**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc194915909)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc194915910)

[2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 10](#_Toc194915911)

[3. ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ 15](#_Toc194915912)

[4. РАЗРАБОТКА БАЗЫ ДАННЫХ ИНТЕРНЕТ-ПРИЛОЖЕНИЯ 19](#_Toc194915913)

[5. WEB-ПРИЛОЖЕНИЕ САЙТА ФИРМЫ ТОРГУЮЩЕЙ ЗАПЧАСТЯМИ ДЛЯ АВТОМОБИЛЕЙ 24](#_Toc194915914)

[6. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 33](#_Toc194915915)

[ЗАКЛЮЧЕНИЕ 36](#_Toc194915916)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 38](#_Toc194915917)

[ЛИСТИНГ А 40](#_Toc194915918)

# **ВВЕДЕНИЕ**

В последние десятилетия интернет стал неотъемлемой частью нашей повседневной жизни, в том числе и в сфере торговли. Особенно это актуально для отрасли, связанной с продажей автомобильных запчастей. Учитывая быстрое развитие электронной коммерции и стремление пользователей к удобству, многие компании перешли на онлайн-продажу товаров, в том числе автозапчастей. В связи с этим возникает необходимость в создании качественных и удобных web-интерфейсов, которые обеспечивают пользователям доступ к информации, возможности поиска и заказа товаров, а также возможности для удобной оплаты и отслеживания заказов.

Проектирование web-интерфейса для сайта фирмы, торгующей запчастями для автомобилей, представляет собой важную задачу, поскольку от качества интерфейса зависит не только удобство пользователя, но и успешность бизнеса. Правильное и продуманное проектирование интерфейса влияет на восприятие компании, повышает лояльность клиентов и способствует увеличению объема продаж. Важно, чтобы веб-сайт был не только визуально привлекательным, но и удобным для навигации, с возможностью быстрого поиска необходимых товаров и оформления заказов.

Целью данной курсовой работы является разработка и реализация web-интерфейса для сайта фирмы, специализирующейся на продаже автозапчастей. В рамках работы будет проведено проектирование интерфейса, выбранных компонентов и функционала, а также рассмотрены вопросы выбора технологий для реализации, обеспечения безопасности и оптимизации пользовательского опыта. Особое внимание будет уделено разработке удобной системы поиска и фильтрации товаров, оформления заказов и управления данными пользователей и товарами.

Таким образом, основными задачами работы являются:

* анализ потребностей пользователей и особенностей бизнеса;
* проектирование структуры и функционала интерфейса;
* выбор технологий для разработки;
* реализация и тестирование интерфейса сайта.

Актуальность данной работы заключается в том, что качественное проектирование и реализация web-интерфейса является важным инструментом для развития онлайн-бизнеса и повышения конкурентоспособности компании на рынке автозапчастей.

# **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 Состояние вопроса**

Сфера продаж автозапчастей в последние годы активно развивается, что связано с ростом числа владельцев автомобилей, увеличением объема рынка и активным переходом бизнеса в интернет-среду. В связи с этим, компании, занимающиеся продажей автозапчастей, все чаще обращаются к созданию веб-сайтов для онлайн-продаж, что позволяет расширить рынок сбыта, улучшить доступность товаров и улучшить взаимодействие с клиентами.

Процесс выбора и приобретения автозапчастей часто становится для конечных потребителей сложной задачей, так как рынок наполнен большим количеством различных производителей, моделей и видов запчастей. Веб-сайт с удобным интерфейсом может значительно упростить этот процесс, позволяя пользователю быстро и точно найти необходимую деталь, а также ознакомиться с подробными характеристиками и условиями доставки.

На текущий момент большинство крупных компаний и автосервисов, торгующих запчастями, уже имеют свои онлайн-магазины, однако качество веб-интерфейсов и функционала сильно варьируется. Важным аспектом является создание интерфейса, который будет учитывать потребности различных групп пользователей, таких как автолюбители, автосервисы и администраторы сайта.

Был проанализирован рынок сайтов, предоставляющих услуги продажи автозапчастей, сайты «autocompas», «mirzap» и «avtoall».

«autocompas» — предоставляют удобный и понятный интерфейс, блок поиска товаров по автомобилю, вин коду и т.п. Так же имеет поиск запчастей по артикулу, красивый каталог товаров, предоставляющий все расходники, ходовые и другие запчасти на автомобиль. Ещё есть функция по подбору автозапчастей от экспертов, которая помогает не знающему пользователю, подобрать нужные ему детали.

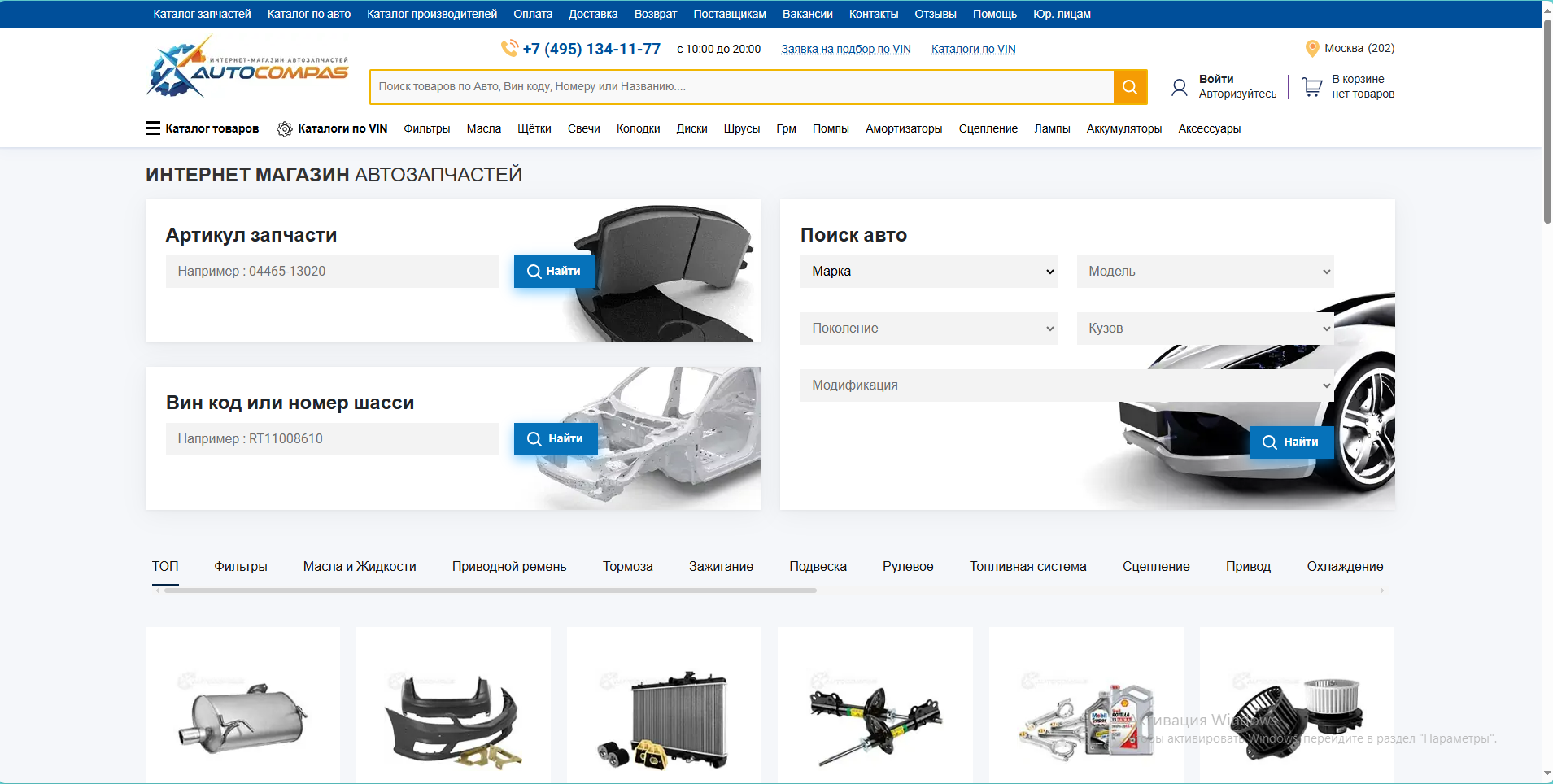


Рисунок 1 — Сайт "autocompas"

«mirzap» — предоставляет немного улучшенный дизайн сайта в отличии от «autocompas» и «avtoall», имея округлённые, приятные для глаза блоки, так же имеется поиск по вин коду, по марке автомобиля и имеет удобный и широкий каталог товаров, как и в «autocompas». Преимущества данного сайта в том, что он предоставляет продажу оригинальных автозапчастей оптом, в то время как в «autocompas» и «avtoall», такого нет, но и на данном сайте имеются минусы, такие как, отсутствие консультации в подборе автозапчастей пользователя как в «autocompas».

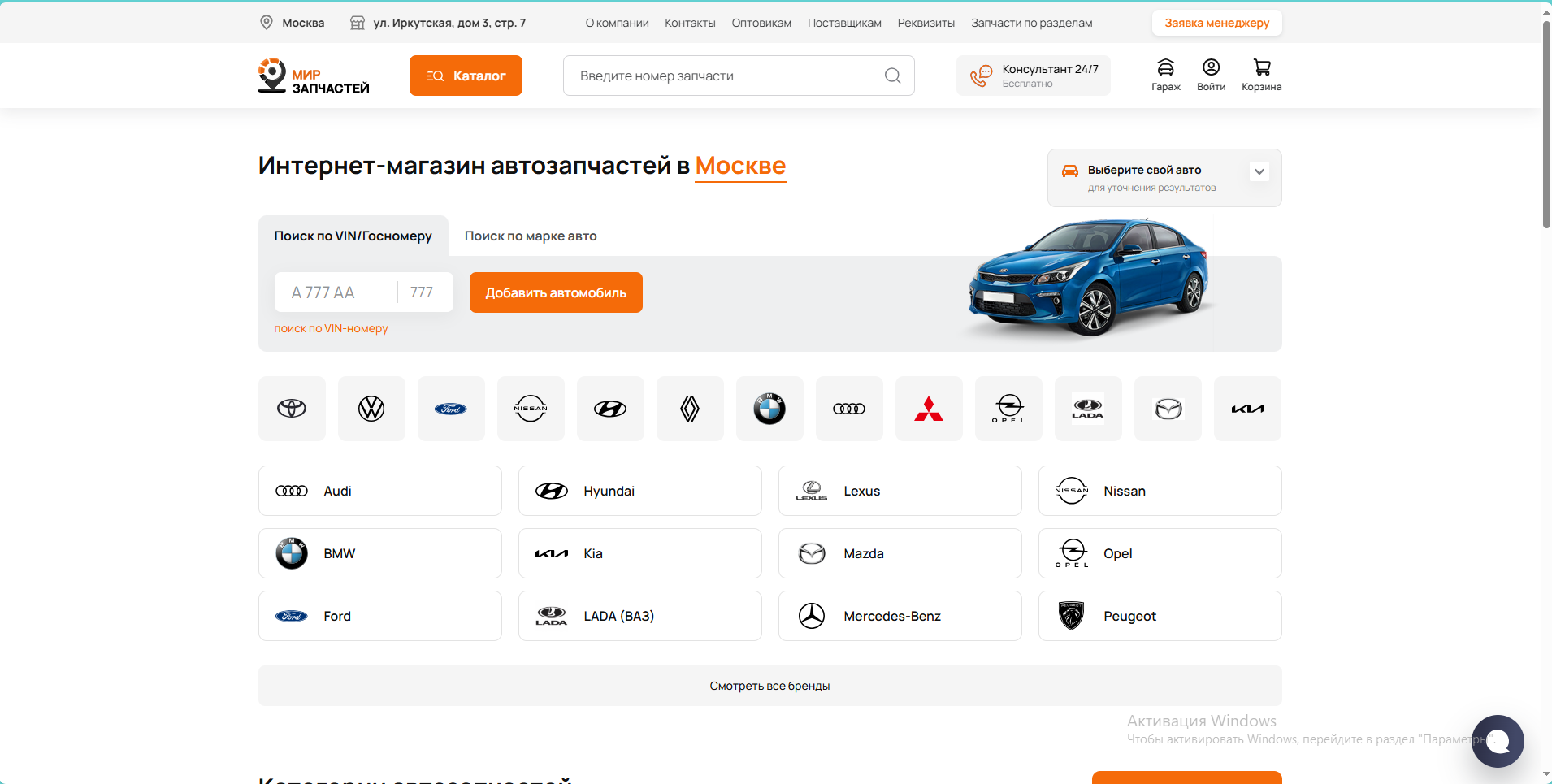


Рисунок 2 — Сайт "mirzap"

«avtoall» — предоставляет ужаснейший интерфейс сайта по сравнению с «autocompas» и «mirzap», имеются ненужные кнопки, которые не работаю, блок «каталог товаров» вообще съезжает при наведении на него. Сам сайт работает с очень большой задержкой, текст на сайте не стилизован, выглядит некорректно и вообще не подходит к данному сайту.

