Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**Веб-служба по продаже игр**

БГУИР КП 6-05-0612-01-058 ПЗ

Студент Чилюк И.А.

Руководитель Болтак С.В.

Минск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc199096055)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc199096056)

[1.1 Обзор аналогов 6](#_Toc199096057)

[1.2 Постановка задачи 8](#_Toc199096058)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА](#_Toc199096059) 10

[2.1 Структура программы 10](#_Toc199096060)

[2.2 Проектирование интерфейса программного средства 11](#_Toc199096061)

[2.3 Проектирование функционала программного средства 13](#_Toc199096062)

[3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 20](#_Toc199096063)

[3.1 Интерфейс пользователя и взаимодействие с системой 20](#_Toc199096064)

[3.2 Обработка заказов и генерация ключей 21](#_Toc199096065)

[3.3 Аутентификация и безопасность 22](#_Toc199096066)

[3.4 Работа с базой данных 24](#_Toc199096067)

[4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 26](#_Toc199096068)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 27](#_Toc199096069)

[5.1 Интерфейс программного средства 27](#_Toc199096070)

[5.2 Главное окно пользователя 28](#_Toc199096071)

[5.3 Магазин игр 29](#_Toc199096072)

[5.4 Корзина 30](#_Toc199096072)

[5.5 Техническая поддержка 30](#_Toc199096072)

[ЗАКЛЮЧЕНИЕ 32](#_Toc199096073)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc199096074)

[ПРИЛОЖЕНИЕ А. Исходный код программы 34](#_Toc199096075)

# ВВЕДЕНИЕ

Компьютерные игры занимают важное место в современной цифровой индустрии, представляя собой не только форму развлечения, но и мощный инструмент для развития креативного мышления, коммуникации и командной работы. С ростом популярности цифровой дистрибуции всё больше пользователей отдают предпочтение покупке игр через онлайн-платформы. Такие сервисы обеспечивают удобный доступ к игровому контенту, мгновенную доставку ключей активации и поддержку различных платёжных систем, что делает процесс покупки максимально простым и доступным.

Создание собственного веб-сайта по продаже цифровых копий игр позволяет реализовать индивидуальный подход к оформлению витрины, внедрить уникальные механизмы работы с заказами и гибкую систему скидок. В отличие от крупных универсальных площадок, специализированный ресурс предоставляет возможность адаптировать функциональность под конкретную целевую аудиторию — будь то казуальные игроки, поклонники инди-игр или коллекционеры. Кроме того, наличие собственного сервиса снижает зависимость от сторонних платформ и позволяет контролировать политику ценообразования, ассортимент и пользовательский опыт.

Одной из ключевых задач при разработке такого сайта является обеспечение надёжной системы управления цифровыми ключами, отслеживания заказов и защиты пользовательских данных. Не менее важна реализация интуитивно понятного интерфейса, позволяющего быстро находить интересующие игры, оформлять покупки и получать ключи для активации на популярных платформах, таких как Steam, Origin, Uplay и другие.

Целью данного курсового проекта является разработка веб-приложения для продажи цифровых копий игр с возможностью регистрации пользователей, формирования корзины, осуществления оплаты и автоматической выдачи ключей. В процессе работы будут рассмотрены принципы построения e-commerce систем, методы взаимодействия с базами данных, механизмы безопасности и интеграции с внешними API. Итогом проекта станет функциональный онлайн-магазин, обеспечивающий удобную покупку и управление цифровыми продуктами.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Обзор аналогов

На сегодняшний день существует огромное количество сайтов/приложений, позволяющих купить интересующую человека игру. Примерами являются playerok.com, steambuy.com, funpay.com.

В первую очередь стоит рассмотреть сайт playerok.com, так как он является одним из самых популярных аналогов. Внешний вид веб-интерфейса playerok.com представлен на рисунке 1.1.

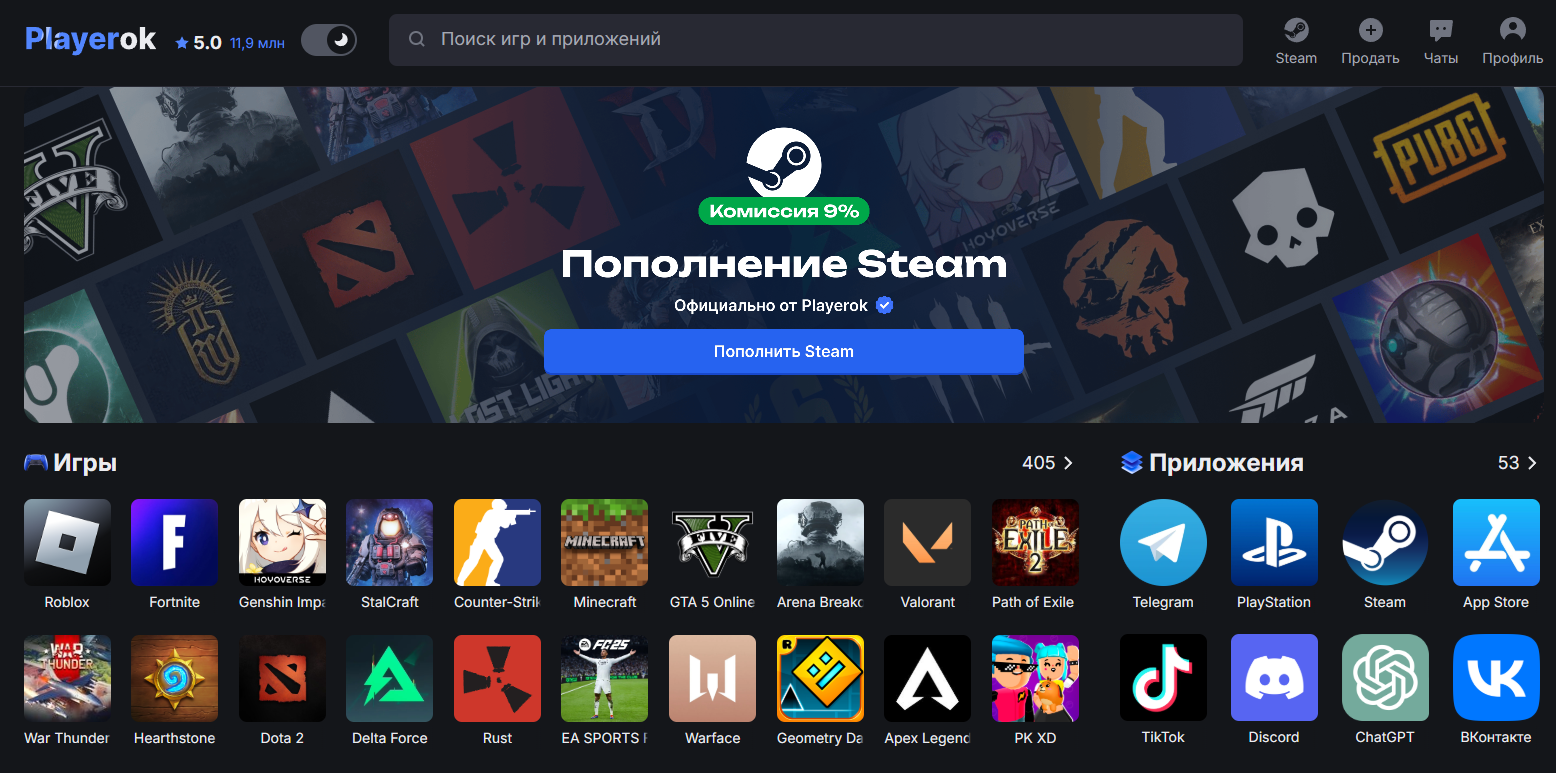


Рисунок 1.1 – Маркетплейс Playerok

Данный маркетплейс предлагает пользователям приобретение различных игр, а также услуг связанных с играми. На данном маркетплейсе можно приобрести подписки в разных играх, звезды в телеграм и т.п. В особенности данного сайта можно отнести возможность купить услуги по действительно дешевым ценам, на официальных портал данная стоимость в 2-3 раза выше.

Следующим маркетплейсом, заслуживающим внимания, является steambuy.com. Внешний вид данного приложения представлен на рисунке 1.2.

