МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ СОЦИОКУЛЬТУРНЫХ КОММУНИКАЦИЙ
Кафедра информационных технологий

ИНТЕРНЕТ-ВИТРИНА МАГАЗИНА ИНТЕРЬЕРНЫХ ЧАСОВ

Курсовой проект

Шик Карины Вадимовны
Студентки 4 курса,
специальности
«Прикладная информатика»
Научный руководитель:
доцент, к.т.н.
Царик С.В.

Минск, 2020

СОДЕРЖАНИЕ

РЕФЕРАТ

Курсовой проект содержит: 26 страниц, 14 иллюстраций, 8 листингов, 12 источников, 2 приложения.

Ключевые слова: ИНТЕРНЕТ-ВИТРИНА, JAVA, SPRING FRAMEWORK, APACHE MAVEN FRAMEWORK, POSTGRESQL.

Цель работы: разработка адаптивной интернет-витрины магазина интерьерных часов.

Результатом курсового проекта является адаптивная интернет-витрина магазина интерьерных часов с реализацией просмотра, добавления, редактирования и удаления товаров администратором сайта.

РЭФЕРАТ

Курсавой праект змяшчае: 26 старонак, 14 ілюстрацый, 8 лістынгаў, 12 крыніц, 2 прыкладання.

Ключавыя словы: ІНТЭРНЭТ-ВІТРЫНА, JAVA, SPRING FRAMEWORK, APACHE MAVEN FRAMEWORK, POSTGRESQL.

Мэта работы: распрацоўка адаптыўнай інтэрнэт-вітрыны крамы інтэр'ерных гадзін.

Вынікам курсавога праекта з'яўляецца адаптыўная інтэрнэт-вітрына крамы інтэр'ерных гадзін з рэалізацыяй прагляду, дадання, рэдагавання і выдалення тавараў адміністратарам сайта.

ABSTRACT

The course project includes: 26 pages, 14 illustrations, 8 listings, 12 sources, 2 appendices.

Keywords: WEB STORE, JAVA, SPRING FRAMEWORK, APACHE MAVEN FRAMEWORK, POSTGRESQL.

The aim of this work is to develop an interior clocks’ adaptive web store.

The result of the course project is the interior clocks’ adaptive web store with the implementation of viewing, adding, editing and deleting products by the site administrator.

ВВЕДЕНИЕ

Эффективным инструментом продвижения товаров и услуг соответствующей организации является наличие веб-сайта, основным назначением которого является связь покупателя с производителем. Актуальность создания веб-сайта обусловлена тем, что существует возможность ознакомиться с перечнем предоставляемых производителем товаров и услуг в любое удобное пользователю время.

Целью курсового проекта является разработка адаптивной интернет-витрины магазина интерьерных часов с использованием следующих языков программирования и технологий: язык программирования Java, Spring Framework, Apache Maven Framework, система управления базами данных PostgreSQL, Bootstrap Framework.

Достижение заданной цели осуществимо в ходе выполнения следующих задач:

1. Провести обзор следующих инструментальных средств и технологий: язык программирования Java, Spring Framework, Spring MVC, Apache Maven Framework, система управления базами данных PostgreSQL, Bootstrap Framework.
2. С помощью языка программирования Java и системы управления базами данных PostgreSQL разработать серверную часть проекта.
3. В соответствии с запросами `GET`, `POST`, `PUT` и `DELETE`, поступающими от пользователя, реализовать отображение всех товаров на сайте, отображение определенного товара, возможность добавления и редактирования товара, удаление товара.
4. Каталог товаров отобразить со следующей обязательной информацией: название, категория, цена и фотография товара.
5. Права на добавление, редактирование и удаление товаров предоставить администратору сайта.
6. Реализацию пользовательского интерфейса осуществить с помощью фреймворка Bootstrap.

ГЛАВА 1 ОБЗОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ И ТЕХНОЛОГИЙ

1.1 Язык программирования Java

Язык программирования Java – Си-подобный объектно-ориентированный язык программирования, исполняемость кода которого не зависит от операционной системы или установленного программного обеспечения.

Основная особенность языка программирования Java, которая позволяет решать проблемы обеспечения безопасности и переносимости программ, состоит в том, что компилятор данного языка программирования выдает не исполняемый код, а так называемый байт-код – оптимизированный набор инструкций, предназначенных для выполнения в исполняющей системе Java, называемой виртуальной машиной Java [1, с. 44]. Программы, написанные на данном языке программирования, транслируются в байтовый код (этап компиляции), который затем исполняется виртуальной машиной Java (этап интерпретации).

Для разработки программ на языке программирования Java необходим специальный комплект разработки, именуемый «Java Development Kit (JDK)». Комплект разработки JDK – комплект разработчика приложений на языке программирования Java, включающий в себя компилятор, стандартные библиотеки классов языка программирования Java, примеры, документацию, различные утилиты и исполнительную систему Java, или Java Runtime Environment (JRE) [2].

Язык программирования Java предлагает специальную форму синтетических метаданных – аннотации – которая может быть добавлена в исходный код программы. Аннотации представляют из себя дескрипторы, включаемые в текст программы, и используются для хранения метаданных программного кода, необходимых на разных этапах жизненного цикла программы [3]. Информация, хранимая в аннотациях, может использоваться соответствующими обработчиками для создания необходимых вспомогательных файлов или для маркировки классов, полей, методов и так далее.

Аннотации помечаются символом «@». Например, стандартная аннотация `@Override` гарантирует, что метод подкласса переопределяет метод родительского класса, в противном случае возникает ошибка времени компиляции.

Пример использования аннотации `@Override` приведен в листинге 1.1, где переопределяется метод `method()` класса `AnotherClass`, наследуемого от класса `SomeClass`.

Листинг 1.1 – Аннотация `@Override`

```
class SomeClass {
     void method() {
       System.out.println("Работает метод родительского класса.");
    }
}
class AnotherClass extends SomeClass {
     @Override
     void method() { // переопределяем метод
       System.out.println("Работает метод класса-потомка.");
    }
}
```

Так, например, если в имени переопределяемого метода разработчик допустит опечатку, компилятор учтет аннотацию `@Override` и выдаст соответствующую ошибку, в противном случае – создаст новый метод.

Таким образом, аннотация `@Override` никак не влияет на переопределение метода в процессе написания исходного кода, но позволяет контролировать успешность переопределения при компиляции или сборке.

1.2 Семейство фреймворков Spring

Семейство фреймворков Spring включает в себя следующие программные платформы (фреймворки):

- Spring Boot – фреймворк, целью которого является упрощение создания приложений на основе Spring. Данный фреймворк позволяет наиболее простым способом создать веб-приложение, требуя минимум усилий по настройке проекта и написанию кода.
- Spring Framework – фреймворк, предоставляющий комплексную модель программирования и конфигурации для современных корпоративных приложений на основе языка программирования Java на любой платформе развертывания [4].
- Spring MVC – фреймворк, ориентированным на запросы.
- Spring Data – фреймворк, предоставляющий единую модель программирования для доступа к данным, сохраняя при этом черты базового хранилища [5].

- Spring Security – среда аутентификации и контроля доступа, которая фокусируется на обеспечении как аутентификации, так и авторизации для приложений, написанных на языке программирования Java [6].

Рассмотрим два наиболее популярных фреймворка семейства Spring – Spring Framework и Spring MVC.

Как видно на рисунке 1.1, Spring Framework имеет модульную структуру, что позволяет подключать только те модули, которые необходимы для разрабатываемого приложения.

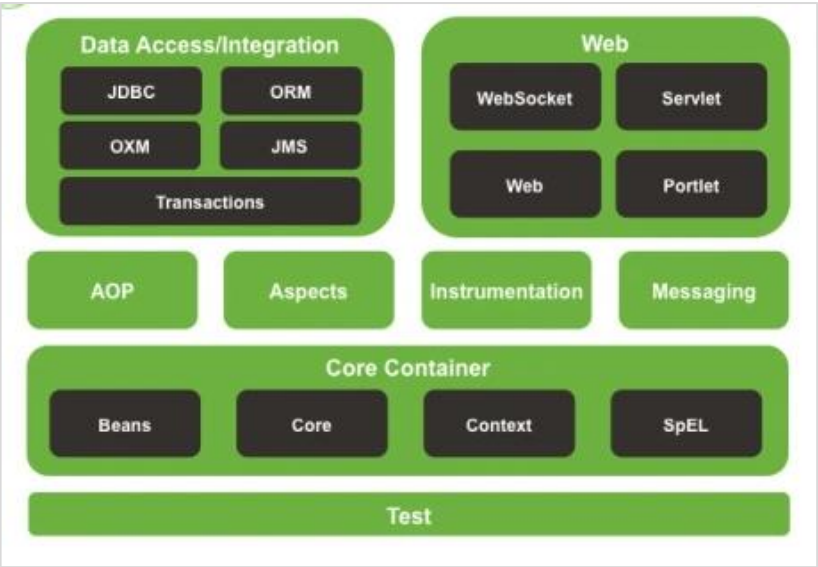

Рисунок 1.1 – Структура Spring Framework

Spring Framework содержит множество функций, организованных примерно в двадцать модулей. Модули группируются в следующие основные группы: «Основной контейнер», «Доступ к данным и интеграция», «Веб», «Тестирование», «Аспектно-ориентированное программирование» и другие.

Центральной частью Spring Framework является контейнер «Inversion of Control» («Инверсия контроля»), который предоставляет средства конфигурирования и управления объектами языка программирования Java с помощью рефлексии. Контейнер «Inversion of Control» отвечает за управление жизненным циклом объектов: создание, вызов методов инициализации и конфигурирование. Создаваемые контейнером объекты именуются управляемыми объектами, или «Beans». Конфигурирование контейнера осуществляется путем загрузки XML-файлов, содержащих определение управляемых объектов и предоставляющих информацию, необходимую для создания управляемых объектов [7, с. 431]. Управляемые объекты могут быть получены с помощью поиска или внедрения зависимости.

Spring MVC является фреймворком, обеспечивающим архитектуру паттерна «Model – View – Controller» («Модель – Представление – Контроллер») и разделяющий соответственно логику ввода, бизнес-логику и интерфейс, обеспечивая при этом свободную связь между вышеперечисленными компонентами.

