Geekbrains

**ShopAminute WebSuite: Разработка и Тестирование Высоконадёжного Веб-Приложения на Java**

Программа: Разработчик - Программист

Специализация: Веб-разработка на Java

Вилькс Максим Павлович

Санкт-Петербург

2024

**Содержание**

Введение (4–5 стр.)

Теоретическая глава (5– 21 стр).

Глава 1. Веб-приложения, их связь с базами данных, и как важно защищать информацию в них. (5 cтр).

Глава 1.1. Package “objects” (6 cтр).

Глава 1.2. Package “handlers”. (8 cтр).

Глава 1.3 Package “dataManagers”. (10 cтр).

Глава 1.4 Package “security” (11 стр).

Глава 1.5 Package “settings”. (12 стр).

Глава 1.6 Package “util”. (13 стр).

Глава 1.7. Package “coordinators”. (14 стр).

Базы Данных (14 стр.)

Процесс создания веб-страниц для интернет-магазина **shopAminute** (16 cтр).

Покрытие тестами приложения, важность тестов. (18 стр).

Практическая глава (21- 47 стр).

Заключение (48 стр).

Список используемой литературы (49 стр).

# 

# Введение

Тема проекта: Исследование особенностей организации процесса создания приложения, баз данных и тестов для онлайн-магазина с верификацией на примере проекта **shopAminute.**

Цель: Разработать и проанализировать модель данных для онлайн-магазина, который включает систему верификации пользователей и товаров, с акцентом на безопасность и интеграцию с электронной коммерцией. А также обеспечить покрытие тестами более 80%

Какую проблему решает: Необходимость создания надежной и безопасной базы данных для онлайн-магазинов, которая обеспечивает верификацию и защиту данных.

Задачи:

1. Анализ требований к базам данных в контексте онлайн-магазинов с верификацией.
2. Разработка модели данных, обеспечивающей надежную верификацию и защиту данных.
3. Исследование методов защиты данных от несанкционированного доступа.
4. Проектирование и реализация базы данных shopAminute с использованием современных технологий и подходов.
5. Интеграция системы верификации с интерфейсом пользователя для обеспечения подлинности товаров и данных пользователей.
6. Тестирование и оценка эффективности разработанной системы.

Инструменты: Maven, Spring Boot, Spring Security, Spring Data JPA, Thymeleaf, PostgreSQL, JUnit, Mockito.

Состав команды: Вилькс М.П. (Разработчик).

В данном проекте особое внимание уделяется выбору технологий и инструментов, которые способствуют созданию безопасного и удобного в использовании онлайн-магазина. Использование артефактов Maven, таких как spring-boot-starter-security и spring-boot-starter-web, подчеркивает стремление к созданию надежного продукта с акцентом на безопасность и качество в сфере онлайн-торговли. В процессе разработки осуществляется тщательное тестирование всех компонентов системы, что позволяет гарантировать высокий уровень безопасности и функциональности проекта **shopAminute**.

# Теоретическая глава.

Глава 1. Веб-приложения, их связь с базами данных, и как важно защищать информацию в них.

Веб-приложение — это программное обеспечение, которое использует веб-технологии и веб-браузеры для выполнения задач через Интернет. Они могут выполнять широкий спектр функций и предоставлять интерактивные пользовательские интерфейсы, аналогичные настольным приложениям, но с тем преимуществом, что их можно использовать из любой точки мира, где есть доступ в Интернет.

Базы данных — это структурированные наборы данных, которые хранятся и управляются электронно. Они являются центральным компонентом большинства веб-приложений, поскольку обеспечивают хранение, поиск и управление данными, которые приложение собирает и генерирует.

Взаимосвязь между веб-приложениями и базами данных критически важна, потому что:

* Веб-приложения полагаются на базы данных для хранения информации, такой как пользовательские данные, транзакции, история покупок и многое другое.
* Базы данных позволяют веб-приложениям извлекать, обновлять и удалять данные в ответ на запросы пользователей.
* Эффективность и масштабируемость веб-приложений часто зависят от структуры и производительности базы данных.

Защита данных важна по нескольким причинам:

* Конфиденциальность: Пользовательские данные часто содержат личную информацию, которая должна оставаться конфиденциальной.
* Безопасность: Незащищенные данные могут стать целью для хакеров, что может привести к краже данных, мошенничеству и другим злоупотреблениям.
* Соответствие законодательству: Многие страны имеют законы, регулирующие хранение и обработку личных данных, такие как GDPR в Европейском Союзе.
* Доверие клиентов: Защита данных помогает поддерживать доверие клиентов и сохранять репутацию компании.

Поэтому разработчики веб-приложений должны уделять особое внимание защите данных, используя методы шифрования, безопасные протоколы передачи данных и следуя лучшим практикам безопасности.

Создание приложения я начал с модели данных, ниже мы рассмотрим это создание поэтапно.

Глава 1.1.

Package “objects”.

Классы, представленные ниже , являются частью модели данных для веб-приложения, использующего Java Persistence API (JPA) с Hibernate в качестве провайдера. Они представляют собой сущности, которые будут отображаться в таблицы базы данных. Вот краткое описание каждого класса и аннотаций:

1. Basket:
   * @Entity и @Table(name = "product\_cart") указывают, что класс является сущностью и будет отображаться в таблицу product\_cart.
   * @Id и @GeneratedValue(strategy = GenerationType.IDENTITY) определяют первичный ключ и автоматическую генерацию его значения.
   * @Column аннотации указывают на соответствующие столбцы в таблице.
   * Конструкторы и геттеры/сеттеры позволяют создавать и управлять экземплярами класса.
2. Category:
   * Аналогично использует @Entity для определения сущности.
   * @OneToMany(mappedBy = "category", fetch = FetchType.EAGER) указывает на однонаправленную связь с множеством продуктов, принадлежащих категории.
3. Image:
   * @ManyToOne(fetch = FetchType.EAGER, optional = false) устанавливает связь многие-к-одному с продуктом, каждый образ может быть связан только с одним продуктом.
4. Order:
   * Содержит поля для управления заказами, включая связи с продуктом и клиентом.
   * @PrePersist используется для инициализации даты и времени перед сохранением.
5. Person:
   * Использует валидационные аннотации @NotEmpty и @Size для проверки полей логина и пароля.
   * @ManyToMany() и @OneToMany() аннотации указывают на связи с продуктами в корзине и заказами соответственно.
6. Product:
   * Содержит информацию о продукте и связи с категорией, изображениями и заказами.
   * @ManyToMany() и @OneToMany() аннотации используются для управления связями с пользователями и заказами.

Эти классы и аннотации обеспечивают связь между объектами Java и таблицами базы данных, позволяя эффективно управлять данными в приложении. Они отвечают за отображение, связи, валидацию и жизненный цикл сущностей.

Глава 1.2.

Package “handlers”.

Класс AdminController является частью веб-приложения, созданного с использованием Spring Framework, и предназначен для управления товарами в интернет-магазине. Он содержит методы для добавления, редактирования и удаления продуктов, а также для загрузки изображений продуктов. Вот основные аспекты этого класса:

* Конструктор: Принимает сервис ProdService и хранилище ClassificationStorage, которые используются для взаимодействия с базой данных и управления категориями товаров соответственно.
* Методы addProduct: Позволяют добавлять новые товары в интернет-магазин. Включают в себя загрузку изображений и привязку товара к категории. Проверка данных на валидность производится с помощью аннотации @Valid.
* Метод admin: Отображает страницу администратора с перечнем всех товаров.
* Методы deleteProduct и editProduct: Предоставляют функциональность для удаления и редактирования информации о товарах.

Этот класс важен для управления содержимым интернет-магазина, так как он обеспечивает интерфейс для выполнения операций CRUD (создание, чтение, обновление, удаление) над товарами. Он также обеспечивает интеграцию с системой хранения файлов для загрузки изображений, что является ключевым аспектом представления товаров в интернет-магазине. Защита данных в таких операциях критически важна для обеспечения безопасности и конфиденциальности информации о товарах и пользователях.

Классы AuthenticationController, MainController и ProductController являются частью структуры веб-приложения на базе Spring Framework и предназначены для выполнения различных функций в контексте интернет-магазина.

AuthenticationController:

* Этот класс содержит один метод login, который обрабатывает GET-запросы по пути /authentication. Метод возвращает имя представления authentication, которое, связано с процессом аутентификации пользователей.

MainController:

* Конструктор: Инициализирует репозитории и сервисы, необходимые для работы с товарами, пользователями и заказами.
* Метод index: Определяет роль пользователя и перенаправляет его на соответствующую страницу (администратора или пользователя).
* Методы registration и resultRegistration: Обрабатывают процесс регистрации новых пользователей.
* Метод infoProduct: Предоставляет информацию о продукте по его идентификатору.
* Метод productSearch: Реализует поиск товаров с фильтрацией по различным критериям.
* Методы addProductInCart, cart, deleteProductFromCart: Управляют корзиной пользователя, добавляя товары, отображая корзину и удаляя товары из неё.
* Методы order и orderUser: Обрабатывают создание заказов и отображают список заказов пользователя.

ProductController:

* Конструктор: Инициализирует репозиторий и сервис для работы с товарами.
* Метод getAllProduct: Возвращает список всех товаров.
* Метод infoProduct: Предоставляет информацию о конкретном товаре.
* Метод productSearch: Реализует поиск товаров с возможностью сортировки и фильтрации.

Эти классы играют ключевую роль в управлении функциональностью интернет-магазина, включая аутентификацию пользователей, управление товарами, корзиной и заказами. Они используют аннотации Spring MVC для маршрутизации запросов и взаимодействия с моделью данных и представлениями, что позволяет создать гибкую и масштабируемую архитектуру веб-приложения.

Глава 1.3.

Package “dataManagers”.

В мире кода, где каждая строка таит в себе мощь создания и управления, интерфейсы репозиториев **shopAminute** стоят как стражи данных, обеспечивая порядок и структуру в хранилище информации интернет-магазина.

**BasketStorage** - хранитель корзины, ведёт учёт желаний покупателей. С его помощью можно отыскать все сокровища, которые пользователь желает приобрести, а также с лёгкостью удалить из корзины ненужный товар, словно стирая следы в песке.

**ClassificationStorage** - знаток категорий, он знает каждую категорию товаров, как старый библиотекарь знает каждую книгу. Он может найти нужную категорию по имени, словно волшебник, находящий заклинание в древнем фолианте.

**IndividualStorage** - хранитель личностей, бережно собирает в себе информацию о каждом пользователе. Он может отыскать человека по логину, как детектив, который находит ключевую улику в запутанном деле.

**ProductRepo** - сокровищница товаров, в ней хранится вся информация о продуктах, доступных для покупки. Она умеет искать товары по названию, сортировать их по цене, как алхимик, который сортирует ингредиенты для создания зелья.

**PurchaseStorage** - архив заказов, ведёт записи о каждой покупке, словно летописец, который не упустит ни одного важного события.

Эти интерфейсы - волшебные инструменты в руках разработчиков, позволяющие взаимодействовать с базой данных, как музыкант взаимодействует со струнами инструмента, создавая гармонию данных, которая поддерживает жизнь интернет-магазина **shopAminute** .

Глава 1.4.

Package “security”.

В виртуальном пространстве **shopAminute** , где безопасность является краеугольным камнем, класс PersonDetails и конфигурация **shopAminute** выступают как надёжные стражи пользовательских данных и их защиты.

**PersonDetails** - это персональный паспорт пользователя в системе. Он несёт в себе уникальные данные о пользователе, такие как логин и пароль, а также его роль в системе. Этот класс обеспечивает интеграцию с Spring Security, предоставляя информацию о правах доступа пользователя, что позволяет системе определить, какие двери в мире **shopAminute** открыты перед ним.

SecurityConfig - это архитектор безопасности, который проектирует защитные стены и определяет правила доступа. С помощью аннотации @Configuration, он настраивает фильтры безопасности, управляет процессом аутентификации и авторизации, и определяет, какие пути открыты для всех, а какие только для избранных. Этот класс использует PasswordEncoder для шифрования паролей, делая их непонятными для посторонних глаз, и SecurityFilterChain, чтобы установить правила, которые как магический барьер защищают от нежелательных гостей.

Вместе эти классы создают мощную систему защиты для **shopAminute**, гарантируя, что каждый пользователь может чувствовать себя в безопасности, зная, что его личные данные и действия на сайте находятся под надёжной защитой. Это как замок и ключ к нему, где PersonDetails - это ключ, который открывает двери, а SecurityConfig - это замок, который их защищает.

Глава 1.5.

Package “settings”.

Класс Config, реализующий интерфейс WebMvcConfigurer, и класс SpringConfig с аннотацией @ComponentScan, а также перечисление Status, играют важную роль в архитектуре безопасности и функциональности **shopAminute**.

**Config**:  
Этот класс отвечает за конфигурацию ресурсов веб-приложения. С помощью аннотации @Value, он извлекает путь к директории, где хранятся загружаемые файлы, и настраивает обработчики ресурсов таким образом, чтобы статические ресурсы (в данном случае изображения) были доступны по определённому URI-пути.

**SpringConfig**:  
Этот класс использует @ComponentScan для автоматического обнаружения и регистрации бинов в указанном пакете, что упрощает настройку и управление компонентами приложения.

**Status**:  
Перечисление Status определяет возможные статусы заказа в системе. Это позволяет стандартизировать обработку заказов и обеспечивает ясность в отслеживании их состояния на всех этапах обработки.

Все эти компоненты вместе создают упорядоченную и эффективную систему, которая поддерживает целостность и безопасность данных, а также обеспечивает удобство и скорость работы веб-приложения **shopAminute**. Они как кирпичики в фундаменте, каждый из которых несёт свою функцию для обеспечения стабильности и надёжности всего здания.

Глава 1.6.

Package “util”.

Класс IdentityVerifier играет ключевую роль в системе безопасности **shopAminute**, выполняя функцию валидатора данных пользователя. Он реализует интерфейс Validator из Spring Framework, что позволяет ему интегрироваться с другими компонентами Spring для проверки объектов на соответствие определённым требованиям.

