[Введение 1](#_Toc1036506672)

[1. Обзор программных средств. 3](#_Toc424889277)

[1.1. Выбор языков программирования. 3](#_Toc1187073811)

[1.2. Выбор инструментов реализации. 5](#_Toc1035858396)

[2. Практическая реализация целей и задач. 5](#_Toc1373938736)

[2.1. Планирование приложения. 5](#_Toc351867995)

[2.2. Дизайн приложения. 6](#_Toc1722980444)

[2.3. Работа с базой данных. 7](#_Toc1369926315)

[2.4. Работа над структурой приложения. 8](#_Toc75613650)

[2.5. Реализация классов. 10](#_Toc1587494095)

[2.6. Тестирование. 18](#_Toc1236413473)

[Заключение 23](#_Toc633680021)

[Список литературы 24](#_Toc447174471)

# Введение

В современном цифровом мире технологии играют все более значимую роль в различных аспектах нашей повседневной жизни. Одной из сфер, которая ощущает сильное влияние современных технологий, является искусство. Ранее выбор, продажа и приобретение произведений искусства требовали физического присутствия в галереях, аукционах или узкоспециализированных магазинах. Однако с появлением веб-приложений появилась возможность создать платформу, которая позволит людям из разных уголков мира наслаждаться и покупать произведения искусства онлайн.

Цель данной курсовой работы заключается в разработке MVC веб-приложения для помощи в продаже предметов искусства. Оно позволит любителям искусства исследовать, выбирать и приобретать различные произведения. Приложение будет предоставлять широкий спектр произведений искусства, включая картины, скульптуры, фотографии, антиквариат и другие формы искусства. Приложение предоставит удобный и надежный инструмент для коллекционеров, позволяя им исследовать рынок антиквариата, находить ценные предметы, связываться с другими коллекционерами, а также продавать со, а также продавать собственные предметы искусства.

Приложение станет безопасной средой для обмена информацией между ценителями прекрасного.

Актуальность разработки этого веб-сервиса заключается в нескольких ключевых аспектах:

1. Растущий интерес к антиквариату: сегодня все больше людей проявляют интерес к коллекционированию искусства. Развитие цифровых технологий и доступ к Интернету позволяют коллекционерам находить и приобретать предметы антиквариата в онлайн-режиме. Разработка веб-сервиса для их продажи позволит соединить спрос и предложение в одной удобной и доступной платформе.
2. Расширение рынка: традиционно продажа и покупка требовали присутствия на аукционах, ярмарках или специализированных магазинах. Разработка веб-сервиса позволит преодолеть географические и временные ограничения, предоставляя возможность коллекционерам со всего мира находить и продавать предметы искусства без необходимости физического присутствия.
3. Удобство и эффективность: веб-сервис предоставит коллекционерам удобный способ искать, оценивать и приобретать интересующие их экспонаты. Они смогут легко находить интересующие их предметы, получать детальную информацию о них, а также взаимодействовать с продавцами. Это позволит сократить время и усилия, затрачиваемые на поиск и покупку.
4. Сохранение и продвижение культурного наследия: разработка веб-сервиса, который собирает и предлагает антикварные предметы, способствует сохранению и продвижению культурных ценностей, что, что всегда является актуальным вопросом в обществе.
5. Доступность и разнообразие: веб-сервисы для продажи предметов антиквариата, предметов искусства делают эту сферу доступной для широкой аудитории. Коллекционеры со всех уголков мира могут искать и приобретать предметы, которые ранее были недоступны из-за географических ограничений. Кроме того, веб-сервисы предлагают разнообразие экспонатов, позволяя коллекционерам выбирать из большого количества категорий, эпох и стилей.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Выбор программных инструментов, с помощью которых будет реализована цель проекта, и их изучение.
2. Выбор СУБД для работы.
3. Проработать дизайн приложения.
4. Разработать прототип приложения на основе паттерна MVC.

# 1. Обзор программных средств.

## 1.1. Выбор языков программирования.

Для выполнения задач, поставленных в рамках этого курсового проекта, будет использован язык программирования Java. А также Spring Framework.