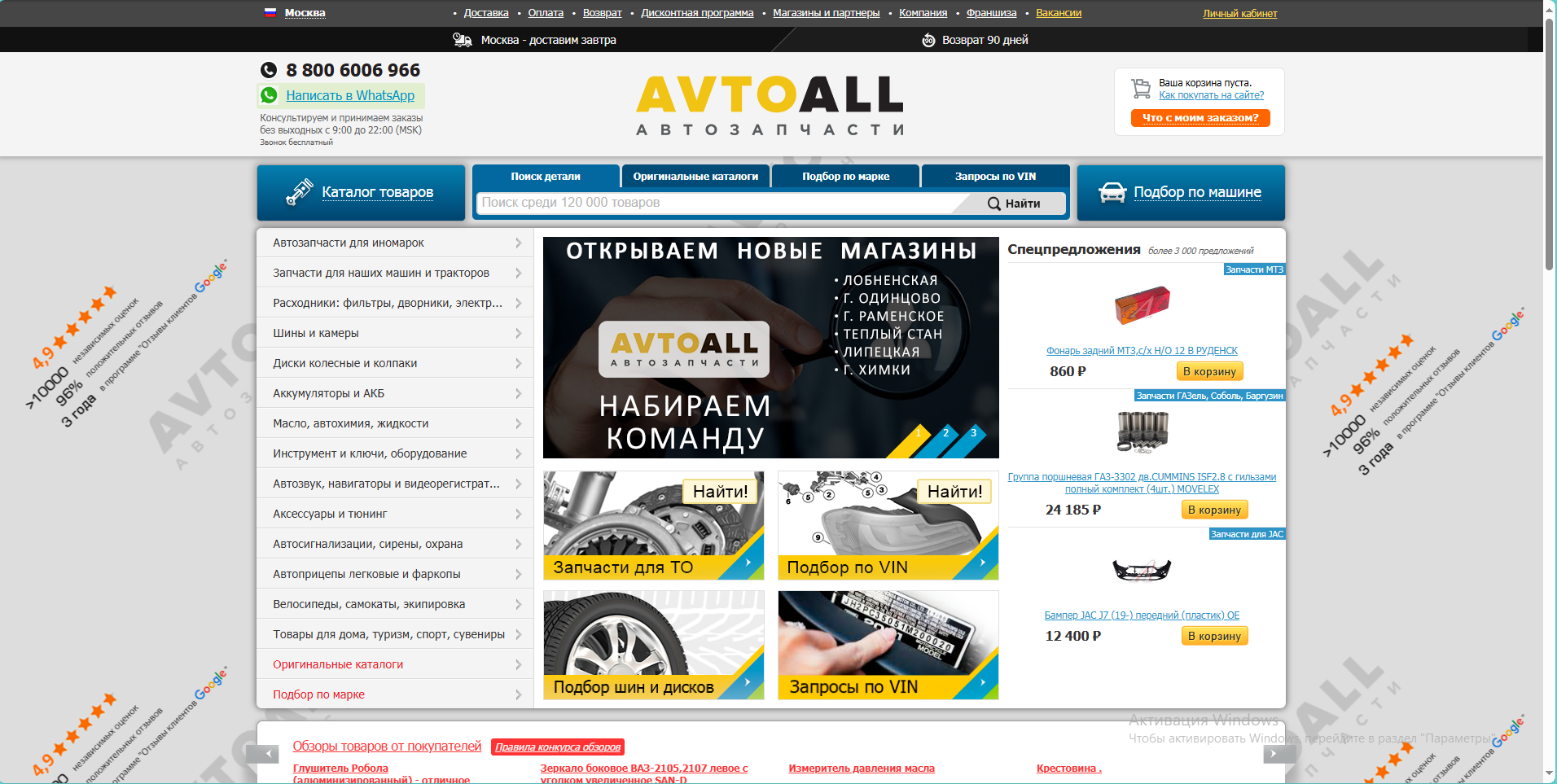


Рисунок 3 — Сайт "avtoall"

Современные технологии разработки позволяют создать динамичные и адаптивные сайты с функционалом поиска, фильтрации, оплаты и отслеживания заказов. В то же время, существует необходимость в интеграции с внешними системами — для работы с логистикой, платежными системами, а также для обеспечения безопасности персональных данных пользователей.

**1.2 Актуальность и цель работы**

Актуальность: с развитием интернет-коммерции, продажа автозапчастей через веб-сайты стала важным и необходимым инструментом для компаний, работающих в этой области. В условиях высокой конкуренции, качественный и удобный интерфейс является важным фактором, определяющим успех бизнеса. Удобство и интуитивность интерфейса, быстрота поиска и оформления заказа, а также безопасность и надежность системы — все это критически важно для привлечения клиентов и их удержания. Поэтому проектирование и реализация web-интерфейса для сайта фирмы, торгующей автозапчастями, является актуальной задачей, которая поможет бизнесу улучшить взаимодействие с пользователями и повысить его эффективность.

Цель работы: целью курсовой работы является проектирование и реализация web-интерфейса для сайта фирмы, торгующей автозапчастями, который будет учитывать потребности всех категорий пользователей и обеспечивать эффективное взаимодействие с системой, включая поиск и оформление заказов, управление товарами и заказами, а также обработку платежей и доставку.

Для достижения этой цели необходимо:

* проанализировать требования к web-интерфейсу с учетом различных групп пользователей;
* спроектировать структуру сайта, определить его основные разделы и функциональные компоненты;
* реализовать интерфейс с использованием современных веб-технологий;
* обеспечить безопасность данных пользователей и защиту их личной информации;
* провести тестирование системы на удобство и функциональность.

Решение этих задач позволит разработать функциональный, удобный и безопасный веб-интерфейс для онлайн-продаж автозапчастей, способствующий улучшению работы компании и повышению удовлетворенности клиентов.

# **2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Описание области применения и исходных данных приложения**

**Область применения:** разрабатываемый web-интерфейс предназначен для использования в интернет-магазине автозапчастей, торгующем запчастями для автомобилей различных марок и моделей. Сайт будет служить основной платформой для онлайн-продаж, обеспечивая покупателям удобный доступ к широкому ассортименту автозапчастей, возможностям поиска и фильтрации товаров, а также функционалу оформления заказов и оплаты.

Платформа будет обслуживать как индивидуальных клиентов, так и бизнес-клиентов (автосервисы и оптовых покупателей), предоставляя различные уровни доступа и функционала.

**Исходные данные:**

* перечень автозапчастей (название, описание, цена, характеристики);
* информация о товарах, включая совместимость с марками и моделями автомобилей;
* данные о поставках и наличии товара на складе;
* информация о клиентах (имя, фамилия, адрес, контактный телефон, история заказов);
* список доступных способов оплаты и доставки;
* данные о системах доставки (курьерские компании, почта и т.д.).

**2.2 Требования к пользовательским интерфейсам**

**Общие требования:**

* **интуитивно понятный и удобный интерфейс** с минимальной нагрузкой на пользователя;
* **адаптивный дизайн**, чтобы сайт корректно отображался на разных устройствах (ПК, планшетах, смартфонах);
* **многоязычность** (например, русский и английский языки интерфейса);
* **высокая производительность** (быстрое реагирование на действия пользователя, минимизация времени ожидания).

**Основные страницы интерфейса:**

1. **Главная страница**: должна содержать информацию о компании, основные предложения, баннеры с акциями, ссылки на популярные категории товаров и новинки.
2. **Каталог товаров**: должен включать фильтры по марке и модели автомобиля, категории товаров, сортировку по цене и другим параметрам.
3. **Карточка товара**: на странице товара должна быть представлена детальная информация, включая фото, описание, цену, характеристики и доступность на складе.
4. **Корзина покупок**: пользователь должен легко видеть товары в корзине, изменять количество, удалять товары, а также переходить к оформлению заказа.
5. **Оформление заказа**: включает поля для ввода данных клиента, выбора способа доставки и оплаты, а также подтверждение заказа.
6. **Личный кабинет**: для зарегистрированных пользователей с историей заказов, возможностью изменения личных данных и восстановления пароля.
7. **Административная панель**: для управления товарами, заказами, клиентами, а также генерации отчетов и анализа данных.

**Интерфейс должен поддерживать:**

* **поиск товаров** по названию, артикулу, марке и модели автомобиля;
* **фильтрацию товаров** по различным критериям (тип детали, цена, производитель, наличие на складе);
* **уведомления** (о подтверждении заказа, доставке и т.д.).

**2.3 Требования к аппаратным и программным интерфейсам**

**Аппаратные требования:**

* **серверная инфраструктура** для размещения сайта и обработки запросов пользователей. Рекомендуемые параметры для хостинга:
  + процессор с многозадачностью и высокой тактовой частотой;
  + минимум 8 GB оперативной памяти;
  + SSD-хранилище для быстрой работы с базой данных;
  + доступ к интернет-каналу с пропускной способностью не менее 100 Мбит/с для стабильной работы.
* **Клиентские устройства**:
  + для пользователей: ПК, ноутбуки, смартфоны и планшеты с браузерами, поддерживающими HTML, CSS и JavaScript;
  + для администраторов: ПК с доступом в интернет и современным браузером (Google Chrome, Firefox, Edge).

**Программные требования:**

* **операционная система сервера**: Windows 10;
* **веб-сервер**: Python;
* **база данных**: SQLite;
* **языки программирования**: HTML, CSS, JavaScript;
* **интеграция с платежными системами**: SberPay, MirPay, СПБ, Яндекс.Деньги, и другие;
* **система контроля версий**: Git;
* **система автоматизированного тестирования**: для обеспечения качества кода.

**2.4 Требования к пользователям продукта**

**Типы пользователей:**

1. **Гости (не зарегистрированные пользователи)**:
   * просмотр товаров и их характеристик;
   * добавление товаров в корзину;
   * оформление заказа без регистрации (с возможностью ввести данные вручную).
2. **Зарегистрированные пользователи**:
   * доступ к личному кабинету;
   * история заказов;
   * возможность отслеживания статуса текущих заказов;
   * управление личными данными (адрес, телефон, email);
   * уведомления о новых поступлениях товаров, акциях и скидках.
3. **Администраторы**:
   * управление каталогом товаров (добавление, редактирование, удаление товаров);
   * управление заказами (изменение статуса заказов, отправка уведомлений клиентам);
   * управление пользователями (блокировка, изменение данных);

**Требования к пользователям:**

* пользователи должны иметь базовые навыки работы с интернет-браузерами и способность совершать онлайн-покупки;
* для оформления заказа необходимы базовые контактные данные: имя, адрес доставки, контактный телефон.

**2.5 Функции продукта**

**Основные функции:**

1. **Поиск и фильтрация товаров**:
   * поиск по наименованию, артикулу, марке автомобиля, типу запчасти;
   * фильтрация по характеристикам, цене, наличию и производителю.
2. **Просмотр и выбор товара**:
   * детальная информация о товаре (фото, описание, совместимость с автомобилями, характеристики);
   * возможность добавления товара в корзину.
3. **Оформление заказа**:
   * ввод контактных данных и данных для доставки;
   * выбор способа оплаты (кредитная карта, электронные кошельки и другие);
   * подтверждение заказа и его обработка.
4. **Личный кабинет пользователя**:
   * управление данными профиля (изменение адреса, телефона, пароля);
   * история заказов и статусы текущих заказов.
5. **Административная панель**:
   * управление товарами, заказами и клиентами;
6. **Обработка заказов**:
   * изменение статуса заказа (обработан, отправлен, доставлен);

**2.6 Ограничения**

1. **Ограничения по времени отклика**:
   * время отклика сайта должно не превышать 3 секунд при стандартных запросах;
   * время отклика при оформлении заказа — не более 5 секунд.
2. **Ограничения по количеству пользователей**:
   * сайт должен обеспечивать одновременное обслуживание до 500 пользователей.
3. **Ограничения по функционалу**:
   * на первом этапе реализации не предусмотрены функции оплаты через криптовалюты;
   * осуществляется базовая интеграция с несколькими платёжными системами (SberPay, MirPay, СПБ, ЯндексДеньги и т.д.);
   * прототип сайта не включает в себя интеграцию с мобильным приложением.
4. **Ограничения по безопасности**:
   * данные клиентов должны быть защищены в соответствии с требованиями GDPR (General Data Protection Regulation) и локальными законами о защите персональных данных;
   * сайт должен быть защищен от основных видов атак (SQL-инъекции, XSS, CSRF).
5. **Ограничения по доступности**:
   * сайт должен быть доступен не менее чем 99.9% времени в месяц.