Логика работы Spring MVC построена вокруг сервлета «DispatcherServlet», который принимает и обрабатывает все HTTP-запросы пользователя и ответы на них. Процесс обработки HTTP -запроса сервлетом приведен на рисунке 1.2:

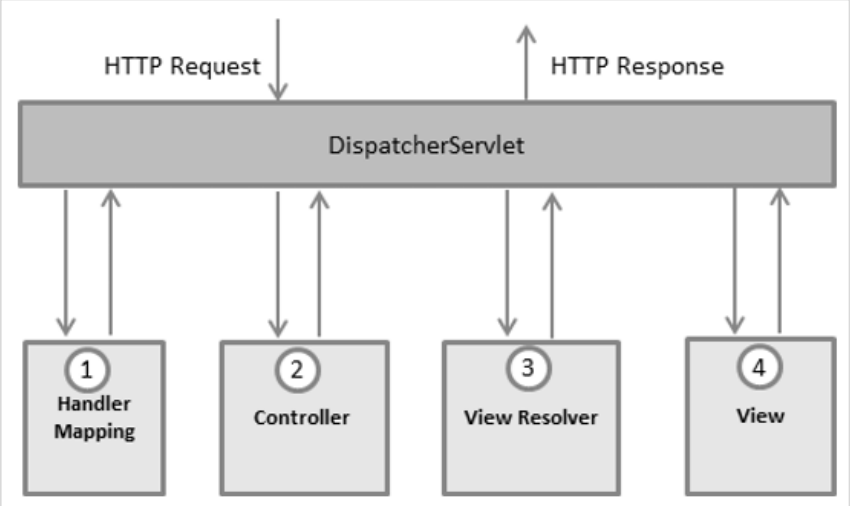

Рисунок 1.2 – Процесс обработки HTTP -запроса сервлетом «DispatcherServlet»

Последовательность событий, соответствующая входящему HTTP-запросу заключается в следующем:

- После получения HTTP-запроса сервлет «DispatcherServlet» обращается к интерфейсу «HandlerMapping», который определяет, какой контроллер должен быть вызван. Далее сервлет отправляет запрос в нужный контроллер.
- Контроллер в свою очередь принимает запрос и вызывает соответствующий служебный метод, основанный на методах `GET` или `POST`. Вызванный метод определяет данные модели, основанные на определенной бизнес-логике, и возвращает в сервлет имя соответствующего представления.
- С помощью интерфейса «ViewResolver» сервлет определяет, какое представление необходимо использовать на основании полученного имени.
- Далее сервлет отправляет данные модели в виде атрибутов в соответствующее представление, который в конечном итоге отображается в браузере.

1.3 Apache Maven Framework

Apache Maven Framework – инструмент, предназначенный для автоматизации процесса сборки проектов на основе описания структуры проектов в файле на языке POM (Project Object Model), который является подмножеством формата XML [8]. Apache Maven Framework обеспечивает декларативную сборку проекта, что позволяет прописывать в файл описания проекта не отдельные команды выполнения, а соответствующие спецификации.

Apache Maven Framework обладает собственной терминологией. Ключевыми понятиями Apache Maven Framework являются:

- «Артефакт» – ресурс, сгенерированный проектом Apache Maven Framework.
- «Зависимость», представляющая собой библиотеку, которая используются в проекте для компиляции или тестирования кода.
- «Плагин», необходимый при сборке проекта.
- «Архетип» – стандартная компоновка каталогов и файлов в проектах различного типа. В соответствии с архетипом Apache Maven Framework строит необходимую структуру проекта.

На рисунке 1.3 представлена стандартная структура Maven-проекта. В зависимости от типа приложения (консольное, интерфейсное, web и так далее) структура проекта может отличаться.

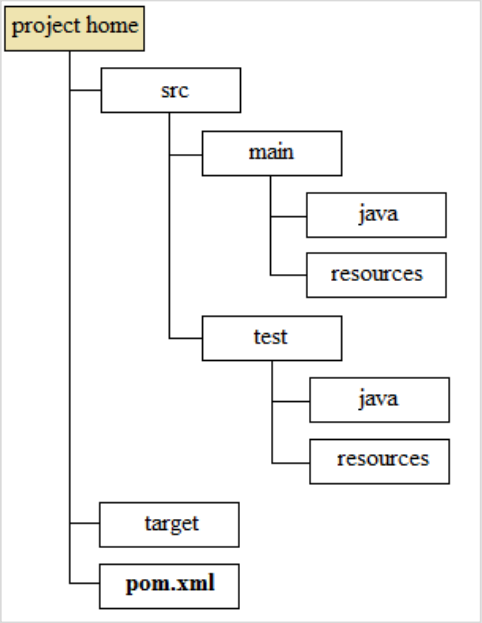

Рисунок 1.3 – Структура простого Maven-проекта

Структура Maven-проекта представляет собой:

- Исходные файлы проекта, расположенные в директории «scr».
- Исходные коды проекта, расположенные в директории «scr/main».
- Исходные файлы с расширением *.java, расположенные в директории «src/main/java».
- Ресурсные файлы, которые используются при компиляции или исполнении, расположенные в директории «src/main/resources».
- Исходные файлы для организации тестирования, расположенные в директории «src/test».
- JUnit-тест-задания для автоматического тестирования, расположенные в директории «src/test/java».
- Файлы для сборки проекта, расположенные в директории «target».
- Проектный файл «pom.xml» (файл объектной модели проекта), расположенный в корне каталога проекта.

Файл «pom.xml» является ядром конфигурации проекта в Apache Maven Framework. В файле «pom.xml» описываются все зависимости проекта, а для каждого используемого в проекте артефакта указываются параметры «groupId», «artifactId» и «version».

Apache Maven Framework базируется на плагин-архитектуре, которая позволяет использовать плагины для задач компиляции, сборки, а также тестирования проекта.

1.4 Система управления базами данных PostgreSQL

Система управления базами данных PostgreSQL – свободная объектно-реляционная система управления базами данных, базирующаяся на языке SQL.

Фундаментальной характеристикой объектно-реляционной базы данных является поддержка пользовательских объектов и поведения объектов, включая типы данных, функции, операции, домены и индексы, что делает систему управления базами данных PostgreSQL невероятно гибким инструментом. Система управления базами данных PostgreSQL позволяет создавать, хранить и извлекать сложные структуры данных.

Система управления базами данных PostgreSQL поддерживает обширный список типов данных. Кроме числовых, с плавающей точкой, текстовых, булевых и других типов данных и множества соответствующих вариаций, система управления базами данных PostgreSQL может похвастаться поддержкой денежного, перечисляемого, геометрического и бинарного типов,

сетевых адресов, битовых строк, текстового поиска, XML, JSON, массивов, композитных типов и диапазонов [9]. Тип данных JSON, например, обеспечивает проверку корректности JSON, который позволяет использовать специализированные JSON-операторы и функции, встроенные в систему управления базами данных PostgreSQL для выполнения запросов и манипулирования данными. Система управления базами данных PostgreSQL поддерживает также тип JSONB – двоичную разновидность формата JSON, у которой удаляются пробелы и не сохраняется сортировка объектов, а хранение данных происходит наиболее оптимальным способом: сохраняется только последнее значение для ключей-дубликатов. Формат JSONB обычно является предпочтительным форматом, поскольку требует меньше места для объектов, может быть проиндексирован и обрабатывается быстрее, так как не требует повторного синтаксического анализа.

1.5 Bootstrap Framework

Bootstrap Framework – открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой верстки адаптивных дизайнов сайтов и веб-приложений [10].

Основной областью применения Bootstrap Framework является «front-end» разработка сайтов и панелей администратора. Bootstrap Framework представляет собой набор *.css- и *.js-файлов, после подключения которых становятся доступны инструменты данного фреймворка: колоночная система, классы и компоненты.

Основное отличие верстки с помощью Bootstrap Framework от верстки нативными средствами заключается в том, что данный фреймворк предоставляет возможность использования компонентов. Компоненты представляют из себя часто используемые готовые HTML-блоки с предопределенными стилями. Bootstrap Framework предоставляет возможность использования готового компонента со стандартными стилями, а также возможность определения внешнего вида компонентов с помощью изменения значений соответствующих переменных.

Bootstrap Framework состоит из следующих компонентов:

- Инструменты для создания макета (оберточные контейнеры, система сеток, гибкие медиа-объекты, адаптивные утилитные классы).
- Классы для стилизации базового контента – текста, изображений, кода, таблиц и фигур.

- Готовые компоненты (кнопки, формы, горизонтальные и вертикальные навигационные панели, слайдеры, выпадающие списки, аккордеоны, модальные окна, всплывающие подсказки).
- Утилитные классы для решения традиционных задач наиболее часто возникающими перед веб-разработчиками (выравнивание текста, отображение и скрытие элементов, задание цвета, фона, отступов).

Преимуществами Bootstrap Framework являются:

- Высокая скорость создания качественной адаптивной верстки начинающими веб-разработчиками благодаря использованию готовых классов и компонентов, созданных профессионалами.
- Кроссбраузерность и кроссплатформенность (корректное отображение и работа сайта во всех поддерживаемых данным фреймворком браузерах и операционных системах).
- Наличие большого количество готовых хорошо продуманных компонентов, протестированных на различных устройствах.
- Возможность настройки под пользовательские проекты.
- Однородность и согласованность дизайна между различными компонентами.

Из недостатков Bootstrap Framework можно выделить следующие пункты:

- Сложность использования фреймворка для создания проектов с уникальным дизайном.
- Большой размер *.css- и *.js-файлов проектов, поскольку стили фреймворка содержат универсальный код.

ГЛАВА 2 РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОЙ ЧАСТИ ПРОЕКТА

2.1 Информационная структура

Определение информационной структуры является важным этапом в процессе разработки веб-сайта. Согласно структуре разрабатываемой интернет-витрины магазина интерьерных часов, изображенной на рисунке 2.1, авторизованный пользователь, находясь на главной странице, может перейти на пять основных страниц веб-сайта:

- «Каталог» – страница со всеми товарами магазина интерьерных часов.
- «Карточка товара», где согласно идентификационному номеру представлена подробная информация о соответствующем товаре (название, описание, цена, фотография, категория, механизм, материал, вес, размеры, дополнительная информация).
- «Контакты».
- «Аккаунт», на которой располагается панель администратора для пользователя с соответствующей ролью. Неавторизованным пользователям данная страница недоступна. Если неавторизованный пользователь пожелает перейти на страницу аккаунта, произойдет перенаправление на страницу авторизации.
- «Авторизация» – страница с полем авторизации.

