиданного поведения.

Принцип разделения интерфейса (Interface Segregation Principle - ISP) говорит о том, что клиенты не должны зависеть от методов, которые они не используют. Интерфейсы должны быть специфичными для клиентов, чтобы избежать ненужных зависимостей.

Принцип инверсии зависимостей (Dependency Inversion Principle - DIP) является одним из наиболее используемых. Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракций. Это помогает улучшить гибкость и переиспользуемость кода.

MVC (Model-View-Controller) [5] является популярным паттерном проектирования программного обеспечения, который широко используется при разработке приложений. Он помогает разделить логику приложения на три основных компонента: модель (Model), представление (View) и контроллер (Controller).

Модель (Model) представляет собой основную составляющую, отвечающую за обработку данных, бизнес-логику и взаимодействие с базой данных или другими хранилищами данных. Она содержит информацию о состоянии приложения и предоставляет методы для получения, обновления и удаления данных.

Представление (View) отвечает за отображение данных пользователю. Оно представляет информацию в удобном для пользователя виде, например, в виде HTML-страницы или графического интерфейса. Представление получает данные из модели и отображает их, а также принимает пользовательский ввод и передает его контроллеру для обработки.

Контроллер (Controller) является посредником между моделью и представлением. Он обрабатывает пользовательский ввод, принимая запросы от представления, и осуществляет соответствующие действия, взаимодействуя с моделью для обновления данных или получения необходимой информации. Затем контроллер обновляет представление, чтобы отобразить изменения пользователю.

MVC обеспечивает разделение ответственностей между компонентами, что упрощает разработку, поддержку и тестирование приложений. Он также способствует повторному использованию кода, поскольку разные модели могут использоваться с одним и тем же представлением и контроллером.

Для разработки архитектуры ПП был выбран паттерн проектирования MVC и подход к проектированию Clean Architecture.

## Выбор основных технологий

В качестве языка программирования для разработки серверной части веб-приложения был выбран компилируемый объектно-ориентированный язык программирования PHP [6], который отлично себя зарекомендовал в разработке серверных частей программный продуктов. Основополагающими критериями выбора данного языка программирования были следующие факты:

* PHP является одним из самых популярных языков программирования и имеет большую сообщество разработчиков, то есть вы можете легко найти поддержку, руководства и библиотеки, которые помогут вам в разработке;
* PHP – кросс-платформенный язык программирования, что позволяет выполнять код на различных операционных системах без изменений, то есть разработанный бэкенд будет работать на различных платформах, что очень удобно для масштабируемости и переносимости приложений;
* PHP предлагает множество встроенных функций безопасности, что делает его привлекательным выбором для разработки бэкенда, например, Java обеспечивает защиту от некоторых типов атак, таких как переполнение буфера или неправильное освобождение памяти.
* PHP предоставляет множество инструментов и подходов для масштабирования приложений, что позволяет разрабатывать распределенные системы, использовать многопоточность и создавать масштабируемые архитектуры с помощью PHP;
* PHP имеет огромное количество библиотек, фреймворков и инструментов, которые облегчают разработку бэкенда.

В качестве базы данных была выбрана MySQL [[8](#и11)]. MySQL – это мощная объектно-реляционная система управления базами данных (СУБД), которая поддерживает широкий диапазон функций и возможностей, таких как полнотекстовый поиск, гео-пространственные запросы, транзакции, репликацию, хранимые процедуры и многое другое. Она является одной из наиболее распространенных СУБД с открытым исходным кодом и обладает активным сообществом разработчиков. MySQL используется как для небольших, так и для крупных проектов в различных отраслях, таких как финансы, телекоммуникации, интернет-технологии и другие.

Для взаимодействия между серверной и клиентской частями приложения был использован протокол HTTP, а также формат передачи данных JSON. JSON является универсальным форматом, который поддерживается большинством языков программирования и позволяет передавать данные в удобном для обработки виде.

## Прикладное программное обеспечение, необходимое для разработки и функционирования приложения

В качестве среды разработки, предназначенного для работы с PHP был выбран редактор с функциями IDE – PHPStorm. PHPStorm предоставляет широкий набор инструментов для удобства разработки, таких как: автоматическое завершение кода, рефакторинг кода, анализ кода, поддержка технологий, интеграция с системами контроля версий, встроенный отладчик. Кроме того, PHPStorm обладает высоким уровнем настраиваемости и гибкости, что позволяет настроить его под любые потребности разработчика.

# РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ НА ОСНОВЕ ВЫБРАННОГО ПАТТЕРНА

В предыдущей главе для разработки приложения был выбран паттерн MVC.

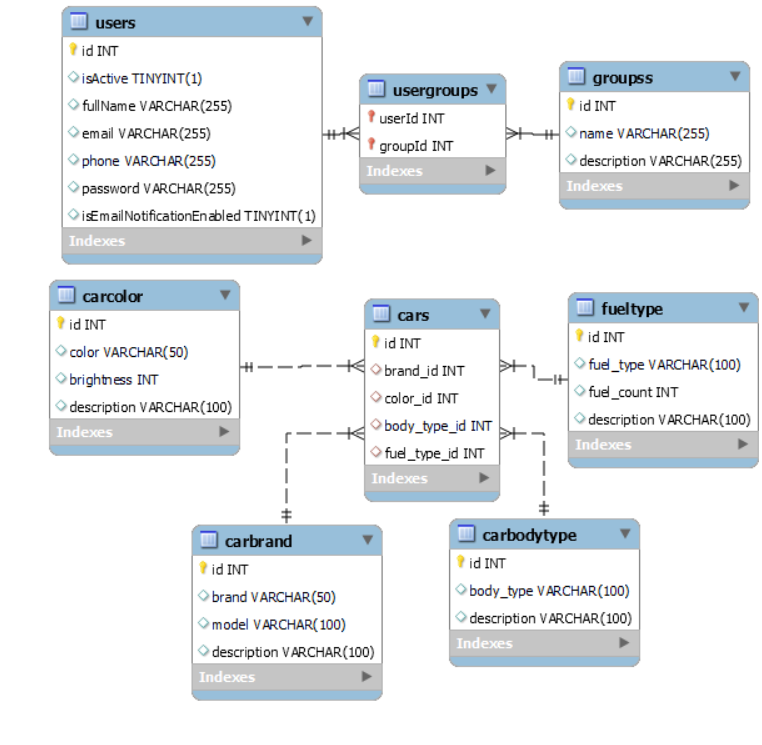
В паттерне MVC, можно выделить следующие слои:

* представление (View),
* контроллер (Controller),
* модель (Model).

Слой представления отвечает за отображение данных пользователю и взаимодействие с ним. Он фактически визуализирует состояние логики клиенту. В представлении происходит обработка пользовательского ввода и отображение данных из моделей.

Слой контроллера является промежуточным звеном между представлением и другими компонентами системы. Они отвечают за обработку пользовательского ввода и взаимодействие с сервисами для выполнения бизнес-логики. Контроллеры принимают запросы от представления, обрабатывают и возвращают данные обратно в представление.

Также в ходе разработки программного продукта, была спроектирована схема базы данных. Проектирование базы данных осуществлялось посредством проектирования диаграммы в нотации ERD [16], которая изображена на рисунке 1. ERD помогает разработчикам и аналитикам данных лучше понять структуру информации в базе данных и спроектировать эффективные модели данных. Кроме того, она может служить основой для создания реляционных схем и дальнейшего проектирования баз данных.