Обновления системы не должны нарушать работу веб-сайта.

# **3. ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ**

В данном разделе обоснованы выборы инструментальных средств, которые будут использоваться для разработки web-интерфейса сайта фирмы, торгующей автозапчастями. Выбранные технологии включают языкипрограммирования CSS, HTML, сервер баз данных SQLite, архитектуру «клиент-сервер» и использование языка Python для хостинга сайта.

**3.1 Используемые языки программирования: CSS, HTML**

HTML (HyperText Markup Language) — это язык разметки, используемый для создания структуры веб-страниц. HTML служит основой всех веб-сайтов, определяя их структуру, в том числе текстовый контент, изображения, ссылки и другие элементы.

Почему выбран HTML?

* + Основной стандарт веб-разработки: HTML является стандартом для всех веб-страниц. Он поддерживается всеми браузерами и устройствами, что гарантирует совместимость с широким спектром платформ.
  + Простота и понятность: HTML — это основа для веб-разработки, и его использование необходимо для любой веб-страницы, что делает его универсальным инструментом для создания структуры сайта.

CSS (Cascading Style Sheets) — язык стилей, отвечающий за внешний вид веб-страниц. CSS позволяет задавать шрифты, цвета, отступы, выравнивание и другие параметры, определяющие визуальное оформление сайта.

Почему выбран CSS?

* + Адаптивность: CSS является основным инструментом для создания адаптивного дизайна. С помощью CSS можно реализовать такие функции, как изменение оформления в зависимости от размера экрана устройства (ПК, планшет, мобильный телефон).
  + Гибкость и расширяемость: CSS позволяет быстро изменять внешний вид всех страниц сайта через центральный стиль, что облегчает управление и масштабирование проекта.
  + Широкая поддержка: CSS широко используется во всех веб-платформах и поддерживается всеми современными браузерами.

**3.2 Сервер баз данных: SQLite**

SQLite — это реляционная система управления базами данных (СУБД), которая широко используется для хранения и обработки структурированных данных. В случае с интернет-магазином автозапчастей, SQLite будет использоваться для хранения информации о товарах, заказах, пользователях, платежах и т. д.

Почему выбран SQLite?

* + Производительность и масштабируемость: SQLite — это одна из самых быстрых и масштабируемых СУБД, которая хорошо справляется с большими объемами данных и высоким количеством запросов, что критично для онлайн-магазинов с большими каталогами товаров.
  + Поддержка транзакций: SQLite поддерживает ACID-транзакции, что гарантирует целостность данных при обработке заказов и финансовых операций.
  + Популярность и поддержка: SQLite является одной из самых популярных СУБД в мире с широким сообществом, множеством инструментов и документации. Это делает разработку и поддержку системы гораздо проще.

**3.3 Архитектура «клиент-сервер»**

Архитектура «клиент-сервер» подразумевает разделение приложения на две части: клиентскую, которая взаимодействует с пользователем, и серверную, которая обрабатывает запросы и управляет данными.

Почему выбрана архитектура «клиент-сервер»?

* + Масштабируемость и производительность: при такой архитектуре нагрузка на сервер можно эффективно распределить, что позволяет масштабировать систему по мере роста числа пользователей и объема данных. Сервер обрабатывает все бизнес-логики, хранит данные и выполняет вычисления, а клиентская часть только отображает результат.
  + Гибкость и безопасность: серверная часть отвечает за обработку чувствительных данных (например, данных о клиентах и транзакциях), что повышает безопасность, так как вся критичная информация хранится и обрабатывается на сервере, а не на устройствах пользователей.
  + Легкость в обновлениях: обновления можно применять только на серверной части, не затрагивая клиентскую сторону, что делает систему более гибкой и легко поддерживаемой.

**3.4 Хостинг сайта на хостовом компьютере с помощью языка Python**

Python — это высокоуровневый язык программирования, который используется для разработки серверной части веб-приложений. Для хостинга сайта с использованием Python будет применяться один из популярных веб-фреймворков, например, Flask.

Почему выбран Python для хостинга сайта?

* + Простота и скорость разработки: Python — это язык, с которым легко работать благодаря его простой синтаксису и широким возможностям. Он позволяет быстро разрабатывать и поддерживать сложные веб-приложения.
  + Широкая поддержка веб-фреймворков: такие фреймворки, как Django и Flask, предоставляют готовые решения для создания сайтов и обработки запросов. Django, например, включает в себя систему для работы с базами данных, механизмы аутентификации пользователей, административную панель и многое другое, что позволяет значительно ускорить разработку.
  + Масштабируемость и производительность: несмотря на то, что Python традиционно считается не самым быстрым языком, использование его для веб-разработки при помощи фреймворков, таких как Django, позволяет достичь высокой производительности при правильной настройке серверной части.
  + Безопасность: Python имеет множество инструментов и библиотек для обеспечения безопасности веб-приложений (например, защита от CSRF-атак, защита от SQL-инъекций и другие).
  + Большое сообщество: Python имеет активное сообщество разработчиков, что означает наличие обширной документации и множества сторонних библиотек для различных задач.

Почему хостинг будет осуществляться на хостовом компьютере?

* Гибкость в настройке: Хостинг на своем сервере с использованием Python и Django позволяет точно настроить серверное окружение и интеграцию с другими системами, такими как базы данных и сторонние API.
* Контроль за серверной частью: Размещение сайта на собственном сервере позволяет обеспечивать полный контроль над безопасностью, конфигурацией и обновлениями системы.

# **4. РАЗРАБОТКА БАЗЫ ДАННЫХ ИНТЕРНЕТ-ПРИЛОЖЕНИЯ**

Разработка базы данных (БД) для интернет-приложения, такого как интернет-магазин автозапчастей, включает несколько этапов: концептуальное, логическое и физическое проектирование. Каждый этап направлен на создание структуры базы данных, которая будет эффективно хранить и обрабатывать данные, обеспечивая высокую производительность и безопасность.

**4.1 Концептуальное проектирование БД**

Концептуальное проектирование базы данных — это этап, на котором определяются основные сущности (объекты) и их взаимосвязи. На этом этапе создается концептуальная модель, которая описывает, какие данные будут храниться в системе и как они будут связаны между собой.

Основные сущности:

Пользователи (Users):

* идентификатор пользователя (UserID)
* имя (FirstName)
* фамилия (LastName)
* адрес электронной почты (Email)
* пароль (PasswordHash)
* телефон (Phone)
* адрес доставки (Address)
* роль (Role: клиент, администратор)

Товары (Products):

* идентификатор товара (ProductID)
* название товара (ProductName)
* описание (Description)
* цена (Price)
* категория (Category)
* производитель (Manufacturer)
* наличие на складе (StockQuantity)
* совместимость с марками и моделями автомобилей (Compatibility)

Заказы (Orders):

* идентификатор заказа (OrderID)
* идентификатор пользователя (UserID)
* дата заказа (OrderDate)
* статус заказа (Status: ожидание, обработка, доставлен)
* общая стоимость (TotalAmount)
* способ оплаты (PaymentMethod)
* адрес доставки (DeliveryAddress)

Элементы заказа (OrderItems):

* идентификатор элемента заказа (OrderItemID)
* идентификатор заказа (OrderID)
* идентификатор товара (ProductID)
* количество (Quantity)
* цена за единицу (UnitPrice)

Категории товаров (Categories):

* идентификатор категории (CategoryID)
* название категории (CategoryName)

Взаимосвязи между сущностями:

* пользователь может создавать несколько Заказов.
* заказ может содержать несколько Элементов заказа.
* товар может принадлежать к одной Категории.

Диаграмма сущностей и связей (ERD):

На этом этапе создается диаграмма, которая визуализирует сущности и их связи. Например:

* пользователь (User) → Заказ (Order)
* заказ (Order) → Элемент заказа (OrderItem)
* товар (Product) → Категория (Category)

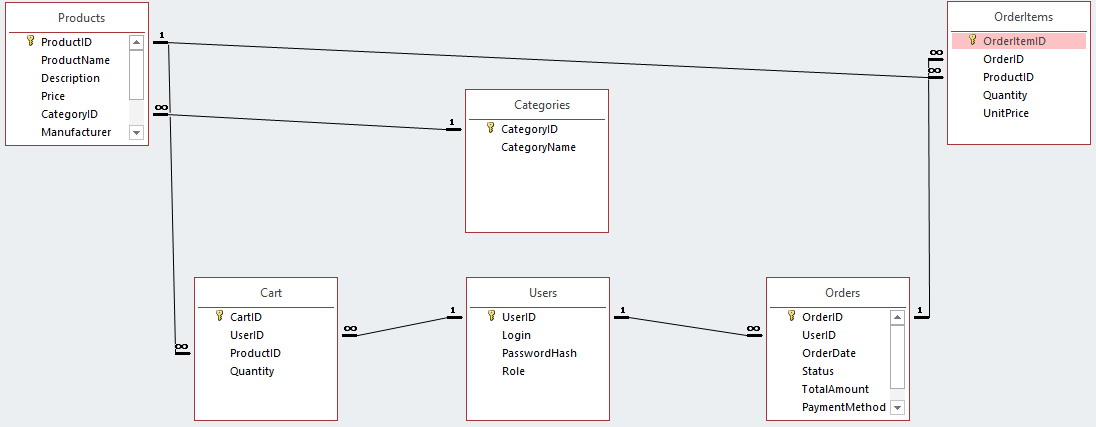


Рисунок 4 — Связи таблицы в БД

**4.2 Логическое проектирование БД**

Логическое проектирование базы данных предполагает преобразование концептуальной модели в логическую структуру, которая будет реализована в конкретной СУБД (например, SQLite). На этом этапе определяются таблицы, их атрибуты (столбцы), типы данных и ключи.

Таблицы и их атрибуты:

1. Таблица Users:

* UserID (INT, PRIMARY KEY, AUTO\_INCREMENT)
* FirstName (VARCHAR(50))
* LastName (VARCHAR(50))
* Email (VARCHAR(100), UNIQUE)
* PasswordHash (VARCHAR(255))
* Phone (VARCHAR(20))
* Address (VARCHAR(255))
* Role (ENUM('client', 'admin'))

1. Таблица Products:

* ProductID (INT, PRIMARY KEY, AUTO\_INCREMENT)
* ProductName (VARCHAR(100))
* Description (TEXT)
* Price (DECIMAL(10, 2))
* CategoryID (INT, FOREIGN KEY REFERENCES Categories(CategoryID))
* Manufacturer (VARCHAR(100))
* StockQuantity (INT)
* Compatibility (TEXT)

1. Таблица Orders:

* OrderID (INT, PRIMARY KEY, AUTO\_INCREMENT)
* UserID (INT, FOREIGN KEY REFERENCES Users(UserID))
* OrderDate (DATETIME)
* Status (ENUM('pending', 'processing', 'delivered'))
* TotalAmount (DECIMAL(10, 2))
* PaymentMethod (VARCHAR(50))
* DeliveryAddress (VARCHAR(255))

1. Таблица OrderItems:

* OrderItemID (INT, PRIMARY KEY, AUTO\_INCREMENT)
* OrderID (INT, FOREIGN KEY REFERENCES Orders(OrderID))
* ProductID (INT, FOREIGN KEY REFERENCES Products(ProductID))
* Quantity (INT)
* UnitPrice (DECIMAL(10, 2))

1. Таблица Categories:

* CategoryID (INT, PRIMARY KEY, AUTO\_INCREMENT)
* CategoryName (VARCHAR(100))

Нормализация базы данных:

На этапе логического проектирования важно провести нормализацию базы данных, чтобы избежать избыточности и аномалий. Например:

* каждая таблица должна иметь уникальный первичный ключ.
* данные должны быть разделены на логические таблицы, чтобы избежать дублирования.
* связи между таблицами должны быть реализованы через внешние ключи.

**4.3 Физическое проектирование БД**

Физическое проектирование базы данных — это этап, на котором логическая модель преобразуется в физическую структуру, которая будет реализована в конкретной СУБД (в данном случае SQLite). На этом этапе определяются индексы, типы данных, параметры хранения и оптимизация производительности.

Типы данных и индексы:

1. Типы данных:

* для числовых данных (например, цены, количество) используются типы DECIMAL и INT.
* для текстовых данных (например, имена, описания) используются типы VARCHAR и TEXT.

Для дат и времени используется тип DATETIME.

1. Индексы:

* первичные ключи (PRIMARY KEY) автоматически индексируются.
* для ускорения поиска по часто используемым полям (например, Email в таблице Users, ProductName в таблице Products) создаются дополнительные индексы (INDEX).
* для внешних ключей (FOREIGN KEY) также создаются индексы для ускорения JOIN-операций.