Фреймворк Spring предоставляет разработчикам инструменты для создания сложных систем, например многопользовательских корпоративных веб-приложений со множеством функций для бизнеса. Он позволяет быстро создавать приложения, которые умеют работать с базами данных и облаками, состоят из разных модулей, обмениваются данными с пользователями через интернет по защищённым каналам. Теоретически всё это можно реализовать в Java и вручную, но Spring даёт разработчикам уже готовые инструменты, которые позволяют писать код гораздо быстрее и концентрироваться не на формальностях, а на уникальных функциях программы.

Spring Framework представляет собой совокупность множество небольших фреймворков, то есть состоит из нескольких модулей, каждый из которых предназначен для решения какой-либо задачи.

Дадим краткую характеристику некоторым модулям Spring:

* Spring Core – ядро платформы, предоставляет базовые средства для создания приложений — управление компонентами (бинами, beans), внедрение зависимостей, MVC фреймворк, транзакции, базовый доступ к БД. В основном это низкоуровневые компоненты и абстракции. По сути, неявно используется всеми другими компонентами;
* Spring MVC – обеспечивает архитектуру паттерна Model-View-Controller при помощи слабо связанных готовых компонентов для разработки веб-приложений;
* Spring Data – обеспечивает доступ к данным: реляционные и не реляционные БД, KV хранилища и т. п.;
* Spring Cloud – используется для микро-сервисной архитектуры;
* Spring Security – авторизация и аутентификация, доступ к данным, методам и т. п.

Подробнее о модуле Spring MVC, с помощью которого будет реализована большая часть веб-приложения:

Фреймворк Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение— Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними.

* Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).
* View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.
* Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Также особенно важным для нас будет модуль Spring Security. Spring Security — это среда, которая позволяет программисту налагать ограничения безопасности на веб-приложения. Кроме того, это библиотека, которую можно использовать и расширять для настройки в соответствии с потребностями программиста. Самым большим преимуществом этой платформы является то, что она является мощной, но при этом легко настраиваемой в реализации. Хотя он следует соглашению Spring по настройке, программисты могут выбирать между условиями по умолчанию или настраивать их в соответствии со своими потребностями. Spring Security работает в двух основных областях: аутентификация и авторизация.

## 1.2. Выбор инструментов реализации.

Так как наше веб-приложения должно быть связано с базой данных, необходимо выбрать СУБД.

Для данного курсового проекта будет выбрана СУБД — MySQL. MySQL — это реляционная база данных (СУБД). Реляционная — значит внутри неё есть данные, которые связаны между собой, и эту связь можно представить в виде таблиц.

Преимуществами MySQL является кроссплатформенность, безопасность, производительность, развитое сообщество, которое не оставит пользователя одного с проблемой. Кроме того, данная СУБД является бесплатной.

Разработка веб-части будет проводиться в среде разработки IntelliJ IDEA.

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java. IntelliJ IDEA продумана в каждом аспекте и готова к использованию сразу после установки. Среда обеспечивает быстрый доступ ко всем функциям и встроенным инструментам, необходимым разработчику, а также широкие возможности индивидуальной настройки.

# 2. Практическая реализация целей и задач.

## 2.1. Планирование приложения.

Программа будет содержать интуитивно понятный веб-интерфейс. Функциональный возможности программы:

* Возможность регистрации и авторизации через email и пароль.
* Поиск объявлений.
* Разделение прав доступа (администратор, гость, пользователь/user).
* Добавление объявлений.
* Возможность их удаления.
* Загрузка изображений к объявлениям.
* Поиск.
* Для администратора существует возможность редактирования пользователей, то есть изменения ролей, а также функция “Бан”).

Приложение должно иметь:

* Страница регистрации и авторизации пользователей.
* Основная страница для просмотра объявлений с кнопкой их поиска, а также с кнопкой входа в аккаунт. Эта страница доступна для всех ролей доступа.
* Страница для просмотра одного объявления с кнопкой перехода на профиль автора.
* Страница профиля пользователя с кнопкой выхода из учетной записи, а также с кнопкой просмотра собственных объявлений.
* Страница администратора с функциями бана и редактирования ролей.
* Страница создания нового объявления пользователя с полями для ввода информации, а также с возможностью добавлять фотографии.

## 2.2. Дизайн приложения.

Дизайн веб-приложения должен быть сделан в минималистичном стиле, чтобы он не отпугивал пользователей своей сложностью и в нем можно было легко разобраться любому, кто будет работать с данной программой. Общая цветовая гамма будет состоять из двух цветов: черный и желтый.