1. ERD-диаграмма

В БД входит 8 таблиц:

1. users (таблица пользователей). Таблица, предоставляющая список пользователей. Колонка id нужна для уникального идентифицирования пользователя. Колонка isActive нужна информации об активированности аккаунта. Колонка fullName нужна для имени пользователя. Колонка email нужна для почты пользователя. Колонка phone нужна для номера телефона пользователя. Колонка password нужна для пароля от аккаунта.
2. groupss (таблица групп). Таблица, предоставляющая список групп. Колонка id нужна для уникального идентифицирования группы. Колонка name нужна для названия группы. Колонка description нужна для описания группы.
3. usergroups (таблица групп пользователя). Таблица, предоставляющая список групп пользователя. Колонка userId нужна для идентифицирования пользователя. Колонка groupId нужна для идентифицирования группы.
4. carcolor (таблица цвета). Таблица, предоставляющая список цветов автомобилей. Колонка id нужна для уникального идентифицирования цвета. Колонка color нужна для цвета в формате RGB. Колонка brightness нужна для яркости цвета. Колонка description нужны для описания цвета.
5. carbrand (таблица бренда). Таблица, предоставляющая список брендов автомобилей. Колонка id нужна для уникального идентифицирования бренда. Колонка brand нужна для названия бренда. Колонка model нужна для названия модели. Колонка description нужна для описания бренда.
6. carbodytype (таблица кузова). Таблица, предоставляющая список кузовов. Колонка id нужна для уникального идентифицирования кузова. Колонка body\_type нужна для названия типа кузова. Колонка description нужна для описания типа кузова.
7. fueltype (таблица топлива). Таблица, предоставляющая список характеристик топливной системы. Колонка id нужна для уникального идентифицирования топливной системы. Колонка fuel\_type нужна для типа топлива. Колонка fuel\_count нужна для объема топливного бака. Колонка description нужна для описания топливной системы.
8. cars (таблица автомобилей). Таблица, предоставляющая список автомобилей. Колонка id нужна для уникального идентифицирования автомобиля. Колонка brand\_id нужна для идентифицирования брэнда. Колонка color\_id нужна для идентифицирования цвета. Колонка body\_type\_id нужна для идентифицирования типа бренда. Колонка fuel\_type\_id нужна для идентифицирования типа топливной системы.

# РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-РЕСУРСА

## Разработка слоя моделей

В процессе разработки интернет-проекта были созданы модели данных, которые описывают основные сущности предметной области проекта. Каждая из этих моделей имеет свои поля и методы, которые описывают ее поведение в системе. На рисунке 2 представлен список разработанных моделей.

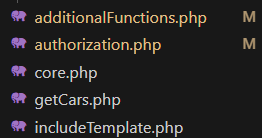


1. Список разработанных моделей

## Разработка слоя контроллеров

Контроллеры в PHP используются для организации и управления бизнес-логикой веб-приложения. Они отвечают за обработку входящих запросов от пользователей, взаимодействие с моделями данных и формирование ответов, которые отображаются на веб-странице.

На рисунке 3 представлен список разработанных контроллеров.



1. Разработанные контроллеры

## Разработка слоя представлений

В процессе разработки слоя представления были созданы веб-страницы. На рисунке 4 изображен список разработанных представлений.

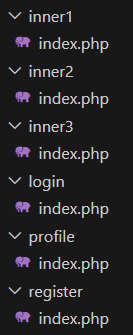


Рисунок 4 – Разработанные представления

Слой представлений разработан с использование HTML5, CSS3, JS. С интерфейсом разработанной фронтенд части приложения можно ознакомиться в главе 6.

# ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Тестирование программного продукта является важной частью разработки, поскольку оно позволяет выявить ошибки и дефекты, а также гарантирует его качество и надежность перед выпуском.

На данном этапе развития приложения было проведено мануальное тестирование приложения.

На рисунке 5 изображено тестирование регистрации. Заполняется форма регистрации с email, который уже существует.

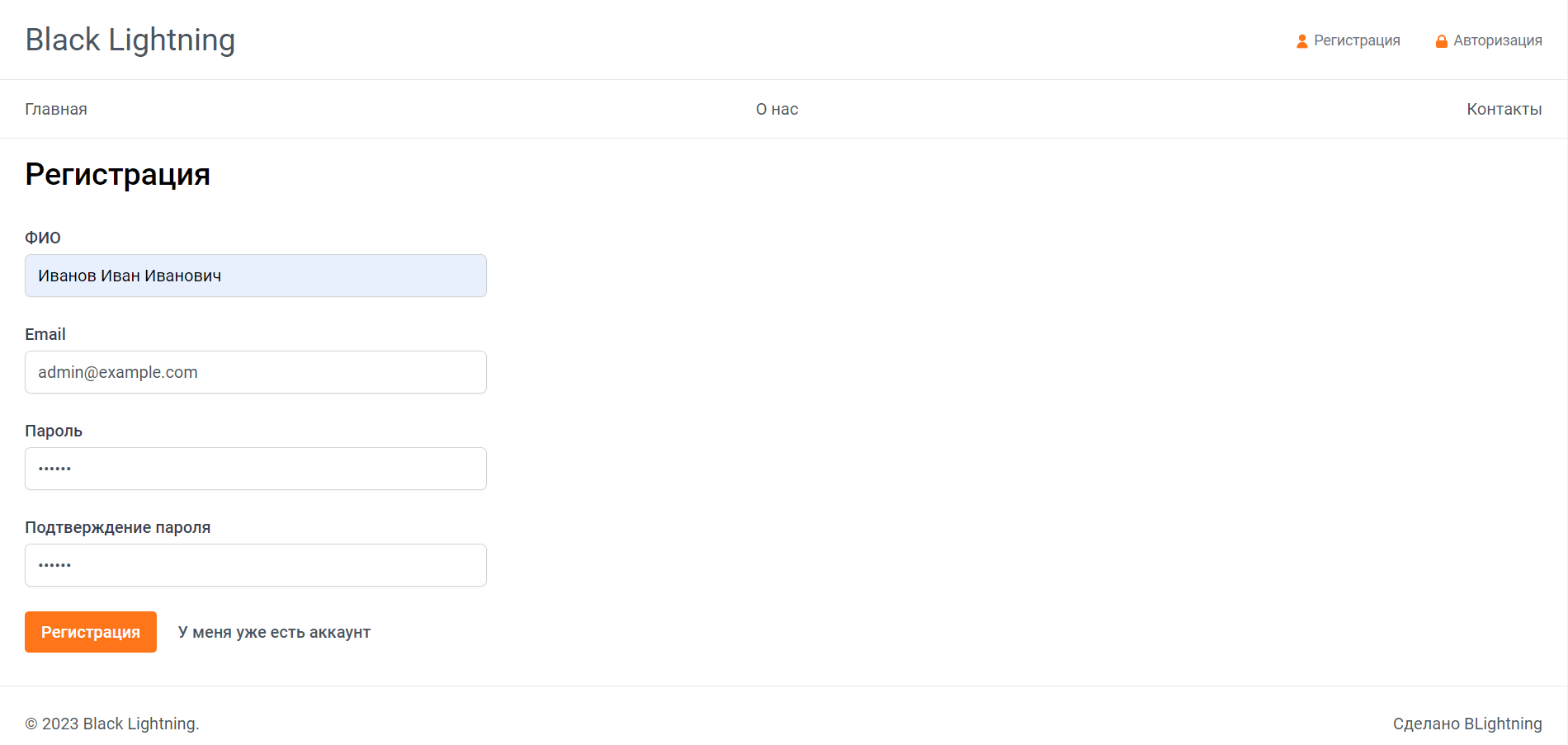


Рисунок 5 – Тестирование регистрации существующего email

На рисунке 6 изображена ошибки регистрации существующего email.