Основные функции и характеристики IdentityVerifier:

* Интеграция с IndividualHandler: Конструктор класса принимает объект IndividualHandler, который предположительно отвечает за управление данными индивидуальных пользователей, такими как поиск пользователя по логину.
* Поддержка класса Person: Метод supports определяет, что этот валидатор поддерживает валидацию объектов типа Person. Это означает, что IdentityVerifier может быть использован для проверки экземпляров Person.
* Валидация логина: В методе validate происходит проверка, не зарегистрирован ли уже в системе пользователь с таким же логином. Если метод findByLogin возвращает не null, это означает, что пользователь с таким логином уже существует, и валидатор добавляет ошибку в объект Errors.

Этот механизм валидации важен для предотвращения дублирования учётных записей и поддержания уникальности логина каждого пользователя, что является важной частью обеспечения безопасности и целостности данных в SecureShopNet. Валидатор IdentityVerifier как бы проверяет паспорта на границе, не позволяя "двойникам" проникнуть в систему. Это ещё один слой защиты, который помогает обеспечить, что каждый аккаунт в системе уникален и принадлежит именно тому, кто за него зарегистрировался.

Глава 1.7.

Package “coordinators”.

**IdentityService**:  
Класс IdentityService реализует интерфейс UserDetailsService, который необходим для интеграции с Spring Security. Он предоставляет механизм загрузки данных пользователя по имени пользователя, что является ключевым для аутентификации.

**IndividualHandler**:  
IndividualHandler управляет операциями, связанными с пользователями, такими как поиск и регистрация. Он использует PasswordEncoder для шифрования паролей, что усиливает безопасность данных.

**ProdService**:  
ProdService отвечает за управление товарами. Он предоставляет методы для получения списка всех продуктов, поиска продукта по идентификатору, сохранения, обновления и удаления продуктов. Использование аннотации @Transactional гарантирует, что операции с базой данных будут выполнены корректно и безопасно.

В целом, эти классы обеспечивают важные функции для работы веб-приложения, такие как безопасное хранение и обработка данных пользователей и продуктов, что способствует созданию надёжной и функциональной системы электронной коммерции. Они взаимодействуют с базой данных и другими компонентами системы, обеспечивая целостность и безопасность данных, а также удобство использования приложения для конечных пользователей.

Базы данных.

Проект базы данных для интернет-магазина **shopAminute**, разработанный с использованием PostgreSQL и инструментов восстановления pg\_restore. Этот проект является ключевым элементом системы электронной коммерции, обеспечивающей безопасное и эффективное управление данными.

**Структура базы данных**:  
База данных спроектирована таким образом, чтобы поддерживать высокий уровень нормализации и обеспечивать целостность данных. Она включает в себя таблицы для категорий товаров (category), изображений (image), заказов (orders), пользователей (person), товаров (product) и корзины покупок (product\_cart). Каждая таблица имеет свой первичный ключ и связанную последовательность для автоматического инкремента идентификаторов.

**Целостность данных**:  
Для обеспечения ссылочной целостности используются внешние ключи, которые связывают данные между таблицами. Это позволяет поддерживать актуальность информации о заказах, пользователях и товарах, а также предотвращает возможность возникновения несогласованных данных.

**Процесс восстановления**:  
С помощью pg\_restore мы можем эффективно восстанавливать структуру базы данных и данные, что критически важно для обеспечения бесперебойной работы интернет-магазина после любых сбоев или переносов системы.

**Безопасность**:  
Безопасность пользовательских данных является приоритетом нашего проекта. Мы используем шифрование паролей и применяем лучшие практики безопасности для защиты личной информации и транзакций.

В общем база данных shopAminute включает в себя следующие таблицы и связанные с ними элементы:

* Таблица category: Хранит информацию о категориях товаров. Имеет первичный ключ category\_pkey и связанную последовательность category\_id\_seq для автоматического создания уникальных идентификаторов.
* Таблица image: Содержит данные об изображениях, которые могут быть связаны с товарами. Имеет первичный ключ image\_pkey и последовательность image\_id\_seq.
* Таблица orders: Отвечает за хранение информации о заказах. Имеет первичный ключ orders\_pkey и последовательность orders\_id\_seq. Включает внешние ключи для связи с другими таблицами.
* Таблица person: Содержит данные о пользователях системы, включая их логины и пароли. Имеет первичный ключ person\_pkey и последовательность person\_id\_seq.
* Таблица product: Хранит информацию о товарах, доступных в интернет-магазине. Имеет первичный ключ product\_pkey, уникальное ограничение uk\_qka6vxqdy1dprtqnx9trdd47c и последовательность product\_id\_seq.
* Таблица product\_cart: Используется для управления корзиной покупок. Имеет первичный ключ product\_cart\_pkey и последовательность product\_cart\_id\_seq.

Каждая таблица имеет свои уникальные идентификаторы, создаваемые с помощью последовательностей, что обеспечивает целостность данных. Внешние ключи (FK CONSTRAINT) используются для обеспечения ссылочной целостности между таблицами, например, связывая заказы с пользователями и товарами.

В заключение, данный проект базы данных является фундаментом для создания надёжной и безопасной платформы электронной коммерции **SecureShopNet**. Он обеспечивает не только хранение данных, но и их целостность, безопасность и доступность для всех компонентов системы.

Процесс создания веб-страниц для интернет-магазина **shopAminute**.

Процесс создания веб-страниц для интернет-магазина **shopAminute** включал несколько ключевых этапов, в том числе разработку фронтенда с использованием HTML и CSS, а также интеграцию с бэкендом на Java.

В рамках дипломного проекта была разработана комплексная система управления контентом для интернет-магазина shopAminute. Проект включает в себя фронтенд и бэкенд части, а также систему аутентификации и управления пользователями.

Фронтенд:

* Разработаны ключевые страницы для пользовательского взаимодействия: страница корзины, главная страница, страница информации о товаре, страница заказов, а также личный кабинет пользователя и администратора.
* Использование Thymeleaf для динамического отображения данных на страницах, что позволяет обновлять информацию в реальном времени без перезагрузки страницы.
* Стилизация выполнена с помощью CSS, обеспечивая современный и адаптивный дизайн.

Бэкенд:

* Реализована логика обработки данных на стороне сервера с использованием Java и Spring Boot.
* Внедрение системы управления базами данных для хранения информации о товарах, заказах и пользователях.
* Обеспечение безопасности данных с помощью системы аутентификации и авторизации.

Аутентификация и управление пользователями:

* Созданы страницы входа и регистрации, позволяющие пользователям создавать личный аккаунт и входить в систему.
* Для администраторов предусмотрен личный кабинет с возможностью добавления, редактирования и удаления товаров.

# В результате была создана полнофункциональная система электронной коммерции, которая обеспечивает удобное управление товарами, заказами и аккаунтами пользователей, а также предоставляет покупателям комфортный и безопасный опыт онлайн-шопинга.

Покрытие тестами приложения, важность тестов.

В рамках дипломного проекта была проведена разработка и тестирование веб-приложения с использованием Spring Framework. Основное внимание уделялось обеспечению качества кода и безопасности приложения, что достигалось за счет комплексного подхода к тестированию.

Создание тестов:  
Тесты были созданы с использованием JUnit и Mockito для имитации зависимостей. Это позволило тестировать каждый компонент в изоляции от остальной части системы. Тесты охватывали следующие аспекты:

1. Контроллеры: Проверка правильности маршрутизации запросов и возвращаемых представлений.
2. Сервисы: Тестирование бизнес-логики и взаимодействия с репозиториями.
3. Репозитории: Проверка запросов к базе данных и корректности возвращаемых данных.
4. Сущности: Тестирование методов equals и hashCode, а также геттеров и сеттеров.
5. Безопасность: Проверка конфигурации безопасности и правильности работы с пользовательскими данными.
6. Валидация: Тестирование ограничений и правил валидации для полей сущностей.

Важность покрытия тестами:  
Покрытие кода тестами является критически важным для обеспечения надежности и безопасности приложения. Тесты позволяют выявить ошибки на ранних этапах разработки, снижают риск регрессии при изменении кода и обеспечивают уверенность в том, что новые функции не нарушают существующую функциональность. Кроме того, хорошее покрытие тестами упрощает поддержку и расширение приложения, так как разработчики могут быть уверены, что их изменения не приведут к непредвиденным последствиям.

Достижение покрытия в 81%:  
В проекте было достигнуто покрытие кода тестами на уровне 81%, что является хорошим результатом для веб-приложения. Это означает, что большая часть кода была проверена тестами, что повышает уровень доверия к работоспособности и безопасности приложения. Покрытие было измерено с помощью инструментов анализа кода, таких как JaCoCo, которые позволяют оценить, какие части кода выполняются при запуске тестов.

В дополнение к общему описанию важности тестирования и достигнутого покрытия, рассмотрим каждый тест более подробно:

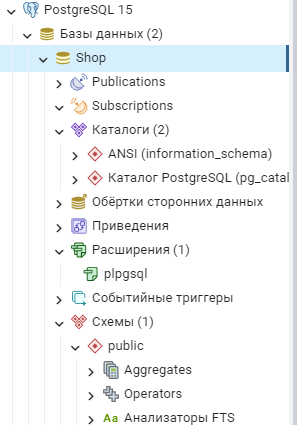
1. AdminControllerTest: Тесты для AdminController проверяют, что методы контроллера возвращают правильные представления и выполняют ожидаемые действия, такие как добавление атрибутов в модель и удаление продуктов.
2. AuthenticationControllerTest: Здесь тестируется, что контроллер аутентификации корректно обрабатывает запросы и возвращает ожидаемое представление при условии, что пользователь аутентифицирован.
3. BasketTests: Эти тесты охватывают проверку конструкторов, геттеров и сеттеров класса Basket, а также метод hashCode.
4. CategoryTests: Тесты для класса Category включают проверку геттеров и сеттеров, а также взаимосвязей между категориями и продуктами.
5. IdentityServiceTest: Тестирование сервиса IdentityService включает проверку загрузки пользовательских данных по имени пользователя и обработку исключения UsernameNotFoundException.
6. IdentityVerifierTest: Здесь проверяется валидация уникальности логина пользователя, чтобы предотвратить создание дублирующих учетных записей.
7. ImageTest: Тесты для класса Image проверяют геттеры и сеттеры, а также конструкторы.
8. IndividualHandlerTest: Тестирование IndividualHandler включает проверку поиска и регистрации пользователей, а также кодирование паролей.
9. OrderTests: Тесты для класса Order охватывают проверку геттеров и сеттеров, а также методы, связанные с жизненным циклом заказа.
10. PersonDetailsTest: Тесты для PersonDetails проверяют интеграцию с Spring Security и правильность предоставления ролей и привилегий пользователя.
11. PersonTests: Тесты для класса Person включают проверку конструкторов, геттеров и сеттеров, а также методов equals и hashCode.
12. ProdServiceTest: Тестирование ProdService включает проверку взаимодействия с репозиторием ProductRepo, включая сохранение и удаление продуктов.
13. ProductControllerTest: Тесты для ProductController проверяют, что контроллер правильно обрабатывает запросы на получение и поиск продуктов, а также корректно обрабатывает ошибки.
14. ProductTests: Тесты для класса Product охватывают проверку геттеров и сеттеров, а также конструкторов.
15. SecurityConfigTest: Тестирование SecurityConfig включает проверку правильности создания PasswordEncoder.
16. ShopAminuteApplicationTests: Этот тест проверяет, что контекст Spring загружается без ошибок.
17. StatusTest: Тесты для перечисления Status проверяют, что оно содержит правильные значения и что порядок значений соответствует ожидаемому.

В заключение данной главы, тестирование является неотъемлемой частью разработки качественного программного обеспечения. Оно позволяет обеспечить стабильность и безопасность приложения, а также упрощает процесс разработки и поддержки. Достижение высокого уровня покрытия тестами подтверждает надежность разработанного веб-приложения.

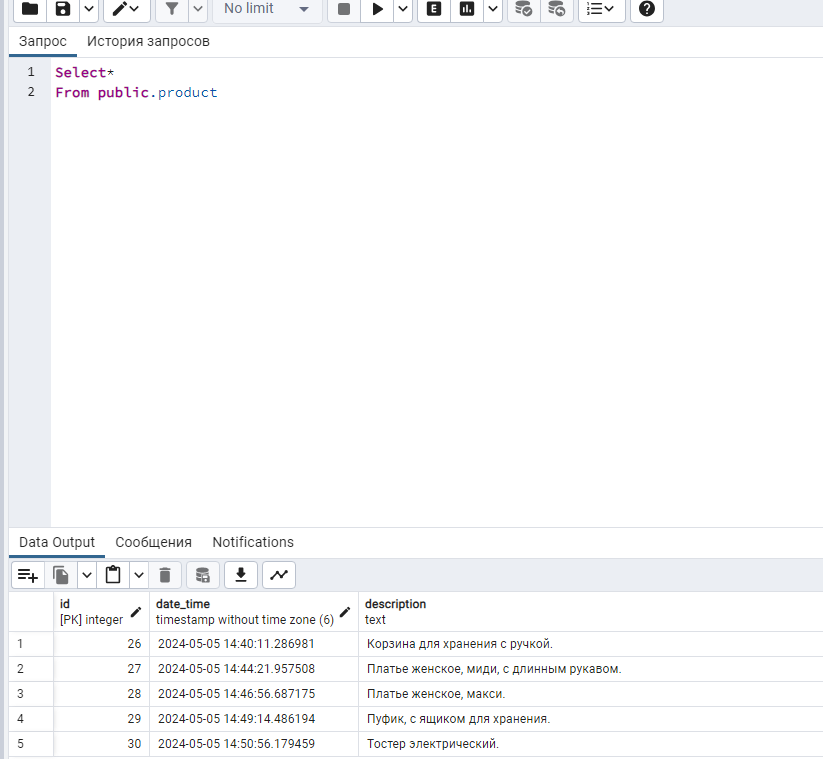
# Практическая глава.

В этой главе будет рассмотрена работа базы данных, работа приложения, а также их взаимодействие между собой, и проведение тестов.

Начнем с запуска базы данных, с помощью приложения pgAdmin4.

‘

База данных называется Shop, проверим ее наполненность простым запросом.