В рамках разработки веб-приложения мы будем использовать современные технологии и инструменты, такие как HTML, CSS и JavaScript, для создания интуитивно понятного пользовательского интерфейса.

Страницы будут сделаны с помощью языка разметки с добавлением механизма шаблонов. В нашем случае будет использован шаблонизатор FreeMarker. Apache FreeMarker — это механизм шаблонов: библиотека Java для генерации текстового вывода (HTML-страницы, xml, файлы конфигурации, исходный код и т. д. На вход подается шаблон, например html в котором есть специальные выражения, подготавливаются данные, соответствующие этим выражением, а Freemarker динамически вставляет эти данные из модели и получается динамически заполненный документ.

Для оформления UI веб-приложения будет использован Bootstrap. Bootstrap — это бесплатный CSS-фреймворк с открытым исходным кодом, предназначенный для быстрой вёрстки адаптивных интерфейсов сайтов и веб-приложений с ориентацией на мобильные устройства. Bootstrap используется многими веб-разработчиками по всему миру. Основная область его применения – это фронтенд разработка сайтов и интерфейсов. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным. С помощью предоставляемых шаблонов мы сможем быстро спроектировать пользовательский интерфейс нашему веб-приложению.

Каждая страница приложения имеет панель навигации и тело. На панели находится кнопка для входа в аккаунт и регистрации пользователя. В теле сайта расположена вся необходимая информация.

## 2.3. Работа с базой данных.

Важным шагом является установка и подключения СУБД.

Мы должны установить MySQL Workbench. Данная программа является инструментом для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

Следующим шагом будет создания схемы.

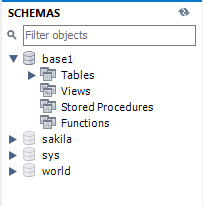


Рис. 1 - Создание схемы в MySQL Workbench.

В файле application.properties укажем подключение к БД. Данный файл является настроечным файлом, где прописываются некоторые механизмы работы.

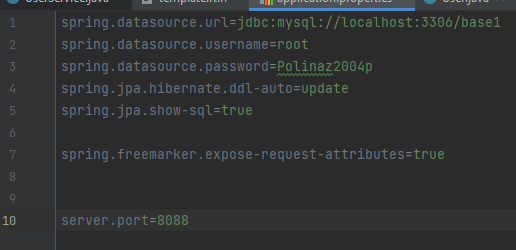


Рис. 2 - Файл application.properties.

## 2.4. Работа над структурой приложения.

Работа над веб-частью приложения начинается с проектирования его структуры.

Основные пакеты:

* сonfigmvc - Конфигурация MVC и безопасности;
* сontrollers - Контроллеры;
* models - Классы-модели;
* repositories - Интерфейсы для работы с классами;
* services - классы, которые отвечают за бизнес-логику;
* templates - представления страниц (ftlh);

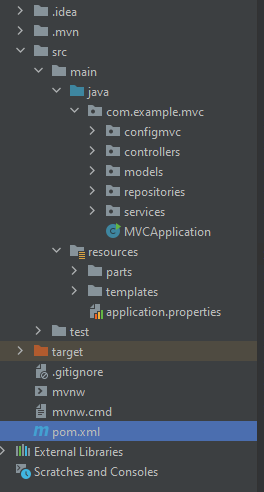
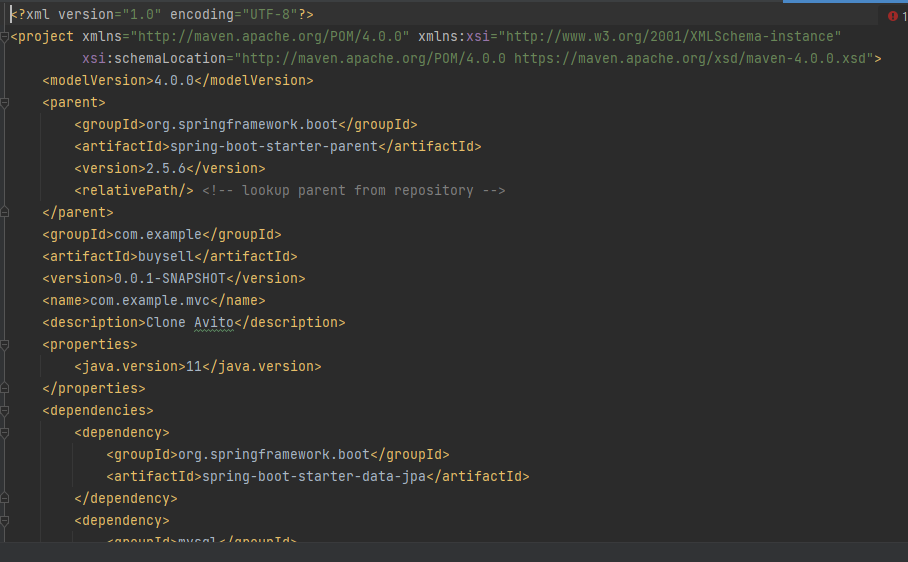