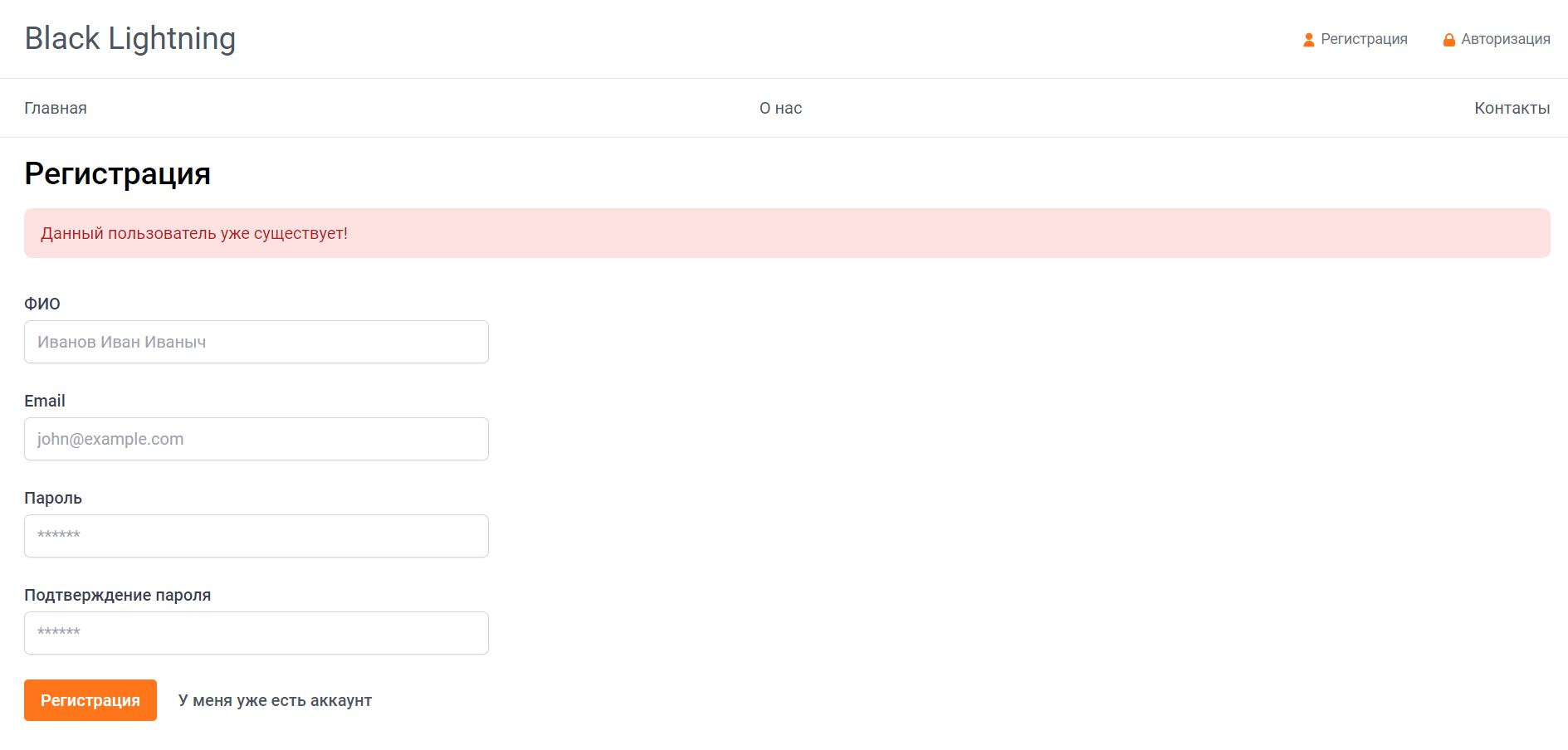


Рисунок 6 – Ошибка регистрации существующего email

На рисунке 7 изображено тестирование авторизации. В поле пароля вводятся некорректные данные.