Сама же главная страница разрабатываемого веб-сайта содержит такие информационные блоки, как «Товары», «Акционные товары», «Отзывы», «Новые поступления» и «Акции».

Рисунок 2.1 – Структура интернет-витрины магазина интерьерных часов

2.2 Разработка визуального оформления

Поскольку на всех страницах интернет-витрины магазина интерьерных часов отображаются шапка и подвал (рисунок 2.2 – 2.3 соответственно), принято выносить общие структурные элементы в отдельные шаблоны.

Рисунок 2.2 – Шапка интернет-витрины магазина интерьерных часов

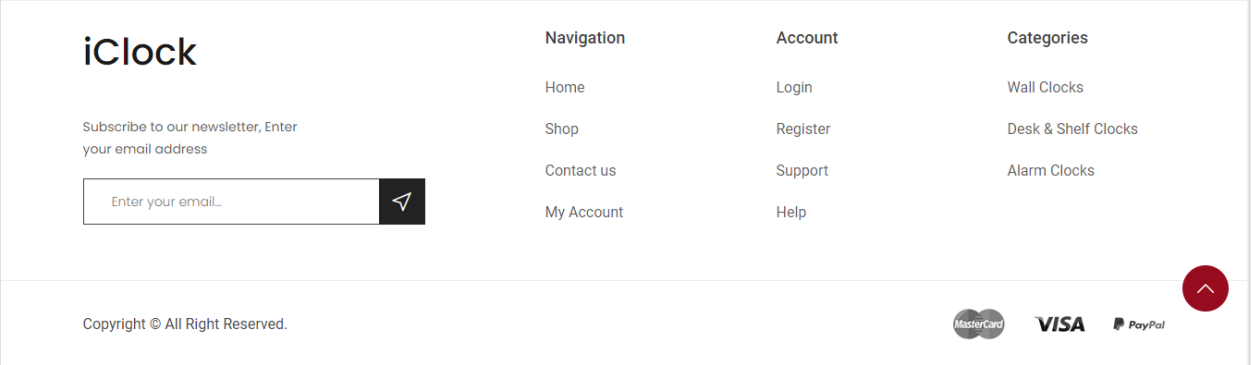

Рисунок 2.3 – Подвал интернет-витрины магазина интерьерных часов

При разработке интернет-витрины магазина интерьерных часов использовался шаблонизатор Thymeleaf – современный серверный механизм шаблонов языка программирования Java как для веб, так и для автономных сред [11]. Шаблонизатор Thymeleaf разработан с учетом стандартов Web, в частности HTML5, что позволяет создавать полностью соответствующие стандарту шаблоны.

Шаблонизатор Thymeleaf предлагает набор интеграций Spring, которые позволяют использовать шаблонизатор как полнофункциональную замену JSP в приложениях Spring MVC. Официальные пакеты интеграции «thymeleaf-spring3» и «thymeleaf-spring4» определяют диалект «SpringStandard Dialect», который в основном совпадает со стандартным диалектом, чтобы лучше использовать некоторые функции Spring Framework [12].

Фрагмент кода подключения шаблона шапки и подвала интернет-витрины магазина интерьерных часов с помощью шаблонизатора Thymeleaf представлен в листинге 2.1.

Листинг 2.1 – Подключение шаблона шапки и подвала интернет-витрины

```
<div th:insert="blocks/header :: header"></div>
<div th:insert="blocks/footer :: footer"></div>
```

На рисунке 2.4 представлен фрагмент главной страницы интернет-витрины магазина интерьерных часов.

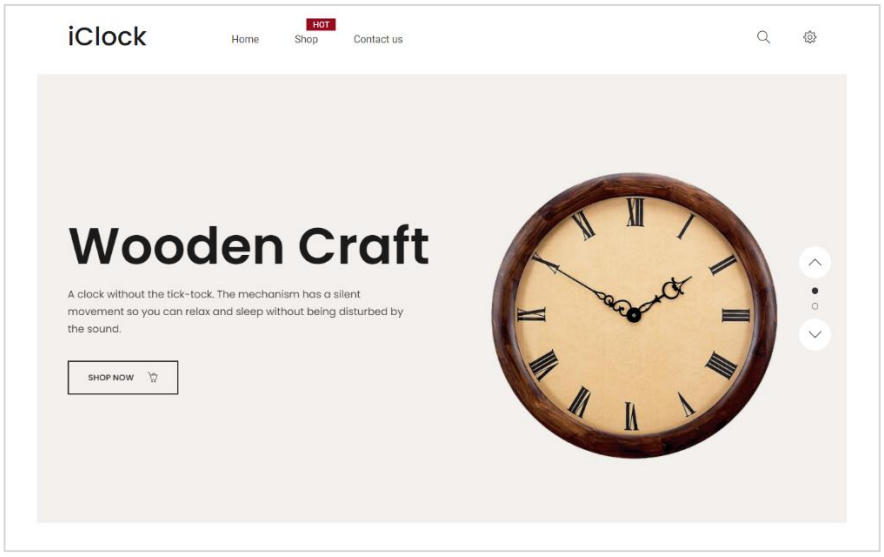

Рисунок 2.4 – Фрагмент главной страницы интернет-витрины магазина интерьерных часов

Поскольку при разработке интернет-витрины использовалась технология Bootstrap Framework, макеты всех веб-страниц являются адаптивными. Фрагменты главной страницы под мобильные и планшетные устройства представлены на рисунках 2.5 – 2.6.

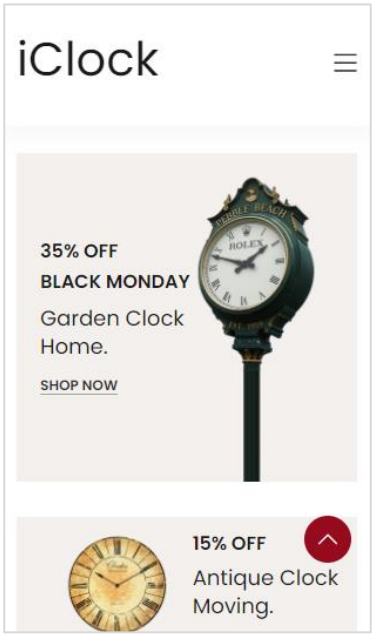

Рисунок 2.5 – Фрагмент главной страницы под мобильные устройства

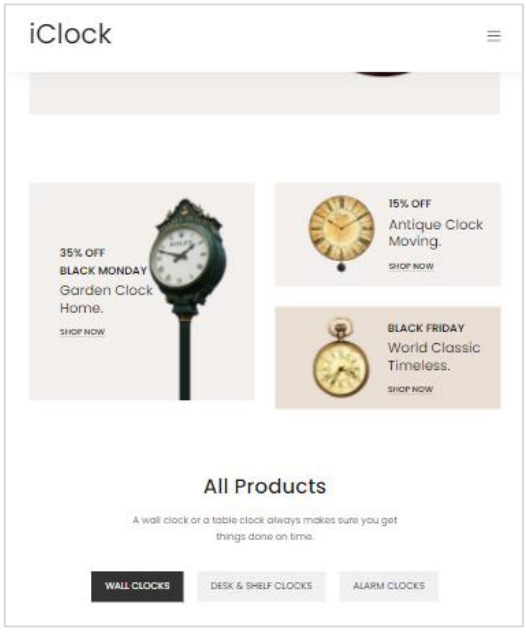

Рисунок 2.6 – Фрагмент главной страницы под планшетные устройства

Следующий информационный блок, визуальное оформление которого представлено на рисунке 2.7, – блок «Товары». Информационный блок «Товары» позволяет выбрать часы соответствующей категории (настенные, настольные и будильники), а также перейти на страницу карточки товара, нажав левой кнопкой мыши по соответствующей фотографии или названию товара.

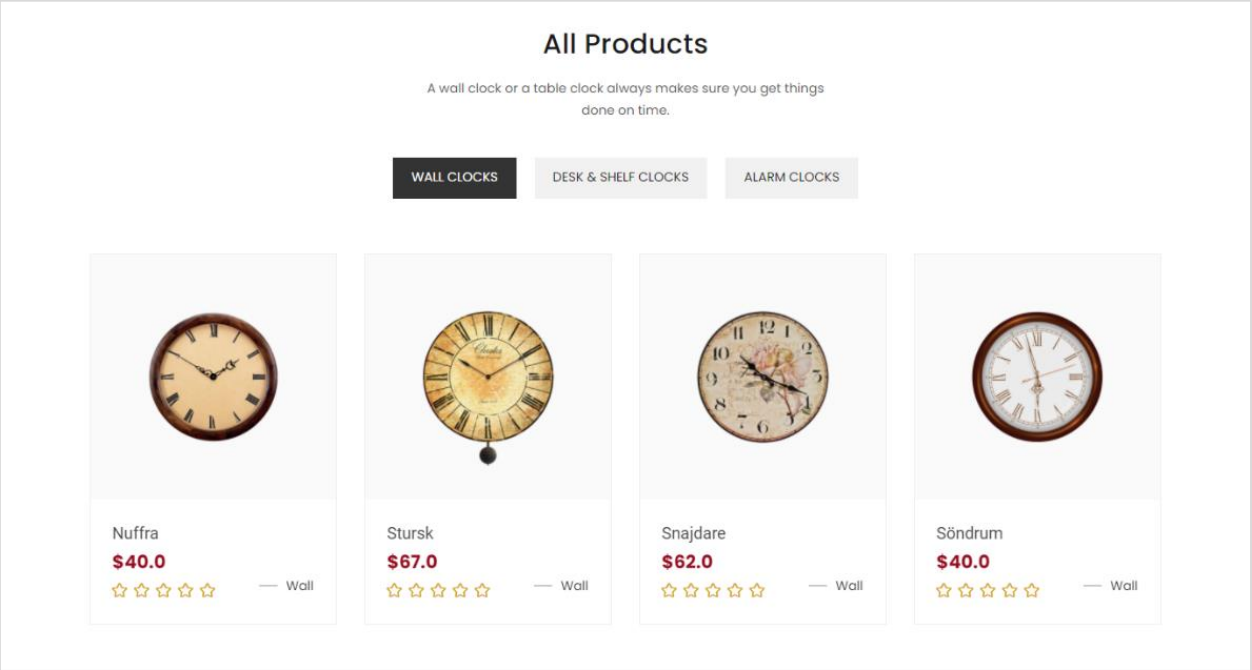

Рисунок 2.7 – Информационный блок «Товары» главной страницы интернет-витрины магазина интерьерных часов

Фрагмент кода вывода часов категории «Настенные» с помощью шаблонизатора Thymeleaf представлен в листинге 2.2.