Рис. 3 - Структура приложения.

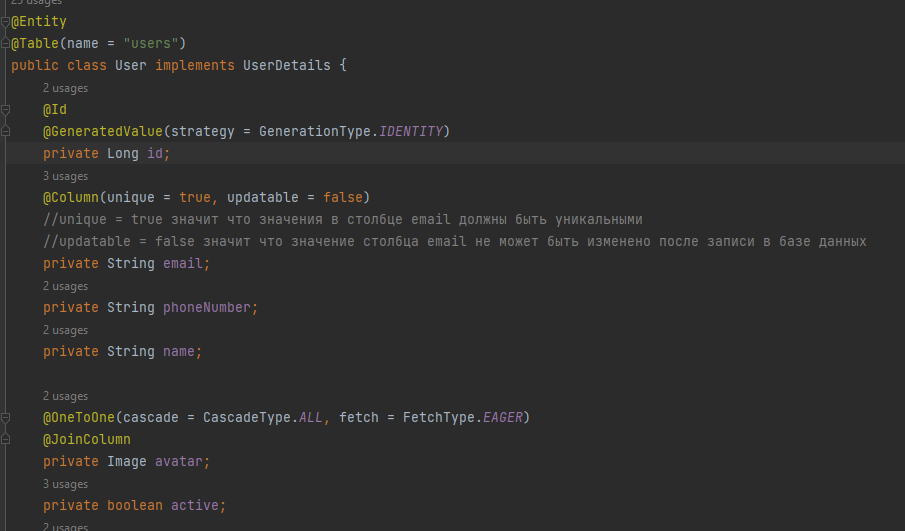
Файл pom.xml содержит информацию об используемых библиотеках, зависимостях. Основные maven зависимости:

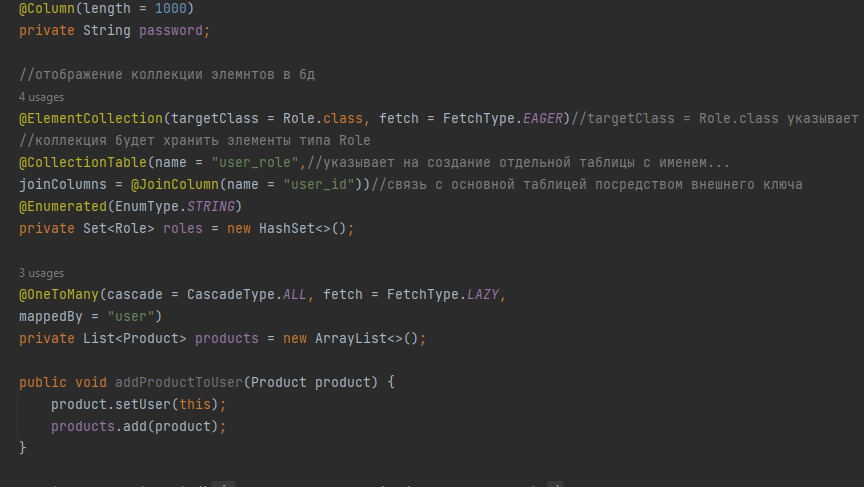
* spring-boot-starter-data-jpa;
* mysql-connector-java;
* spring-boot-starter-freemarker;
* spring-boot-starter-security;
* spring-boot-starter-web;
* Lombok;

Рис. 4 – pom.xml

## 2.5. Реализация классов.

Необходимо создать модель данных, которые будут использоваться для авторизации пользователя и его роли в приложении.