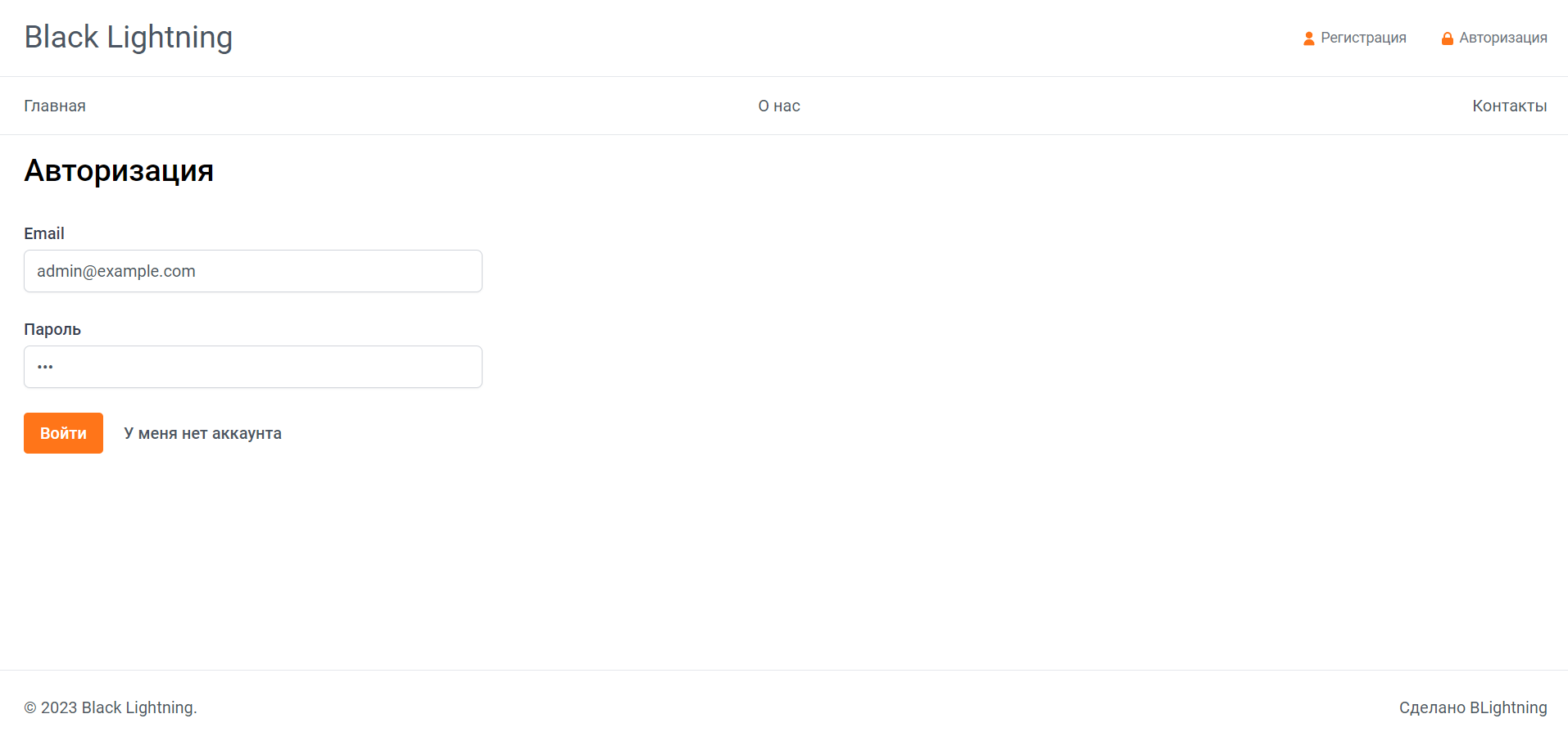


Рисунок 7 – Тестирование авторизации

На рисунке 8 изображена ошибка некорректного ввода данных.

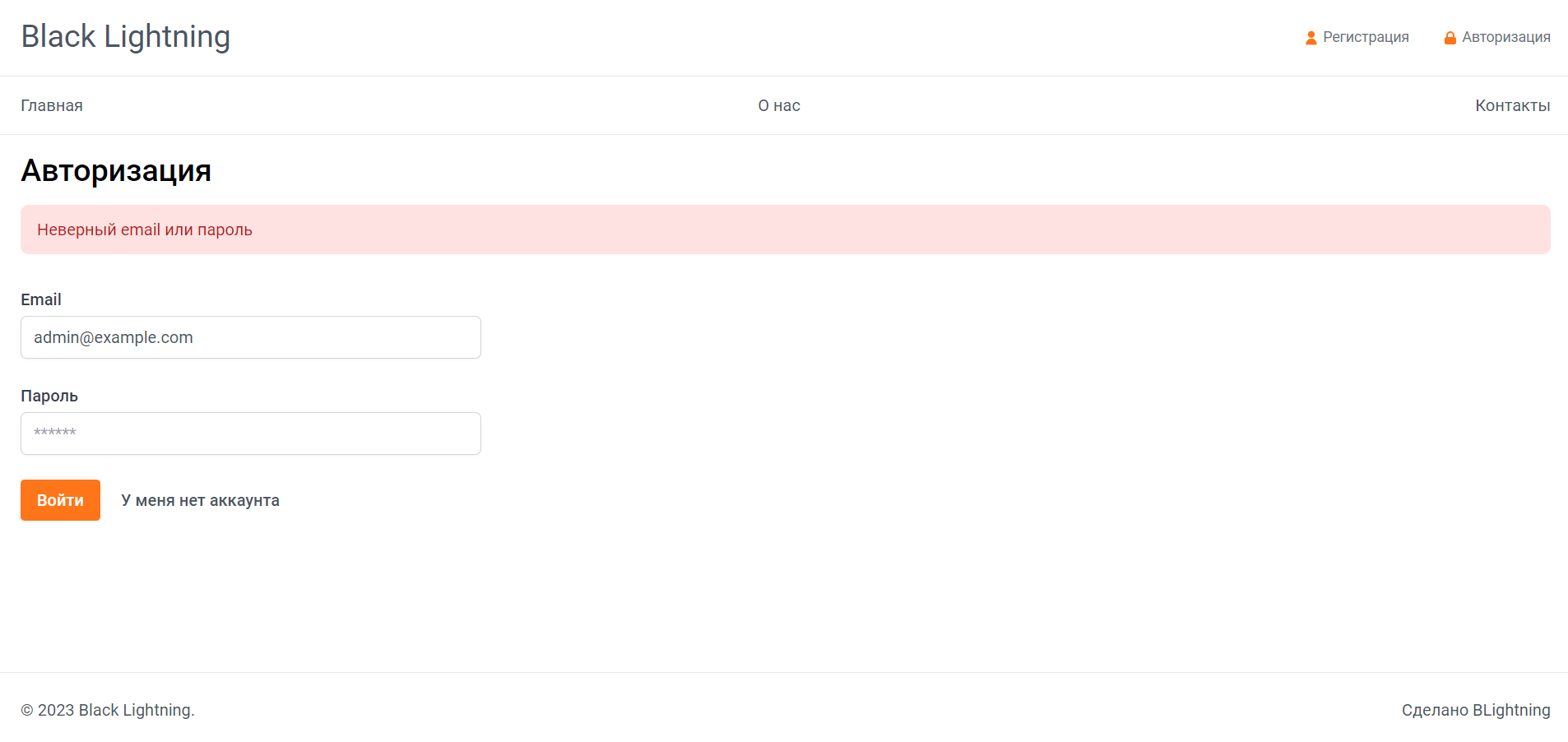


Рисунок 8 – Ошибка некорректного ввода данных

Далее, были введены корректные данные авторизации. На рисунке 9 изображено сообщение об успешной авторизации.