Оптимизация производительности:

* кэширование запросов: для часто выполняемых запросов (например, поиск товаров по категории) можно использовать кэширование.
* партиционирование таблиц: для больших таблиц (например, Orders) можно использовать партиционирование по дате для ускорения выборки данных.
* репликация и шардирование: для масштабируемости можно настроить репликацию базы данных и шардирование (разделение данных между несколькими серверами).

# **5. WEB-ПРИЛОЖЕНИЕ САЙТА ФИРМЫ ТОРГУЮЩЕЙ ЗАПЧАСТЯМИ ДЛЯ АВТОМОБИЛЕЙ**

**5.1 Входные и выходные данные**

Входные данные:

1. Данные пользователей:

* регистрация:

Пользователь вводит логин и пароль через форму в register.html. Эти данные отправляются на сервер методом POST.



Рисунок 5 — Регистрация

Обработка на сервере (app.py):

* + проверка уникальности логина:



Рисунок 6 — Проверка уникальности логина

* + хеширование пароля с помощью werkzeug.security.generate\_password\_hash:



Рисунок 7 — Хэширование пароля

* + сохранение пользователя в БД:



Рисунок 8 — Сохранение пользователя в БД

* оформление заказа:

Пользователь указывает адрес доставки и способ оплаты в форме checkout.html:



Рисунок 9 — Оформление заказа

Обработка на сервере:

* + данные извлекаются из запроса:



Рисунок 10 — Данные извлекаются из запроса

* + сохраняются в таблицу Orders вместе с общей суммой заказа.

1. Данные товаров:

* добавление товара администратором:

Форма в admin\_panel.html включает поля для названия, цены, категории и изображения:



Рисунок 11 — Добавление товара администратором

Обработка на сервере (app.py, маршрут /add\_product):

* + загрузка изображения:

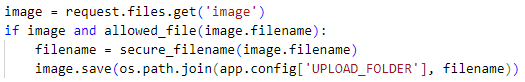


Рисунок 12 — Загрузка изображения

* + сохранение товара в БД:

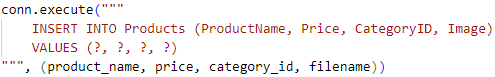


Рисунок 13 — Сохранение товара в БД

1. Поисковые запросы:

* пользователь вводит ключевые слова в строку поиска (catalog.html):



Рисунок 14 — Пользователь вводит ключевые слова в строку поиска

Обработка на сервере (app.py, маршрут /search):

* + SQL-запрос с фильтрацией:

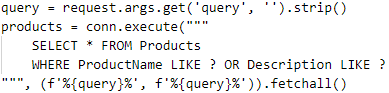


Рисунок 15 — SQL-запрос с фильтрацией

Выходные данные:

1. Каталог товаров (catalog.html):

* динамическое отображение товаров на основе данных из БД:



Рисунок 16 — Динамическое отображение товаров на основе данных из БД

Как это работает:

* + сервер получает список товаров из БД.
  + шаблон Jinja2 генерирует HTML-код для каждого товара, подставляя данные (название, цену, изображение).

1. Корзина (cart.html):

* отображение выбранных товаров и общей суммы:

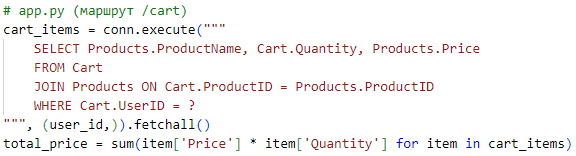


Рисунок 17 — Отображение выбранных товаров и общей суммы

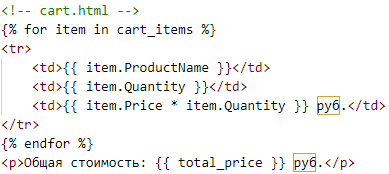


Рисунок 18 — Отображение выбранных товаров и общей суммы

1. Подтверждение заказа (order\_confirmation.html):

* отображение деталей заказа:



Рисунок 19 — Отображение деталей заказа

Как это работает:

* + после оформления заказа сервер создает запись в таблице Orders.
  + пользователь перенаправляется на страницу подтверждения, где данные заказа подтягиваются из БД.

**5.2 Структура веб-приложения**

Архитектура

Приложение построено по модели MVC (Model-View-Controller):

* Model (Модель):
  + база данных (SQLite) с таблицами:
    - Users: Хранение данных пользователей.
    - Products: Информация о товарах.
    - Orders: Заказы.
    - OrderItems: Состав заказов.
    - Categories: Категории товаров.
    - Cart: Временные данные корзины.
* View (Представление):
  + HTML-шаблоны с использованием Jinja2 для динамического отображения данных.
  + пример: catalog.html отображает товары, cart.html — корзину.
* Controller (Контроллер):
  + маршруты Flask (app.py), обрабатывающие HTTP-запросы.
  + пример: маршрут /add\_to\_cart добавляет товар в корзину.Компоненты системы

Детальное описание компонентов

1. Маршруты Flask (app.py):

* аутентификация:
  + /register:
    - принимает POST-запрос с логином и паролем.
    - проверяет уникальность логина.
    - хеширует пароль и сохраняет пользователя в БД.
* /login:
  + - проверяет введенные данные:



Рисунок 20 — Проверяет введённые данные

* работа с товарами:
  + /catalog:
    - получает все товары из БД:



Рисунок 21 — Получает все товары из БД

* + - передает их в шаблон catalog.html.
* /product/<int:product\_id>:
  + - получает детали товара по ID:



Рисунок 22 — Получает детали товара по ID

* корзина:
  + /add\_to\_cart/<int:product\_id>:
    - проверяет наличие товара на складе:



Рисунок 23 — Проверяет наличие товара на складе

* + - добавляет товар в таблицу Cart.
* /cart:
* вычисляет общую стоимость:



Рисунок 24 — Вычисляет общую стоимость

* администрирование:
  + /admin:
    - проверяет роль пользователя:



Рисунок 25 — Проверяет роль пользователя

* + - возвращает списки товаров и категорий:



Рисунок 26 — Возвращает списки товаров и категорий

2. База данных:

* схема таблицы Products:

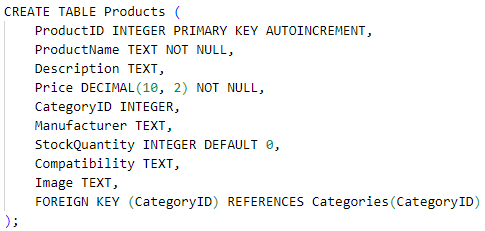


Рисунок 27 — Схема таблицы Products

* связи между таблицами:
  + каждый заказ (Orders) связан с пользователем (Users) через UserID.
  + элементы заказа (OrderItems) связаны с заказом (Orders) и товаром (Products).

3. Взаимодействие компонентов

* пример: Оформление заказа

1. Пользователь нажимает «Оформить заказ» в cart.html.
2. Сервер (app.py, маршрут /checkout):
   * + создает запись в Orders:

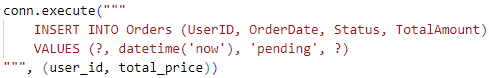


Рисунок 28 — Создает запись в Orders

* + - переносит товары из Cart в OrderItems:

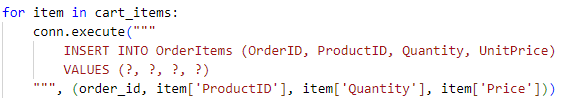


Рисунок 29 — Переносит товары из Cart в OrderItems

* + - обновляет остатки товаров:

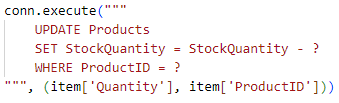


Рисунок 30 — Обновляет остатки товаров

1. Пользователь перенаправляется на order\_confirmation.html, где видит детали заказа.

**5.3 Описание алгоритма работы и SQL-запросов**

Ключевые алгоритмы

1. Добавление товара в корзину:

* проверка авторизации пользователя.
* получение количества товара из формы.
* проверка остатка на складе (StockQuantity).
* обновление таблицы Cart.

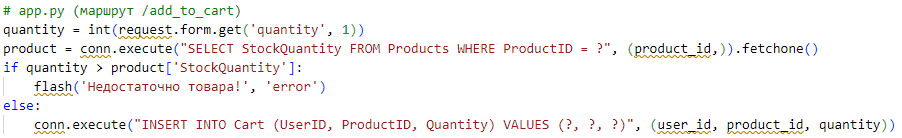


Рисунок 31 — Добавление товара в корзину

1. Оформление заказа:

* создание записи в Orders.
* перенос товаров из Cart в OrderItems.
* обновление остатков в Products.

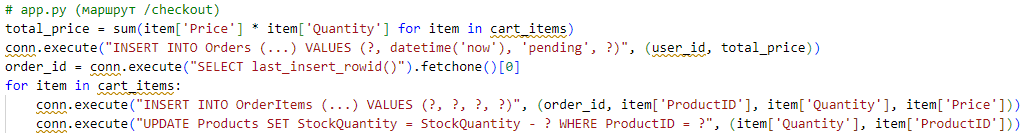


Рисунок 32 — Оформление заказа

Примеры SQL-запросов

1. Поиск товаров по категории:



Рисунок 33 — Поиск товаров по категории

1. Получение истории заказов пользователя:



Рисунок 34 — Получение истории заказов пользователя

1. Обновление роли пользователя (администратором):



Рисунок 35 — Обновление роли пользователя

1. Удаление товара из корзины:



Рисунок 36 — Удаление товара из корзины

Обработка ошибок

* некорректный ввод:



Рисунок 37 — Некорректный ввод

* попытка доступа к админ-панели без прав:

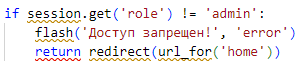


Рисунок 38 — Попытка доступа к админ-панели без прав

# **6. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

Тестирование веб-приложения проводилось для проверки корректности работы функционала, безопасности и удобства интерфейса. Основные этапы и методы тестирования представлены ниже.

**6.1 Виды тестирования**

1. Модульное тестирование (Unit Testing):

* проверка отдельных компонентов приложения:
  + хеширование паролей (generate\_password\_hash и check\_password\_hash);
  + добавление товара в корзину с проверкой остатков на складе;
  + обновление статуса заказа администратором.
* инструменты: pytest для автоматизации тестов.

1. Интеграционное тестирование (Integration Testing):

* проверка взаимодействия модулей:
  + оформление заказа: связь между таблицами Cart, Orders, OrderItems и обновление StockQuantity;
  + поиск товаров с фильтрацией по категориям.
* пример SQL-запроса:



1. Системное тестирование (System Testing):

* проверка всего приложения в сценариях:
  + регистрация → Добавление товара → Оформление заказа → Отслеживание статуса.
  + работа административной панели: добавление/удаление товаров, управление пользователями.

1. Тестирование безопасности:

* проверка на уязвимости:
  + SQL-инъекции в формах поиска и авторизации.
  + XSS-атаки через поля ввода комментариев.
* инструменты: OWASP ZAP для автоматического сканирования.

1. Юзабилити-тестирование (Usability Testing):

* оценка удобства интерфейса:
  + навигация между разделами (каталог → корзина → оформление заказа).
  + адаптивность дизайна на мобильных устройствах.

**6.2 Тестовые сценарии и результаты**

Таблица 1 — Тестовые сценарии и результаты

|  |  |  |  |
| --- | --- | --- | --- |
| Сценарий | Ожидаемый результат | Фактический результат | Статус |
| Регистрация с уникальным логином | Успешное создание аккаунта | Аккаунт создан | ✅ Успешно |
| Регистрация с существующим логином | Ошибка: "Логин уже занят" | Сообщение выводится | ✅ Успешно |
| Добавление товара в корзину | Товар отображается в корзине | Товар добавлен | ✅ Успешно |
| Оформление заказа с пустой корзиной | Ошибка: "Корзина пуста" | Сообщение выводится | ✅ Успешно |
| Обновление роли пользователя (админ) | Роль изменена в БД | Данные обновлены | ✅ Успешно |
| Ввод SQL-инъекции в поле поиска | Запрос отклонен | Уязвимость не обнаружена | ✅ Успешно |

**6.3 Выявленные дефекты и их устранение**

1. Проблема: при добавлении товара в корзину не учитывался остаток на складе.

* решение: добавлена проверка if quantity > product['StockQuantity'] в метод add\_to\_cart.

1. Проблема: Уязвимость к XSS через поле "Адрес доставки".

* решение: санитизация ввода с помощью secure\_filename и экранирование HTML-тегов в шаблонах Jinja2.

1. Проблема: некорректное отображение заказов в личном кабинете при большом количестве записей.