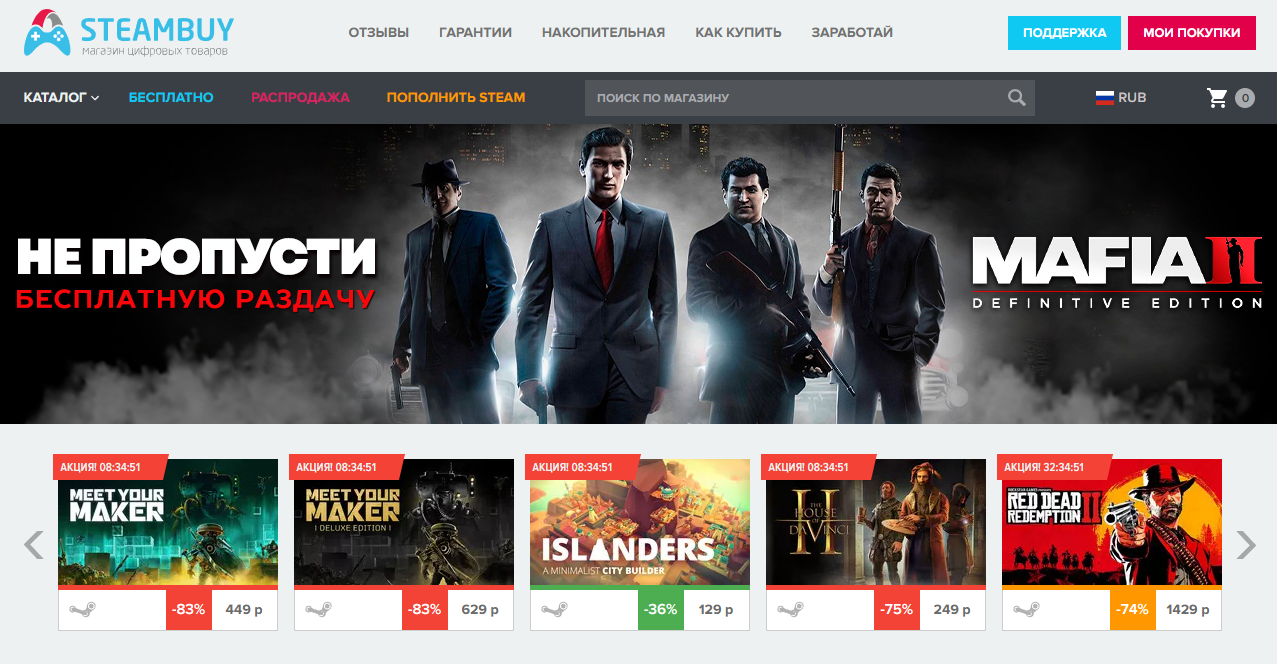


Рисунок 1.2 – Маркетплейс steambuy

Отличие маркетплейса steambuy от playerok заключается в том, что на сайте steambuy отсутствует возможность купить услуги в разных играх. Сайт предоставляет возможность исключительно покупки ключей. Среди особенностей можно выделить возможность покупки ключей Steam из всех регионов и стран, так как многие страны блокируют возможность покупки из ряда стран. Из минусов можно выделить отсутствие возможности покупки игр для Epic Games, Uplay, т.к. есть возможность покупки только для steam.

Еще одним крупным игроком на рынке маркетплейсов является FunPay. Его веб-интерфейс показан на рисунке 1.3.

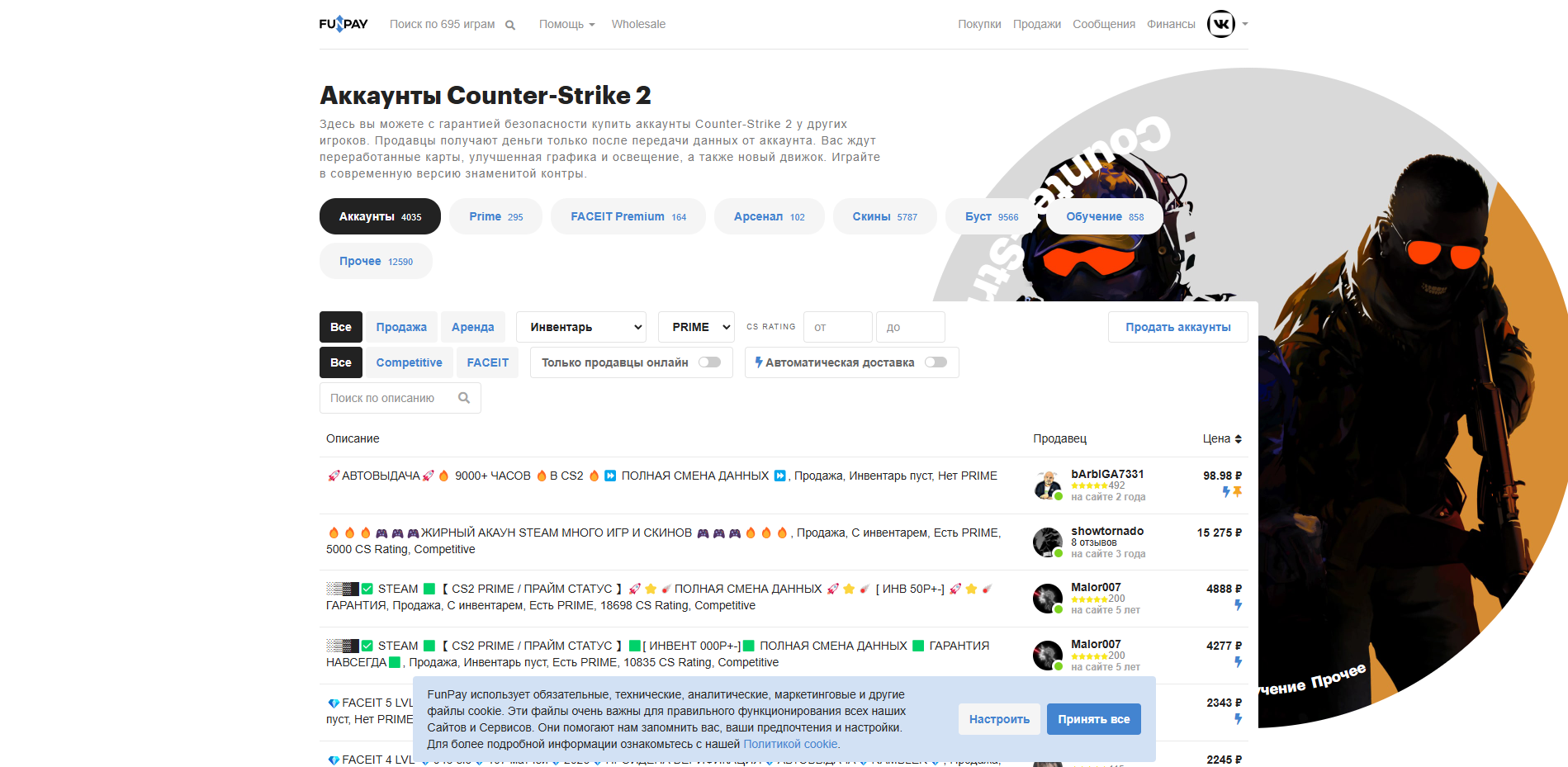


Рисунок 1.3 – Маркетплейс FunPay

FunPay является лучшим аналогом из вышеперечисленных, связано это с тем, что FunPay предоставляет возможность покупки игр на разных площадках, а также различных услуг связанных с играми. Продажа внутриигровой валюты, подписок, прокачек аккаунтов и т.п.

Таким образом, существующие маркетплейсы обладают схожими базовыми возможностями. На основе анализа аналогов можно выделить ключевые требования к разрабатываемому маркетплейсы «Hoofman Store», включая возможность заказа, оплаты покупки игры, связью с тех. поддержкой и личным профилем пользователя.

## Постановка задачи

В рамках данной курсовой работы необходимо разработать веб-приложение — интернет-магазин цифровых копий игр. Данное программное средство должно обеспечивать удобный пользовательский интерфейс, надёжную обработку заказов и автоматическую выдачу цифровых ключей. Основное внимание уделяется безопасности данных, простоте взаимодействия и корректной обработке покупок в режиме онлайн.

Система должна включать следующие основные функции:

– Регистрация и авторизация пользователей;

– Просмотр каталога игр с фильтрацией и пагинацией;

– Добавление игр в корзину и оформление заказа;

– Интеграция с базой данных цифровых ключей (keys) и их автоматическая выдача при оплате;

– Генерация и скачивание .txt файла с купленными ключами после покупки;

– Панель администратора для управления товарами, категориями и ключами;

– Отслеживание истории заказов и статистики покупок;

– Поддержка различных жанров и категорий игр;

– Защита данных пользователей и контроль доступа к административной части;

– Адаптивный интерфейс для корректной работы на различных устройствах.

Для реализации проекта используется язык программирования PHP с применением фреймворка Laravel, что обеспечивает устойчивую структуру кода, безопасность и масштабируемость. Взаимодействие с интерфейсом реализовано при помощи Blade-шаблонов. В качестве базы данных используется MySQL, обеспечивающая надёжное хранение информации о пользователях, товарах, заказах и ключах активации. Данные ключи хранятся в отдельной таблице и связываются с заказами на основе идентификаторов игр.

Особенностью реализации является автоматическая привязка цифровых ключей к заказу и формирование .txt файла для скачивания. Такой подход исключает ручную обработку и повышает удобство использования. В административной панели реализована возможность загрузки новых ключей, редактирования карточек игр, управления категориями и просмотров заказов.

Разрабатываемое программное средство ориентировано на цифровую дистрибуцию и обеспечивает быстрый и безопасный способ приобретения игр. Реализация проекта позволяет пользователю получить игру сразу после покупки.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Структура программы

Разрабатываемое веб-приложение HStore реализовано с использованием фреймворка Laravel и состоит из следующих ключевых модулей:

* **Controllers** — отвечают за обработку пользовательских запросов и логику взаимодействия с данными:
  + BasketController.php — управление корзиной пользователя;
  + ProductController.php — отображение списка игр и управление товарами;
  + ProfileController.php — управление профилем пользователя;
  + SupportTicketController.php — обработка обращений в техническую поддержку;
  + Auth\Controller.php — базовый контроллер для остальных контроллеров аутентификации.
* **Middleware** — промежуточные обработчики запросов, реализующие безопасность и фильтрацию:
  + Authenticate.php — проверка авторизации пользователя;
  + EncryptCookies.php — шифрование cookie;
  + PreventRequestsDuringMaintenance.php — блокировка запросов при техническом обслуживании;
  + RedirectIfAuthenticated.php — редирект при повторной авторизации;
  + TrimStrings.php, TrustHosts.php, TrustProxies.php — утилиты для обработки запросов;
  + ValidateSignature.php, VerifyCsrfToken.php — валидация и защита от CSRF-атак.
* **Requests**:
  + Kernel.php — регистрация middleware и маршрутов приложения.

Также в проекте задействованы дополнительные слои:

* **Миграции и модели Eloquent** для взаимодействия с базой данных (пользователи, товары, корзина, ключи и заказы);
* **Blade-шаблоны** для отображения страниц (главная, корзина, авторизация, каталог и т.д.);
* **Маршруты** (web.php, api.php) для обработки пользовательских действий.