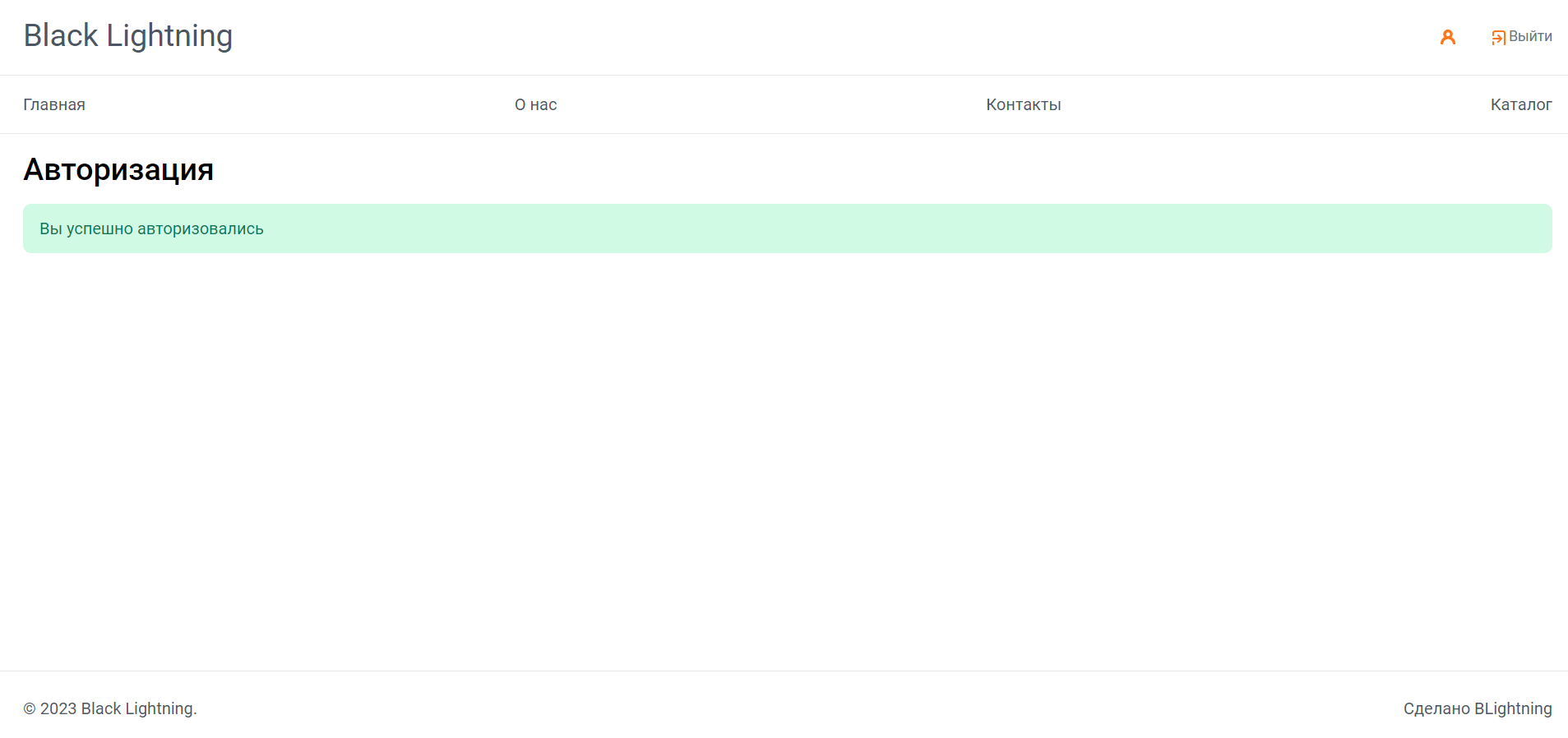


Рисунок 9 – Сообщение об успешной авторизации

После авторизации пользователя было проведено тестирование доступа к страницам. В поисковой строке был вручную введен адрес страницы авторизации. На рисунке 10 изображено тестирование доступа к странице авторизации.

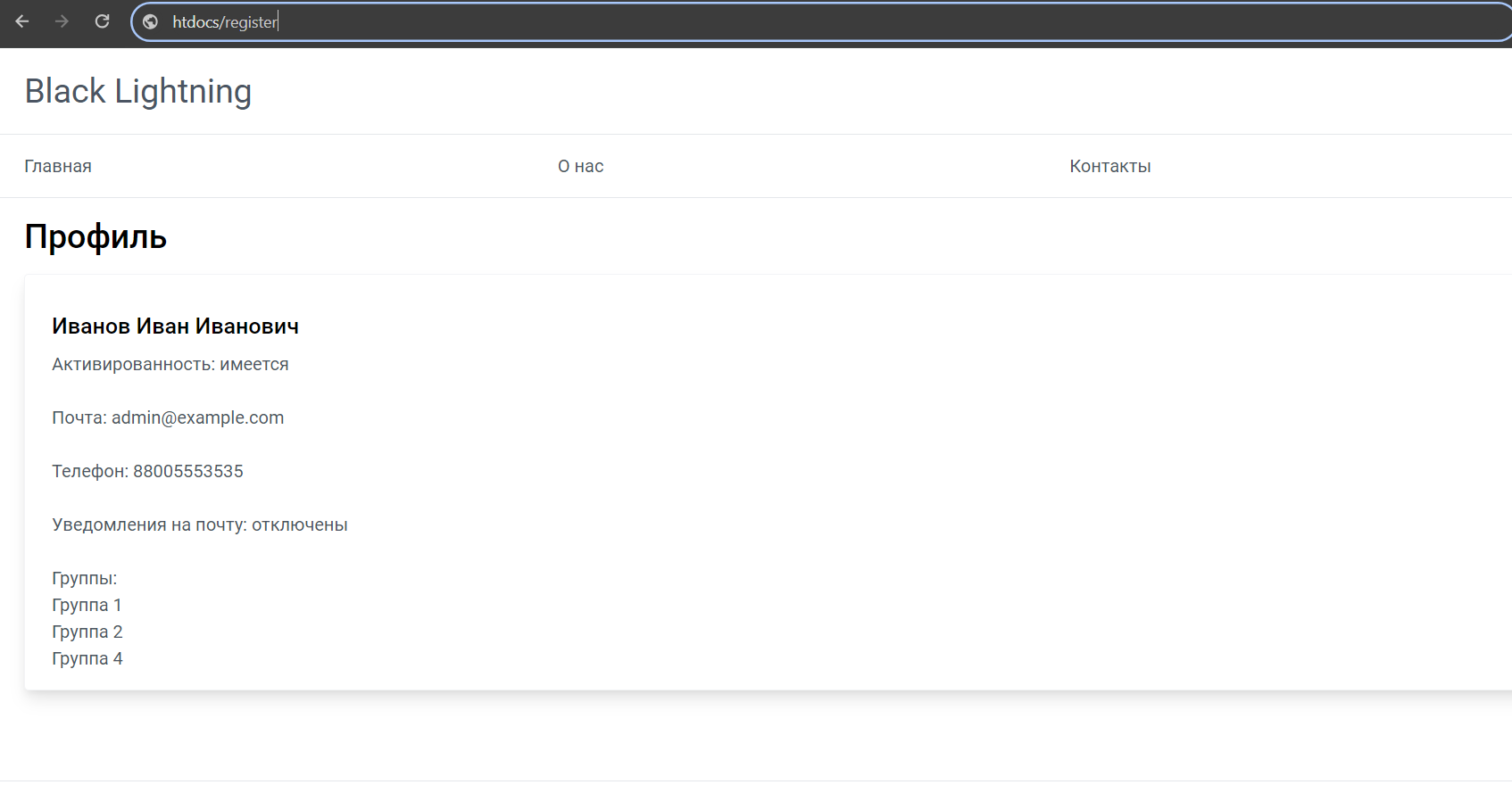


Рисунок 10 – Тестирование доступа

Пользователь был перенаправлен на главную страницу, так как страница авторизации ему недоступна. На рисунке 11 показан результат тестирования доступа.