* решение: оптимизация SQL-запроса с использованием LIMIT и OFFSET.

**6.4 Ограничения тестирования**

* тестирование проводилось только в браузерах Chrome и Firefox;
* не охвачены все возможные комбинации фильтров в каталоге;
* не проведено нагрузочное тестирование для проверки работы при 500+ пользователях.

**6.5 Рекомендации**

1. Внедрить автоматизированные тесты для CI/CD (GitHub Actions).
2. Провести нагрузочное тестирование с использованием JMeter.
3. Расширить покрытие тестами для мобильных устройств.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана и реализована система интернет-магазина автозапчастей, включающая в себя клиентскую и административную части. Приложение предоставляет пользователям возможность регистрации, поиска и фильтрации товаров, оформления заказов, а администраторам — управление товарами, категориями и пользователями.

Основные результаты работы:

1. Реализованный функционал:

* удобный каталог товаров с фильтрацией по категориям и поиском;
* корзина с возможностью изменения количества товаров и оформления заказа;
* личный кабинет пользователя с историей заказов;
* административная панель для управления контентом и пользователями;
* защита данных (хеширование паролей, проверка прав доступа).

1. Использованные технологии:

* Backend: Python (Flask) для серверной логики;
* Frontend: HTML, CSS, Jinja2 для динамического отображения данных;
* база данных: SQLite (с возможностью масштабирования на MySQL).

1. Тестирование:

* проведено модульное, интеграционное и системное тестирование;
* устранены критические уязвимости (SQL-инъекции, XSS);
* подтверждена корректность работы основных сценариев использования.

Перспективы развития:

1. Расширение функционала:

* добавление онлайн-оплаты через интеграцию с платежными системами;
* внедрение системы отзывов и рейтингов товаров;
* разработка мобильного приложения для повышения удобства пользователей.

1. Оптимизация производительности:

* переход на более мощную СУБД (PostgreSQL или MySQL);
* кэширование данных для ускорения загрузки страниц.

1. Масштабирование:

* поддержка мультиязычности для расширения аудитории;
* развертывание на облачном хостинге для обеспечения высокой доступности.

Вывод:

Разработанное веб-приложение соответствует поставленным задачам и демонстрирует стабильную работу. Оно предоставляет удобный инструмент для онлайн-продажи автозапчастей, а его модульная архитектура позволяет легко расширять функционал в будущем. Дальнейшая работа будет направлена на улучшение пользовательского опыта и масштабирование системы.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Официальная документация и руководства:

* Flask Documentation [Электронный ресурс]. – Режим доступа: https://flask.palletsprojects.com/
* SQLite Documentation [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/docs.html
* Jinja2 Template Documentation [Электронный ресурс]. – Режим доступа: https://jinja.palletsprojects.com/

1. Учебные и справочные материалы:

* Гринберг М. Разработка веб-приложений с использованием Flask на Python. – СПб.: Питер, 2021. – 352 с.
* Лутц М. Изучаем Python. – 5-е изд. – СПб.: Символ-Плюс, 2022. – 1648 с.
* Таненбаум Э., Уэзеролл Д. Компьютерные сети. – 5-е изд. – СПб.: Питер, 2020. – 960 с.

1. Статьи и онлайн-ресурсы:

* MDN Web Docs: HTML, CSS, JavaScript [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/
* OWASP Security Guidelines [Электронный ресурс]. – Режим доступа: https://owasp.org/www-project-top-ten/
* Real Python Tutorials [Электронный ресурс]. – Режим доступа: https://realpython.com/

1. Программное обеспечение и инструменты:

* Visual Studio Code [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/
* Git – система контроля версий [Электронный ресурс]. – Режим доступа: https://git-scm.com/
* Postman – инструмент тестирования API [Электронный ресурс]. – Режим доступа: https://www.postman.com/

1. Дополнительные источники:

* ГОСТ Р 7.0.5-2008. Библиографическая ссылка. Общие требования и правила составления.
* Стандарты разработки безопасных веб-приложений (OWASP, NIST).