Таким образом, структура приложения обеспечивает модульное разделение ответственности, гибкую архитектуру и удобство в сопровождении и масштабировании проекта.

## 2.2 Проектирование интерфейса программного средства

При разработке программного средства за основу будет взят стандартный внешний вид маркетплейсов и логика работы с базами данных.

2.2.1 Регистрация и вход

Окна регистрации и входа содержат поля для ввода электронной почты и пароля. Пользователь может перейти между формами входа и регистрации с помощью соответствующих ссылок. Внешний вид макета окна регистрации представлен на рисунке 2.1. Внешний вид макета окна авторизации представлен на рисунке 2.2.

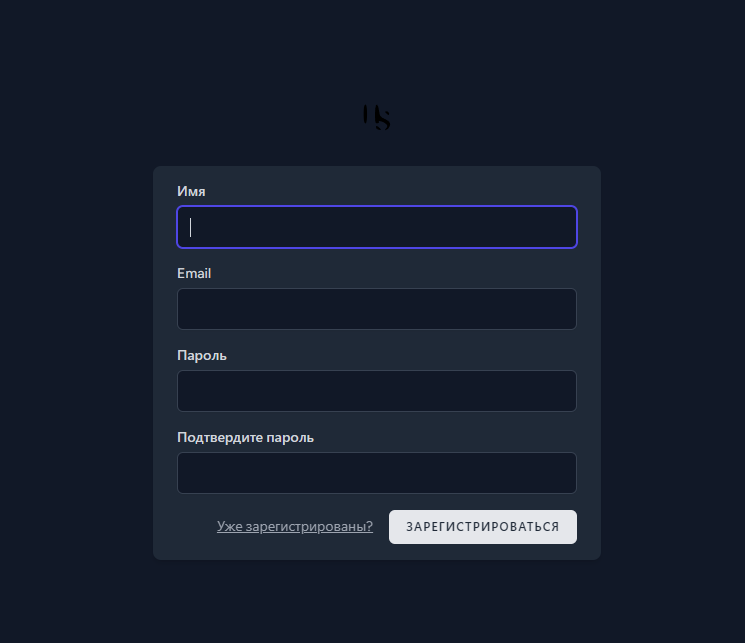


Рисунок 2.1 – Окно регистрации

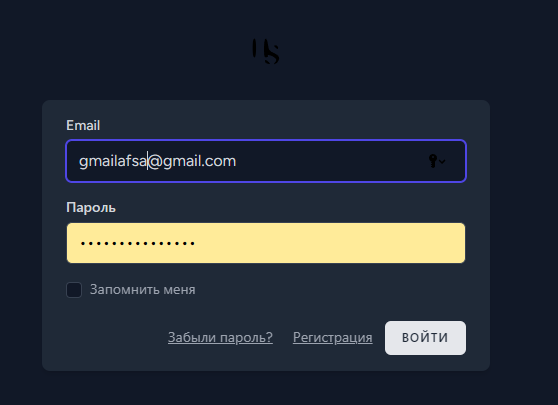


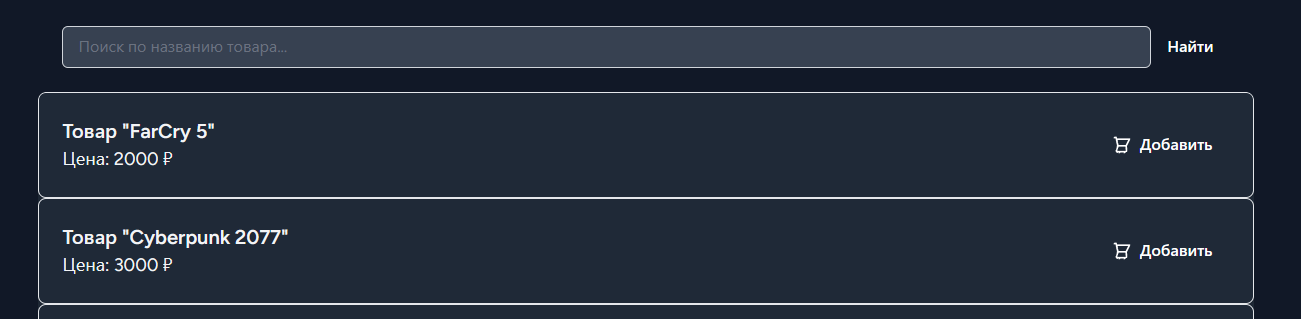
Рисунок 2.2 – Окно авторизации

2.2.2 Элементы управления приложения

Основные элементы управления веб-интерфейсом пользователя:

– Кнопка "Добавить в корзину" – помещает выбранную игру в корзину пользователя;  
– Кнопка "Удалить из корзины" – удаляет игру из корзины на соответствующей странице;  
– Кнопка "Оплатить" – инициирует оформление заказа и получение ключей;  
– Кнопка "Отправить обращение" – используется в разделе поддержки, отправляет сообщение с возможностью прикрепить изображение;  
– Кнопка "Выйти" – завершает текущую сессию пользователя.

Внешний вид элементов управления приложением представлен на рисунке 2.3.





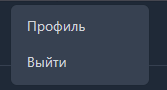
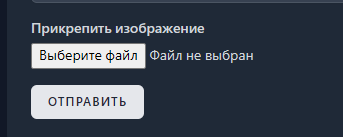


Рисунок 2.3 – Элементы управления приложением

2.2.4 Навигация по сайте

Пользователь может перемещаться между папками с помощью кликов по ссылкам в навигационном меню.

Внешний вид навигации представлен на рисунке 2.4.

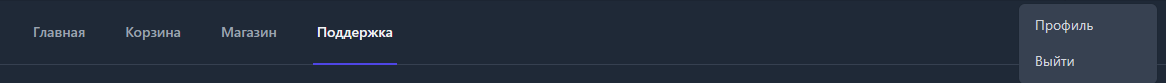


Рисунок 2.4 – Навигация по папкам

2.2.5 Управление профилем пользователя

У программного средства будет реализована возможность редактирования данных пользователя:

* логин
* почта
* пароль
* удаление аккаунта

Внешний вид профиля пользователя представлен на рисунке 2.5.

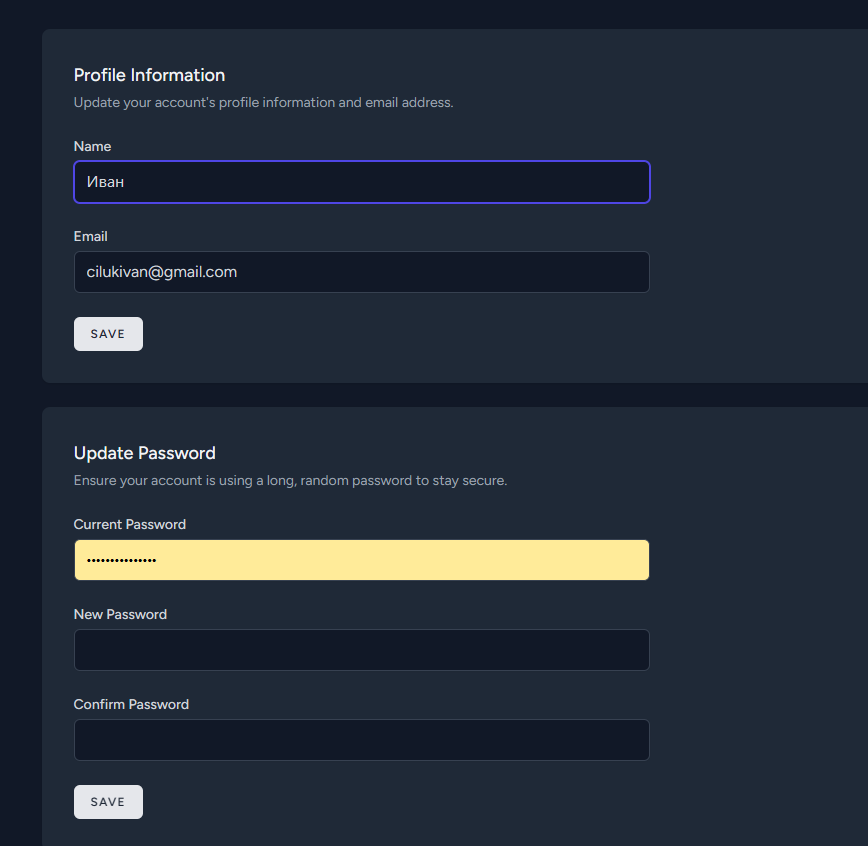


Рисунок 2.5 – Профиль пользователя

## 2.3 Проектирование функционала программного средства

Для обеспечения удобного и эффективного взаимодействия пользователя с распределённым облачным хранилищем приложение должно реализовывать следующий минимальный набор функций:

* Авторизация и регистрация пользователей;
* Полная работа магазина с товаром;
* Удаление товара из корзины, оплата;
* Связь с технической поддержкой;
* Обновление сессии (токена доступа);
* Распределенность хранения данных

2.3.1 Авторизация и регистрация пользователей

Пользователь может зарегистрироваться, указав электронную почту и пароль, после чего войти в систему. При входе выполняется проверка корректности введённых данных. Аутентификация происходит посредством использования Middleware. Блок-схема метода, который осуществляет регистрацию пользователей представлена на рисунке 2.7.

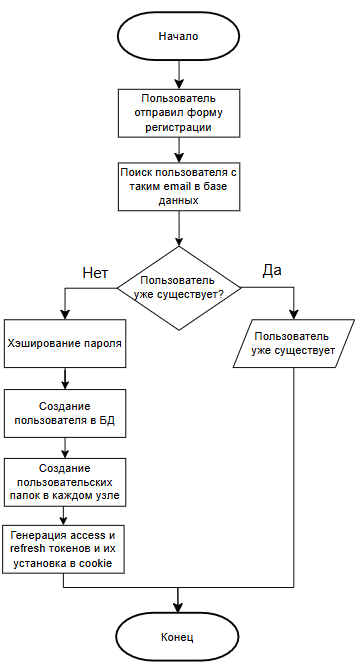


Рисунок 2.6 – Блок-схема регистрации

2.3.2 Работа с магазином

Пользователь при наличии желания купить себе ключ от игры, может это сделать через раздел «Магазин». Блок-схема работы магазина представлена на рисунке 2.8.