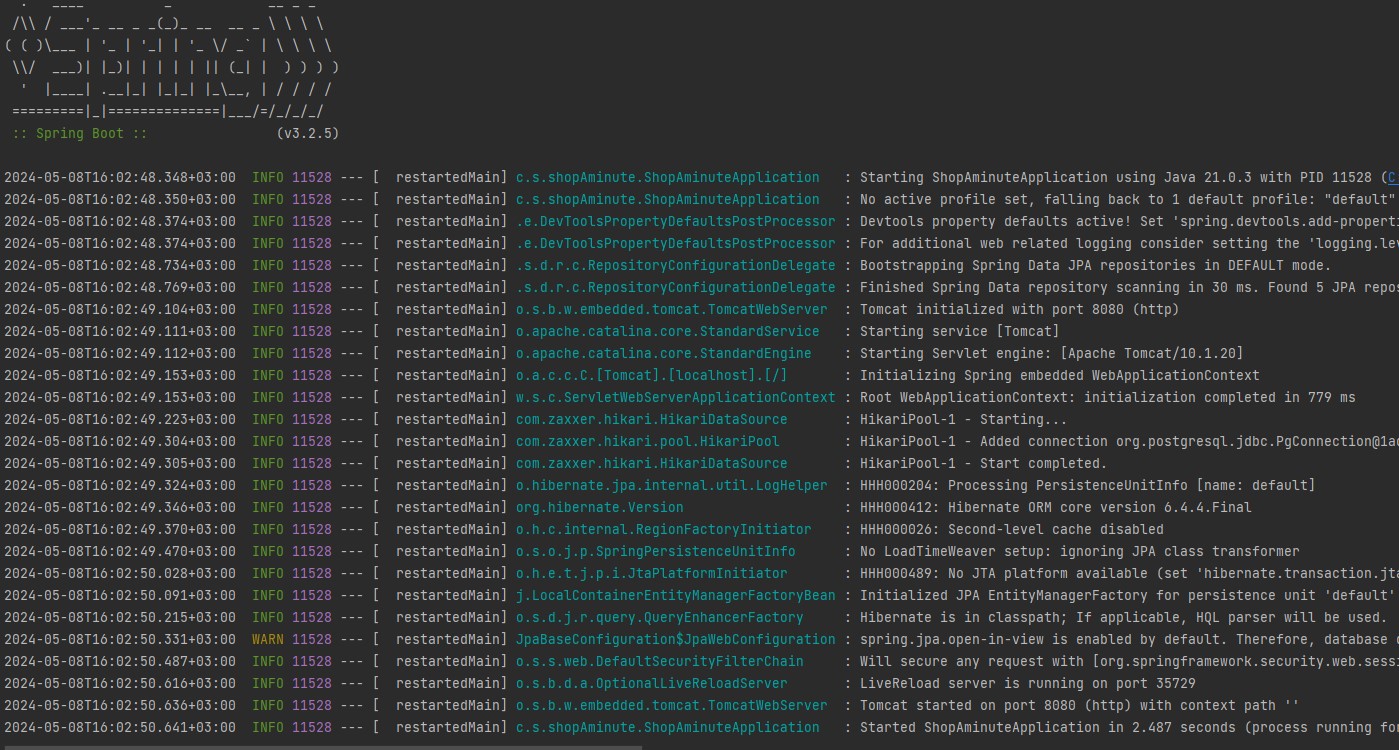
Как мы видим, продукты, присутствуют, база данных в работе.

Проверим как она будет работать с приложением. Запускаем среду разработки.

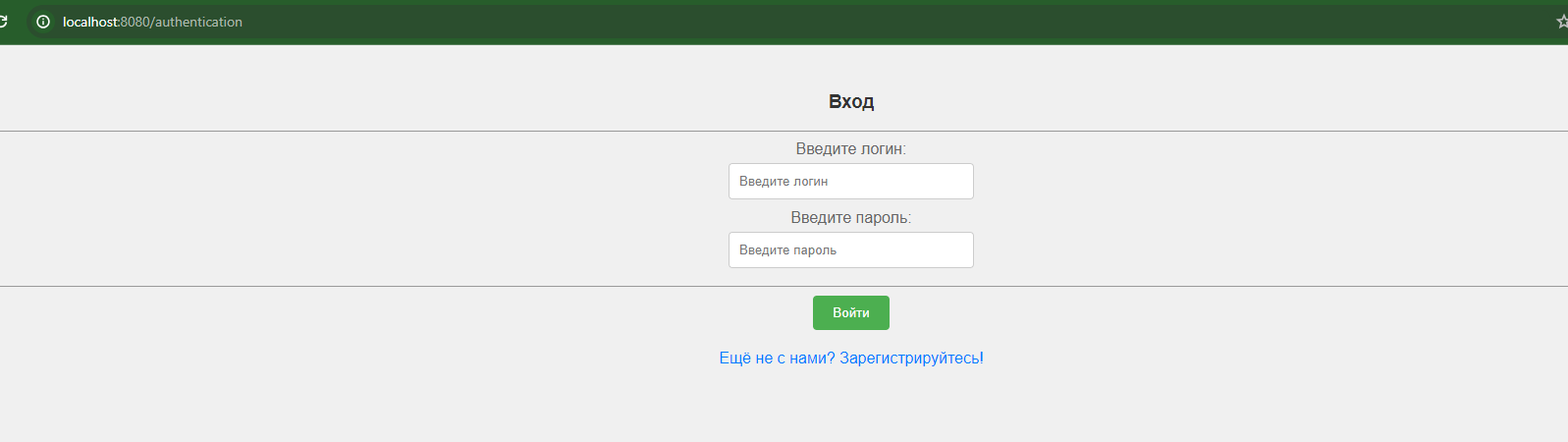
В теоретической части мы рассмотрели каждый класс и интерфейс, поэтому в этой части рассматривать отдельно мы их не будем.



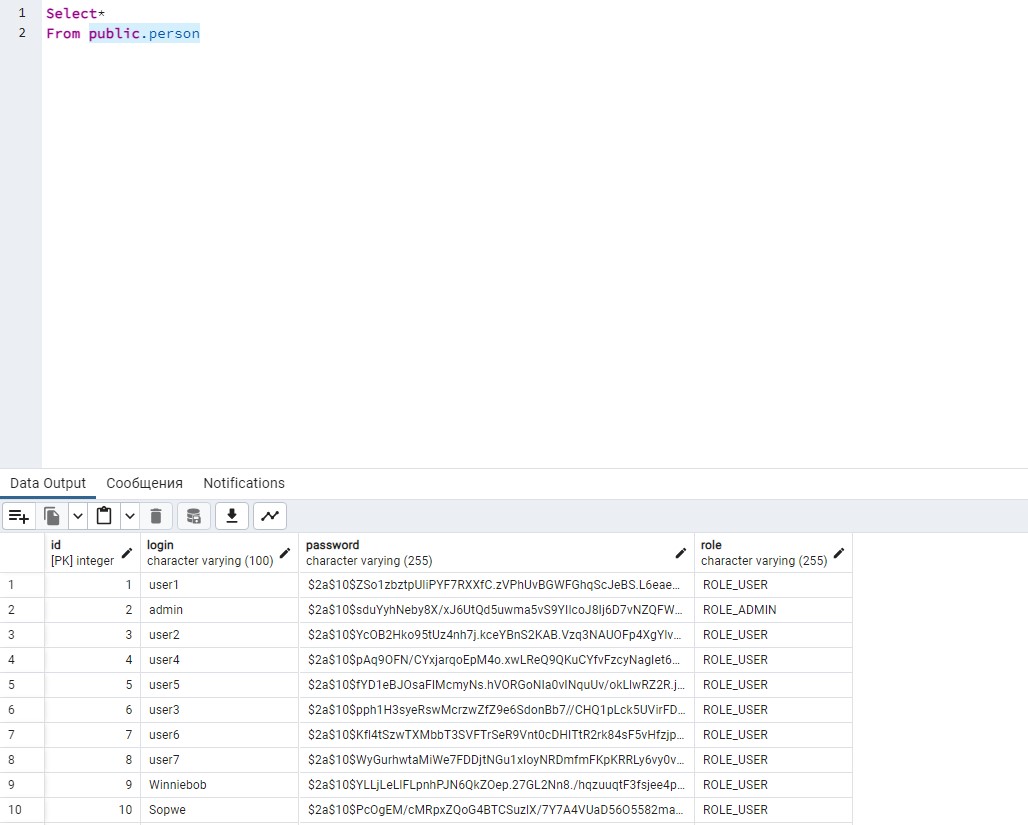
Запустим наше приложение и перейдем на указанный адрес хоста.



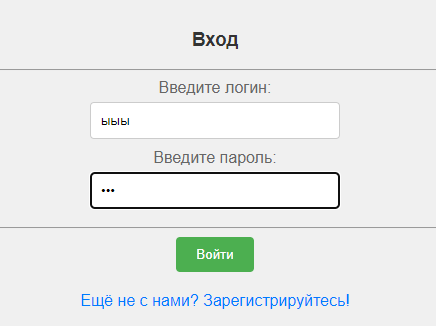
Как мы видим среда разработки не выдала каких либо ошибок, а это значит, что наше приложение работает. Убедимся в этом.

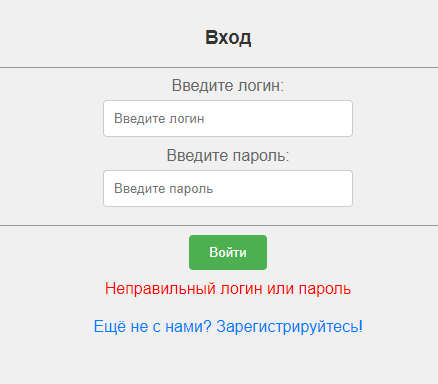


Мы перешил на страничку авторизации пользователя. Они у нас находятся в базе данных, в таблице public.person

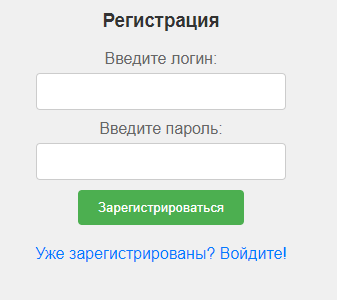


Попробуем ввести несуществующего пользователя

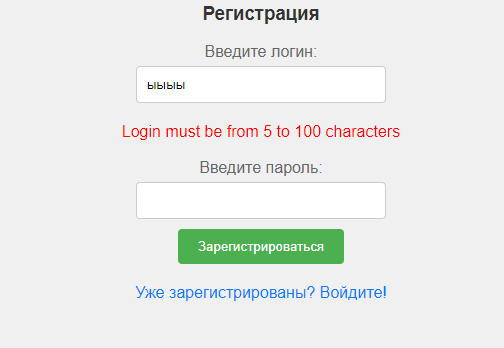




Как и ожидалось, приложение выдало ошибку. Логин или пароль не верны. Давайте попробуем создать пользователя!

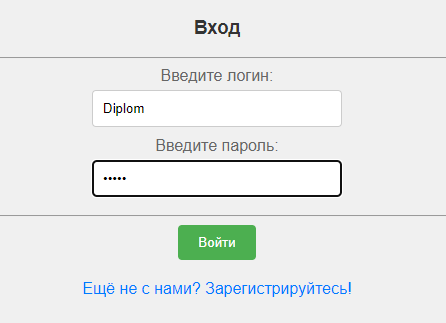


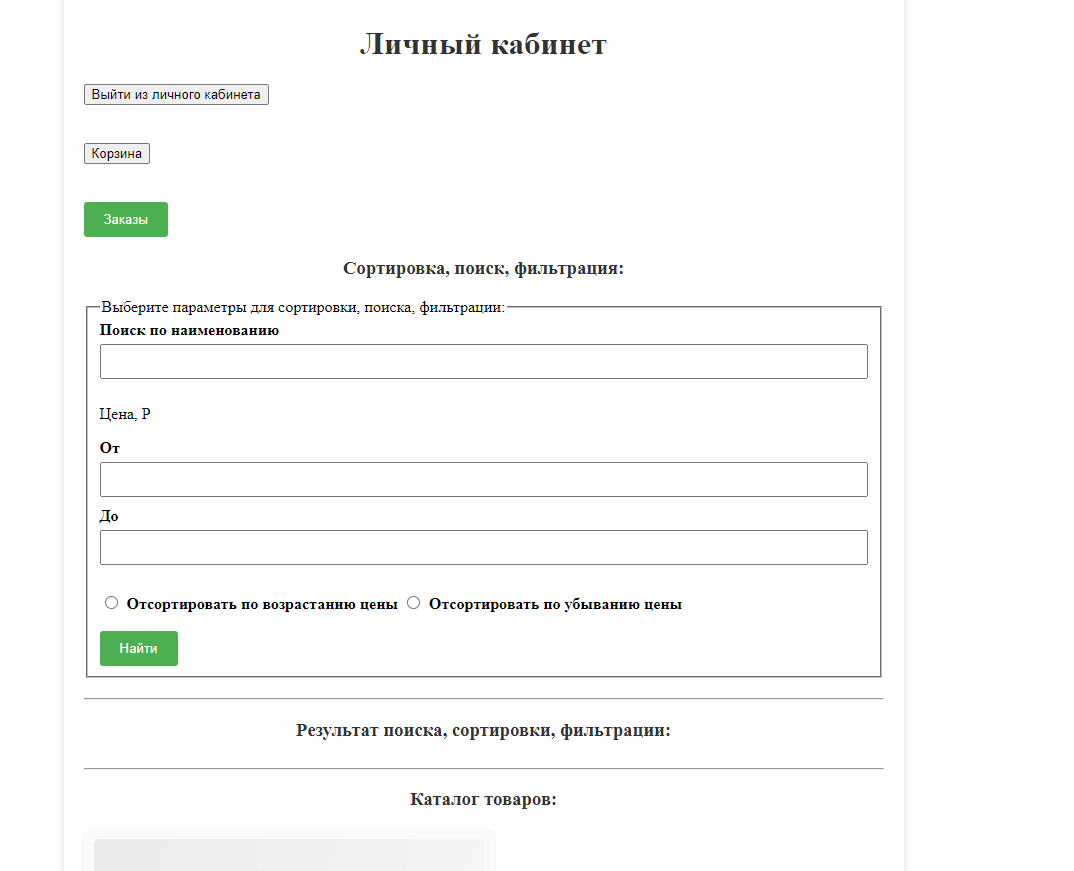
Мы вошли в меня регистрации, вводим логин и пароль.