# **ПРИЛОЖЕНИЕ А**

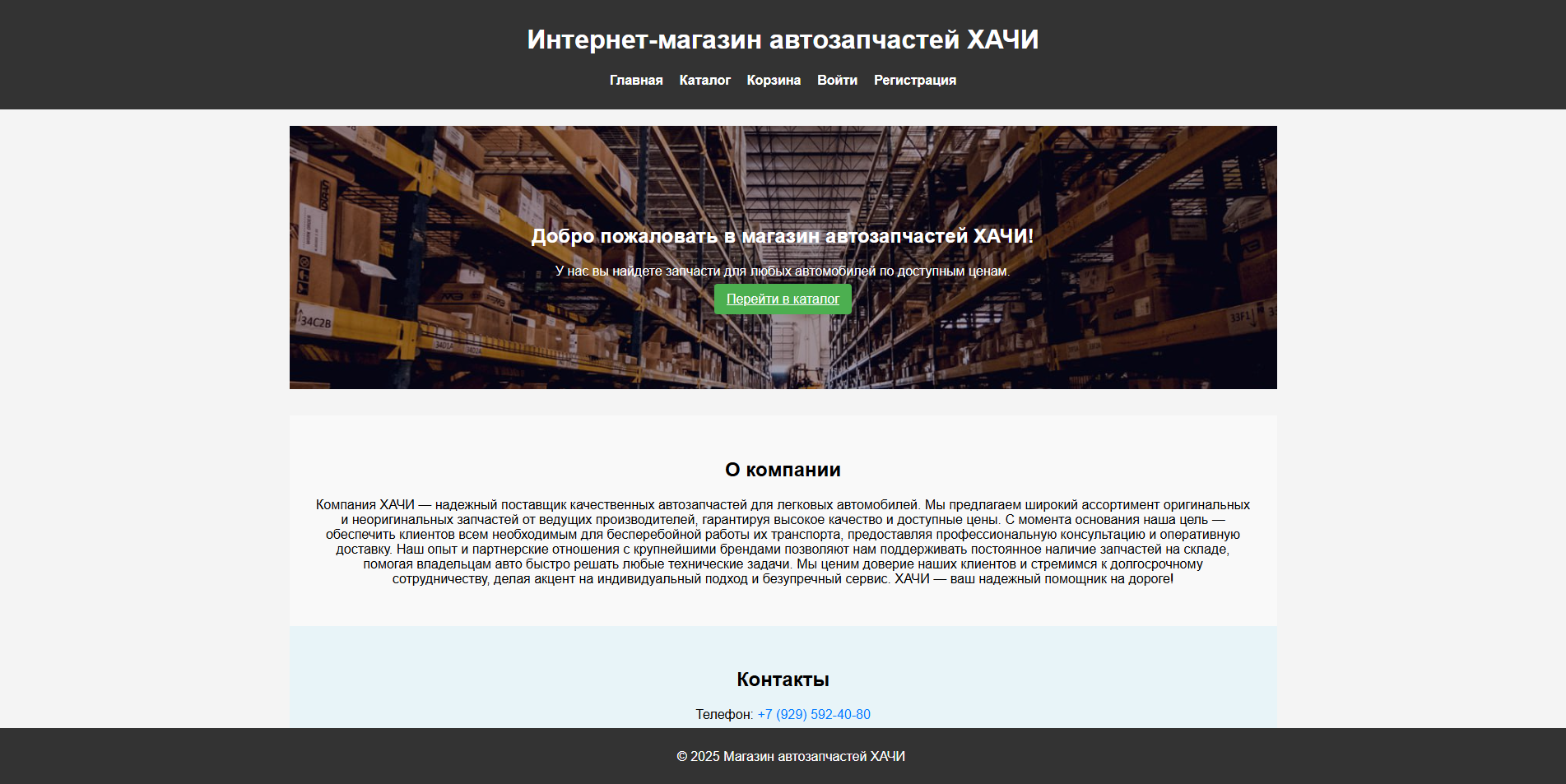


Рисунок А.1 — Страница «Главная» часть 1

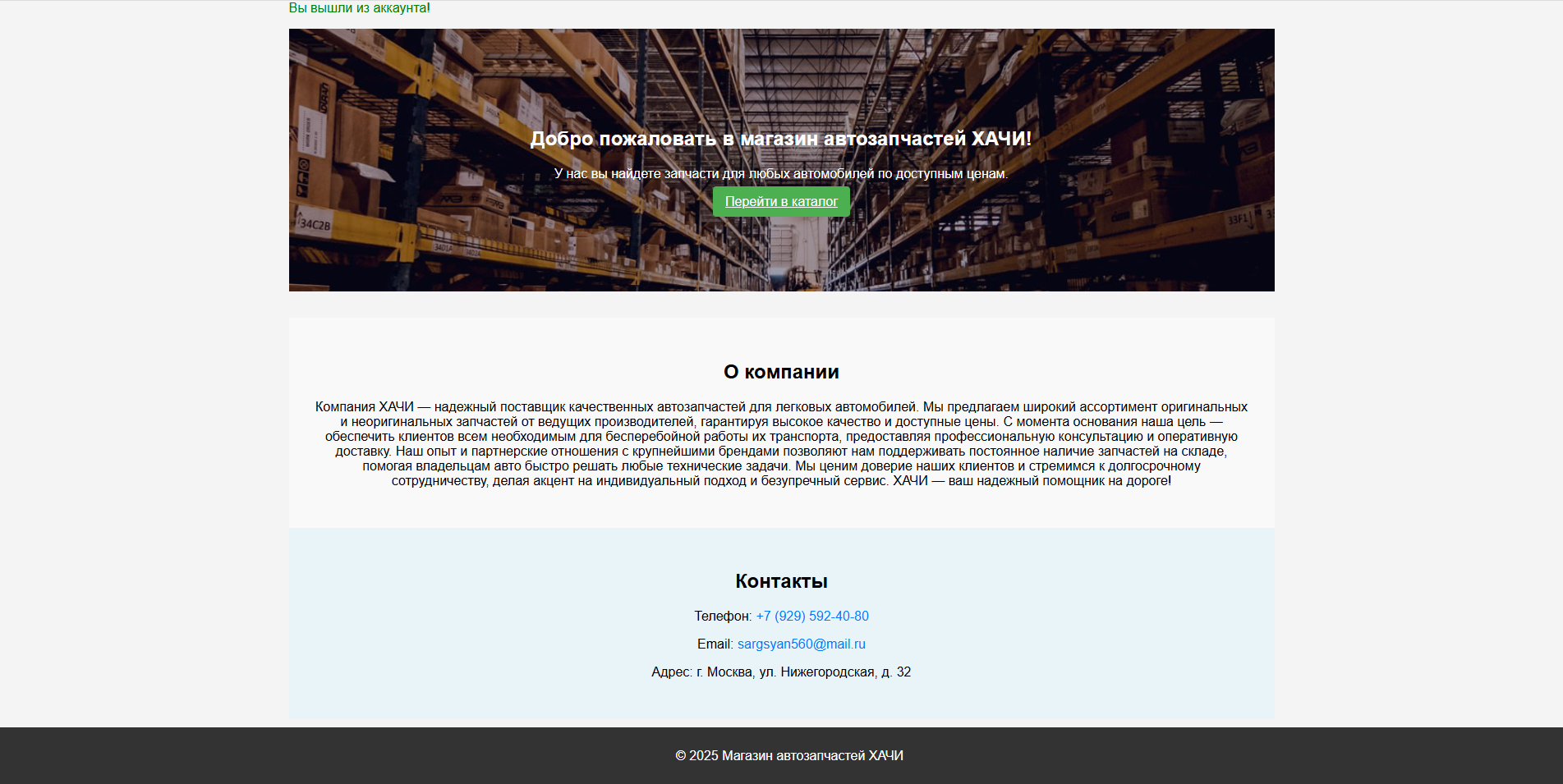


Рисунок А.2 — Страница «Главная» часть 2

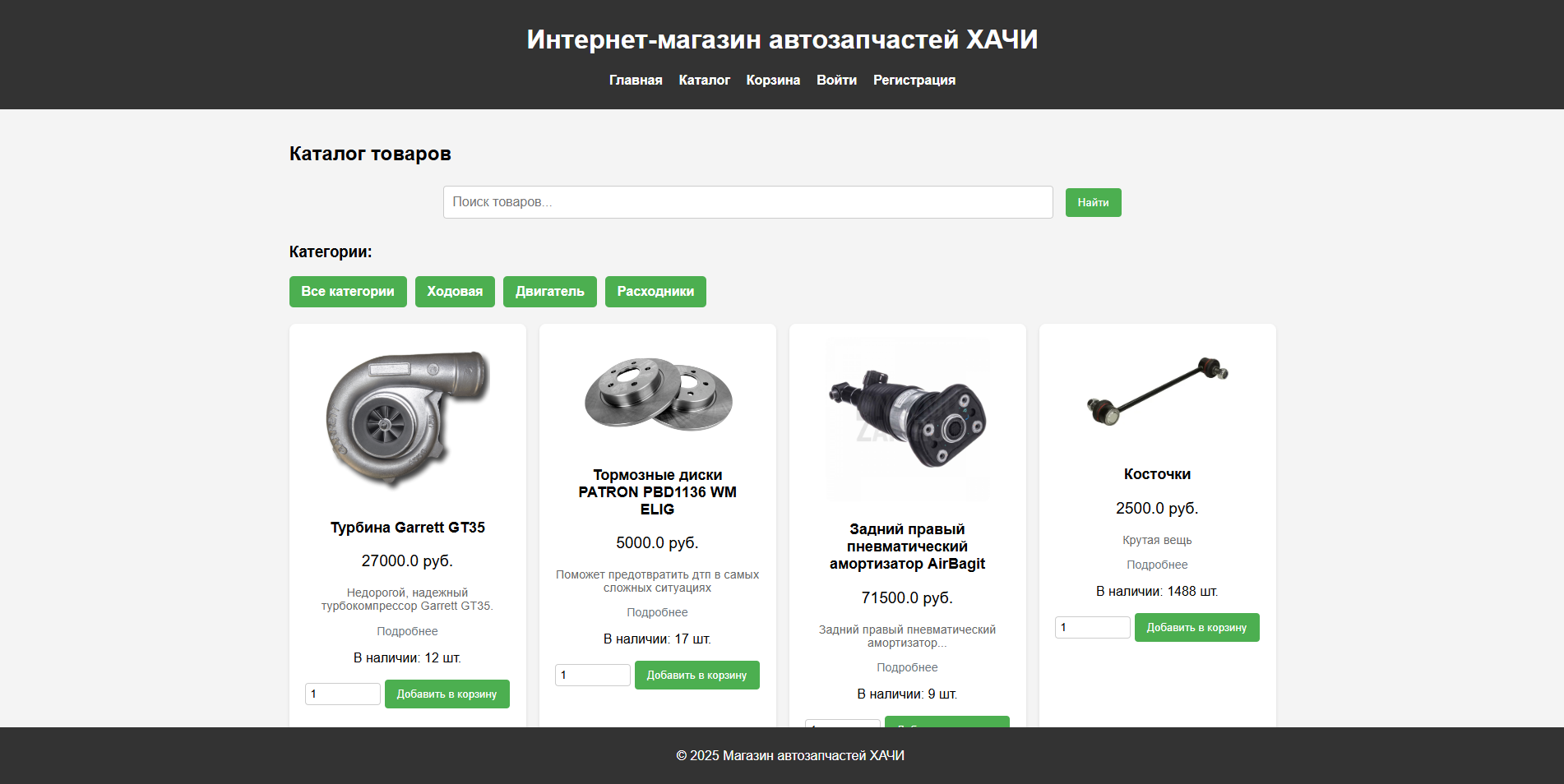


Рисунок А.3 — Страница «Каталог»

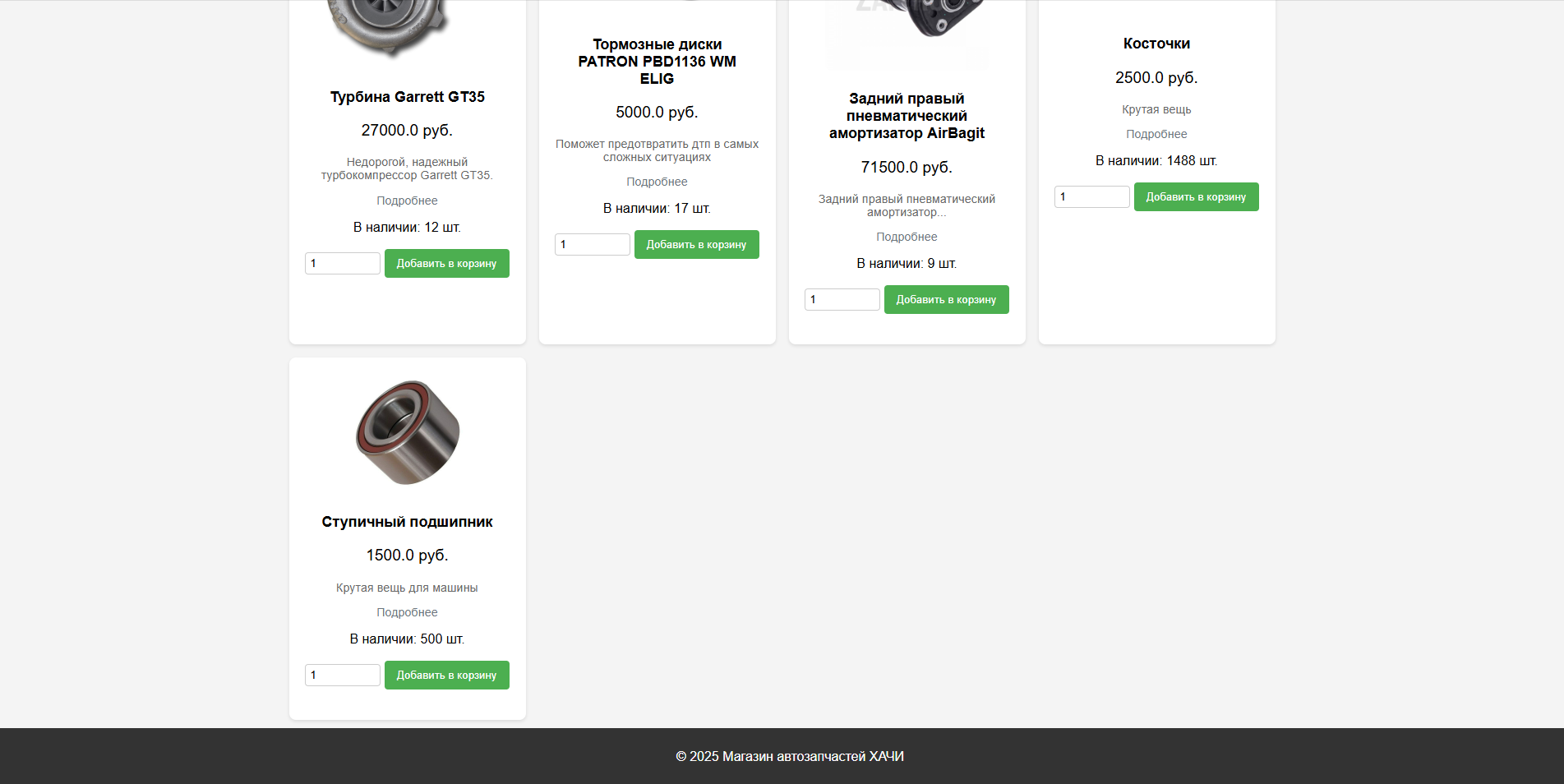


Рисунок А.4 — Страница «Каталог» часть 2

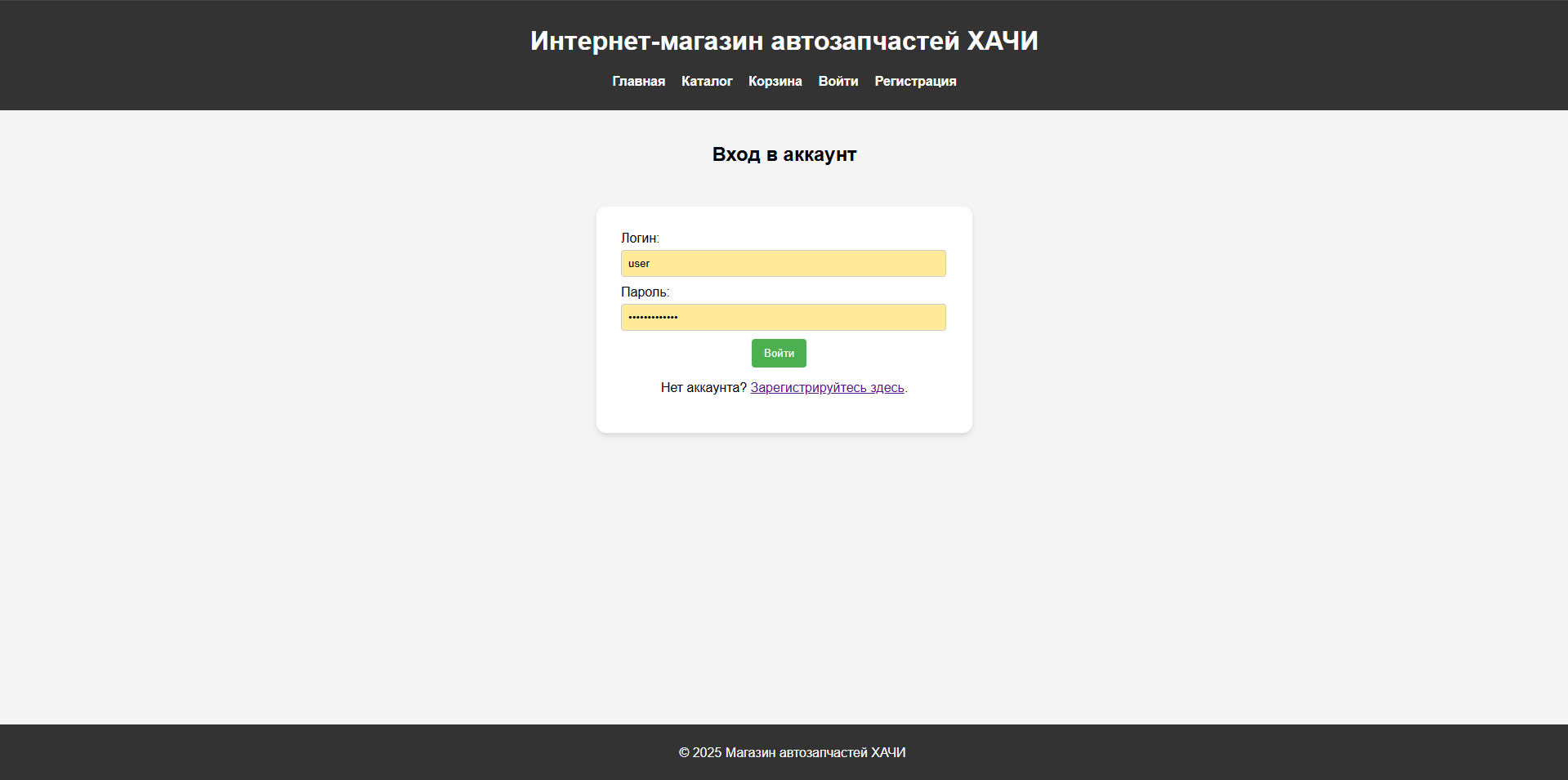


Рисунок А.5 — Страница «Вход»