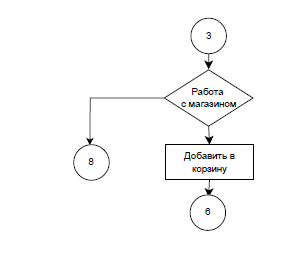
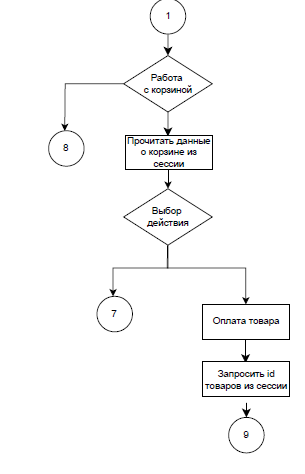




Рисунок 2.7 – Блок-схема работы магазина

2.3.3 Работа с корзиной

Корзина является неотъемлемой частью программы, через корзину появляется возможность оплатить товар, а также просмотреть весь выбранный товар. Блок-схема корзины представлена на рисунке 2.8.



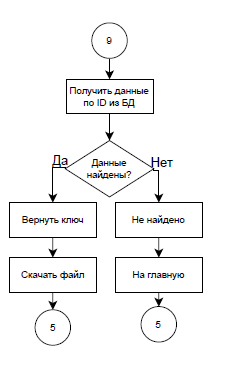


Рисунок 2.8 – Блок-схема корзины

2.3.4 Связь с технической поддержкой

В случае наличия проблем либо вопросов у пользователя, он может обратиться в техническую поддержку. Блок-схема метода, который осуществляет работу с тех. поддержкой представлена на рисунке 2.10.

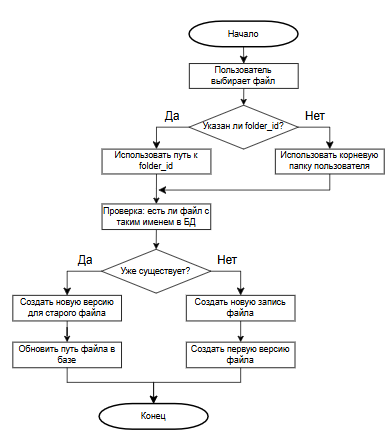


Рисунок 2.9 – Блок-схема загрузки файлов

2.3.6 Обновление сессии

Для предотвращения принудительного выхода по истечению срока действия токена реализован скрипт, предлагающий обновить токен за минуту до окончания сессии. Пользователь может подтвердить или отказаться от обновления. В случае отказа или ошибки происходит перенаправление на страницу входа. Блок-схема метода, который осуществляет обновление сессии представлена на рисунке 2.12.

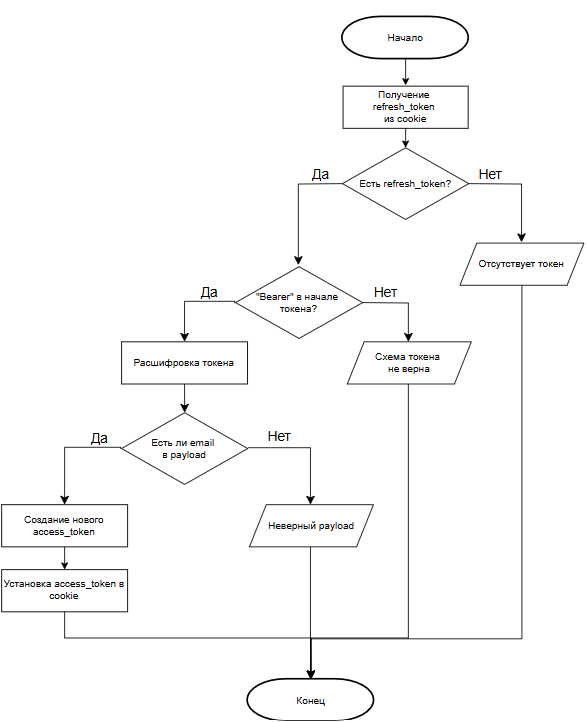


Рисунок 2.12 – Блок-схема обновления access-токена

2.3.7 Распределенность хранения данных

Для обеспечения масштабируемости и надёжности маркетплейса с ключами в программном средстве предусмотрена возможность распределённого хранения данных. Архитектура приложения допускает использование нескольких физических узлов, на которых размещаются файлы пользователей. Модуль storage\_node абстрагирует взаимодействие с файловой системой и позволяет:

* направлять файлы на определённые узлы;
* получать файлы с нужного узла при запросе на скачивание;

Данное решение облегчает расширение системы — при увеличении числа пользователей возможно подключение дополнительных узлов без изменения основной логики. Блок-схема распределения файлов представлена на рисунке 2.13.

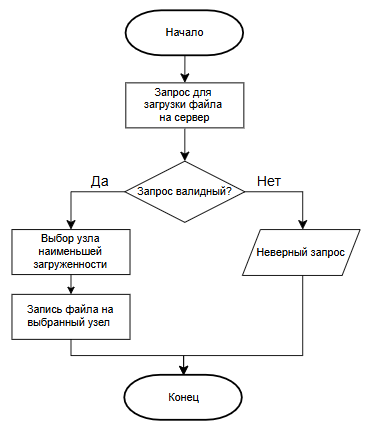


Рисунок 2.13 – Блок-схема распределения файлов

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1 Интерфейс пользователя и взаимодействие с системой

Основной функционал разработанной системы включает просмотр каталога игр, добавление товаров в корзину, оформление заказа, генерацию ключей, управление заказами и обращениями в службу поддержки. Вся работа с данными реализована с помощью Laravel MVC и шаблонов Blade.

3.1.1 Добавление товара в корзину

Пользователь может добавить игру в корзину, нажав на соответствующую кнопку на странице товара. Контроллер сохраняет game\_id в сессии, формируя структуру корзины. Если пользователь авторизован, корзина может быть привязана к пользователю.

**public function addToCart(Request $request, $id)**

**{**

**$cart = session()->get('cart', []);**

**$cart[$id] = true;**

**session()->put('cart', $cart);**

**return redirect()->back()->with('success', 'Товар добавлен в корзину');**

**}**

**3.1.2** Оформление заказа и генерация ключей

При нажатии кнопки "Оплатить", система извлекает все товары из корзины, находит доступные ключи по game\_id, сохраняет заказ в базу и прикрепляет .txt файл с ключами для скачивания.

**public function checkout(Request $request)**

**{**

**$cart = session('cart', []);**

**$order = Order::create([...]);**

**foreach ($cart as $gameId => $\_) {**

**$keys = Key::where('game\_id', $gameId)->where('used', false)->limit(1)->get();**

**...**

**}**

**Storage::put("keys/order\_{$order->id}.txt", $keysString);**

**...**

**}**

Метод получает объект файла по ID из базы данных. При его наличии возвращается файл с ключом от игры. В случае отсутствия файла – клиенту отправляется ошибка 404.

3.1.3 Работа с технической поддержкой

На странице "Поддержка" пользователь может отправить сообщение с текстом и прикрепить изображение. Данные сохраняются в базу, прикреплённый файл — в storage/app/support/.

public function sendSupport(Request $request)

{

$message = SupportMessage::create([

'user\_id' => auth()->id(),

'text' => $request->input('text'),

...

]);

if ($request->hasFile('attachment')) {

$path = $request->file('attachment')->store('support');

$message->update(['attachment\_path' => $path]);

}

}

## 3.2 Обработка заказов и генерация ключей

Система HStore реализует функциональность корзины и автоматическую выдачу ключей после оплаты. Заказы формируются пользователем вручную, а подтверждение покупки запускает процесс генерации текстового файла с ключами.

**3.2.1** Формирование корзины

Пользователь добавляет нужные игры в корзину, каждая позиция содержит идентификатор игры, количество и цену. Все данные сохраняются в сессии браузера или в базе данных (при необходимости). Корзина отображается на отдельной странице с возможностью удаления позиций.

**3.2.2** Завершение заказа и получение ключей

При нажатии на кнопку "Оплатить" формируется заказ. Система автоматически извлекает ключи из таблицы keys, связанные с конкретными game\_id, и создает .txt-файл для скачивания. Пример логики генерации файла:

@$file = fopen(storage\_path("app/public/keys\_$userId.txt"), 'w');

foreach ($cartItems as $item) {

$keys = Key::where('game\_id', $item->game\_id)->take($item->quantity)->get();

foreach ($keys as $key) {

fwrite($file, $key->value . PHP\_EOL);

$key->delete(); // удаляем выданные ключи

}

}

fclose($file);

Файл затем доступен пользователю для скачивания, а корзина очищается. Таким образом, исключается возможность повторной выдачи одного и того же ключа.

## 3.3 Аутентификация и безопасность

Для обеспечения безопасности доступа к заказам и управления ключами в системе реализована полноценная система аутентификации, основанная на Laravel Sanctum.

**3.3.1** Регистрация пользователя

Регистрация пользователей осуществляется через web-интерфейс. При регистрации пользователь вводит имя, email и пароль, после чего данные сохраняются в базе данных с автоматическим хешированием пароля через bcrypt.