Листинг 2.2 – Вывод часов категории «Настенные»

```
<div id="product-1" class="tab-pane active">
    <div class="ht-products product-slider-active owl-carousel">
     <!--Product Start-->
     <div th:each="el : ${wallClocks}" class="ht-product ht-product-
action-on-hover ht-product-category-right-bottom mb-30">
     </div>
     <!--Product End-->
    </div>
</div>
```

Все товары пользователь может просмотреть на странице «Каталог», фрагмент которой представлен на рисунке 2.8.

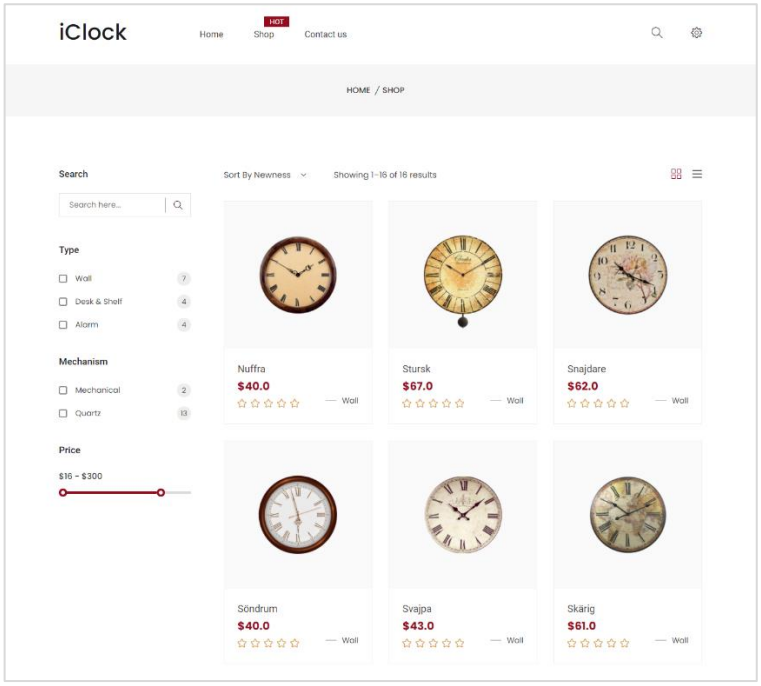

Рисунок 2.6 – Фрагмент страницы «Каталог» интернет-витрины магазина интерьерных часов

По нажатию левой кнопкой мыши по фотографии или названию любого товара, пользователь переходит на страницу «Карточка товара», где указана подробная информация по соответствующему товару: название, описание, цена, фотография, категория, механизм, материал, вес, размеры, дополнительная информация. Фрагмент страницы «Карточка товара» представлен на рисунке 2.7.

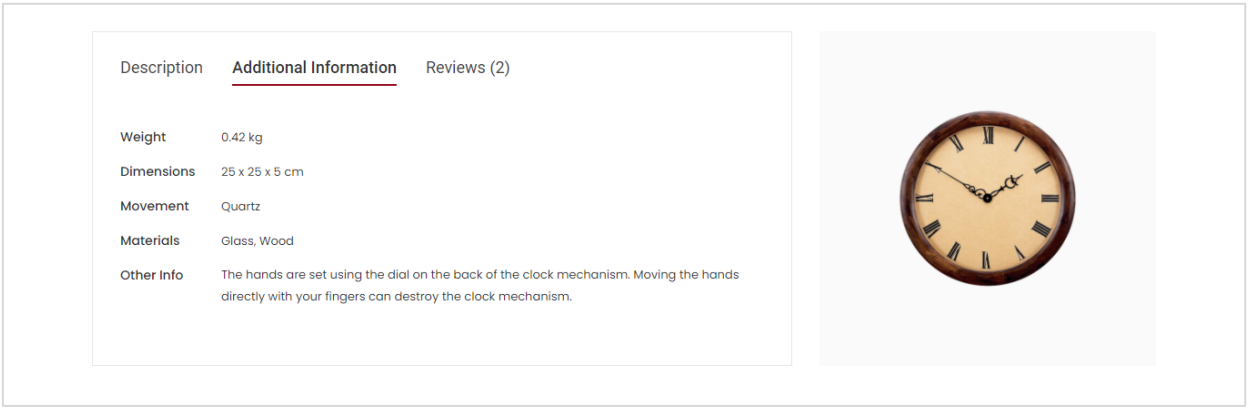

Рисунок 2.7 – Фрагмент страницы «Карточка товара» интернет-витрины магазина интерьерных часов

Визуальное оформление каждой страницы интернет-витрины магазина интерьерных часов представлено в Приложении А.

ГЛАВА 3 РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТА

В процессе разработки серверной части проекта использовались следующие инструментальные средства и технологии: язык программирования Java 11, Spring Framework, Apache Maven Framework, система управления базами данных PostgreSQL.

Администрирование системы управления базами данных PostgreSQL осуществлялось с помощью платформы «pgAdmin».

Разработка программной части проекта велась в интегрированной среде разработки «IntelliJ IDEA» в операционной системе Windows 10.

Отладка проекта в процессе разработки осуществлялась на локальном компьютере.

3.1 Структура проекта

Структура разрабатываемого проекта представлена на рисунках 3.1 – 3.2.

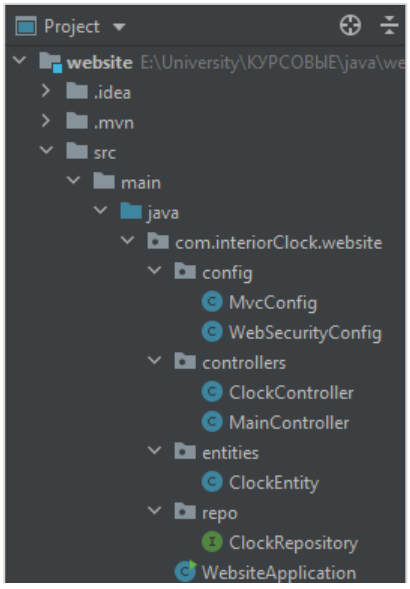

Рисунок 3.1 – Фрагмент №1 структуры проекта

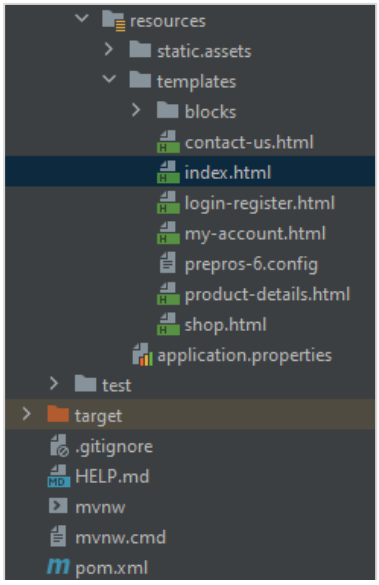

Рисунок 3.2 – Фрагмент №2 структуры проекта

В листинге 3.1 привед код файла «application.properties» проекта, который содержит информацию о подключении к используемой базе данных, имя пользователя, пароль, а также диалект.

Листинг 3.1 – Реализация кода файла «application.properties»

```
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
```

Продолжение листинга 3.1

```
spring.datasource.password=***
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreS
QLDialect
spring.jpa.hibernate.ddl-auto = update
```

База данных `postgres`, используемая в проекте, содержит таблицу `clocks`, в которой хранится информация о всех товарах магазина интерьерных часов. Таблица `clocks` содержит следующие 11 полей: идентификатор `clock_id`, название `title`, категория `category`, описание `description`, цена `price`, механизм `movement`, материал `material`, размеры `dimensions`, вес `weight`, название фотографии `img_ref`, дополнительная информация `other_info`. Фрагмент кода класса сущности `ClockEntity` проекта представлен в листинге 3.2.

Листинг 3.2 – Реализация кода класса сущности `ClockEntity`

```
package com.interiorClock.website.entities;
import javax.persistence.*;
@Entity
@Table(name = "clocks")
public class ClockEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer clockId;
    @Column(name = "title", nullable = false)
    private String title;
    @Column(name = "category", nullable = false)
    private String category;
    @Column(name = "description", nullable = false)
    private String description;
    @Column(name = "price", nullable = false)
    private Double price;
    @Column(name = "movement", nullable = false)
    private String movement;
    @Column(name = "material", nullable = false)
    private String material;
    @Column(name = "dimensions", nullable = false)
    private String dimensions;
    @Column(name = "weight", nullable = false)
    private Double weight;
    @Column(name = "img_ref", nullable = false)
    private String imgRef;
    @Column(name = "other_info", nullable = true)
    private String otherInfo;
```

Продолжение листинга 3.2

```
public ClockEntity() { }
    public ClockEntity(String title, String category, String
description, Double price, String movement, String material, String
dimensions, Double weight, String imgRef, String otherInfo) {
        this.title = title; this.category = category;
this.description = description; this.price = price; this.movement =
movement; this.material = material; this.dimensions = dimensions;
this.weight = weight; this.imgRef = imgRef; this.otherInfo = otherInfo;
    }
    public Integer getClockId() { return clockId; }
    public void setClockId(Integer clockId) {this.clockId = clockId; }
    public String getTitle() {return title; }
    public void setTitle(String title) {this.title = title; }
    public String getCategory() {return category; }
    public void setCategory(String category) {this.category = category;
}
    public String getDescription() {return description; }
    public void setDescription(String description) {this.description =
description;}
    public Double getPrice() {return price; }
    public void setPrice(Double price) {this.price = price; }
    public String getMovement() {return movement;}
    public void setMovement(String movement) {this.movement = movement;
}
    public String getMaterial() {return material; }
    public void setMaterial(String material) {this.material = material;
    }
    public String getDimensions() {return dimensions; }
    public void setDimensions(String dimensions) {this.dimensions =
dimensions; }
    public Double getWeight() {return weight; }
    public void setWeight(Double weight) {this.weight = weight; }
    public String getImgRef() {return imgRef; }
    public void setImgRef(String imgRef) {this.imgRef = imgRef;
    }
    public String getOtherInfo() {return otherInfo; }
    public void setOtherInfo(String otherInfo) {this.otherInfo =
otherInfo; }
}
```

Поскольку в дальнейшем в проекте необходимо реализовать возможность добавления, редактирования и удаления товара администратором, необходимо создать пользовательский интерфейс `ClockRepository`, наследуемый от интерфейса `JpaRepository`, реализующий вышеперечисленные операции. Фрагмент кода пользовательского интерфейса `ClockRepository` представлен в листинге 3.3.