Рис. 5 - Создание модели для пользователя

Рис. 6 - Продолжение класса User

**@Entity**- указывает, что класс представляет сущность, которая будет сопоставлена с таблицей в базе данных

Аннотация **@Table**- показывает, к какой таблице в базе данных принадлежит объект, задает имя таблицы в базе, с которой будет связан данный класс.

**@Id**- аннотация для уникального значения для объекта класса.

**@GeneratedValue**- указывает способ генерации значений id при сохранении в базу данных. strategy=GenerationType.**IDENTITY** позволяет автоматически увеличивать столбец значения ключа (автоинкремент).

Необходимо определить репозиторий, через который объекты будут храниться в базе данных.

Аннотация **@OneToOne** означает, что у текущей сущности (класса, содержащего данный код) есть связь "один-к-одному" с другой сущностью (классом).

* **cascade = CascadeType.ALL** указывает, что все операции (вставка, обновление, удаление) должны быть применены к связанной сущности. Это означает, что если происходит операция на текущей сущности, такая как вставка или удаление, то эта операция будет автоматически распространяться на связанную сущность.
* **fetch = FetchType.EAGER** указывает, что связанная сущность должна быть немедленно загружена (извлечена) при извлечении текущей сущности. Это означает, что при запросе или доступе к текущей сущности, связанная сущность будет загружена из базы данных вместе с ней.

Аннотация **@JoinColumn** указывает, что связь будет установлена через столбец в базе данных, который будет использоваться в качестве внешнего ключа.

Аннотация **@Column** используется в JPA (Java Persistence API) для указания отображения поля класса в столбец базы данных.

Аннотация **@CollectionTable** используется в JPA (Java Persistence API) для указания дополнительных настроек таблицы, которая хранит элементы коллекции. В данном случае, name = "user\_role" задает имя таблицы, в которой будут храниться роли пользователей, а joinColumns указывает, что связь с основной таблицей будет осуществляться посредством внешнего ключа user\_id.

Аннотация **@ElementCollection** указывает, что поле является коллекцией значений или встроенных типов, которые должны быть сохранены в отдельной таблице в базе данных.

* В данном случае аннотация **@ElementCollection** указывает, что поле roles является коллекцией значений типа Role, которые будут сохранены в отдельной таблице базы данных.
* **fetch = FetchType.EAGER** указывает, что коллекция roles должна быть немедленно загружена (извлечена) при извлечении родительской сущности.

Аннотация **@Enumerated** используется для указания способа хранения перечислений в базе данных. В данном случае перечисление будет храниться в виде строки (имени перечисления) в базе данных.

С помощью аннотации **@OneToMany** мы указали, что User может хранить у себя много объектов products.

Определяем репозиторий, через который объекты будут храниться базе данных. Поскольку мы используем Spring Data JPA, эта задача состоит в том, чтобы просто создать интерфейс, который расширяет интерфейс JpaRepository. JpaRepository **–** это интерфейс фреймворка Spring Data предоставляющий набор стандартных методов JPA для работы с БД.

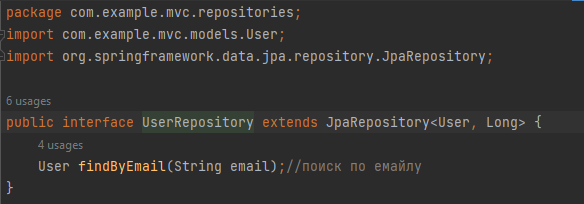
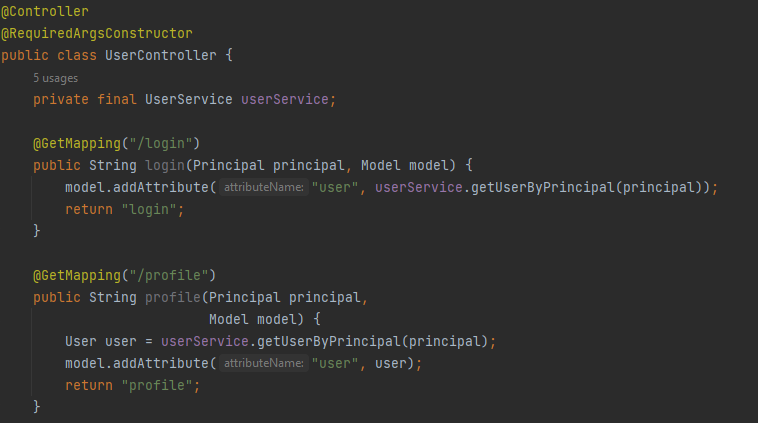


Рис. 7 - Интерфейс для работы с базой данных

Затес нужно создать контроллеры, которые будут обрабатывать веб-запросы от пользователей. Объект контроллера интерпретирует действия и намерения пользователя, выполняемые в объектах просмотра, например, когда пользователь нажимает кнопку или вводит текст в текстовое поле, и передает новые или измененные данные в объекты. Контроллеры являются своеобразным каналом, по которому объекты просмотра узнают об изменениях в объектах модели, и наоборот.