**public function register(Request $request)**

**{**

**$request->validate([**

**'name' => 'required|string|max:255',**

**'email' => 'required|string|email|unique:users',**

**'password' => 'required|string|min:6|confirmed',**

**]);**

**$user = User::create([**

**'name' => $request->name,**

**'email' => $request->email,**

**'password' => Hash::make($request->password),**

**]);**

**Auth::login($user);**

**return redirect()->route('home');**

**}**

После успешной регистрации пользователь автоматически авторизуется и перенаправляется на главную страницу.

**3.3.2** Авторизация

Вход в систему реализован через стандартную Laravel-авторизацию. Пользователь вводит email и пароль, после чего создаётся сессия. После успешной проверки данных пользователя происходит генерация JWT-токена, в который помещается идентификатор пользователя. Токен сохраняется в cookie с флагом HttpOnly, что защищает его от доступа через JavaScript и снижает риск атак типа XSS.

**public function login(Request $request)**

**{**

**$credentials = $request->only('email', 'password');**

**if (Auth::attempt($credentials)) {**

**return redirect()->intended('/');**

**}**

**return back()->withErrors([**

**'email' => 'Неверный email или пароль',**

**]);**

**}**

<?php

use Firebase\JWT\JWT;

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$email = $\_POST["email"];

$password = $\_POST["password"];

$pdo = '';

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

$stmt->execute([$email]);

$user = $stmt->fetch();

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

$secret\_key = random\_bytes(321312);

$jwt = generate\_jwt($user['id'], $secret\_key);

setcookie("access\_token", $jwt, time() + 3600, "/", "", false, true); // HttpOnly

header("Location: /dashboard.php");

exit;

} else {

echo "Неверный логин или пароль.";

}

}

Если введённые данные верны, пользователь попадает в защищённую зону приложения.

**3.3.3** Middleware и защита маршрутов

Доступ к функционалу оформления заказов и генерации ключей ограничен с помощью middleware auth. Незарегистрированные пользователи перенаправляются на страницу входа.

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

**Route::get('/cart', [CartController::class, 'index']);**

**Route::post('/order', [OrderController::class, 'store']);**

**});**

Это гарантирует, что только авторизованные пользователи могут взаимодействовать с корзиной и оформлять заказы. Также реализована проверка через JWT ключ:

$token = $\_COOKIE['access\_token'] ?? null;

if (!$token) {

header("Location: /login.php");

exit;

}

try {

$decoded = JWT::decode($token, new Key($secret\_key, 'HS256'));

$user\_id = $decoded->sub;

// Пользователь аутентифицирован

} catch (Exception $e) {

// Неверный токен

header("Location: /login.php");

exit;

}

**3.3.4** Защита CSRF, JWT и cookie

Для защиты от подделки запросов (CSRF) Laravel автоматически добавляет токены в формы. Кроме того, сессии хранятся в HttpOnly куках, что защищает данные от XSS-атак. Таким образом обеспечивается надёжная защита пользовательских данных и действий в системе. JWT (JSON Web Token) создаётся с помощью библиотеки firebase/php-jwt. В полезную нагрузку (payload) включается информация о пользователе и срок действия токена.

use Firebase\JWT\JWT;

function generate\_jwt($user\_id, $secret) {

$payload = [

'iss' => 'localhost',

'sub' => $user\_id,

'iat' => time(),

'exp' => time() + 3600

];

return JWT::encode($payload, $secret, 'HS256');

}

При выходе из системы JWT токен удаляется.

## 3.4 Работа с базой данных

Вся информация о пользователях, играх, заказах, корзине и сгенерированных ключах хранится в реляционной базе данных MySQL. Laravel предоставляет мощный ORM Eloquent, который используется для взаимодействия с таблицами и автоматического построения SQL-запросов.

**3.4.1** Структура базы данных

База данных включает несколько ключевых таблиц:

– users – содержит данные о зарегистрированных пользователях: имя, email, хэш пароля и дату регистрации;

– games – таблица с данными о доступных играх: название, описание, цена и изображение;

– carts – временное хранилище товаров, добавленных пользователем до оплаты;

– orders – таблица заказов, связанная с пользователями и ключами;

– keys – содержит лицензионные ключи, привязанные к играм и заказам.

**class Order extends Model**

**{**

**use HasFactory;**

**protected $fillable = ['user\_id'];**

**public function keys()**

**{**

**return $this->hasMany(Key::class);**

**}**

**public function user()**

**{**

**return $this->belongsTo(User::class);**

**}**

**}**

Модель реализует связи один-ко-многим (пользователь → заказы → ключи), что упрощает доступ к связанным данным.

**3.4.2** Взаимодействие с базой данных в коде

Laravel позволяет удобно работать с базой через методы Eloquent, без ручного написания SQL. Например, при создании заказа данные сохраняются следующим образом:

**$order = Order::create(['user\_id' => auth()->id()]);**

Для генерации и прикрепления ключей:

**foreach ($selectedKeys as $key) {**

**$key->update(['order\_id' => $order->id]);**

**}**

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В процессе тестирования облачного маркетплейса были выявлены и устранены различные ошибки и недочёты, касающиеся как клиентской, так и серверной части программного средства.

В ходе разработки и тестирования проекта HStore было выявлено несколько важных проблем, требующих корректировки логики и интерфейса. Одной из таких проблем стала некорректная работа корзины: на начальном этапе корзина не сохраняла добавленные товары между сессиями, а при попытке оформить заказ возникали ошибки из-за отсутствия связей между товарами и пользователем. Причина заключалась в неправильной инициализации пользовательского сеанса и отсутствии валидации в BasketController.

После внесения изменений в маршруты и добавления логики сохранения в базу данных, корзина начала функционировать корректно: товары сохраняются в таблице baskets с привязкой к user\_id, корректно отображаются в интерфейсе и доступны для оформления заказа.

Также в процессе тестирования были выявлены и устранены следующие проблемы:

* **Ошибки валидации на фронтенде**: при пустом значении в форме поддержки или неверно введённом email пользователь не получал обратной связи. После внедрения проверки и всплывающих уведомлений ошибки стали наглядными.
* **Неработающие формы поддержки**: отправленные сообщения не сохранялись в базу данных из-за отсутствия необходимых миграций. Проблема решена добавлением миграции для таблицы support\_tickets и настройкой маршрута в SupportTicketController.
* **Неправильная авторизация после регистрации**: пользователь не попадал в профиль, потому что не происходил автоматический вход. Логика регистрации была дополнена функцией Auth::login() после успешного создания пользователя.

В результате модульного и ручного тестирования была обеспечена корректная работа всех основных компонентов проекта: корзины, профиля, формы поддержки, авторизации и отображения каталога. Система демонстрирует стабильность и удобство взаимодействия с пользователем.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 5.1 Интерфейс программного средства

5.1.1 Окно регистрации и авторизации

При первом запуске веб-приложения пользователю отображается приветственное сообщение и кнопки регистрации и входа. При нажатии на кнопку регистрации пользователь заполняет форму регистрации, формы регистрации и авторизации включают в себя:

* адрес электронной почты;
* логин (регистрация);
* пароль.

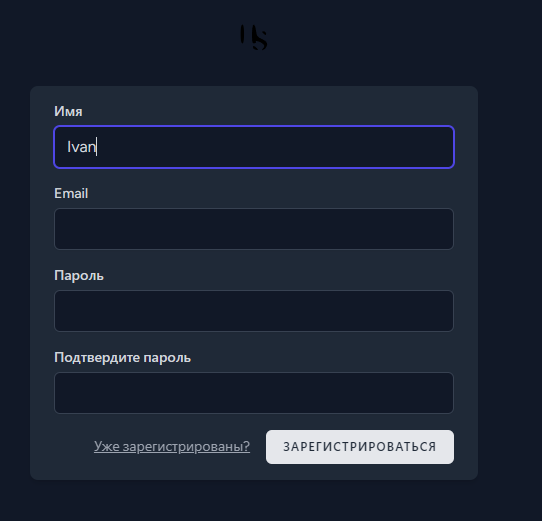


Рисунок 5.1 – Форма регистрации

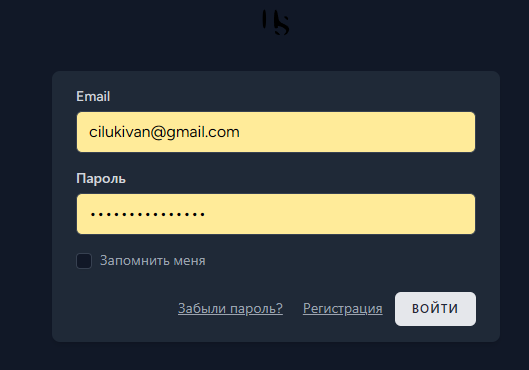


Рисунок 5.2 – Форма авторизации

## 5.2 Главное окно пользователя

После авторизации пользователю открывается интерфейс с возможностью переключения между главной страницей, магазином и корзиной.

5.2.1 Структура интерфейса

Интерфейс состоит из следующих элементов:

* Заголовок сайта: отображает название магазина и приветствие для авторизованных пользователей.
* Навигационное меню: включает ссылки на главную страницу, каталог игр, корзину, профиль пользователя и страницу поддержки.
* Кнопка выхода: позволяет пользователю завершить сеанс работы с аккаунтом.
* Каталог игр: представлен в виде карточек с изображением, названием, описанием и ценой каждой игры.
* Корзина: отображает список выбранных игр с возможностью изменения количества или удаления позиций.
* Форма оформления заказа: включает подтверждение покупки и выбор способа оплаты.
* История заказов: предоставляет пользователю список ранее приобретённых игр с возможностью повторного скачивания ключей.
* Форма обращения в поддержку: позволяет пользователю отправить сообщение администрации с возможностью прикрепления файлов.

**Доступные действия для пользователя:**

* Регистрация и вход в аккаунт: создание нового пользователя и аутентификация.
* Просмотр каталога игр: ознакомление с доступными для покупки играми.
* Добавление игр в корзину: выбор интересующих игр для последующей покупки.
* Оформление заказа: подтверждение покупки и получение ключей.
* Просмотр истории заказов: доступ к ранее приобретённым играм и ключам.
* Обращение в поддержку: отправка сообщений администрации для решения возникающих вопросов.