Ловим исключение, слишком короткий логин. Будем внимательнее!

Назовемся Diplom и дадим ему пароль 12345

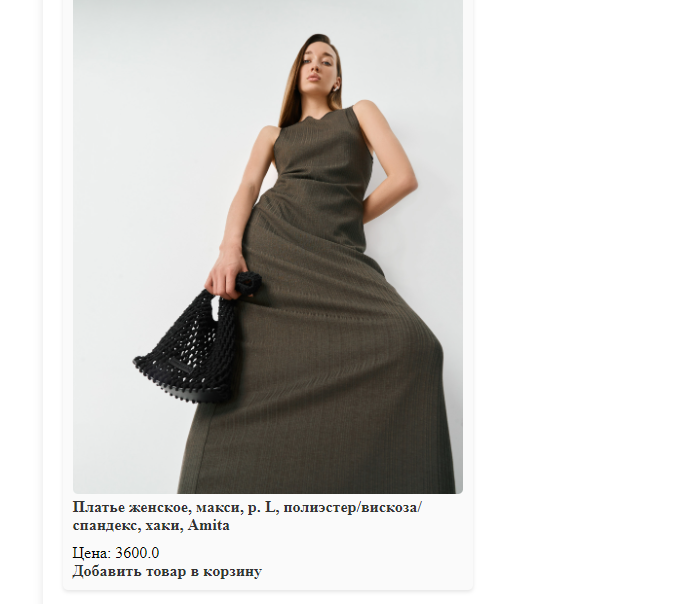




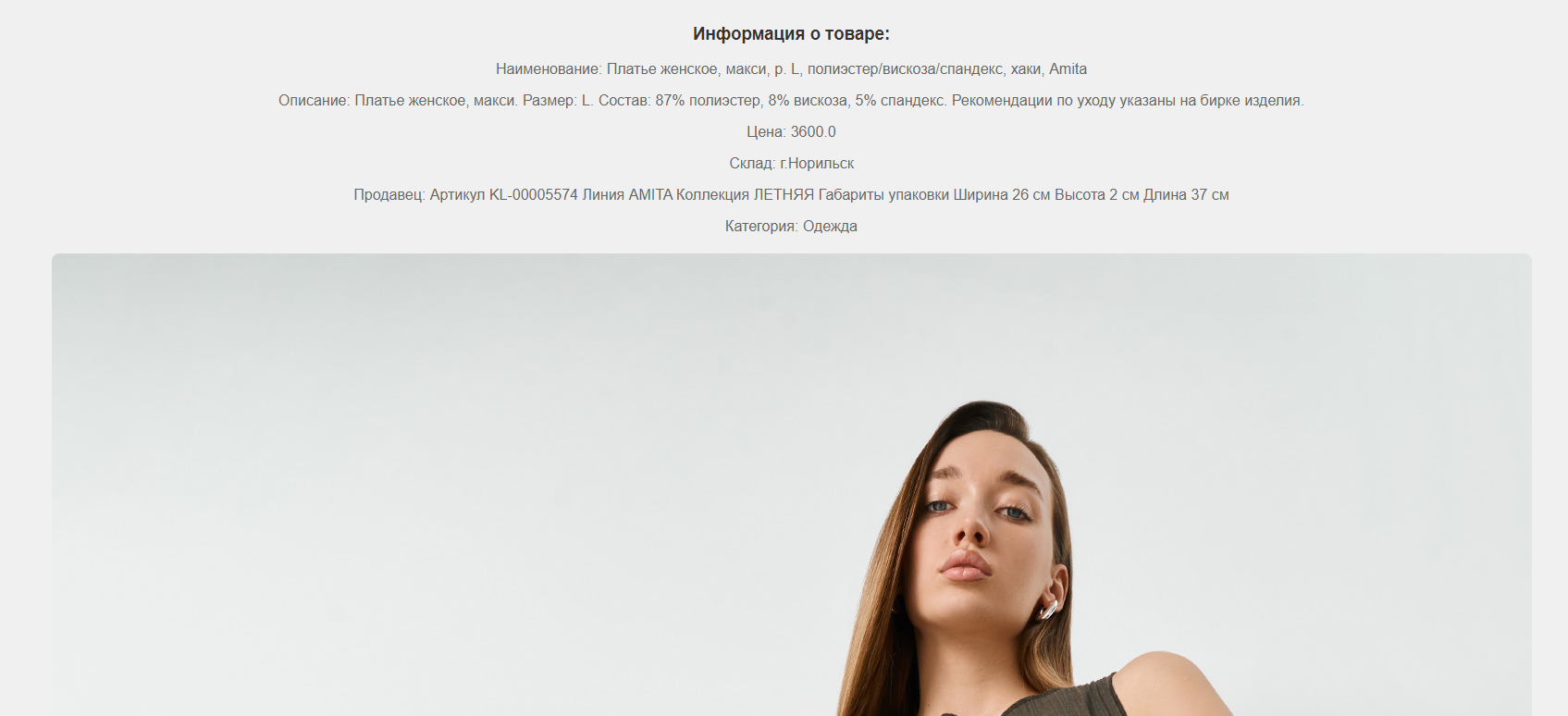
Вот мы и вошли под правами пользователя на нашу страничку. Дальше давайте проверим сохранился ли наш пользователь в базе данных, а также будет ли видно наш пароль там!

C:\Users\vilks\OneDrive\Изображения\Снимки экрана\Снимок экрана 2024-05-08 161045.png

Как и ожидалось, наш хитрый пароль скрыт, а мы были созданы в системе. Попробуем, что-нибудь заказать! Например это прекрасное платье. Но нужно сначала с ним ознакомится ! Кликаем по надписи “ платье женское”

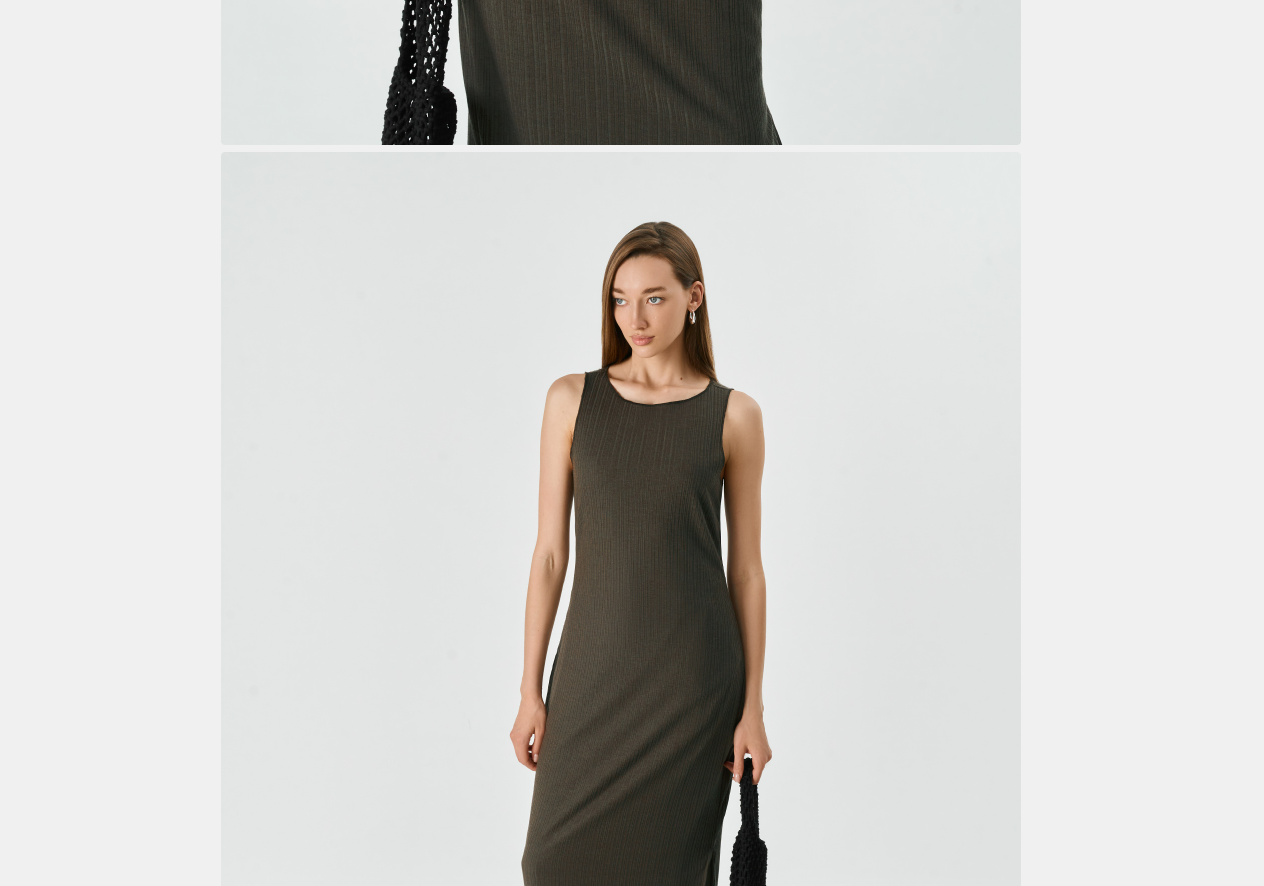


Кликаем по надписи “ платье женское” и переходим на страничку информация о товаре!



Вуаля, картинка уменьшена в Word, но на страничке ее отлично видно, так же можно пролистать ниже и посмотреть фотографии товара.

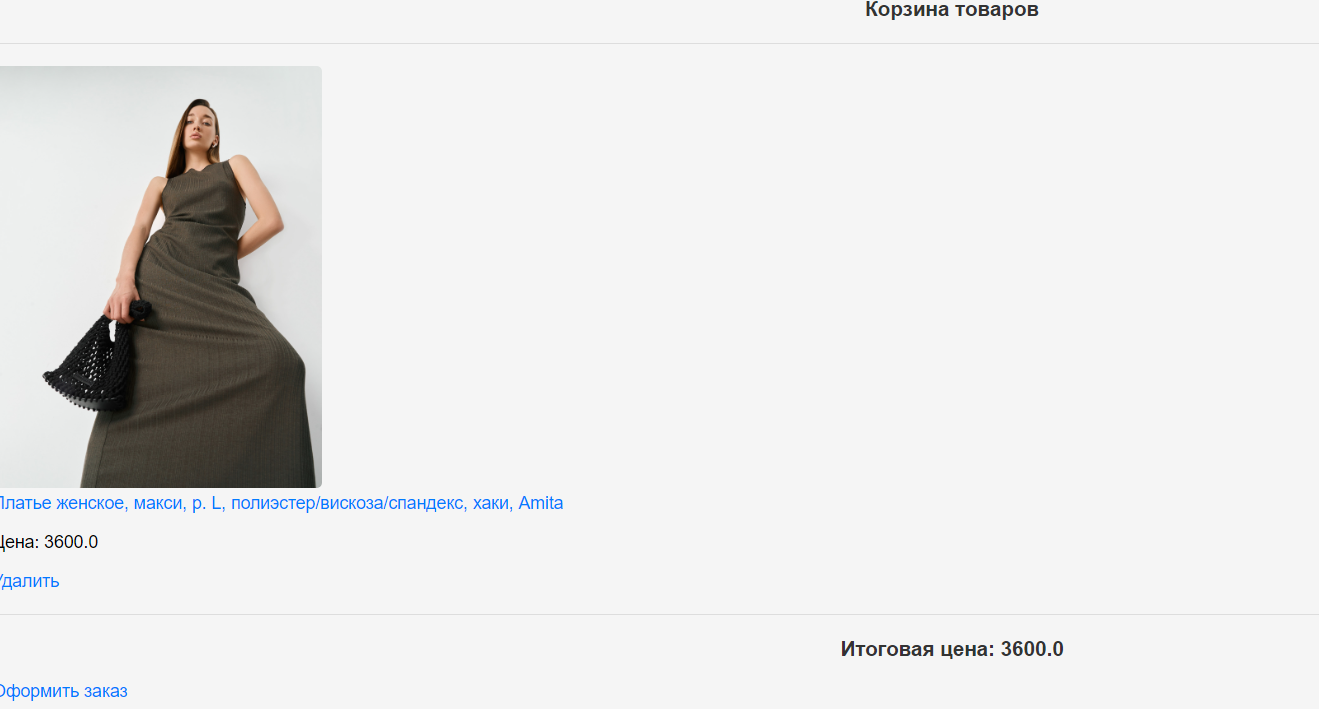




Товар нам понравился, давайте его закажем, кликаем на добавить товар в корзину, что находится под ценой!