Листинг 3.3 – Реализация кода интерфейса `ClockRepository`

```
package com.interiorClock.website.repo;
import com.interiorClock.website.entities.ClockEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface ClockRepository extends JpaRepository<ClockEntity,
Integer> {
    List<ClockEntity>        findAllByCategory(String        category,
org.springframework.data.domain.Sort sort);
}
```

В интерфейсе `ClockRepository` также определен пользовательский метод `findAllByCategory()`, позволяющий найти все товары в соответствии с указанной категорией.

3.2 Клиент-серверное взаимодействие в стиле REST

Взаимодействие клиента с программным интерфейсом приложения основано на использовании архитектурного стиля Representational State Transfer, или REST. Фактически, REST API является набором веб-адресов, обращаясь к которым с помощью HTTP-запросов, клиент получает в ответ информацию от сервера в формате JSON, XML, HTML.

В программном интерфейсе REST для манипулирования ресурсами используются стандартные HTTP-методы:

- GET – для получения текущего представления ресурсов.
- POST – для создания новых ресурсов.
- PUT – для изменения существующего ресурса.
- DELETE – для удаления существующего ресурса.

Реализация вышеперечисленных операций в отношении товаров магазина интерьерных часов требует создания класса-контроллера `MainController`, фрагмент кода которого представлен в листинге 3.4. Полная реализация класса-контроллера `MainController` представлена в Приложении Б.

Листинг 3.4 – Фрагмент кода класса-контроллера `MainController`

```
< package com.interiorClock.website.controllers;
import com.interiorClock.website.entities.ClockEntity;
import com.interiorClock.website.repo.ClockRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Controller;
```

Продолжение листинга 3.4

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.time.Clock;
import java.util.ArrayList;
import java.util.Optional;

@Controller
public class MainController {

    @Autowired
    ClockRepository clockRepository;

    @GetMapping("/")
    public String Home(Model model) {
        model.addAttribute("pageTitle", "Home");

        Iterable<ClockEntity>               wallClocks               =
clockRepository.findAllByCategory("Wall",   Sort.by(Sort.Direction.ASC,
"clockId"));
        Iterable<ClockEntity>               deskClocks               =
clockRepository.findAllByCategory("Desk",   Sort.by(Sort.Direction.ASC,
"clockId"));
        Iterable<ClockEntity>               alarmClocks              =
clockRepository.findAllByCategory("Alarm",  Sort.by(Sort.Direction.ASC,
"clockId"));

        model.addAttribute("wallClocks", wallClocks);
        model.addAttribute("deskClocks", deskClocks);
        model.addAttribute("alarmClocks", alarmClocks);

        Iterable<ClockEntity>                  clocks                =
clockRepository.findAll(Sort.by(Sort.Direction.DESC, "clockId"));
        ArrayList<ClockEntity> arr = new ArrayList<>();
        Integer i = 7;

        for(ClockEntity clock : clocks) {
           if(i > 0) {
                arr.add(clock); i--;
           }
        }
        model.addAttribute("newClocks", arr);
        return "index";
    }
}
```

3.3 Авторизация пользователей

Поскольку выполнение операций по добавлению, редактированию и удалению товаров предоставляется только администратору сайта, необходимо реализовать авторизацию пользователей на сайте магазина интерьерных часов, а также добавить ограничения на доступ к определенным ссылкам всех пользователей за исключением администратора сайта. Реализация кода вышеперечисленных задач описывается в файле «WebSecurityConfig.java» и представлена в листинге 3.5.

Листинг 3.5 – Реализация кода файла «WebSecurityConfig.java»

```
package com.interiorClock.website.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.config.annotation.web.builders.HttpSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.Enabl
eWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSe
curityConfigurerAdapter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http
                .cors().and().csrf().disable()
                .authorizeRequests()
                    .antMatchers(HttpMethod.GET, "/**/*.css",
"/**/*.js", "/**/*.scss", "/**/*.png", "/**/*.svg", "/**/*.gif",
"/**/*.woff", "/**/*.woff2", "/**/*.ttf").permitAll()
                    .antMatchers("/", "/shop",
"/shop/product/{id}","/contact").permitAll()
```

Продолжение листинга 3.5

```
                    .antMatchers("/api/**",
"/account/admin/**").hasRole("ADMIN")
                    .anyRequest().authenticated()
                .and()
                    .formLogin()
                        .loginPage("/authorization")
                        .permitAll()
                .and()
                    .logout()
                        .permitAll();
    }

    @Bean
    @Override
    public UserDetailsService userDetailsService() {
        UserDetails admin =
                User.withDefaultPasswordEncoder()
                        .username("admin")
                        .password("123")
                        .roles("ADMIN")
                        .build();

        UserDetails user =
                User.withDefaultPasswordEncoder()
                        .username("user")
                        .password("123")
                        .roles("USER")
                        .build();