## 5.3 Магазин игр

Основа сайта – магазин с играми, в нем можно просмотреть список игр, сделать поиск по магазину и найти желаемые игры. На рисунке 5.3 можно просмотреть магазин.

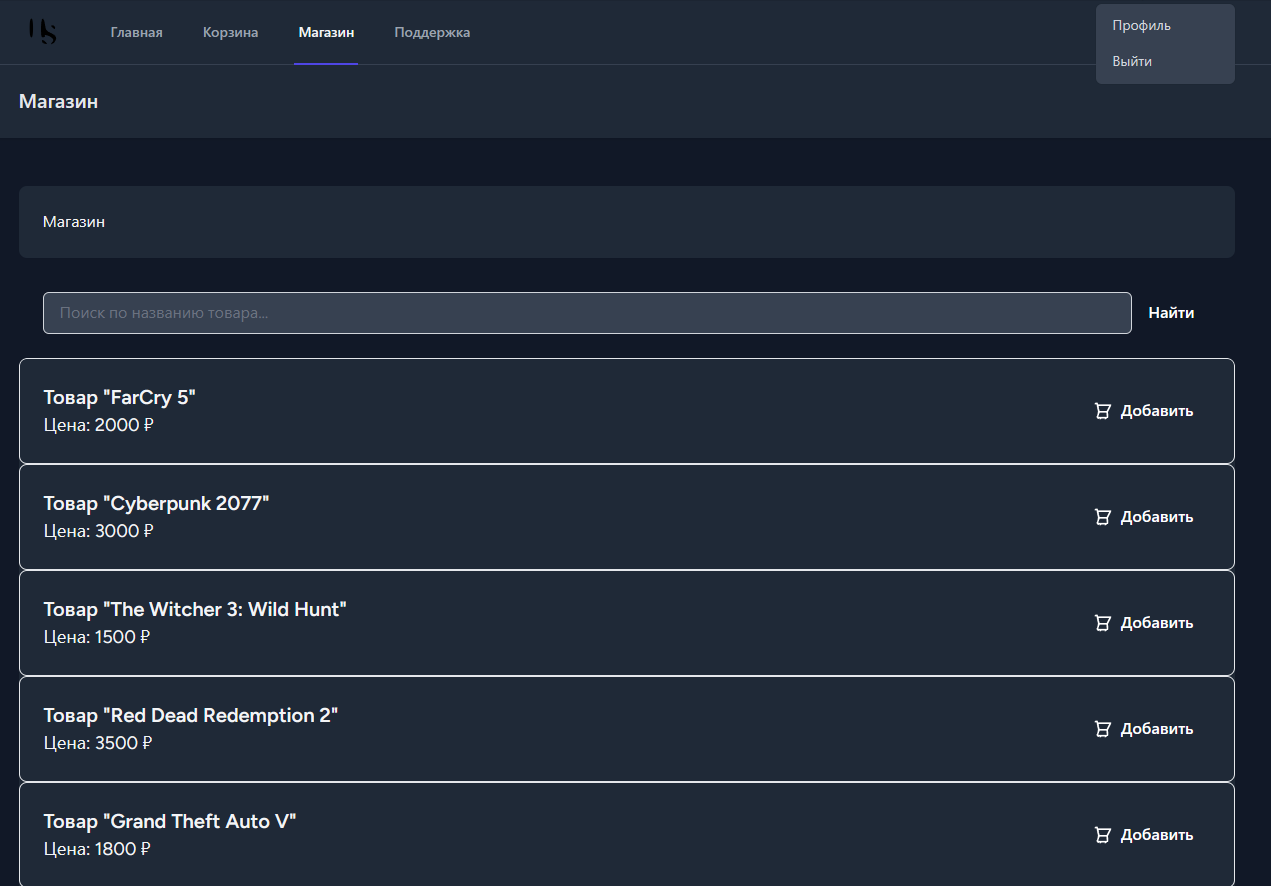


Рисунок 5.3 – Магазин

5.4 Корзина

После добавления товара в корзину можно ознакомится с ним в разделе «корзина», найти его можно в верхней части сайта. Корзина позволяет ознакомится со всем выбранным товаром, удалить его, а также оплатить и в следствие получить ключ. Интерфейс корзины предоставлен на рисунке 5.4.

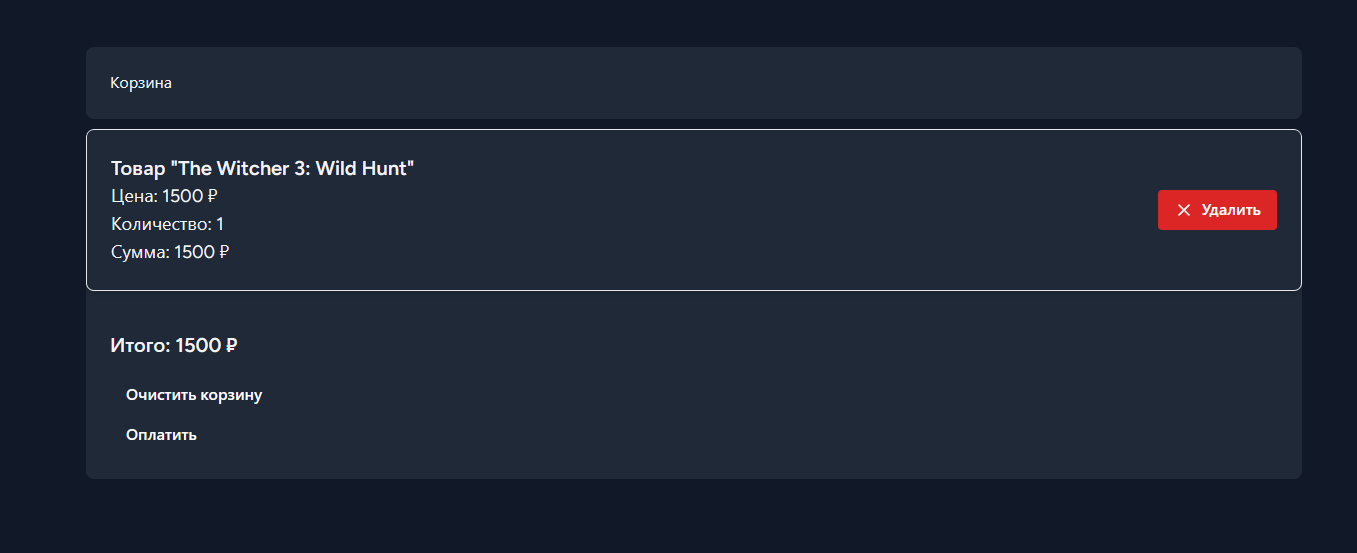


Рисунок 5.4 – Корзина

При оплате корзины будет загружен ключ с нужной игрой, активировать его можно в Steam. На картинке 5.5 можно ознакомится с документом с ключем.

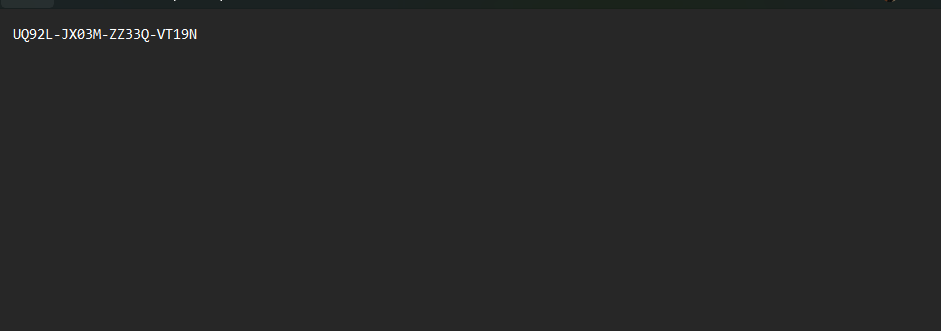


Рисунок 5.5 - Ключ

5.5 Техническая поддержка

В случае наличия технических проблем/вопросов у пользователя, пользователь может обратится в техническую поддержку. На рисунке 5.6 можно ознакомится с разделом поддержка.



Рисунок 5.6 – Поддержка

Если пользователь захочет завершить сессию, то он может нажать на кнопку «Выйти»

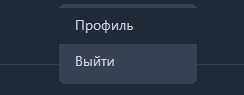


Рисунок 5.10 – Выход из аккаунта

# ЗАКЛЮЧЕНИЕ

В рамках курсового проекта был разработан полнофункциональный веб-магазин цифровых игр, обеспечивающий пользователю простой и удобный способ приобретения ключей для популярных игровых продуктов. Система включает в себя такие основные функции, как регистрация и авторизация, добавление товаров в корзину, оформление заказа, автоматическая генерация текстового файла с ключами после покупки, а также управление ассортиментом со стороны администратора.

При реализации использовались современные технологии, включая фреймворк Laravel, шаблонизатор Blade, а также подключение к базе данных MySQL. В системе предусмотрено разграничение прав доступа, валидация пользовательских данных, система уведомлений и базовая защита от некорректных действий.

Во время тестирования были выявлены и устранены ряд ошибок, связанных с некорректным сохранением корзины и проблемами при генерации файлов с ключами. После доработки проект продемонстрировал стабильную работу всех основных компонентов и корректную обработку пользовательских сценариев.

Таким образом, цели проекта были достигнуты: реализован полноценный магазин цифровых игр, который может служить основой для дальнейшего масштабирования, расширения функциональности (добавление онлайн-оплаты, отзывов, статистики) и внедрения в реальную коммерческую среду.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Власов Д. PHP 8. Разработка веб-приложений. – СПб.: Питер, 2021. – 384 с.