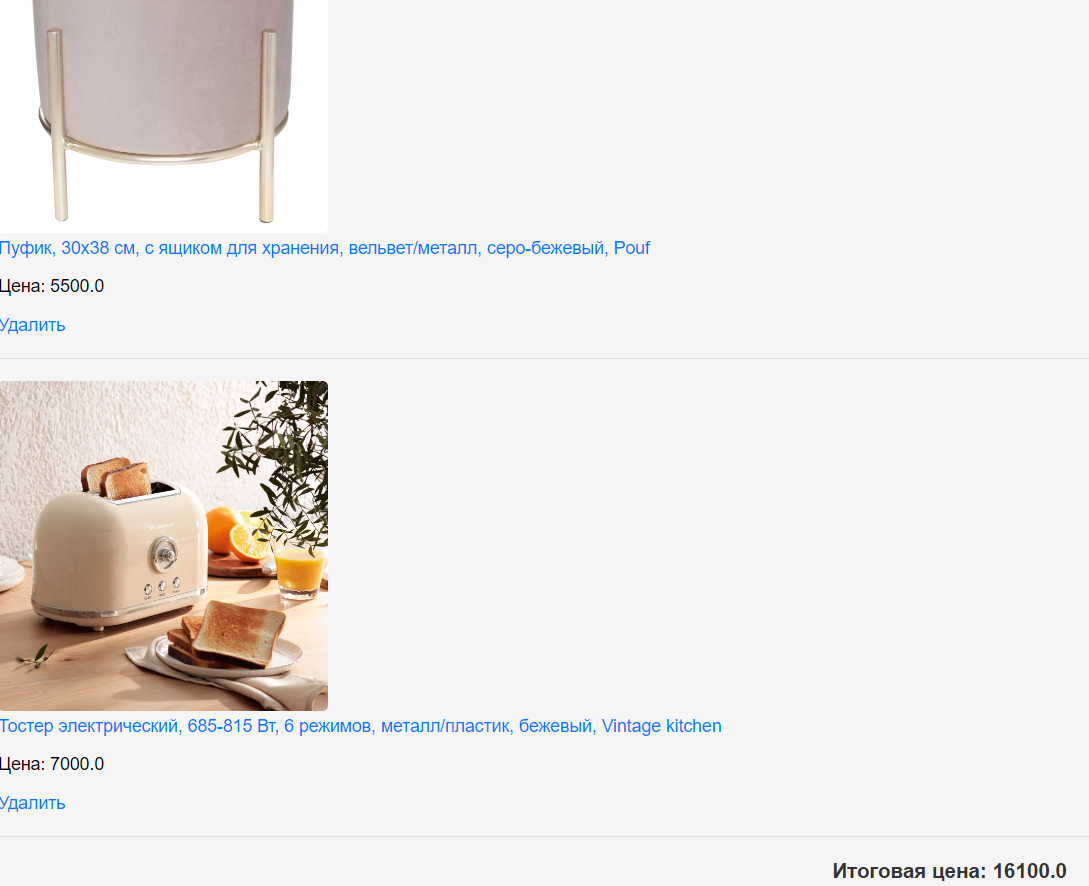
C:\Users\vilks\OneDrive\Изображения\Снимки экрана\Снимок экрана 2024-05-08 161525.png

Переходим в корзину товаров. Тут мы можем удалить товар, если он нам разонравился, а также посмотреть его повторно и оформить заказ. Так же производится итоговый расчет цены!



Давайте добавим еще пару товаров!

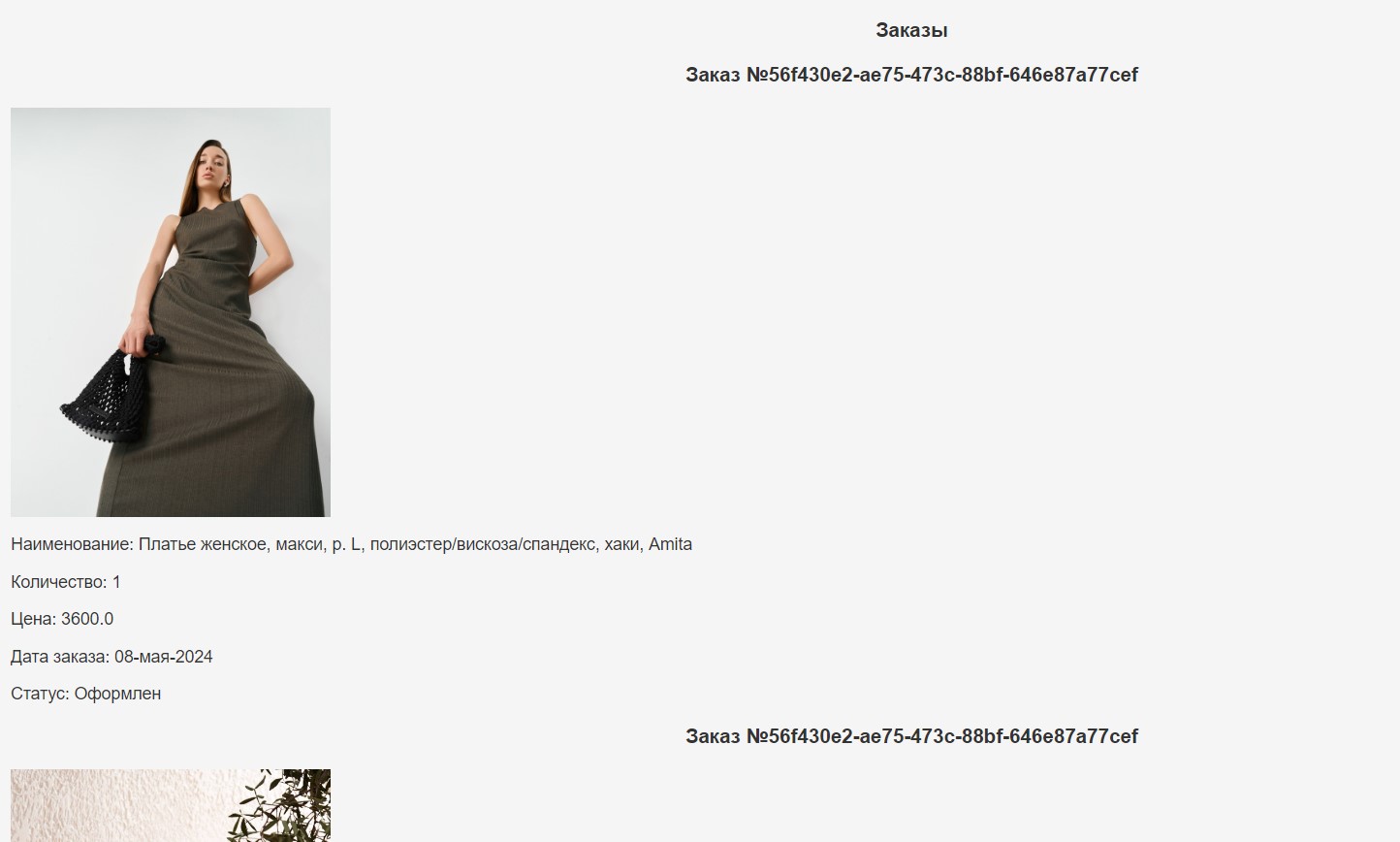
Чудесный тостер и пуфик, появились на нашей страничке, но что такое, итоговая цена не укладывается в наш бюджет! Давайте удалим товар из корзины!



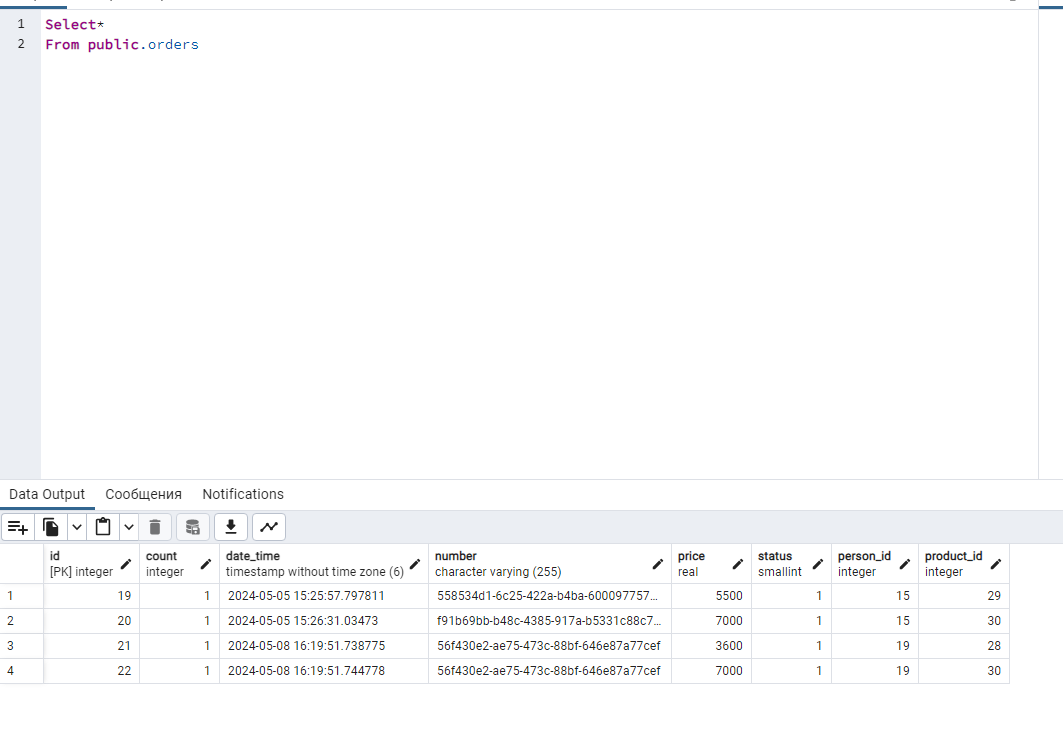
После нажатие на удалить пуфик исчезает из нашей корзины, а итоговая цена становится оптимальной.



Давайте уже оформим наш заказ!



Как мы видим, вещи разбились на отдельные заказы, чтобы их проще было отслеживать. Так же указана дата заказа, его статус и наименование. Можем убедится в базе данных, что заказ появился !

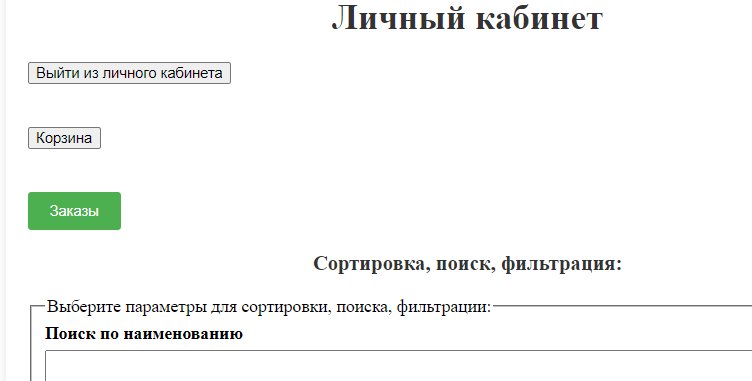


Так точно, вот два свежих заказа, имеющие свои индивидуальные номера и в тоже время заказанные одним пользователем!

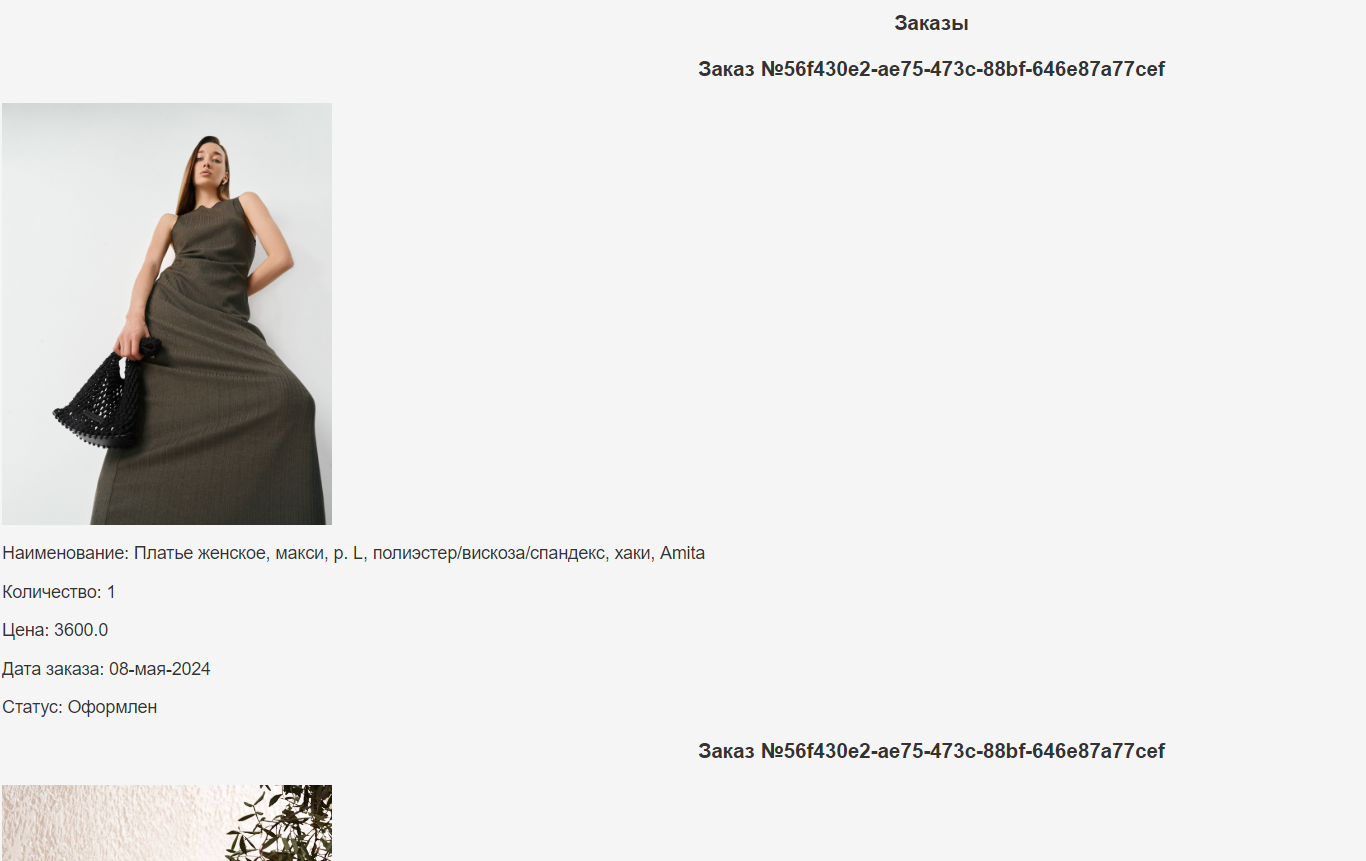
Вернемся в нашу корзину и убедимся, что товаров там больше нет.