        return new InMemoryUserDetailsManager(admin, user);
    }
}
```

Таким образом, доступ неавторизованного или не имеющего полномочий пользователя к определенным ресурсам будет запрещен.

ЗАКЛЮЧЕНИЕ

Результатом курсового проекта является адаптивная интернет-витрина магазина интерьерных часов, разработанная с помощью следующих языков программирования и технологий: язык программирования Java, Spring Framework, Spring Security, Apache Maven Framework, система управления базами данных PostgreSQL, Bootstrap Framework.

В ходе выполнения курсового проекта были осуществлены следующие задачи:

1. Проведен обзор следующих инструментальных средств и технологий: язык программирования Java, Spring Framework, Apache Maven Framework, система управления базами данных PostgreSQL, Bootstrap Framework.
2. С помощью языка программирования Java и системы управления базами данных PostgreSQL разработана серверная часть проекта.
3. В соответствии с запросами `GET`, `POST`, `PUT` и `DELETE`, поступающими от пользователя, реализовано отображение всех товаров на сайте, отображение определенного товара, возможность добавления и редактирования товара, удаление товара.
4. Каталог товаров отображается со следующей обязательной информацией: название, категория, цена и фотография товара.
5. Права на добавление, редактирование и удаление товаров предоставлены администратору сайта.
6. Реализация пользовательского интерфейса осуществлена с помощью фреймворка Bootstrap.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Шилдт, Г. Java. Полное руководство / Г. Шилдт. – СПб.: ООО «Альфа-книга», 2018 – 1488 с.

2. Технология Java [Электронный ресурс] – Режим доступа: https://websofter.ru/technologiya-java/ – Дата доступа: 07.11.2020.

3. Аннотация методов, annotation [Электронный ресурс] – Режим доступа: http://java-online.ru/java-annotation.xhtml – Дата доступа: 09.11.2020.