[2] Кузнецов С. Laravel для начинающих. – М.: БХВ-Петербург, 2022. – 288 с.

[3] Шмаков Д.Ю. SQL. Полное руководство. – М.: Диалектика, 2020. – 480 с.

[4] Троелсен Э. MySQL. Базы данных от простого к сложному. – М.: ДМК Пресс, 2019. – 304 с.

[5] Документация PHP [Электронный ресурс]. – Режим доступа: https://www.php.net/manual/ru/

[6] Документация Laravel [Электронный ресурс]. – Режим доступа: https://laravel.ru/docs

[7] Документация MySQL [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/refman/8.0/ru/

# ПРИЛОЖЕНИЕ А. Исходный код программы

Routes/web.php:

<?php

use App\Http\Controllers\BasketController;

use App\Http\Controllers\ProfileController;

use App\Http\Controllers\ProductController;

use App\Http\Controllers\SupportTicketController;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider and all of them will

| be assigned to the "web" middleware group. Make something great!

|

\*/

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

return view('dashboard');

})->middleware(['auth', 'verified'])->name('dashboard');

Route::get('/shop', [ProductController::class, 'index'])->middleware(['auth'])->name('shop');

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

return view('support');

})->middleware(['auth'])->name('support');

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

Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');

Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');

Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');

Route::get('/basket', [BasketController::class, 'index'])->name('basket.index');

Route::post('/basket/add/{id}', [BasketController::class, 'add'])->name('basket.add');

Route::post('/basket/remove/{id}', [BasketController::class, 'remove'])->name('basket.remove');

Route::post('/basket/clear', [BasketController::class, 'clear'])->name('basket.clear');

Route::post('/basket/pay', [BasketController::class, 'pay'])->name('basket.pay');

Route::post('/support', [SupportTicketController::class, 'store'])->name('support.store');

});

require \_\_DIR\_\_ . '/auth.php';

app/Http/Controllers:

<?php

namespace App\Http\Controllers;

use App\Models\Product;

use App\Models\Key;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Response;

class BasketController extends Controller

{

// Отображение корзины

public function index()

{

$basket = session()->get('basket', []);

return view('basket', compact('basket'));

}

// Добавление товара в корзину

public function add($id)

{

$product = Product::findOrFail($id);

$basket = session()->get('basket', []);

if (isset($basket[$id])) {

$basket[$id]['quantity']++;

} else {

$basket[$id] = [

'name' => $product->name,

'price' => $product->price,

'quantity' => 1,

'game\_id' => $product->id // добавляем game\_id для поиска ключей

];

}

session()->put('basket', $basket);

return redirect()->back()->with('success', 'Товар добавлен в корзину!');

}

// Удаление товара из корзины

public function remove($id)

{

$basket = session()->get('basket', []);

if (isset($basket[$id])) {

unset($basket[$id]);

session()->put('basket', $basket);

}

return redirect()->back()->with('success', 'Товар удален из корзины.');

}

// Очистка корзины

public function clear()

{

session()->forget('basket');

return redirect()->back()->with('success', 'Корзина очищена.');

}

// Метод для оплаты и загрузки ключей

public function pay(Request $request)

{

// Получаем корзину

$basket = session()->get('basket', []);

$gameIds = collect($basket)->pluck('game\_id')->unique();

// Получаем ключи, которые будут использоваться (где actual = 1)

$keysQuery = Key::whereIn('game\_id', $gameIds)

->where('actual', 1);

// Получаем сами ключи

$keys = $keysQuery->get();

// Если ключей нет — ошибка

if ($keys->isEmpty()) {

return redirect()->route('basket.index')->with('error', 'Нет доступных ключей для оплаты.');

}

// Генерация содержимого txt

$keyValues = $keys->pluck('key')->toArray();

$fileContent = implode("\n", $keyValues);

$filePath = storage\_path('app/keys.txt');

file\_put\_contents($filePath, $fileContent);

// Обновляем actual = 0 для всех использованных ключей

$keysQuery->update(['actual' => 0]);

// Очищаем корзину

session()->forget('basket');

// Отдаём файл на скачивание

return response()->download($filePath, 'keys.txt', [

'Content-Type' => 'text/plain',

'Content-Disposition' => 'attachment; filename="keys.txt"',

]);

}

}

class ProductController extends Controller

{

public function index(Request $request)

{

$query = $request->input('search');

$products = Product::when($query, function ($q) use ($query) {

$q->where('name', 'like', '%' . $query . '%');

})->get();

return view('shop', compact('products', 'query'));

}

}

class ProfileController extends Controller

{

/\*\*

\* Display the user's profile form.

\*/

public function edit(Request $request): View

{

return view('profile.edit', [

'user' => $request->user(),

]);

}

/\*\*

\* Update the user's profile information.

\*/

public function update(ProfileUpdateRequest $request): RedirectResponse

{

$request->user()->fill($request->validated());

if ($request->user()->isDirty('email')) {

$request->user()->email\_verified\_at = null;

}

$request->user()->save();

return Redirect::route('profile.edit')->with('status', 'profile-updated');

}

/\*\*

\* Delete the user's account.

\*/

public function destroy(Request $request): RedirectResponse

{

$request->validateWithBag('userDeletion', [

'password' => ['required', 'current\_password'],

]);

$user = $request->user();

Auth::logout();

$user->delete();

$request->session()->invalidate();

$request->session()->regenerateToken();

return Redirect::to('/');

}

}

class SupportTicketController extends Controller

{

public function store(Request $request)

{

$request->validate([

'message' => 'required|string',

'attachment' => 'nullable|image|max:2048',

]);

$path = $request->file('attachment')?->store('support\_attachments', 'public');

SupportTicket::create([

'user\_id' => Auth::id(),

'message' => $request->message,

'attachment' => $path,

]);

return redirect()->back()->with('status', 'Обращение отправлено!');

}

}

<x-guest-layout>

<div class="mb-4 text-sm text-gray-600 dark:text-gray-400">

{{ \_\_('Это защищенная область приложения. Пожалуйста, подтвердите свой пароль, прежде чем продолжить.') }}

</div>

<form method="POST" action="{{ route('password.confirm') }}">

@csrf

<!-- Password -->

<div>

<x-input-label for="password" :value="\_\_('Пароль')" />

<x-text-input id="password" class="block mt-1 w-full"

type="password"

name="password"

required autocomplete="current-password" />

<x-input-error :messages="$errors->get('password')" class="mt-2" />

</div>

<div class="flex justify-end mt-4">

<x-primary-button>

{{ \_\_('Подтвердить') }}

</x-primary-button>

</div>

</form>

</x-guest-layout>

<x-guest-layout>

<div class="mb-4 text-sm text-gray-600 dark:text-gray-400">

{{ \_\_('Забыли пароль? Нет проблем. Просто сообщите нам свой адрес электронной почты, и мы вышлем вам ссылку для сброса пароля, которая позволит вам выбрать новый.') }}

</div>

<!-- Session Status -->

<x-auth-session-status class="mb-4" :status="session('status')" />

<form method="POST" action="{{ route('password.email') }}">

@csrf

<!-- Email Address -->

<div>

<x-input-label for="email" :value="\_\_('Email')" />

<x-text-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autofocus />

<x-input-error :messages="$errors->get('email')" class="mt-2" />

</div>

<div class="flex items-center justify-end mt-4">

<x-primary-button>

{{ \_\_('Отправить ссылку для сброса') }}

</x-primary-button>

</div>

</form>

</x-guest-layout>

<x-guest-layout>

<!-- Session Status -->

<x-auth-session-status class="mb-4" :status="session('status')" />

<form method="POST" action="{{ route('login') }}">

@csrf

<!-- Email Address -->

<div>

<x-input-label for="email" :value="\_\_('Email')" />

<x-text-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autofocus autocomplete="username" />

<x-input-error :messages="$errors->get('email')" class="mt-2" />

</div>

<!-- Password -->

<div class="mt-4">

<x-input-label for="password" :value="\_\_('Пароль')" />

<x-text-input id="password" class="block mt-1 w-full"

type="password"

name="password"

required autocomplete="current-password" />

<x-input-error :messages="$errors->get('password')" class="mt-2" />

</div>

<!-- Remember Me -->

<div class="block mt-4">

<label for="remember\_me" class="inline-flex items-center">

<input id="remember\_me" type="checkbox" class="rounded dark:bg-gray-900 border-gray-300 dark:border-gray-700 text-indigo-600 shadow-sm focus:ring-indigo-500 dark:focus:ring-indigo-600 dark:focus:ring-offset-gray-800" name="remember">

<span class="ms-2 text-sm text-gray-600 dark:text-gray-400">{{ \_\_('Запомнить меня') }}</span>

</label>

</div>

<div class="flex items-center justify-end mt-4">

@if (Route::has('password.request'))

<a class="underline text-sm text-gray-600 dark:text-gray-400 hover:text-gray-900 dark:hover:text-gray-100 rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500 dark:focus:ring-offset-gray-800" href="{{ route('password.request') }}" style="margin-right: 20px">

{{ \_\_('Забыли пароль?') }}

</a>

<a class="underline text-sm text-gray-600 dark:text-gray-400 hover:text-gray-900 dark:hover:text-gray-100 rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500 dark:focus:ring-offset-gray-800" href="{{ route('register') }}">

{{ \_\_('Регистрация') }}

</a>

@endif

<x-primary-button class="ms-3">

{{ \_\_('Войти') }}

</x-primary-button>

</div>

</form>

</x-guest-layout>

<x-guest-layout>

<form method="POST" action="{{ route('register') }}">

@csrf

<!-- Name -->

<div>

<x-input-label for="name" :value="\_\_('Имя')" />

<x-text-input id="name" class="block mt-1 w-full" type="text" name="name" :value="old('name')" required autofocus autocomplete="name" />