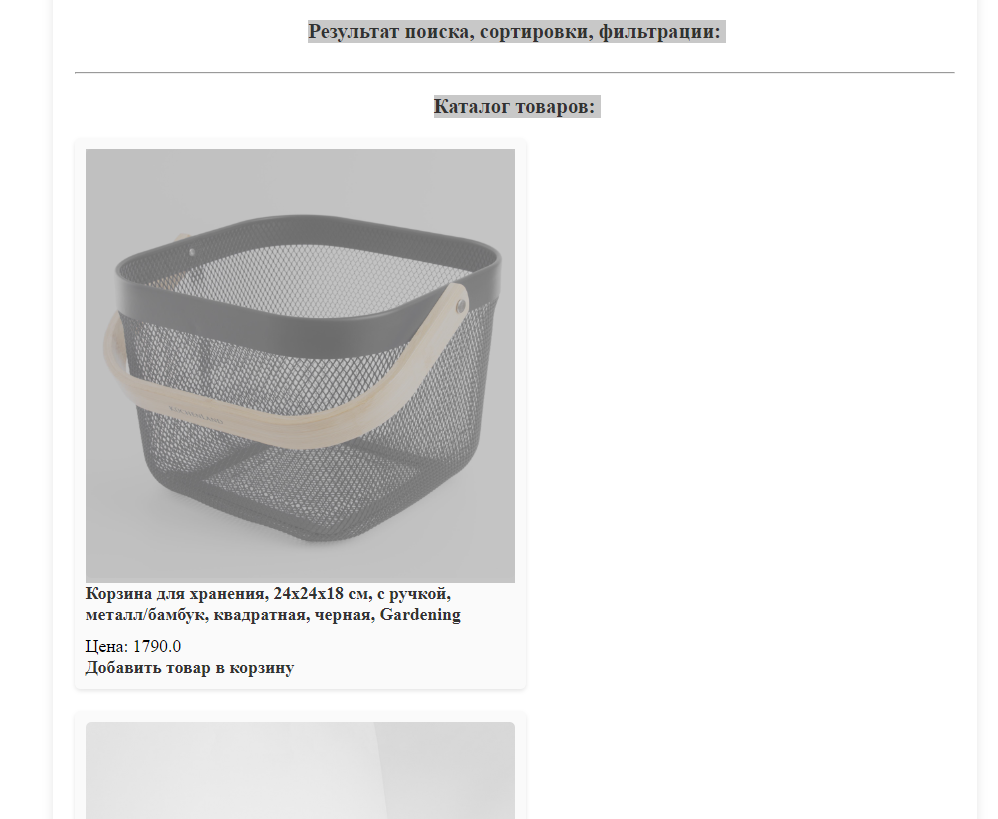
Но где же они? Для этого на главной страничке личного кабинета предусмотрена кнопка Заказы! Кликаем на нее!



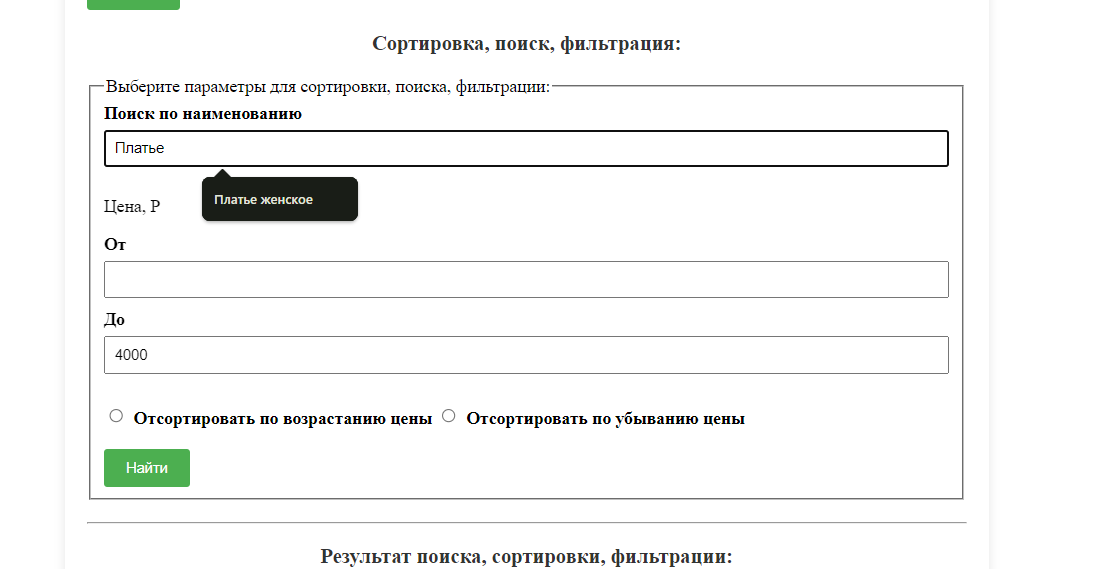
Вот и наши заказы



Пролистав ниже мы можем наткнуться на результат поиска, а также каталог товаров!

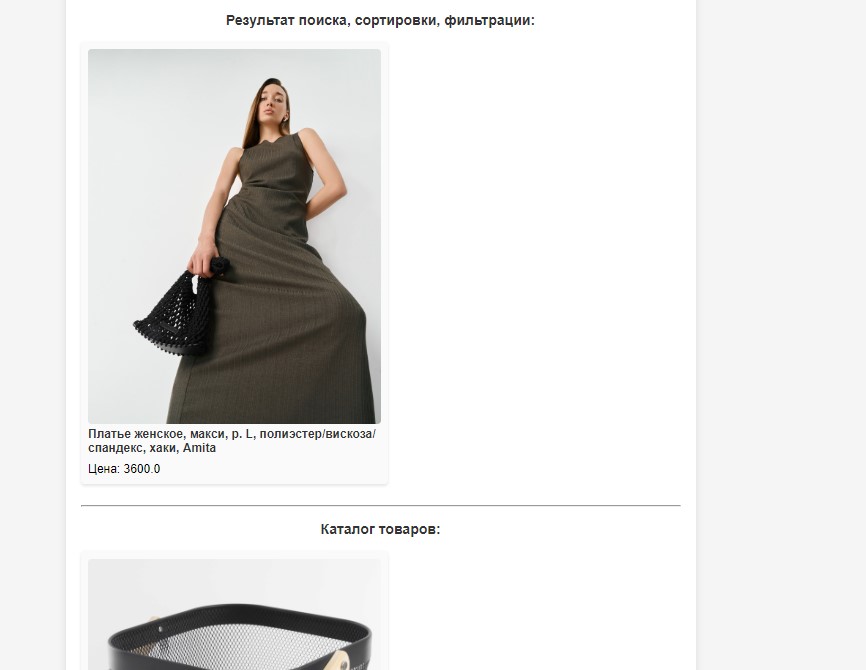


Давайте попробуем воспользоваться поиском и сортировкой!

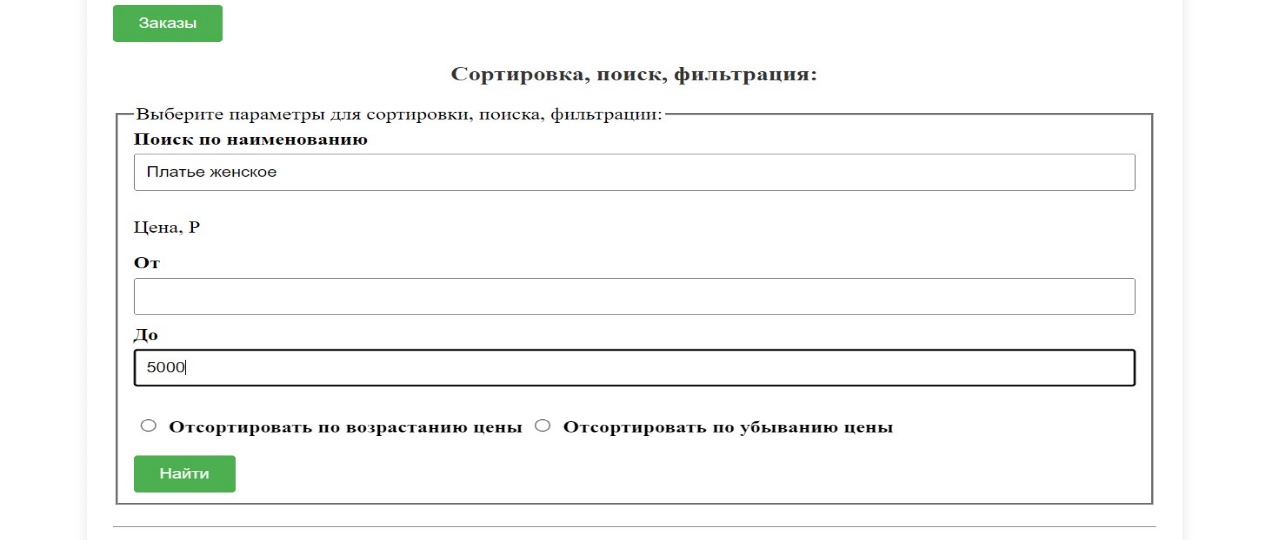


Представим, что у нас есть только 4000 и мы ищем платье!

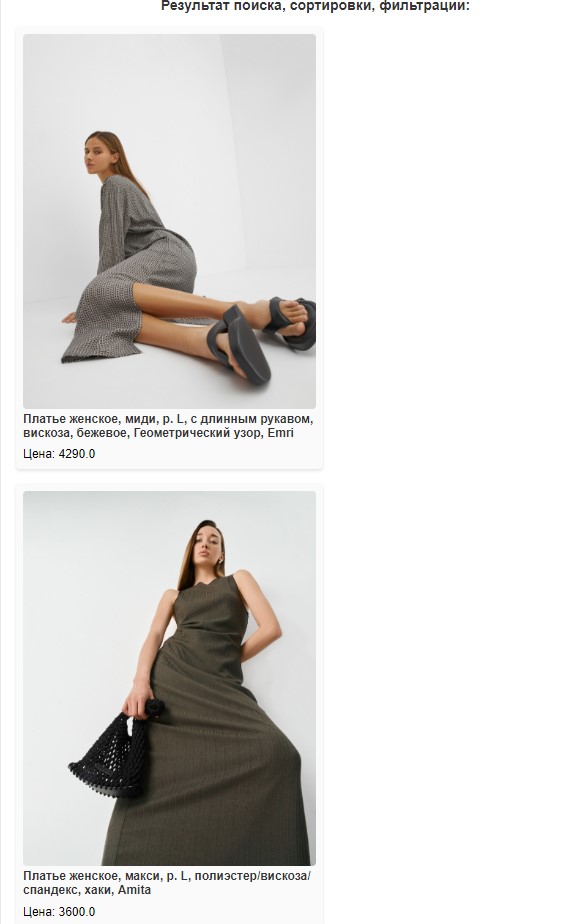
А вот и наше платье! ниже идет уже каталог товаров! Вдруг мы найдем, что еще прикупить!



Нам одолжили еще тысячу, давайте поднимем ценовой предел до 5000!

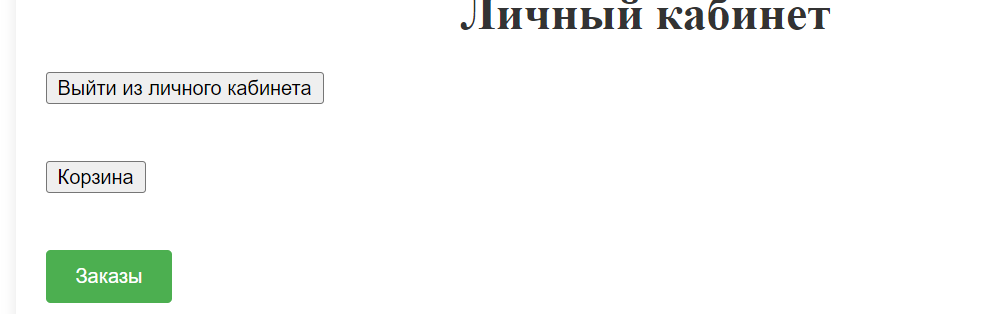


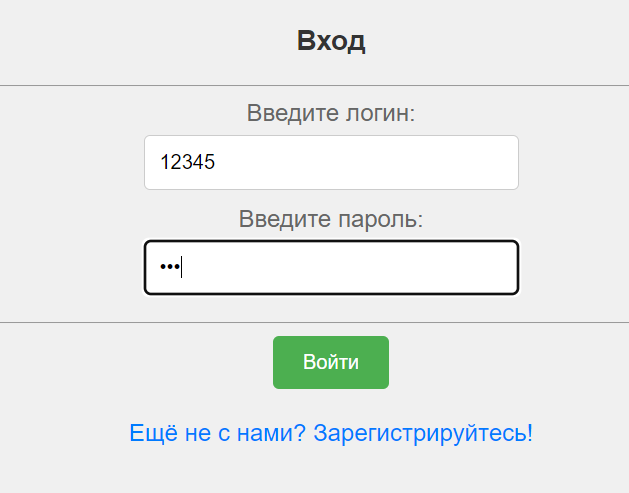
Результат изменился сразу на другой !



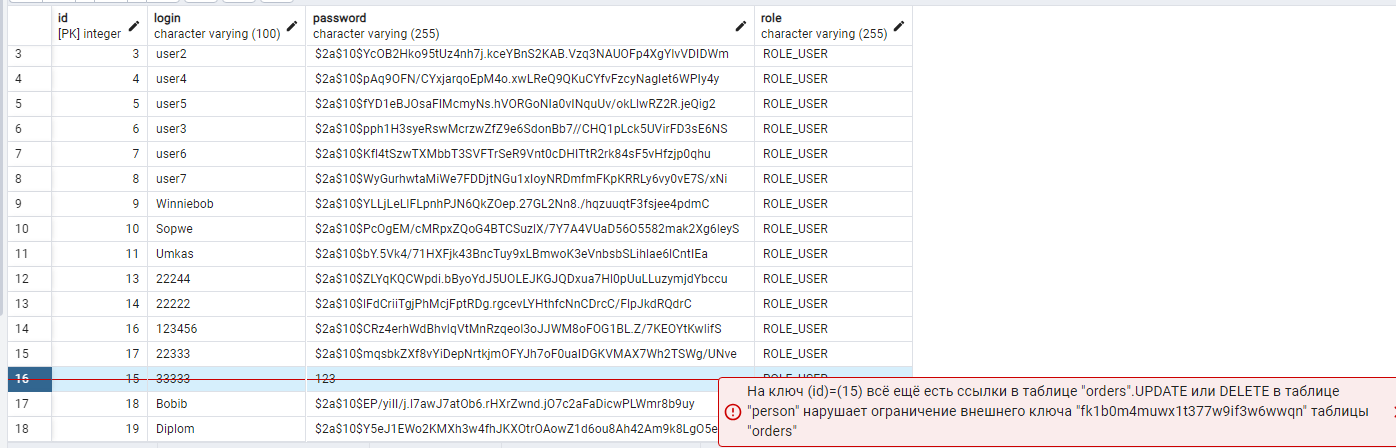
Теперь тут присутствует и платье за 4290!