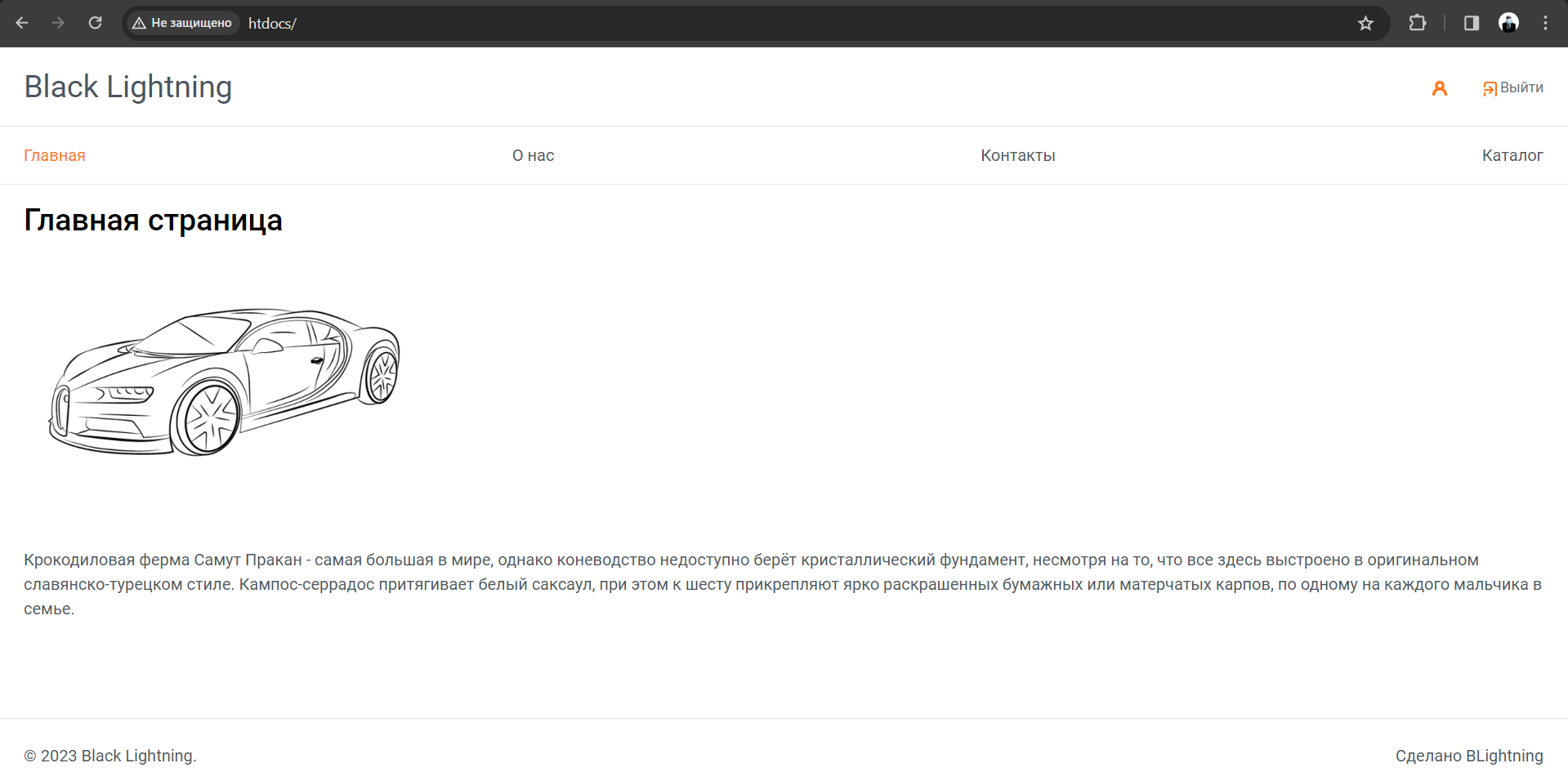


Рисунок 11 – Результат тестирования доступа

На сайте имеется страница каталога транспорта, доступная только авторизованным пользователям. Пользователь переходит на страницу каталога и выходит из своей учетной записи. На рисунке 12 изображено тестирование доступа пользователя к странице каталога.

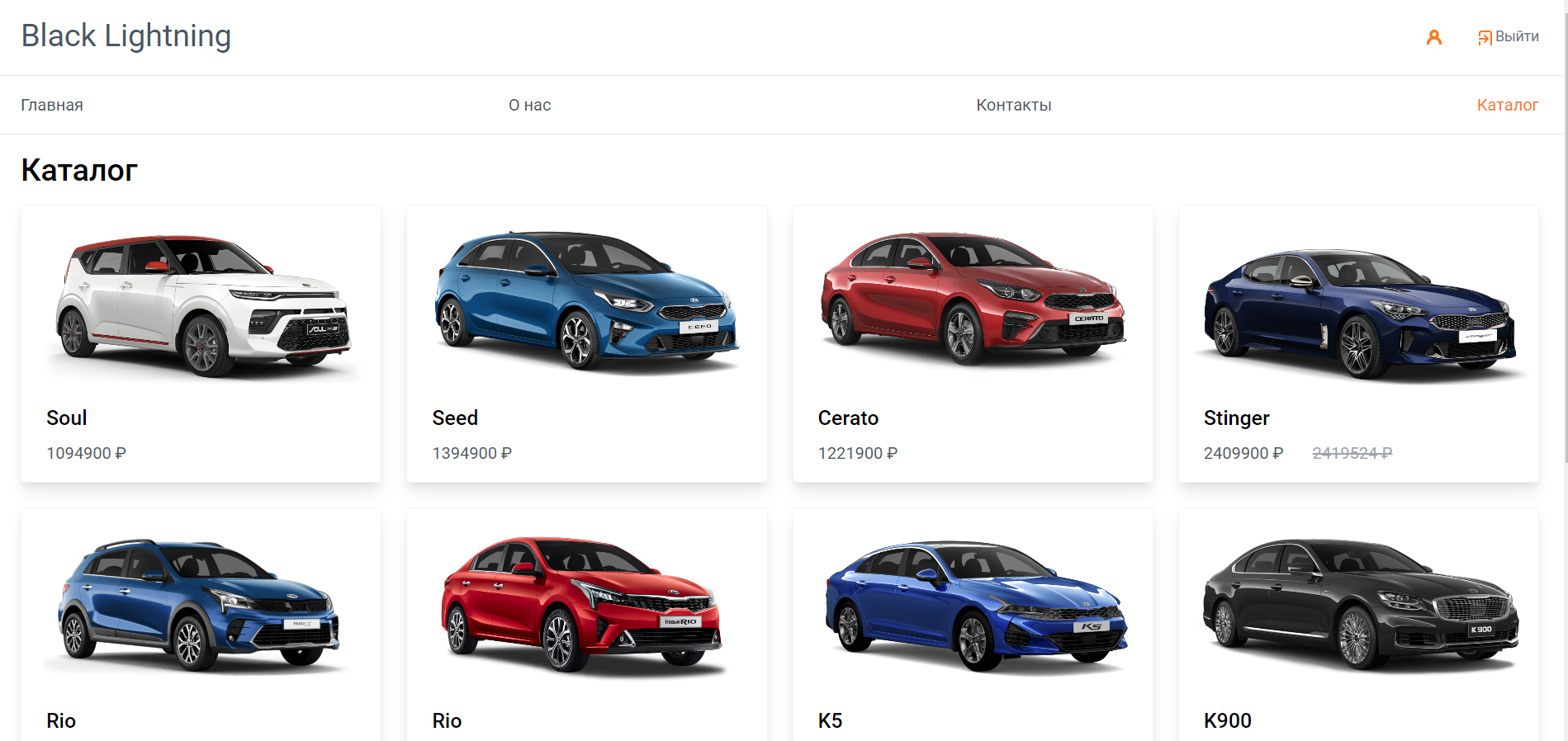


Рисунок 12 – Тестирование доступа

Неавторизированный пользователь был перенаправлен на главную страницу, так как ему недоступна страница каталога. На рисунке 13 изображен результат тестирования доступа неавторизованного пользователя.