<x-input-error :messages="$errors->get('name')" class="mt-2" />

</div>

<!-- Email Address -->

<div class="mt-4">

<x-input-label for="email" :value="\_\_('Email')" />

<x-text-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autocomplete="username" />

<x-input-error :messages="$errors->get('email')" class="mt-2" />

</div>

<!-- Password -->

<div class="mt-4">

<x-input-label for="password" :value="\_\_('Пароль')" />

<x-text-input id="password" class="block mt-1 w-full"

type="password"

name="password"

required autocomplete="new-password" />

<x-input-error :messages="$errors->get('password')" class="mt-2" />

</div>

<!-- Confirm Password -->

<div class="mt-4">

<x-input-label for="password\_confirmation" :value="\_\_('Подтвердите пароль')" />

<x-text-input id="password\_confirmation" class="block mt-1 w-full"

type="password"

name="password\_confirmation" required autocomplete="new-password" />

<x-input-error :messages="$errors->get('password\_confirmation')" class="mt-2" />

</div>

<div class="flex items-center justify-end mt-4">

<a class="underline text-sm text-gray-600 dark:text-gray-400 hover:text-gray-900 dark:hover:text-gray-100 rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500 dark:focus:ring-offset-gray-800" href="{{ route('login') }}">

{{ \_\_('Уже зарегистрированы?') }}

</a>

<x-primary-button class="ms-4">

{{ \_\_('Зарегистрироваться') }}

</x-primary-button>

</div>

</form>

</x-guest-layout>

<x-guest-layout>

<form method="POST" action="{{ route('password.store') }}">

@csrf

<!-- Password Reset Token -->

<input type="hidden" name="token" value="{{ $request->route('token') }}">

<!-- Email Address -->

<div>

<x-input-label for="email" :value="\_\_('Email')" />

<x-text-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email', $request->email)" required autofocus autocomplete="username" />

<x-input-error :messages="$errors->get('email')" class="mt-2" />

</div>

<!-- Password -->

<div class="mt-4">

<x-input-label for="password" :value="\_\_('Пароль')" />

<x-text-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />

<x-input-error :messages="$errors->get('password')" class="mt-2" />

</div>

<!-- Confirm Password -->

<div class="mt-4">

<x-input-label for="password\_confirmation" :value="\_\_('Подтвердите пароль')" />

<x-text-input id="password\_confirmation" class="block mt-1 w-full"

type="password"

name="password\_confirmation" required autocomplete="new-password" />

<x-input-error :messages="$errors->get('password\_confirmation')" class="mt-2" />

</div>

<div class="flex items-center justify-end mt-4">

<x-primary-button>

{{ \_\_('Сбросить пароль') }}

</x-primary-button>

</div>

</form>

</x-guest-layout>

<x-guest-layout>

<div class="mb-4 text-sm text-gray-600 dark:text-gray-400">

{{ \_\_('Thanks for signing up! Before getting started, could you verify your email address by clicking on the link we just emailed to you? If you didn\'t receive the email, we will gladly send you another.') }}

</div>

@if (session('status') == 'verification-link-sent')

<div class="mb-4 font-medium text-sm text-green-600 dark:text-green-400">

{{ \_\_('A new verification link has been sent to the email address you provided during registration.') }}

</div>

@endif

<div class="mt-4 flex items-center justify-between">

<form method="POST" action="{{ route('verification.send') }}">

@csrf

<div>

<x-primary-button>

{{ \_\_('Resend Verification Email') }}

</x-primary-button>

</div>

</form>

<form method="POST" action="{{ route('logout') }}">

@csrf

<button type="submit" class="underline text-sm text-gray-600 dark:text-gray-400 hover:text-gray-900 dark:hover:text-gray-100 rounded-md focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500 dark:focus:ring-offset-gray-800">

{{ \_\_('Log Out') }}

</button>

</form>

</div>

</x-guest-layout>

<x-app-layout>

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">

{{ \_\_('Корзина') }}

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg" style="margin-bottom: 10px">

<div class="p-6 text-gray-900 dark:text-gray-100">

{{ \_\_("Корзина") }}

</div>

</div>

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">

@if(count($basket) > 0)

@foreach($basket as $id => $item)

<div

class="p-6 text-gray-900 dark:text-gray-100 border rounded-lg shadow-md bg-white dark:bg-gray-800 flex items-center justify-between mb-4">

<div>

<h2 class="text-xl font-semibold">Товар "{{ $item['name'] }}"</h2>

<p class="text-lg">Цена: {{ $item['price'] }} ₽</p>

<p class="text-lg">Количество: {{ $item['quantity'] }}</p>

<p class="text-lg">Сумма: {{ $item['price'] \* $item['quantity'] }} ₽</p>

</div>

<form method="POST" action="{{ route('basket.remove', $id) }}">

@csrf

<button

type="submit"

class="flex items-center bg-red-600 hover:bg-red-700 text-white font-semibold py-2 px-4 rounded">

<svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5 mr-2" fill="none"

viewBox="0 0 24 24" stroke="currentColor">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"

d="M6 18L18 6M6 6l12 12"/>

</svg>

{{ \_\_("Удалить") }}

</button>

</form>

</div>

@endforeach

<div class="p-6 text-gray-900 dark:text-gray-100">

<p class="text-xl font-semibold mb-4">Итого:

{{ collect($basket)->sum(fn($item) => $item['price'] \* $item['quantity']) }} ₽

</p>

<form method="POST" action="{{ route('basket.clear') }}">

@csrf

<button

type="submit"

class="bg-gray-700 hover:bg-gray-800 text-white font-semibold py-2 px-4 rounded">

Очистить корзину

</button>

</form>

<form id="pay-form" method="POST" action="{{ route('basket.pay') }}">

@csrf

<button

type="submit"

class="bg-gray-700 hover:bg-gray-800 text-white font-semibold py-2 px-4 rounded">

Оплатить

</button>

</form>

</div>

@else

<div class="p-6 text-gray-900 dark:text-gray-100">

Корзина пуста

</div>

@endif

</div>

</div>

</div>

</x-app-layout>

<x-app-layout>

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">

{{ \_\_('Главная') }}

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">

<div class="p-6 text-gray-900 dark:text-gray-100">

{{ \_\_("Тут будет информация") }}

</div>

</div>

</div>

</div>

</x-app-layout>

<x-app-layout>

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">

{{ \_\_('Магазин') }}

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg" style="margin-bottom: 10px">

<div class="p-6 text-gray-900 dark:text-gray-100">

{{ \_\_("Магазин") }}

</div>

</div>

<div class="p-6">

<form method="GET" action="{{ route('shop') }}" class="flex items-center gap-2">

<input

type="text"

name="search"

value="{{ request('search') }}"

placeholder="Поиск по названию товара..."

class="w-full px-4 py-2 border border-gray-300 rounded-md dark:bg-gray-700 dark:text-white"

/>

<button

type="submit"

class="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2 rounded-md font-semibold">

Найти

</button>

</form>

</div>

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">

@if(count($products)>0)

@foreach($products as $product)

<div

class="p-6 text-gray-900 dark:text-gray-100 border rounded-lg shadow-md bg-white dark:bg-gray-800 flex items-center justify-between">

<div>

<h2 class="text-xl font-semibold">Товар "{{ $product->name }}"</h2>

<p class="text-lg">Цена: {{ $product->price }} ₽</p>

</div>

<form method="POST" action="{{ route('basket.add', $product->id) }}">

@csrf

<button

type="submit"

class="flex items-center bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2 px-4 rounded">

<svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5 mr-2" fill="none"

viewBox="0 0 24 24" stroke="currentColor">

<path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"

d="M3 3h2l.4 2M7 13h10l4-8H5.4M7 13L5.4 5M7 13l-1.34 5.34A1 1 0 007.6 20h8.8a1 1 0 001-1.33L17 13M9 21h.01M15 21h.01"/>

</svg>

{{ \_\_("Добавить") }}

</button>

</form>

</div>

@endforeach

@else

<div class="p-6 text-gray-900 dark:text-gray-100">

Каталог пуст

</div>

@endif

</div>

</div>

</div>

</div>

</x-app-layout>

<x-app-layout>

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">

{{ \_\_('Поддержка') }}

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

<div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">

<div class="p-6 text-gray-900 dark:text-gray-100">

@if (session('status'))

<div class="mb-4 font-medium text-sm text-green-600 dark:text-green-400">

{{ session('status') }}

</div>

@endif

@if ($errors->any())

<div class="mb-4 text-sm text-red-600 dark:text-red-400">

<ul class="list-disc list-inside">

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif

<form method="POST" action="{{ route('support.store') }}" enctype="multipart/form-data">

@csrf

<div class="mb-4">

<label for="message" class="block text-sm font-medium text-gray-700 dark:text-gray-300">

Сообщение

</label>

<textarea id="message" name="message" rows="4" required

class="mt-1 block w-full rounded-md dark:bg-gray-700 border-gray-300 dark:border-gray-600 shadow-sm focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50">{{ old('message') }}</textarea>

</div>

<div class="mb-4">

<label for="attachment" class="block text-sm font-medium text-gray-700 dark:text-gray-300">

Прикрепить изображение

</label>

<input type="file" name="attachment" id="attachment" accept="image/\*"

class="mt-1 block w-full text-sm text-gray-500 dark:text-gray-300 file:mr-4 file:py-2 file:px-4

file:rounded file:border-0 file:text-sm file:font-semibold

file:bg-indigo-50 dark:file:bg-indigo-900 file:text-indigo-700 dark:file:text-indigo-100

hover:file:bg-indigo-100 dark:hover:file:bg-indigo-800">

</div>

<x-primary-button>Отправить</x-primary-button>

</form>

</div>

</div>

</div>

</div>

</x-app-layout>