Мы рассмотрели функционал нашего сайта будучи обычным пользователем, но что если мы зайдем под правами Администратора! Для этого нужно выйти из личного кабинета и ввести новый логин и пароль!

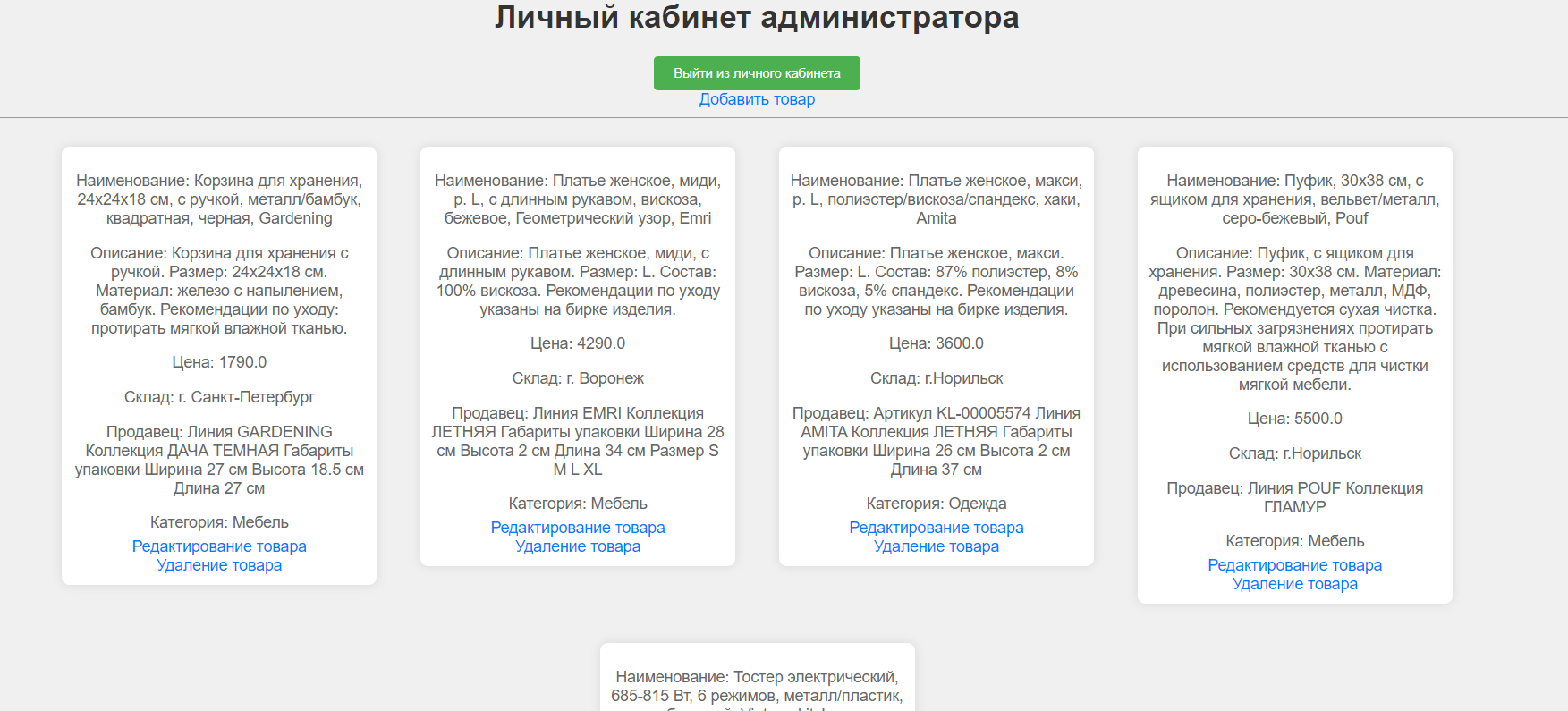




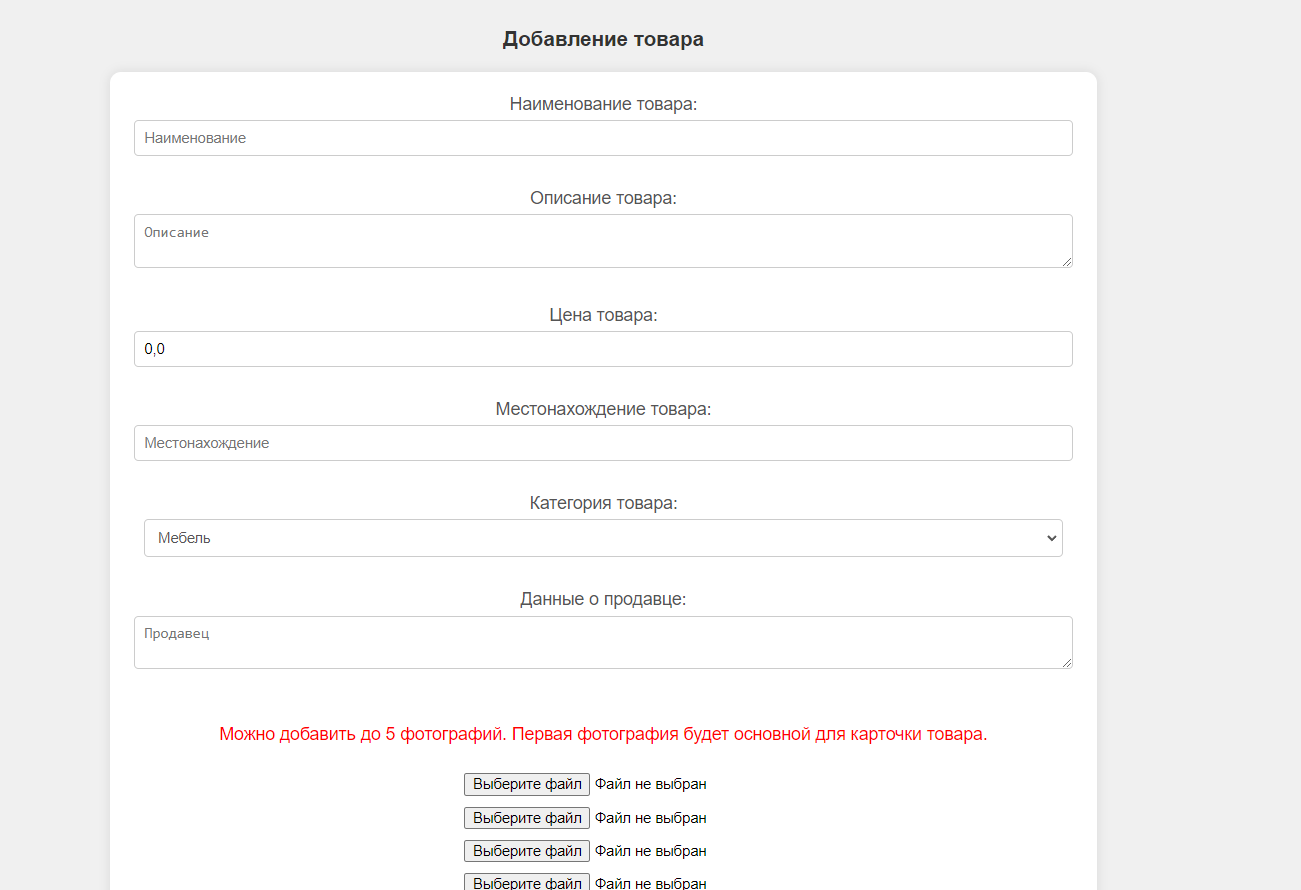
Хотелось бы отметить, что мы не можем удалить пользователя, который сделал заказ из базы данных, так как это нарушает таблицу внешнего ключа заказов!



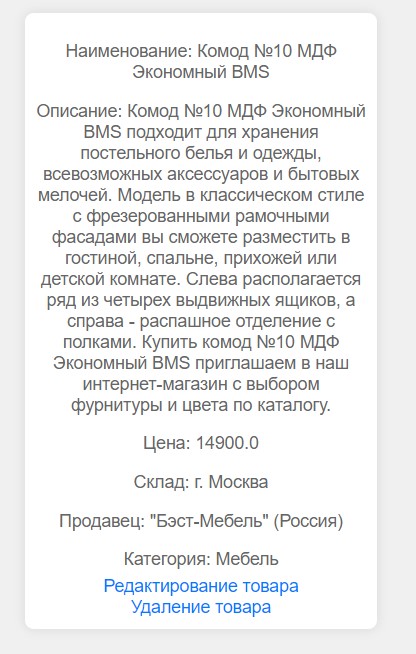
И так мы заходи под администратором и видим у него совершенно другую структуру сайта



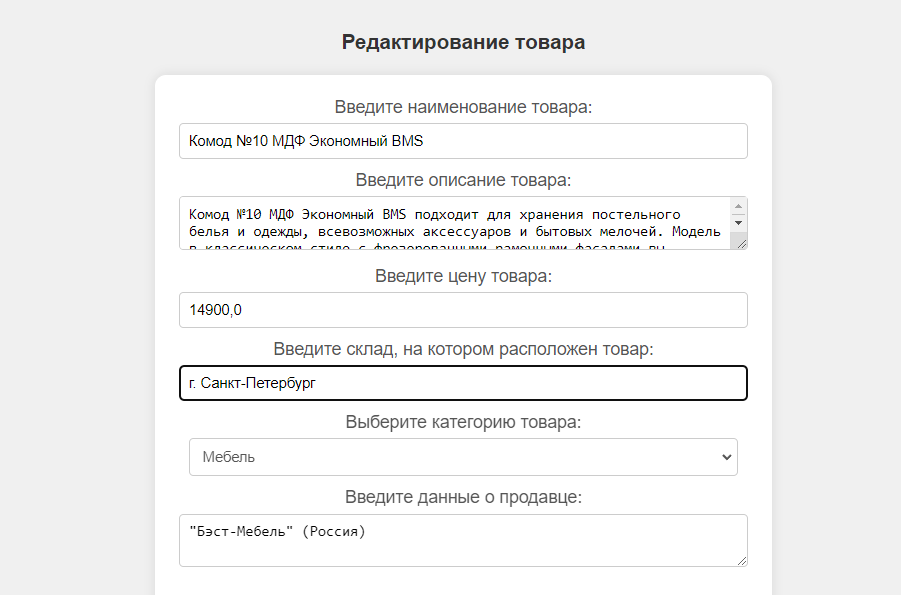
Тут присутствует функция выйти из личного кабинета, добавить товар, а так же его редактировать и удалить. Давайте попробуем выполнить все эти функции. Начнем с создания товара! Жмем “Добавить товар” и переходим на новую страничку!

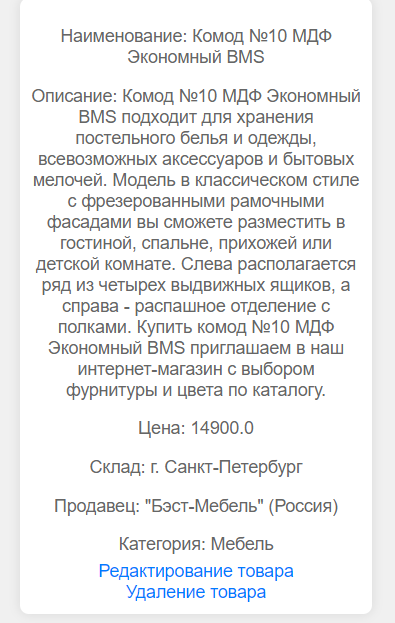


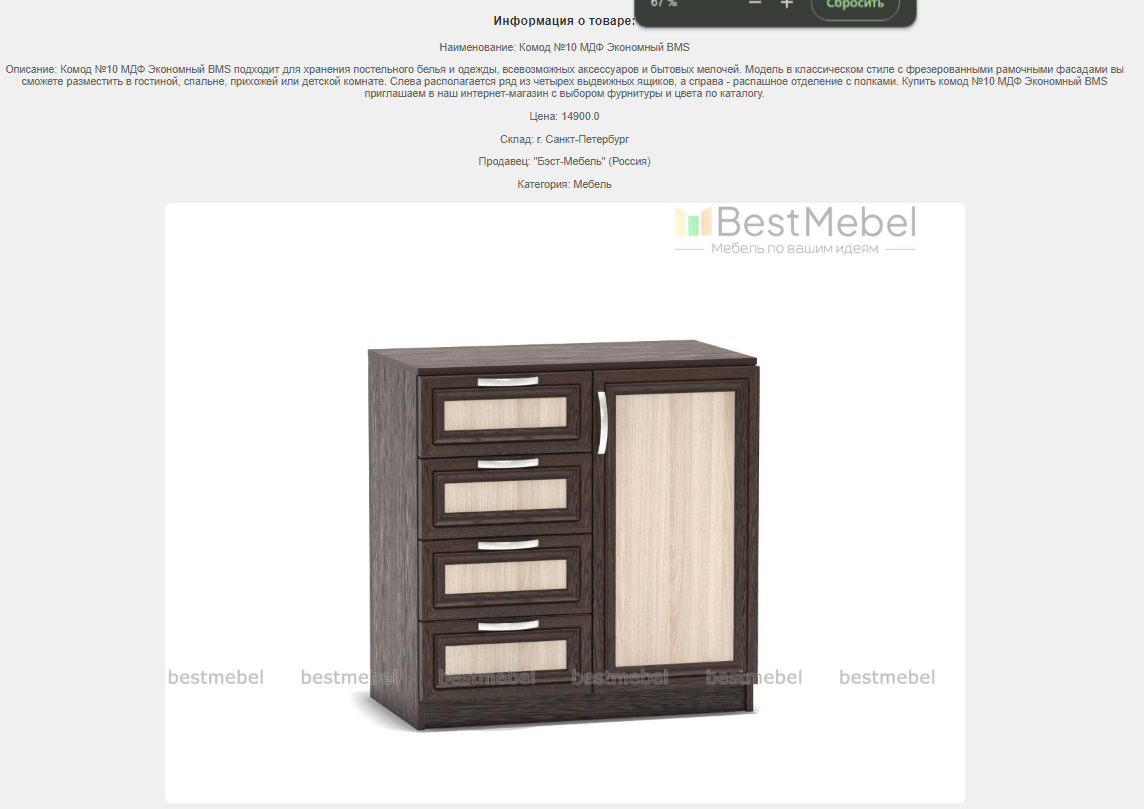
Тут мы видим возможность назвать наш товар, описать его, назначить цену, а также определить его местоположение, добавить его в категорию и написать данные о продавце! Попробуем все это заполнить!