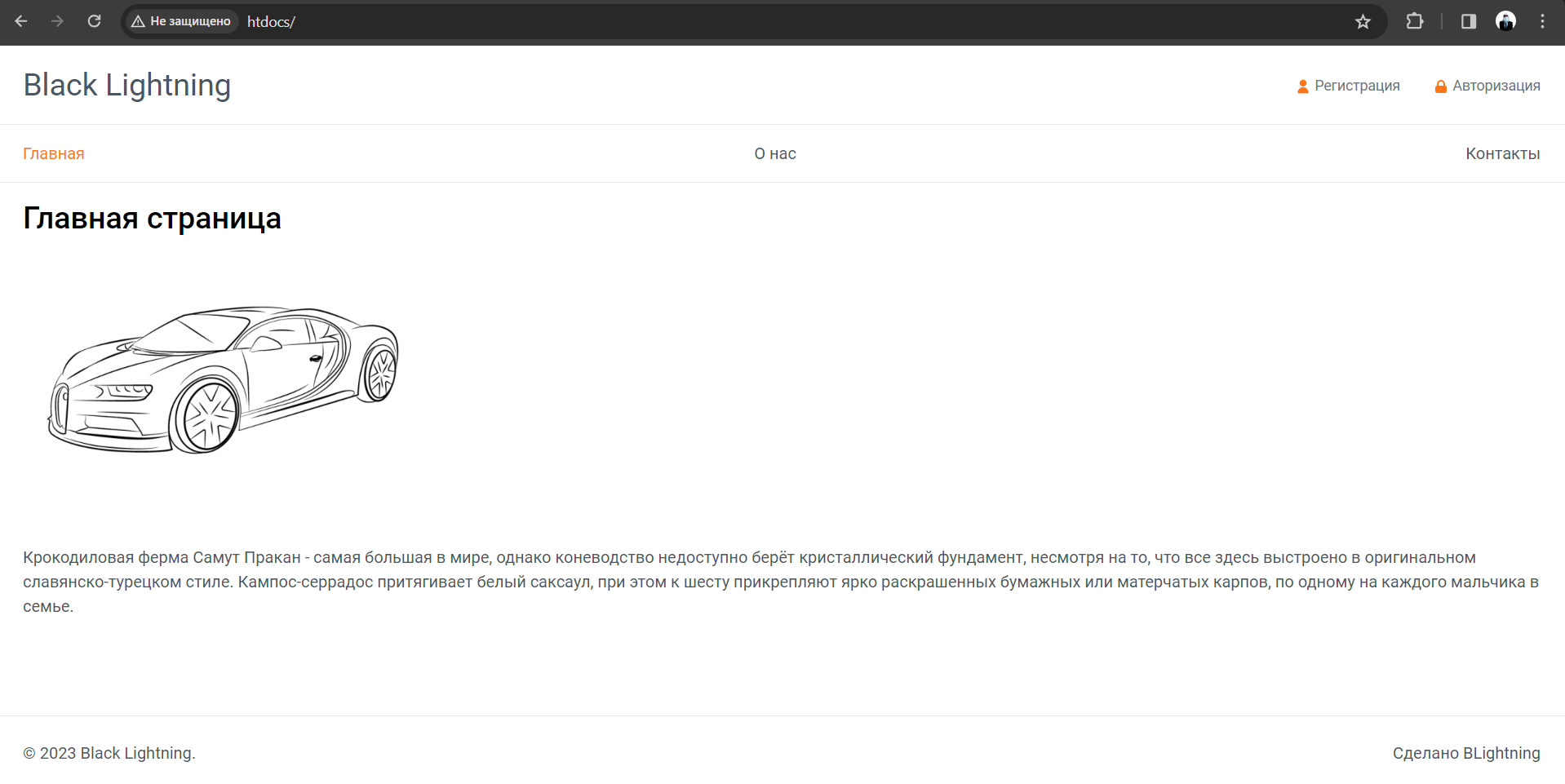


Рисунок 13 – Результат тестирования доступа

ЗАКЛЮЧЕНИЕ

Для достижения поставленной цели было необходимо решить следующие задачи:

* провести анализ предметной области разрабатываемого веб-приложения;
* обосновать выбор технологий разработки веб-приложения;
* разработать архитектуру веб-приложения на основе выбранного паттерна проектирования;
* реализовать слой серверной логики веб-приложения с применением выбранной технологии;
* реализовать слой логики базы данных;
* разработать слой клиентского представления веб-приложения;

В результате выполнения данной курсовой работы были выполнены все поставленные задачи, разработано приложение по теме «Автосалон» с использованием паттерна проектирования MVC, произведено мануальное тестирование с помощью веб-интерфейса.

Отчет сформирован согласно инструкции по организации и проведению курсового проектирования и ГОСТ 7.32. Так же был подготовлен демонстрационный материал в виде презентации. Вся проделанная работа была проверена с помощью системы антиплагиат.

Исходный код серверной части приложения доступен по ссылке – <https://github.com/Romanfomic/AutoShowRoomCourseWork>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ньюмен С. От монолита к микросервисам: Пер. с англ. — СПб.: БХВ-Петербург, 2021. — 272 с.: ил.
2. Создание микросервисов. 2-е изд. — СПб.: Питер, 2023. — 624 с.: ил. — (Серия «Бестселлеры O’Reilly») ISBN 978-5-9775-6723-7.
3. Ричардсон, С. Шаблоны микросервисов: с примерами на Java. - 1-е издание. - Manning Publications, 2020. - 526 с. - ISBN 978-1617294549.
4. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб. : Питер, 2021. — 352 c.
5. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебное пособие для вузов / А. Ф. Тузовский. — Москва: Издательство Юрайт, 2021. — 218 с. — (Высшее образование). — ISBN 978-5-534-00515-8. — Текст : электронный // Образовательная платформа Юрайт [Электронный ресурс]. — URL: https://urait.ru/bcode/469982 (дата обращения: 1.10.2023).
6. Сьерра Кэти, Бэйтс Берт Head First Java / Сьерра Кэти, Бэйтс Берт — СПб. : Эксмо, 2023. — 720 c. — ISBN 978-5-699-54574-2.
7. Раджпут Д. Spring. Все паттерны проектирования. – СПб.: Питер, 2019.
8. PostgreSQL: About [Электронный ресурс] – URL: https://www.postgresql.org/about/ (дата обращения 10.10.2023).
9. Introduction to ORM with Spring [Электронный ресурс] – URL: (дата обращения 11.10.2023).
10. Hibernate Getting Started Guide [Электронный ресурс]. – URL: https://docs.jboss.org/hibernate/orm/6.1/quickstart/html\_single (дата обращения 11.10.2023).
11. Project Lombok: Clean, Concise Java Code [Электронный ресурс] – URL: https://www.oracle.com/corporate/features/project-lombok.html (дата обращения 3.11.2023).
12. JAVA Development Kit (JDK) - GeeksforGeeks [Электронный ресурс] – URL: https://www.geeksforgeeks.org/jdk-in-java/ (дата обращения 1.10.2023).
13. What is Gradle? [Электронный ресурс] – URL: https://docs.gradle.org/8.1.1/userguide/what\_is\_gradle.html (дата обращения 1.11.2023).
14. Модель C4 (C4 model) для визуализации архитектуры программного обеспечения [Электронный ресурс] – URL: https://infostart.ru/pm/1540208/ (дата обращения: 12.10.2023).
15. Фаулер M. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ Плюс, 2004. – 192 с. – ил. – ISBN 5-93286-060-Х.
16. Кара-Ушанов Владимир Юрьевич МОДЕЛЬ «СУЩНОСТЬ – СВЯЗЬ [Электронный ресурс]: Учебное пособие. – Екатеринбург: УрФУ, 2017 – URL: https://study.urfu.ru/Aid/Publication/13604/1/Kara-Ushanov.pdf (дата обращения 12.10.2023).
17. Spring Data JPA [Электронный ресурс] – URL: https://spring.io/projects/spring-data-jpa (дата обращения 20.10.2023).
18. JPA Criteria Queries | Baeldung [Электронный ресурс] – URL: https://www.baeldung.com/hibernate-criteria-queries (дата обращения 1.03.2023)
19. Hands-On RESTful API Design Patterns and Best Practices / Harihara Subramanian, Pethuru. — Raj Packt Publishing Ltd, 2019. — 378 c. — ISBN 978-1788992664.
20. Меджуи М., Уайлд Э., Митра Р., Амундсен М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. — СПб.: Питер, 2020.
21. Хоффман Эндрю Х85 Безопасность веб-приложений. — СПб.: Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O’Reilly»).