4. Spring Framework [Электронный ресурс] – Режим доступа: https://spring.io/projects/spring-framework – Дата доступа: 15.11.2020.

5. Spring DATA [Электронный ресурс] – Режим доступа: https://javastudy.ru/frameworks/spring/spring-data/ – Дата доступа: 19.11.2020.

6. Spring Security [Электронный ресурс] – Режим доступа: https://spring.io/projects/spring-security – Дата доступа: 24.11.2020.

7. Шеффер, К. Spring 4 для профессионалов / К. Шеффер, К. Хо, Р. Харроп. – Москва: Вильямс, 2017 – 752 с.

8. Фреймворк Apache Maven [Электронный ресурс] – Режим доступа: http://java-online.ru/maven-pom.xhtml – Дата доступа: 29.11.2020.

9. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1 [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/282764/ – Дата доступа: 04.12.2020.

10. Что такое Bootstrap и зачем он нужен? [Электронный ресурс] – Режим доступа: https://itchief.ru/bootstrap/introduction – Дата доступа: 06.12.2020.

11. Thymeleaf [Электронный ресурс] – Режим доступа: https://www.thymeleaf.org/ – Дата доступа: 12.12.2020.

12. Учебник Thymeleaf: Глава 1. Знакомство [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/350864/ – Дата доступа: 12.12.2020.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

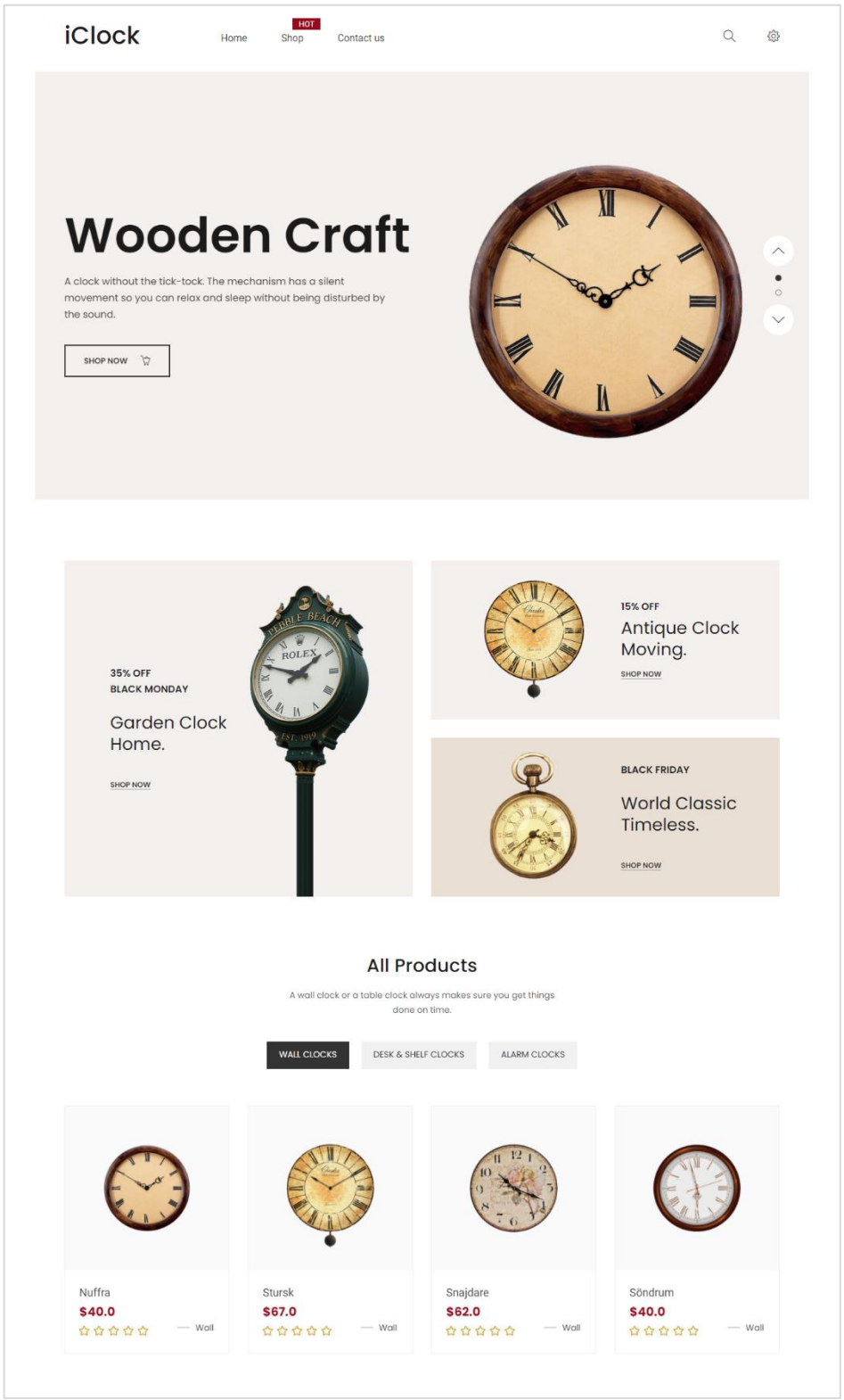

Рисунок А1 – Фрагмент №1 главной страницы веб-сайта

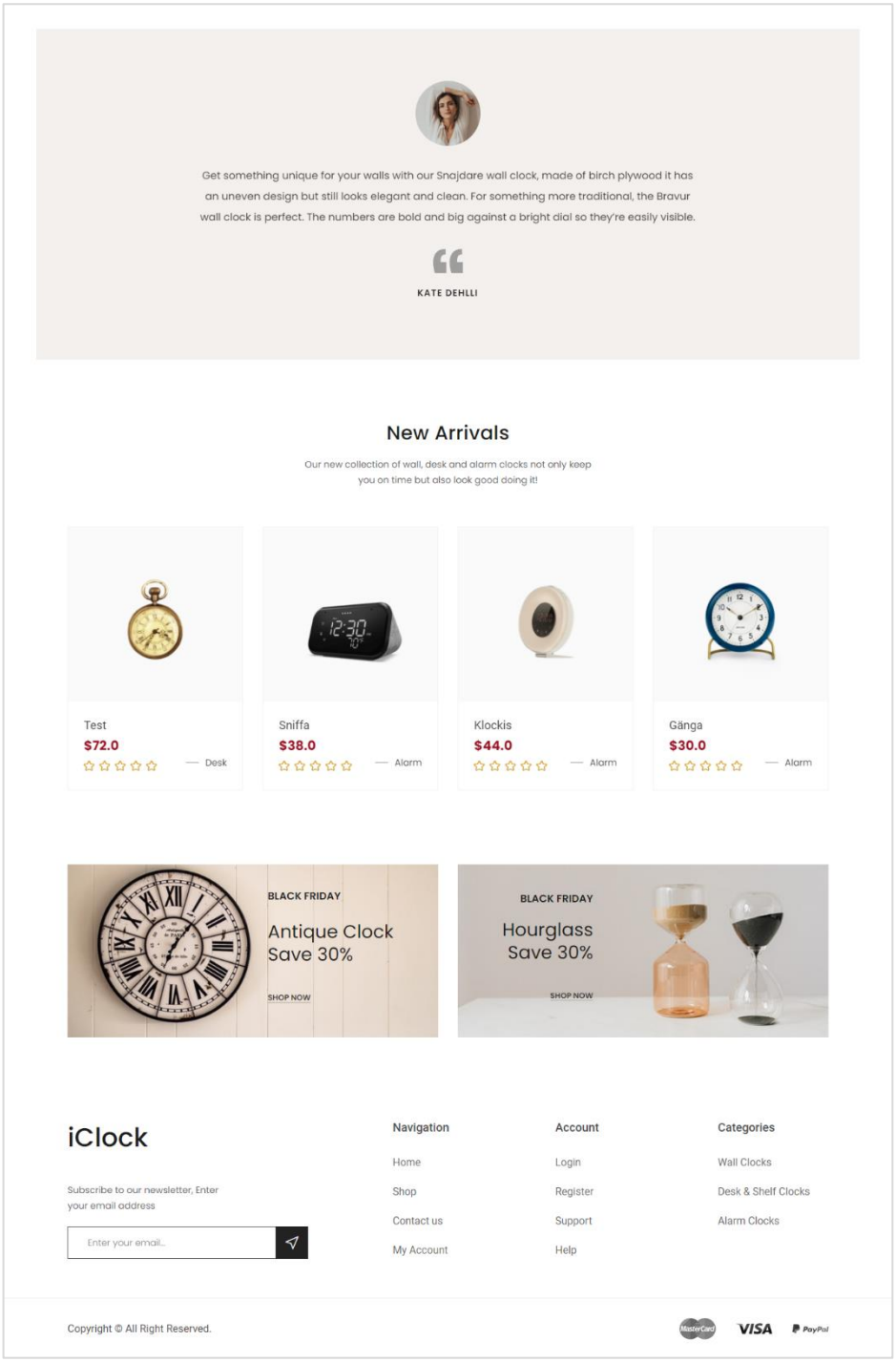

Рисунок А2 – Фрагмент №2 главной страницы веб-сайта

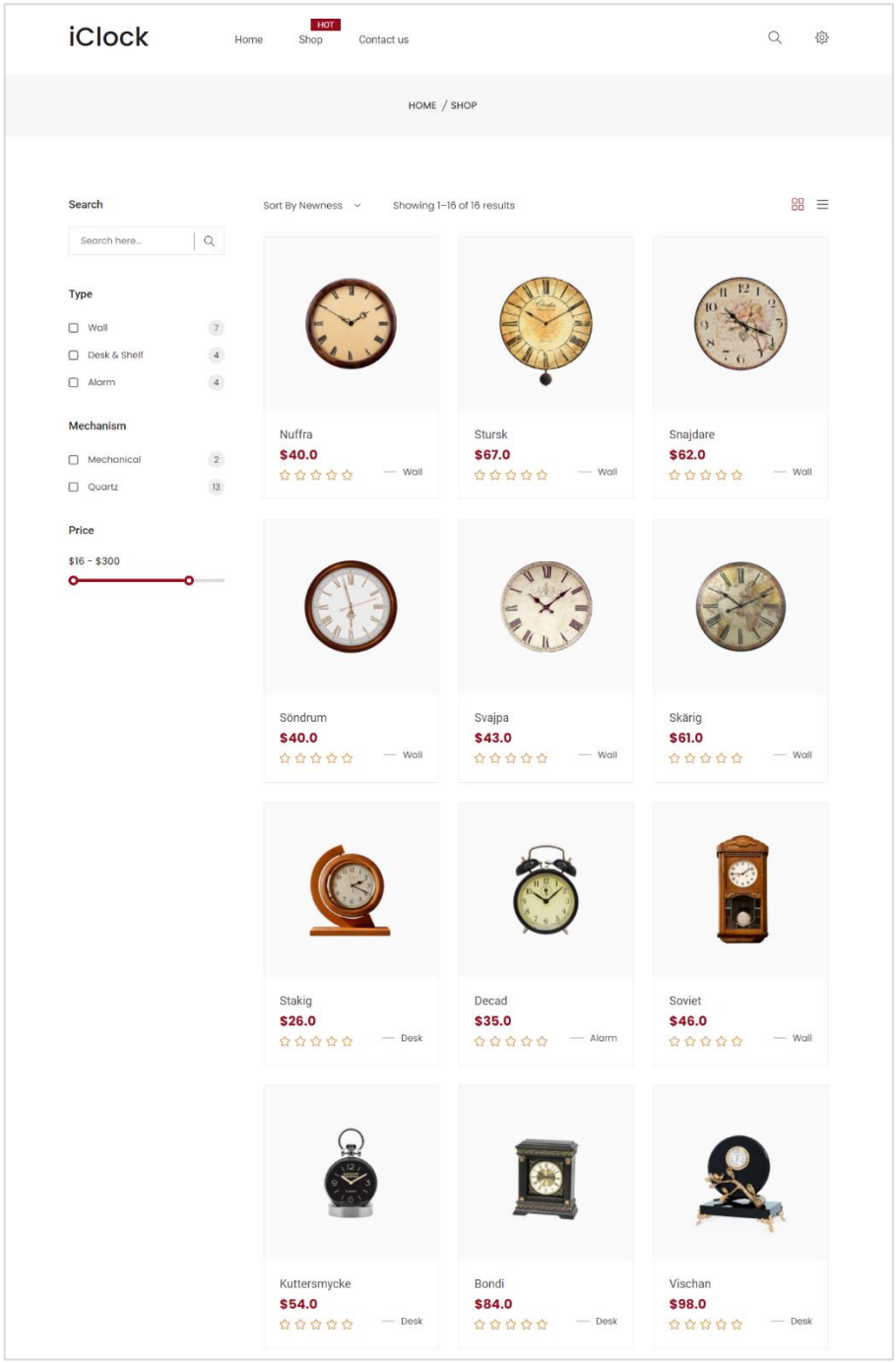

Рисунок А3 – Страница «Каталог» веб-сайта

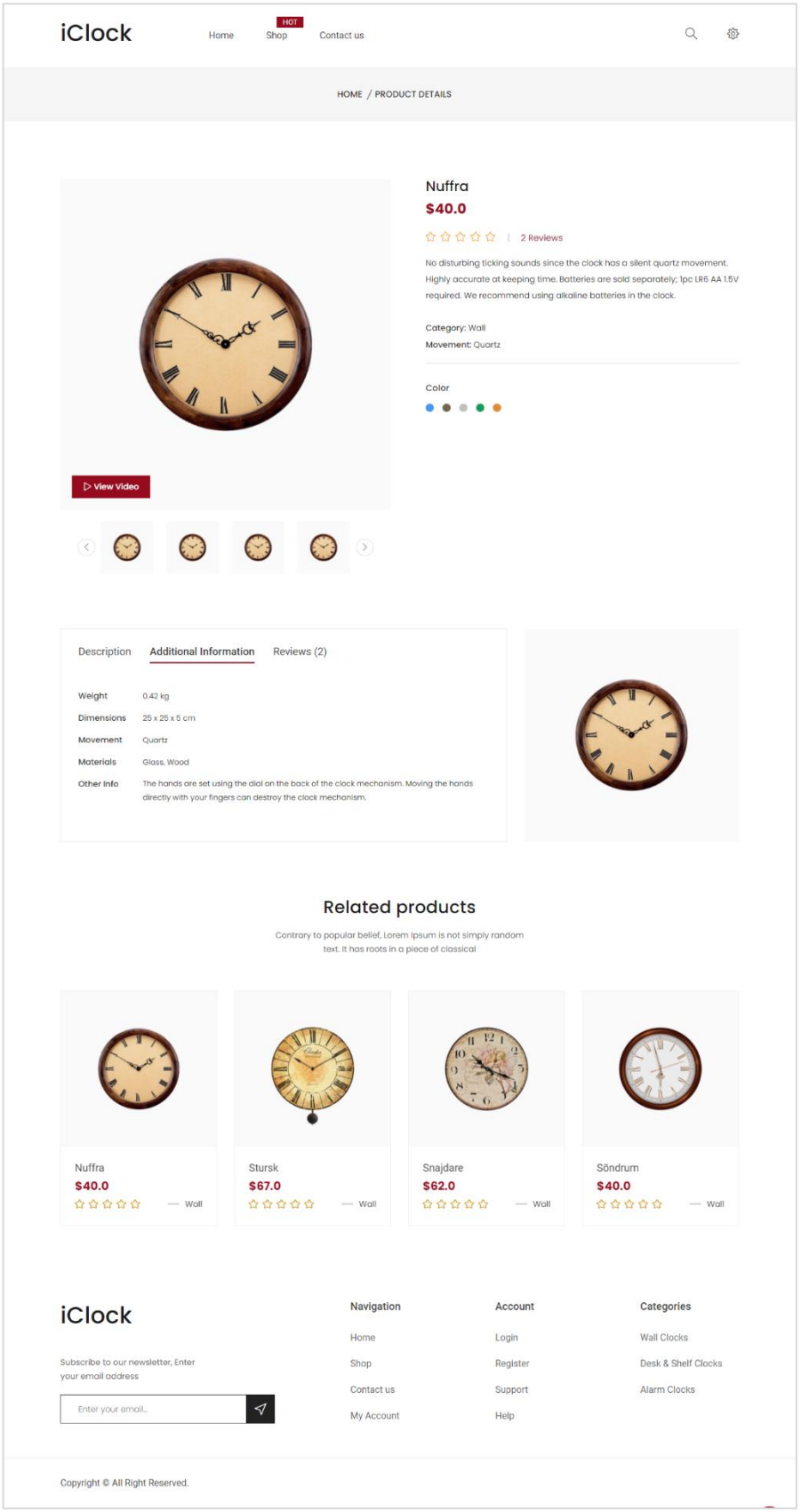

Рисунок А4 – Страница «Карточка товара» веб-сайта

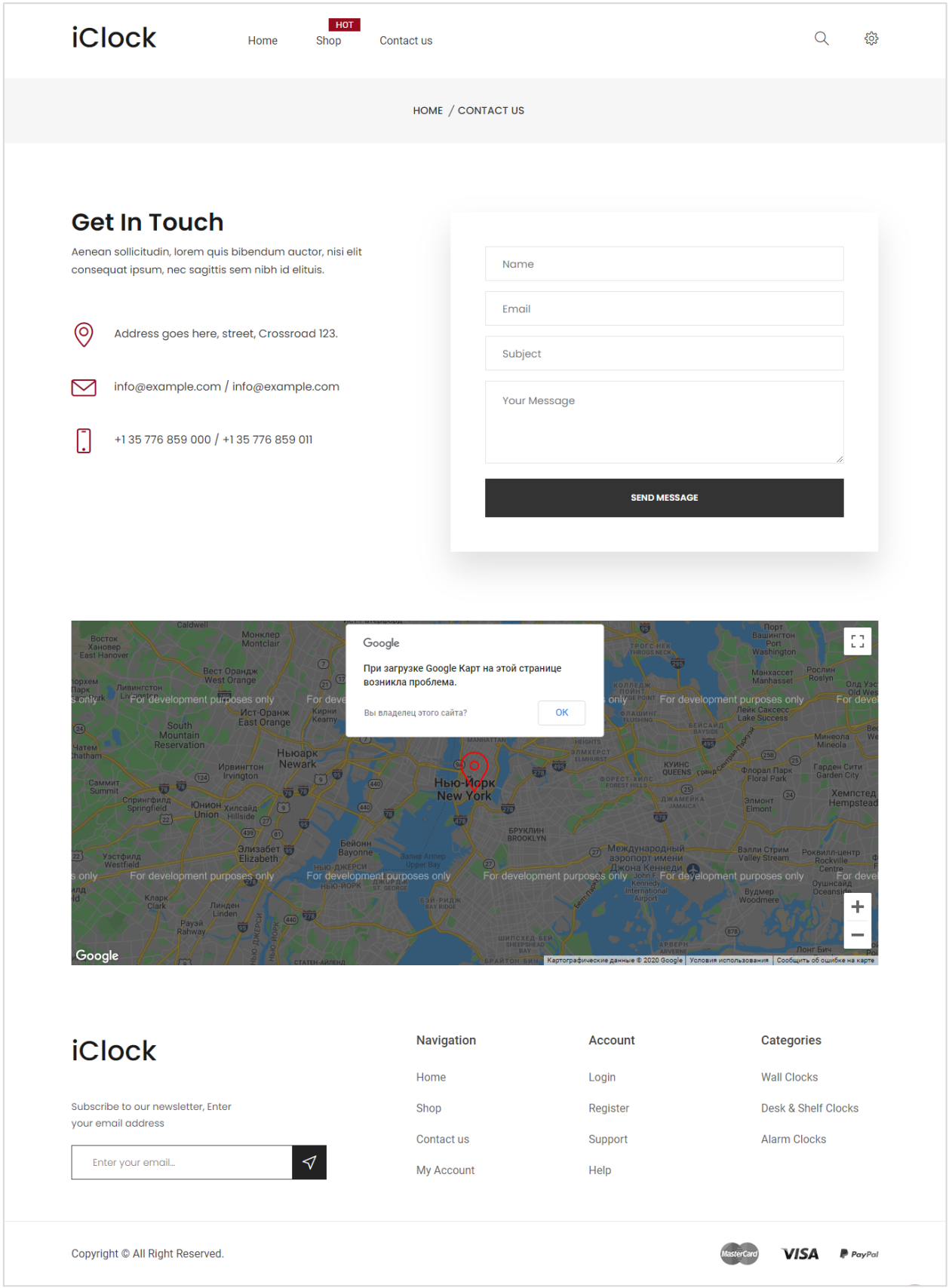

Рисунок А5 – Страница «Контакты» веб-сайта

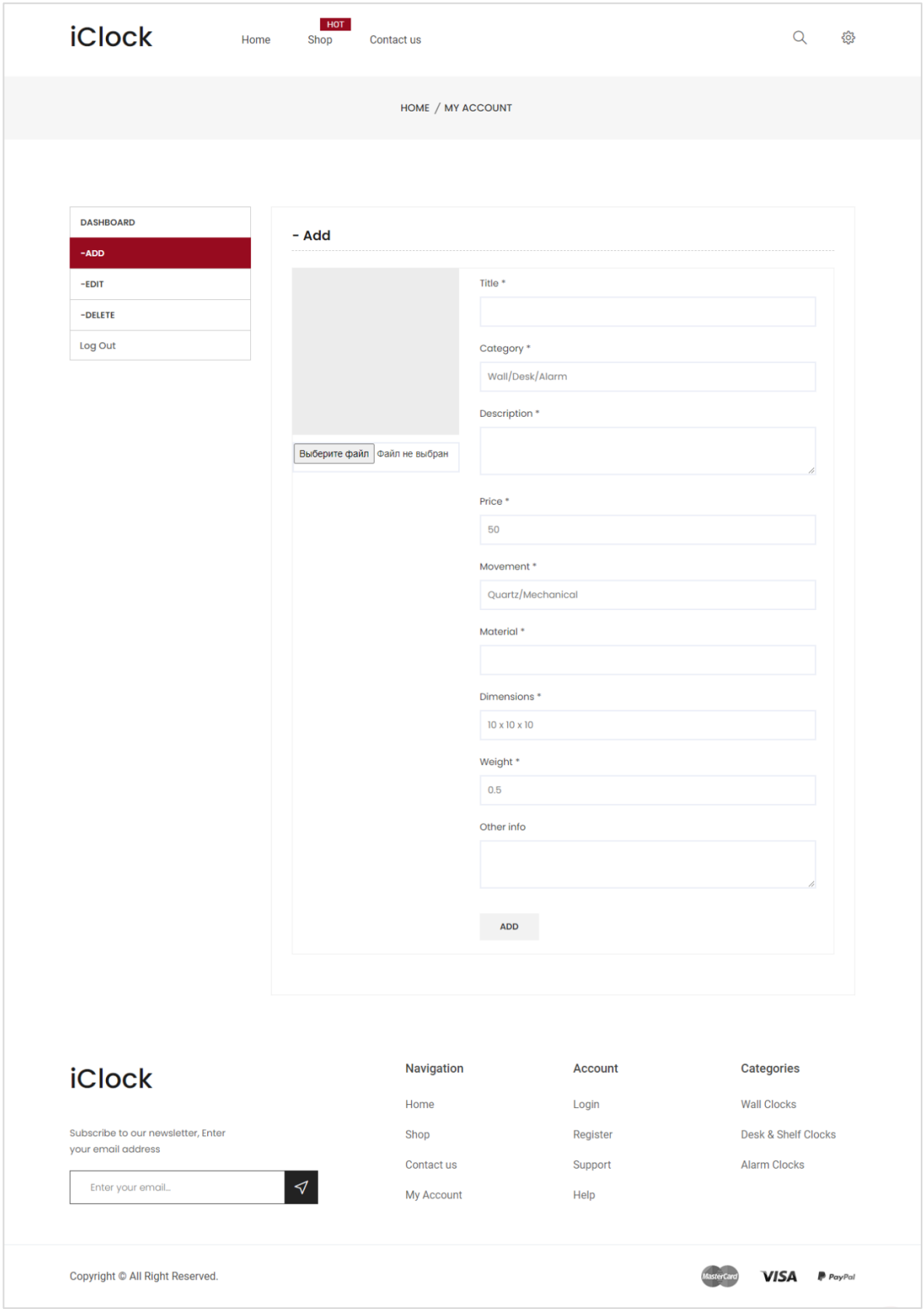

Рисунок А6 – Фрагмент страницы «Аккаунт» веб-сайта

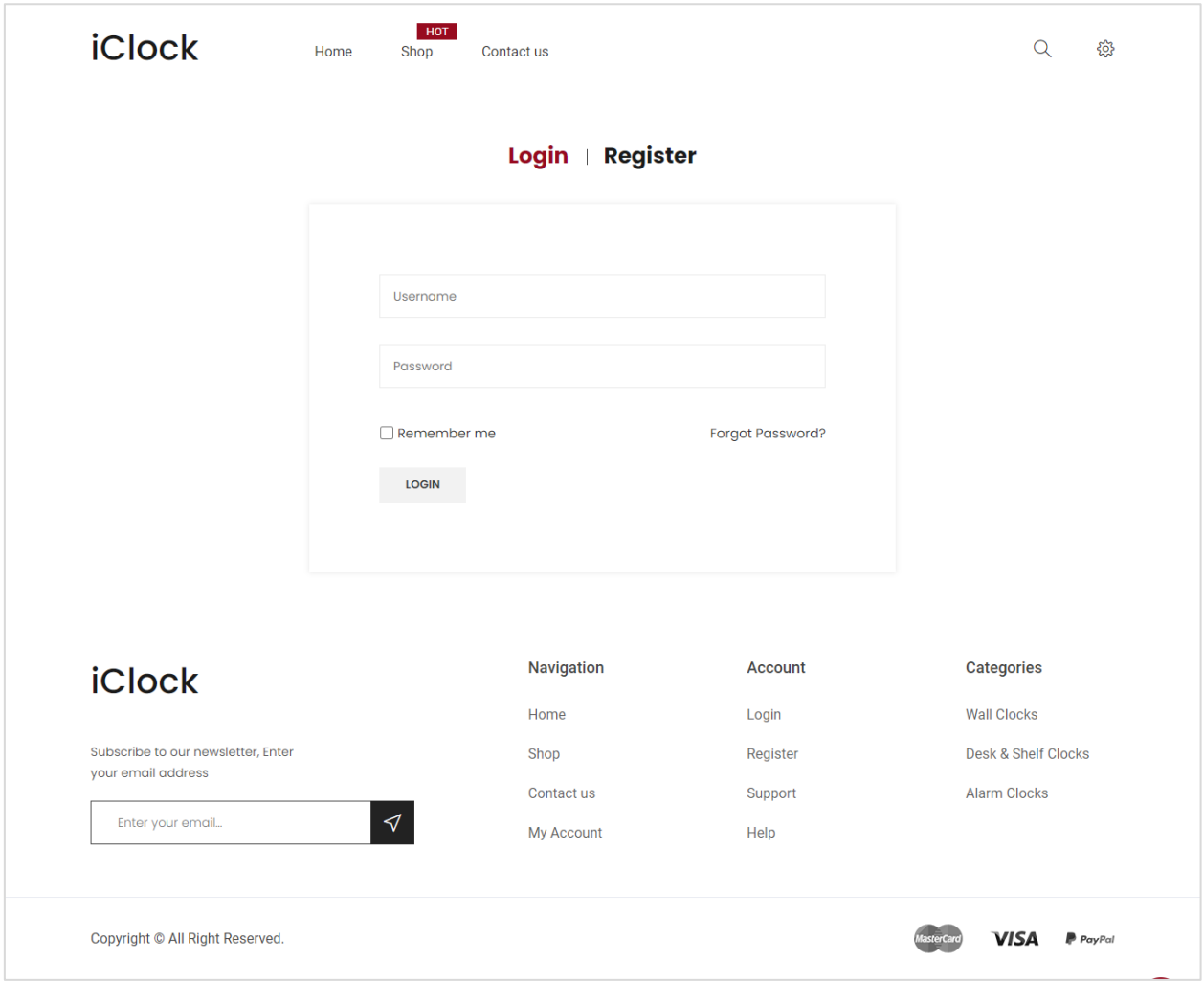

Рисунок А7 – Страница «Авторизация» веб-сайта

Код класса-контроллера `MainController`