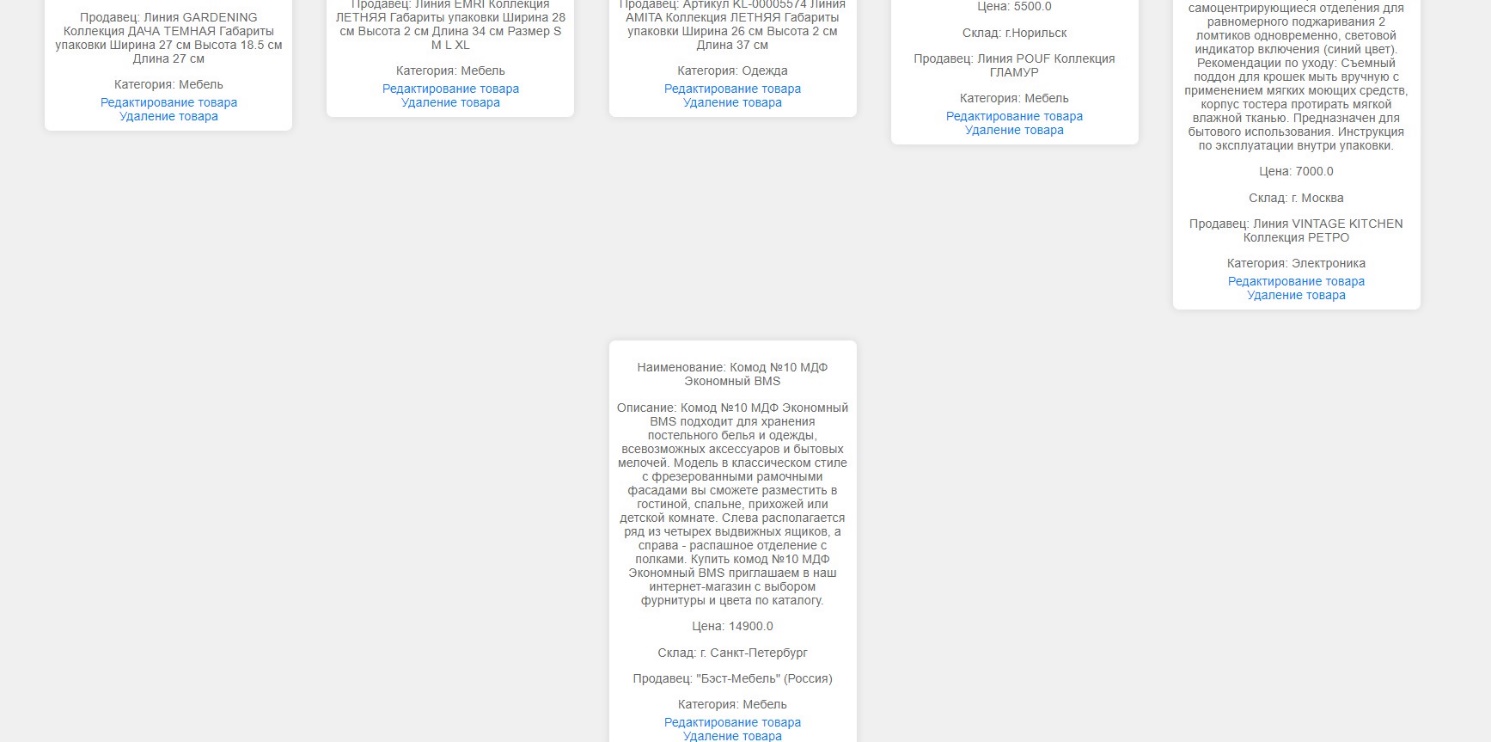
Готово, мы выбрали фотографии и заполнили все поля. Давайте убедимся в этом а заодно и редактируем город, ведь по ошибке мы указали Москву! Хотя товар в Санкт-Петербурге!



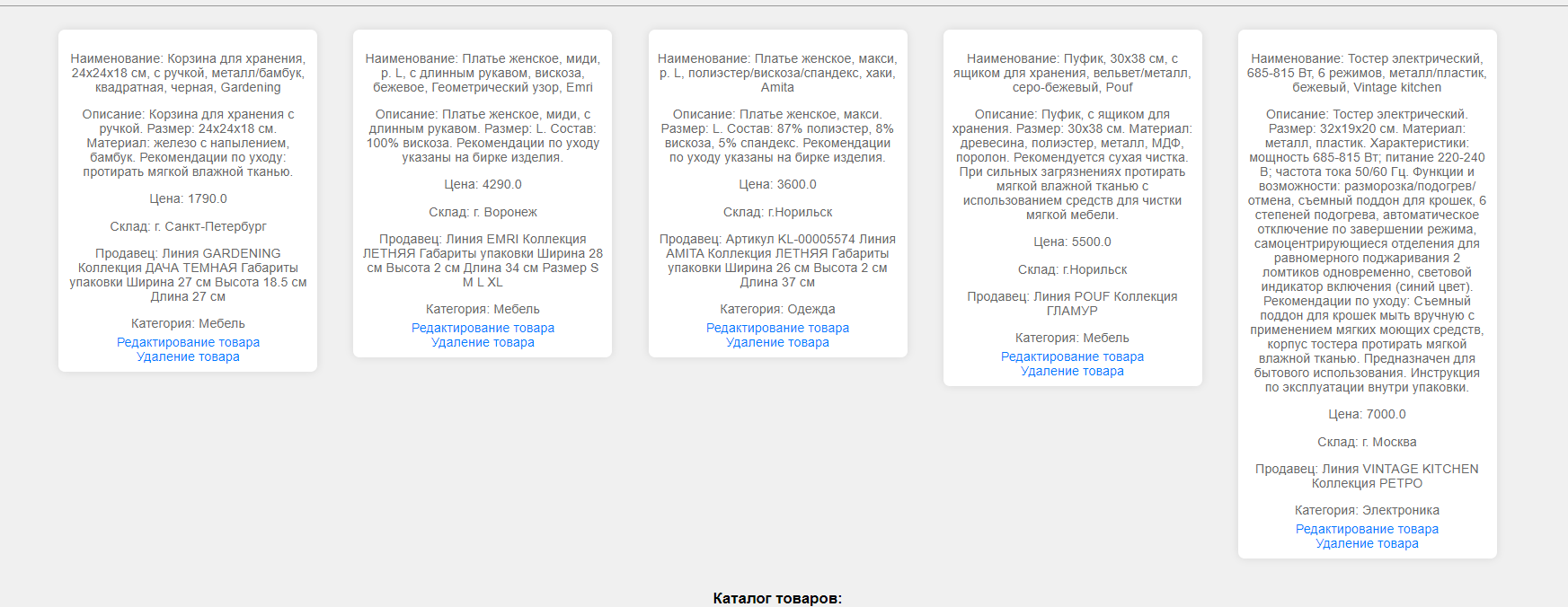




Так то лучше! Но пока мы меняли город, товар разобрали, придется его удалить!

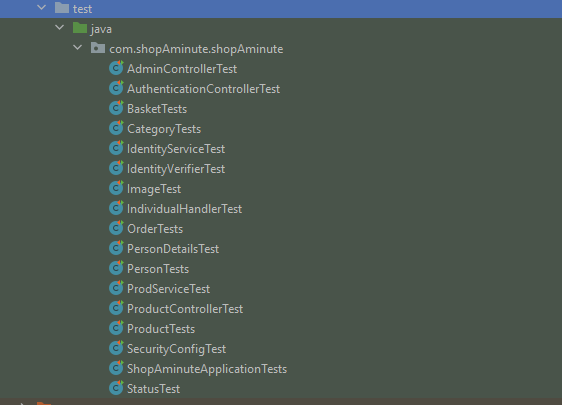


Вот продукт есть.

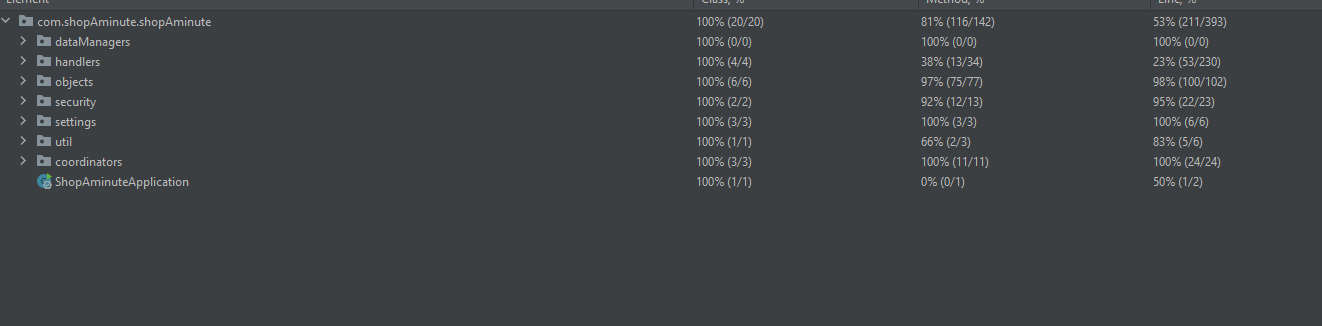


А вот продукта уже не стало.

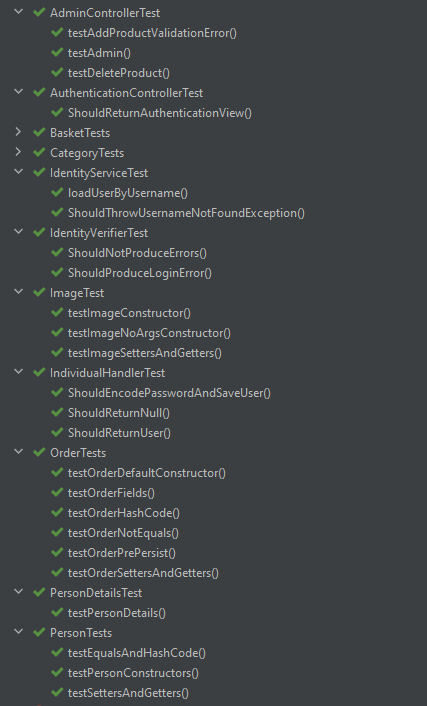
И на последок хотелось бы рассмотреть покрытие тестами наше приложение. В теоретической части мы так же рассмотрели, какой класс и тест за что отвечает, тут просто посмотрим, какого покрытия мне удалось достичь.



Что ж давайте посмотрим.



Ого. 81 % Это не так уж и плохо! А зеленые галочки радуют глаз!



**Заключение.**

В рамках дипломного проекта по разработке веб-приложения на языке Java были достигнуты следующие результаты:

1. **Тестирование и качество программного продукта:**
   * Применение JUnit и Mockito позволило эффективно тестировать отдельные компоненты системы, обеспечив высокий уровень покрытия кода тестами (81%), что способствовало надежности и безопасности приложения.
   * Разработанные тесты охватили ключевые аспекты приложения, включая контроллеры, сервисы, репозитории и сущности.
2. **Связь с базами данных:**
   * Интеграция с системой управления базами данных и использование Hibernate обеспечили высокую производительность и надежность хранения данных.
   * Тестирование взаимодействия с базой данных подтвердило корректность операций CRUD и эффективность выполнения запросов.
3. **Разработка на языке Java:**
   * Выбор Java обусловлен его возможностями, такими как строгая типизация и многопоточность.
   * Применение Spring Framework и Spring Boot позволило создать гибкую архитектуру и ускорить процесс разработки.
4. **Практическая значимость:**
   * Повышение качества и безопасности веб-приложения, что подтверждает практическую значимость проекта.
   * Рекомендации по совершенствованию включают дополнительные инструменты автоматизации тестирования и расширение тестового покрытия.
5. **Общий итог:**
   * Цель проекта достигнута: разработано веб-приложение с высоким уровнем покрытия тестами.
   * Задачи выполнены: созданы тесты для всех ключевых компонентов системы.
   * Гипотеза о том, что комплексное тестирование повышает качество программного продукта, была доказана.

Таким образом, дипломный проект демонстрирует успешное применение современных методик тестирования и разработки на языке Java, обеспечивая качество и безопасность веб-приложения. Это подтверждает его практическую значимость и вклад в развитие области разработки программного обеспечения.

# Список используемой литературы

Здесь нужно будет указать список используемой литературы, ссылки на все ресурсы, которые нужны были для создания проектной работы.

Русскоязычная литература:

1. Блох М. "Java. Эффективное программирование", 3-е изд. - Санкт-Петербург: Символ-Плюс, 2019. - 450 с.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. "Паттерны проектирования". - Москва: Вильямс, 2020. - 368 с.
3. Седжвик Р., Уэйн К. "Алгоритмы на Java", 5-е изд. - Москва: Вильямс, 2018. - 830 с.

Иностранная литература:

1. Bloch J. "Effective Java", 3rd edition. - Boston: Addison-Wesley, 2018. - 416 p.
2. Gamma E., Helm R., Johnson R., Vlissides J. "Design Patterns: Elements of Reusable Object-Oriented Software". - Boston: Addison-Wesley, 1994. - 395 p.
3. Horstmann C.S. "Core Java Volume I--Fundamentals", 11th edition. - Boston: Prentice Hall, 2018. - 928 p.

Интернет-сайты:

1. "Spring Framework Documentation" - [[https://spring.io/docs](](https://spring.io/docs%5D()<https://spring.io/docs> - Дата обращения: 08.05.2024.
2. "JUnit 5 User Guide" - [[https://junit.org/junit5/docs/current/user-guide/](](https://junit.org/junit5/docs/current/user-guide/%5D()<https://junit.org/junit5/docs/current/user-guide/> - Дата обращения: 08.05.2024.
3. "Hibernate ORM Documentation" - [[https://hibernate.org/orm/documentation/](](https://hibernate.org/orm/documentation/%5D()<https://hibernate.org/orm/documentation/> - Дата обращения: 08.05.2024.