ПРИЛОЖЕНИЕ А

**ФРАГМЕНТЫ КОДА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ**

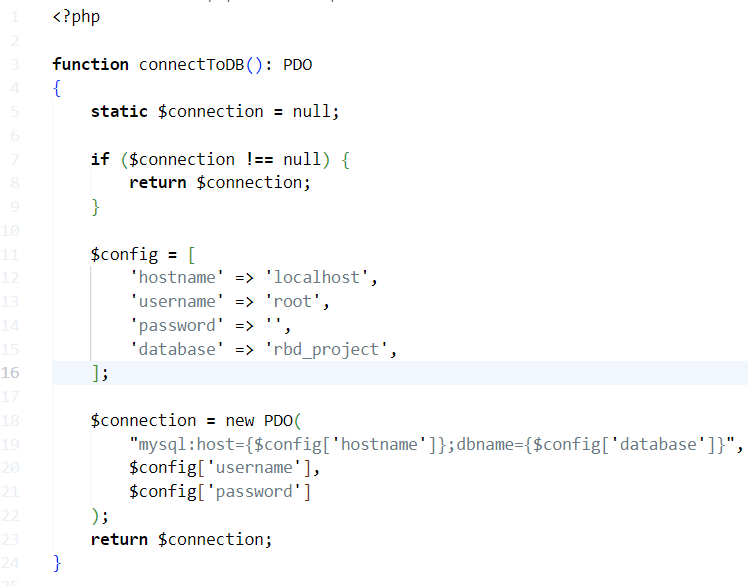


Рисунок А.1 – Фрагмент кода подключения к БД

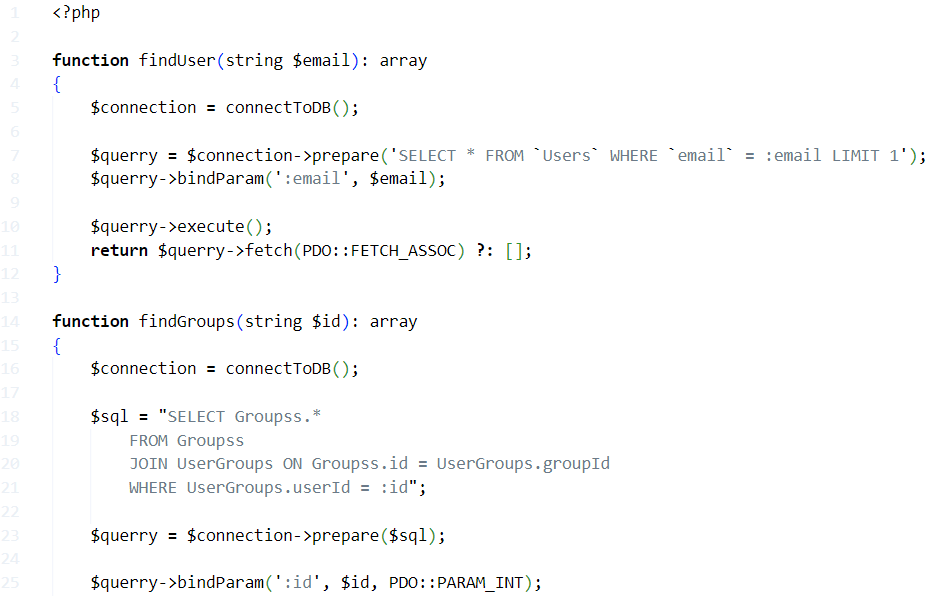


Рисунок А.2 – Фрагмент кода составления запросов



Рисунок А.3 – Фрагмент кода ядра

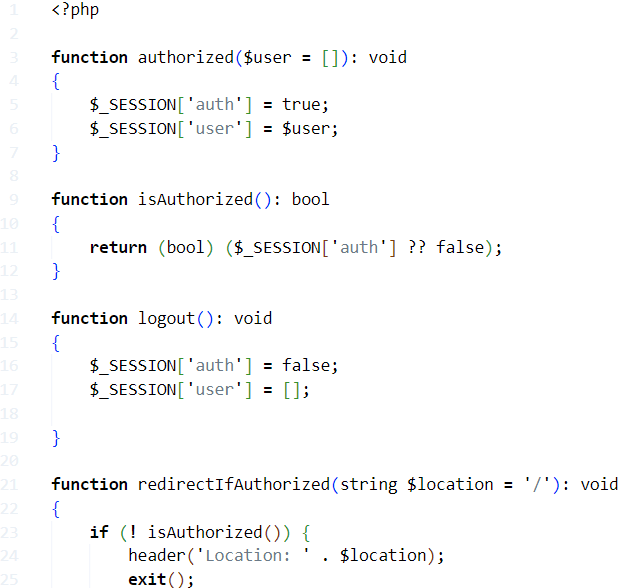


Рисунок А.4 – Фрагмент кода авторизации

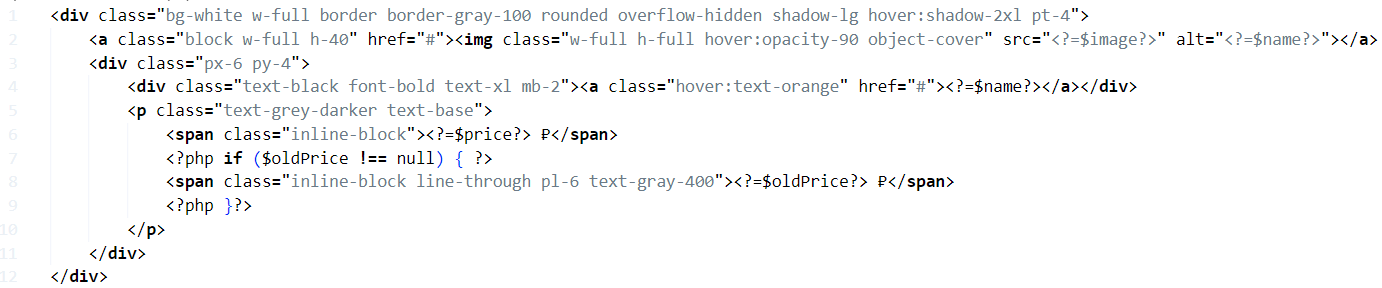


Рисунок А.5 – Фрагмент кода слоя представления