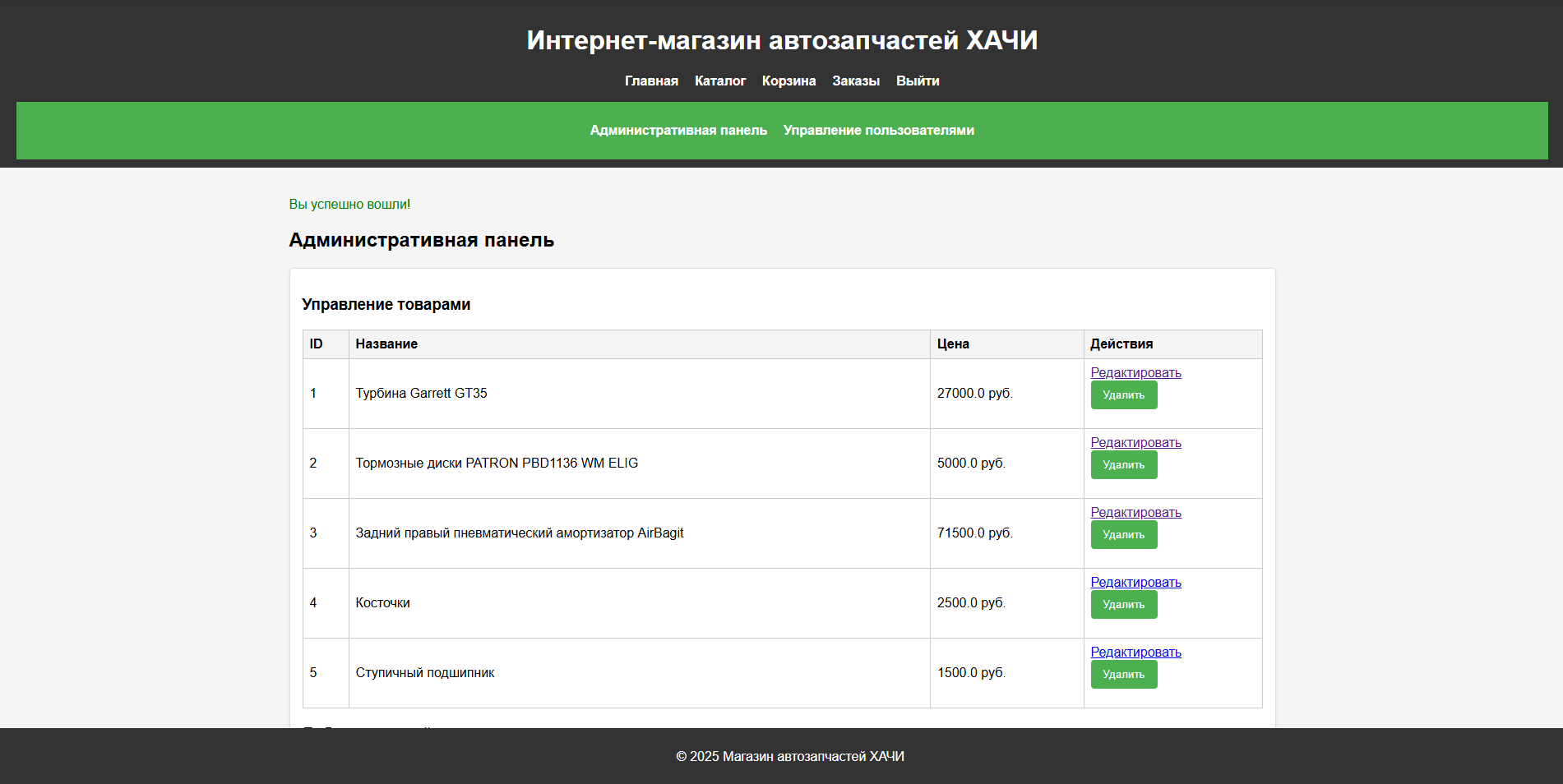


Рисунок А.6 — Страница «Административная панель» часть 1

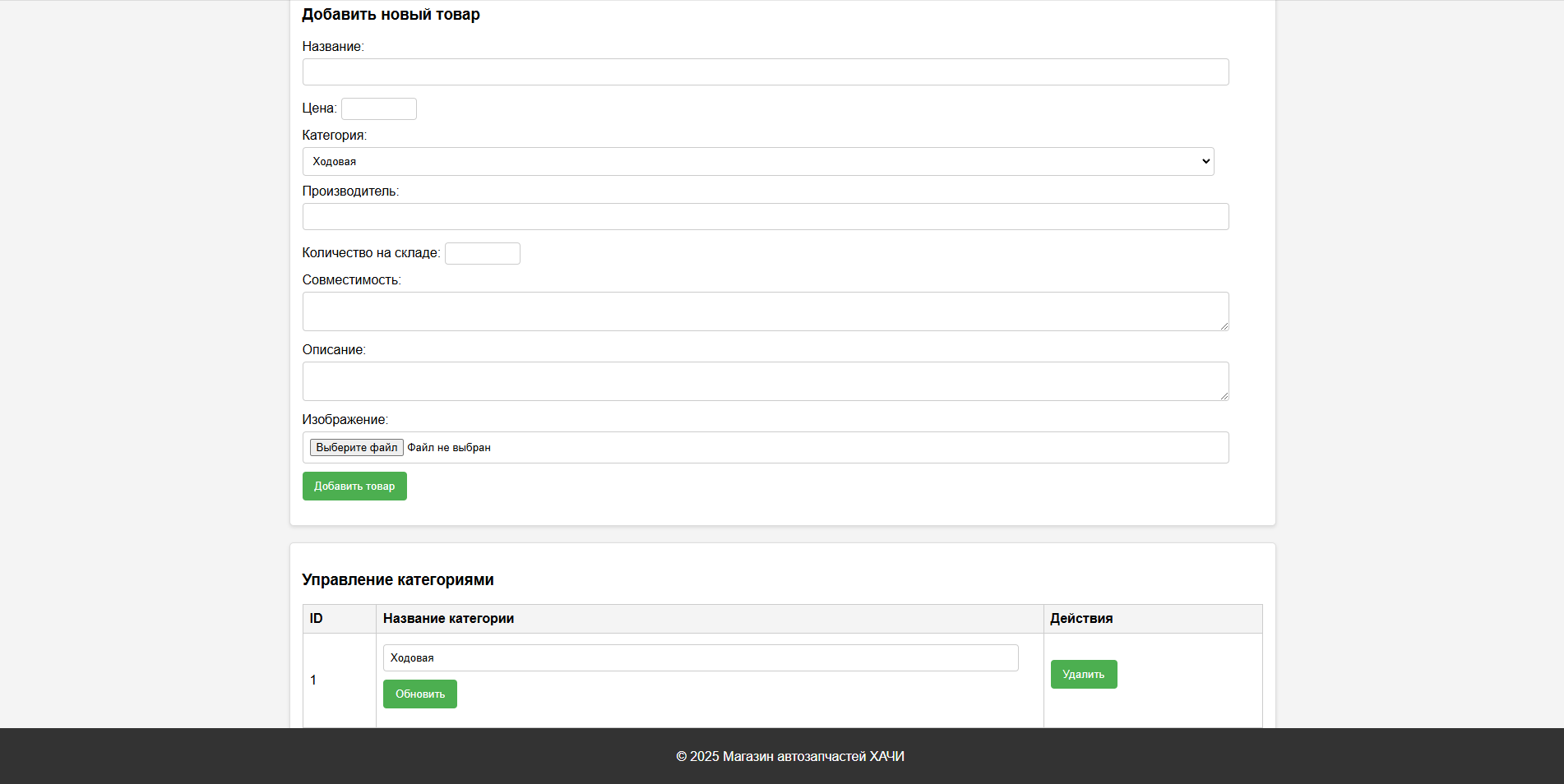


Рисунок А.7 — Страница «Административная панель» часть 2

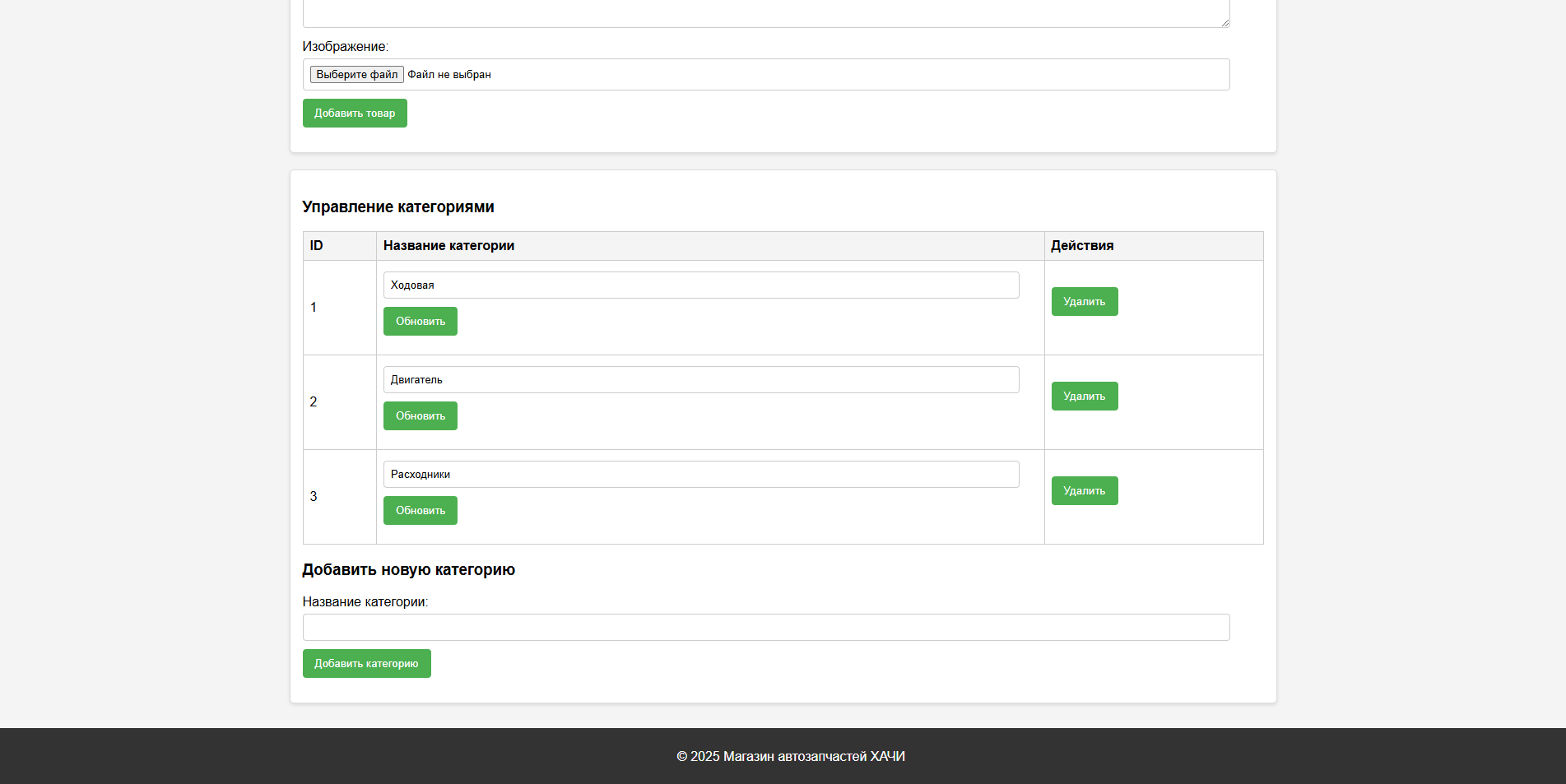


Рисунок А.8 — Страница «Административная панель» часть 3

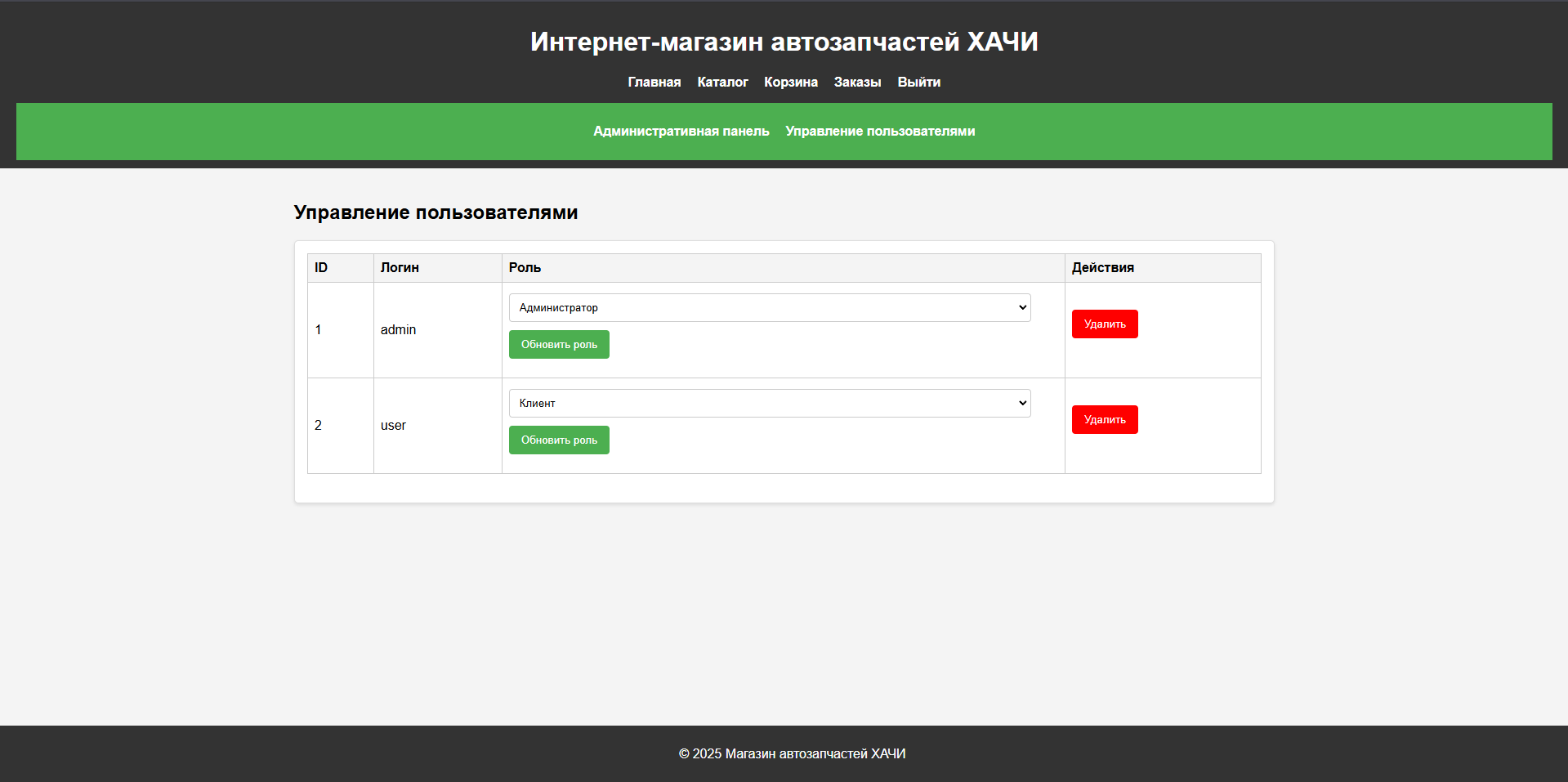


Рисунок А.9 — Страница «Управление пользователями»