```
package com.interiorClock.website.controllers;
import com.interiorClock.website.entities.ClockEntity;
import com.interiorClock.website.repo.ClockRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.time.Clock;
import java.util.ArrayList;
import java.util.Optional;

@Controller
public class MainController {

    @Autowired
    ClockRepository clockRepository;

    @GetMapping("/")
    public String Home(Model model) {
        model.addAttribute("pageTitle", "Home");

        Iterable<ClockEntity> wallClocks =
clockRepository.findAllByCategory("Wall", Sort.by(Sort.Direction.ASC,
"clockId"));
        Iterable<ClockEntity> deskClocks =
clockRepository.findAllByCategory("Desk", Sort.by(Sort.Direction.ASC,
"clockId"));
        Iterable<ClockEntity> alarmClocks =
clockRepository.findAllByCategory("Alarm", Sort.by(Sort.Direction.ASC,
"clockId"));

        model.addAttribute("wallClocks", wallClocks);
        model.addAttribute("deskClocks", deskClocks);
        model.addAttribute("alarmClocks", alarmClocks);

        Iterable<ClockEntity> clocks =
clockRepository.findAll(Sort.by(Sort.Direction.DESC, "clockId"));
        ArrayList<ClockEntity> arr = new ArrayList<>();
        Integer i = 7;

        for(ClockEntity clock : clocks) {
```

```
            if(i > 0) {
                arr.add(clock);
                i--;
            }
        }
        model.addAttribute("newClocks", arr);

        return "index";
    }

    @GetMapping("/shop")
    public String Shop(Model model) {
        model.addAttribute("pageTitle", "Shop");

        Iterable<ClockEntity> clocks =
clockRepository.findAll(Sort.by(Sort.Direction.ASC, "clockId"));
        Long countClocks = clockRepository.count();

        model.addAttribute("clocks", clocks);
        model.addAttribute("countClocks", countClocks);

        return "shop";
    }

    @GetMapping("/shop/product/{id}")
    public String Product(@PathVariable(value = "id") Integer id,
Model model) {
        model.addAttribute("pageTitle", "Product");

        Optional<ClockEntity> clock = clockRepository.findById(id);
        ArrayList<ClockEntity> arr = new ArrayList<>();
        clock.ifPresent(arr::add);
        model.addAttribute("clock", arr);

        String category = clock.get().getCategory();
        Iterable<ClockEntity> relatedClocks =
clockRepository.findAllByCategory(category,
Sort.by(Sort.Direction.ASC, "clockId"));
        model.addAttribute("relatedClocks", relatedClocks);

        return "product-details";
    }

    @GetMapping("/contact")
    public String Contact(Model model) {
        model.addAttribute("pageTitle", "Contact us");
        return "contact-us";
    }
```

```
    @GetMapping("/authorization")
    public String Authorization(Model model) {
        model.addAttribute("pageTitle", "Authorization");
        return "login-register";
    }

    @GetMapping("/account")
    public String Account(Model model) {
        model.addAttribute("pageTitle", "Account");

        Iterable<ClockEntity> clocks =
clockRepository.findAll(Sort.by(Sort.Direction.DESC, "clockId"));
        model.addAttribute("clocks", clocks);

        return "my-account";
    }

    @PostMapping("/account/admin/product/add")
    public String AdminAddClock(@RequestParam String clock_title,
@RequestParam String clock_category,
                                @RequestParam String
clock_description, @RequestParam Double clock_price,
                                @RequestParam String clock_movement,
@RequestParam String clock_material,
                                @RequestParam String
clock_dimensions, @RequestParam Double clock_weight,
                                @RequestParam String clock_img,
@RequestParam String clock_other_info,
                                Model model) {

        ClockEntity newClock = new ClockEntity(clock_title,
clock_category, clock_description,
                clock_price, clock_movement, clock_material,
clock_dimensions, clock_weight, clock_img,
                clock_other_info);

        clockRepository.save(newClock);
        return "redirect:/account";
    }

    @PostMapping("/account/admin/product/{id}/edit")
    public String AdminEditClock(@PathVariable(value = "id") Integer
id, @RequestParam String clock_title,
                                 @RequestParam String
clock_category, @RequestParam String clock_description,
                                 @RequestParam Double clock_price,
@RequestParam String clock_movement,
                                 @RequestParam String
clock_material, @RequestParam String clock_dimensions,
```

```
                                    @RequestParam Double clock_weight,
@RequestParam String clock_img,
                                    @RequestParam String
clock_other_info, Model model) {

        ClockEntity clock =
clockRepository.findById(id).orElseThrow();

        clock.setTitle(clock_title);
        clock.setCategory(clock_category);
        clock.setDescription(clock_description);
        clock.setPrice(clock_price);
        clock.setMovement(clock_movement);
        clock.setMaterial(clock_material);
        clock.setDimensions(clock_dimensions);
        clock.setWeight(clock_weight);
        clock.setImgRef(clock_img);
        clock.setOtherInfo(clock_other_info);

        clockRepository.save(clock);
        return "redirect:/account";
    }

    @PostMapping("/account/admin/product/{id}/delete")
    public String AdminDeleteClock(@PathVariable(value = "id") Integer
id, Model model) {

        ClockEntity clock =
clockRepository.findById(id).orElseThrow();
        clockRepository.delete(clock);
        return "redirect:/account";
    }
}
```