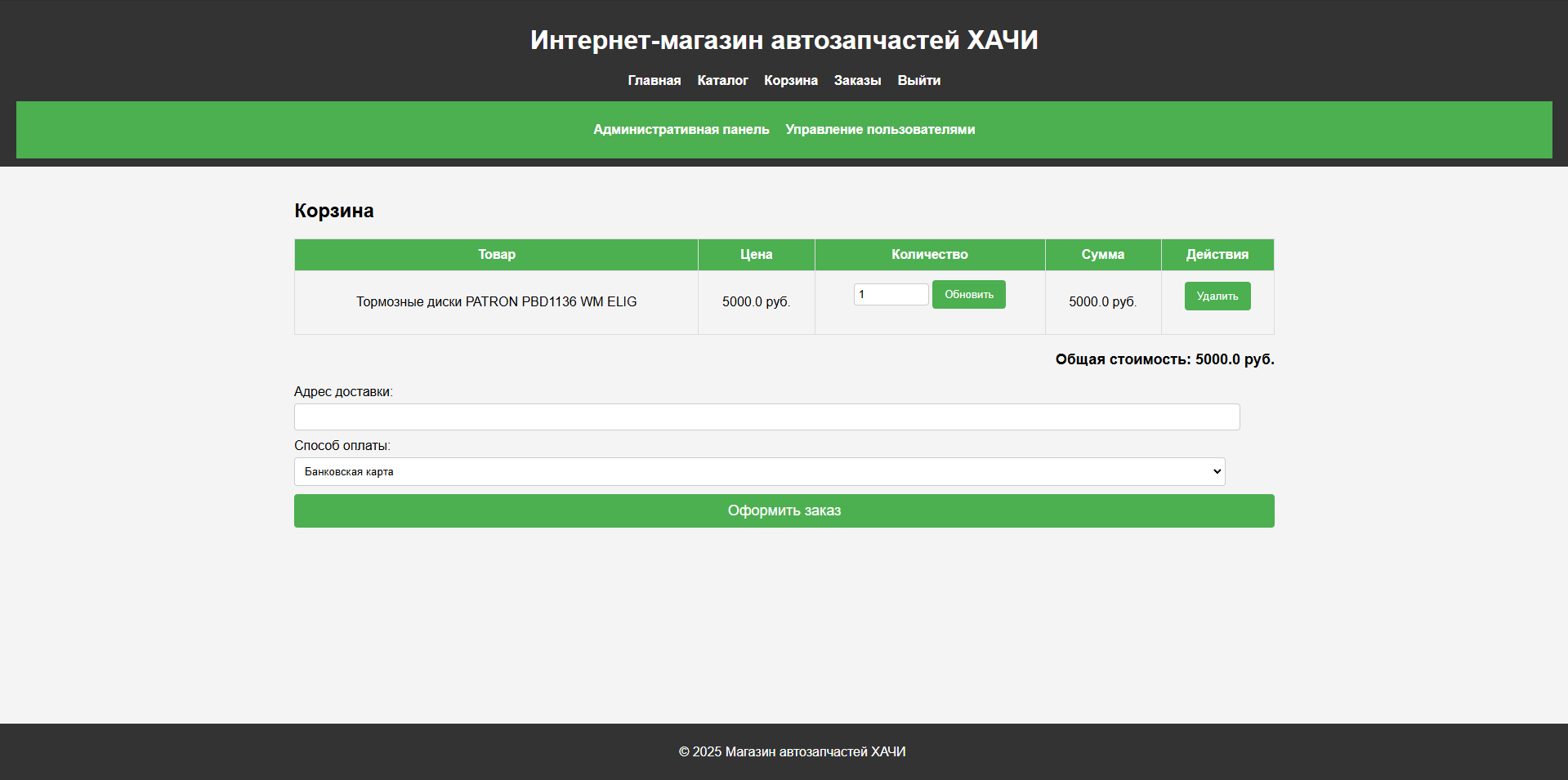


Рисунок А.10 — Страница «Корзина»

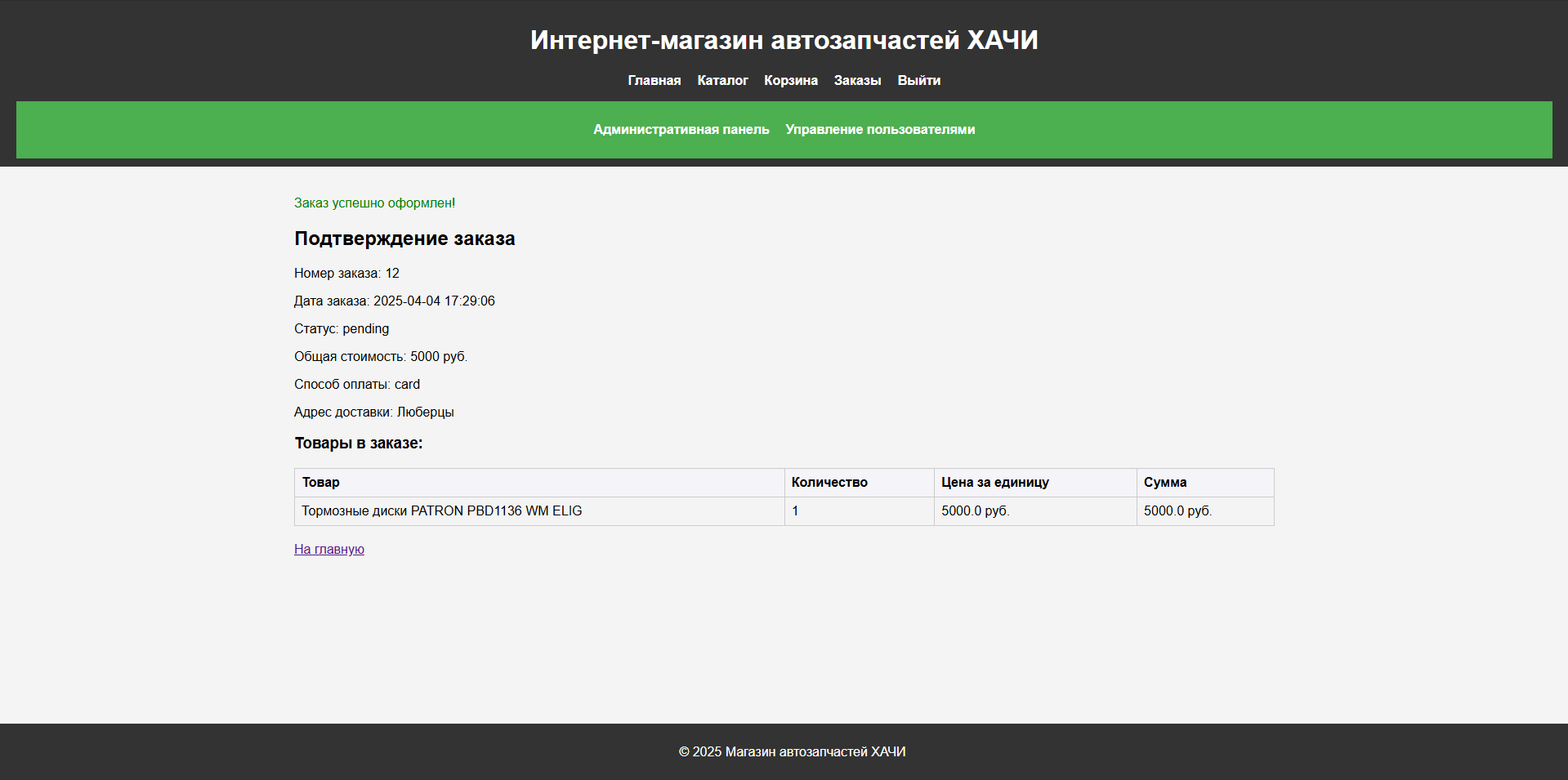


Рисунок А.11 — Страница «Заказ»

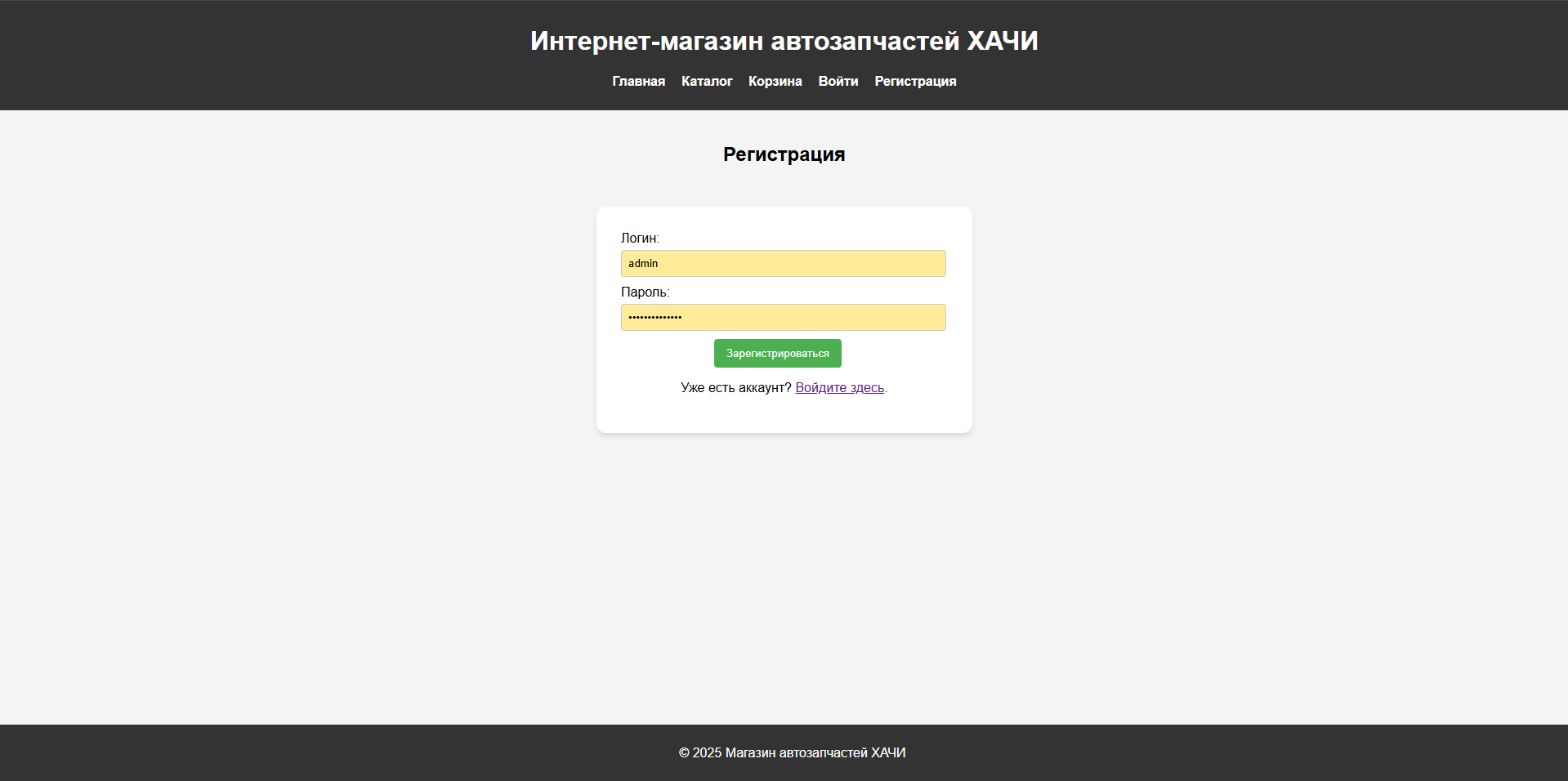


Рисунок А.12 — Страница «Регистрация»