Рис. 8 - Контроллер для реализации функционала регистрации

Классы контроллеров в Spring аннотируются аннотацией **@Controller**.Она помечает классы контроллеров как обработчики запросов, чтобы Spring мог распознать их как службу RESTful во время выполнения.RESTful API — это прикладной программный интерфейс (API), который использует HTTP-запросы для получения, извлечения, размещения и удаления данных.

Рис. 9 - Методы контроллера

Аннотация @RequestMapping используется для отображения запросов на методы контроллеров.

Существуют также специфические для HTTP-метода варианты сокращения аннотации @RequestMapping:

* @GetMapping - Обрабатывает get-запросы;
* @PostMapping - Обрабатывает post-запросы;
* @PutMapping и т. д.

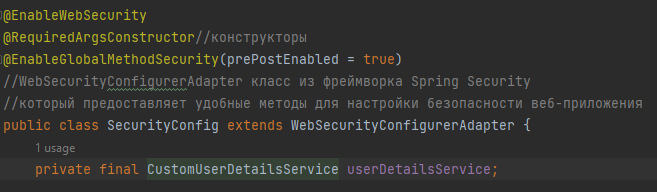
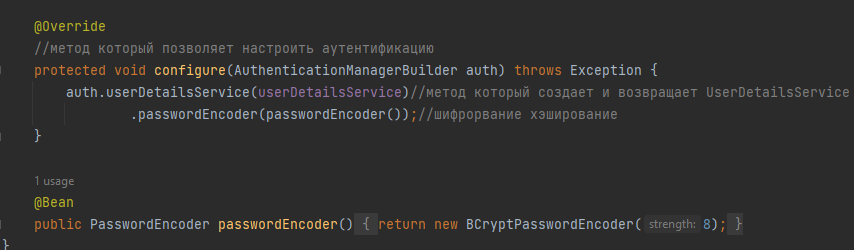
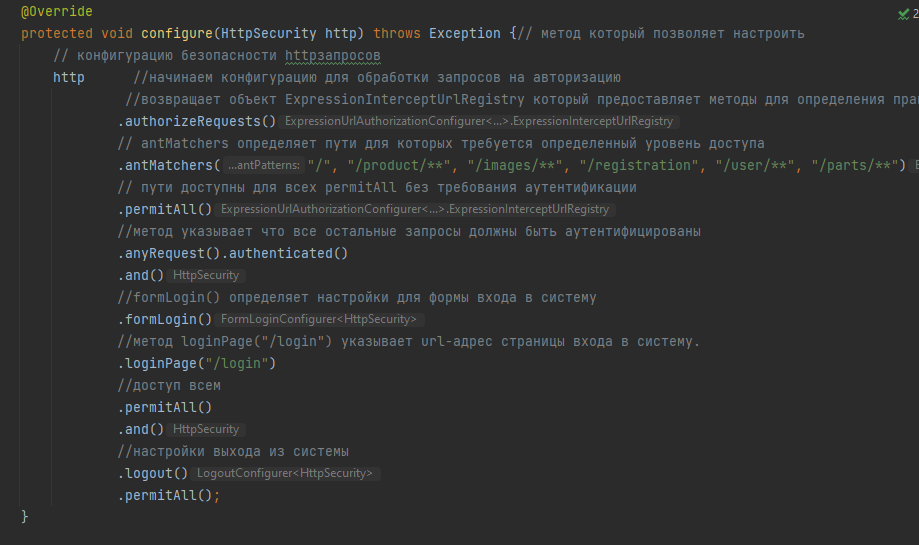


Рис. 10 - класс конфигурации Spring Security

При использовании аннотации **@EnableWebSecurity** вы сообщаете Spring, что вы хотите включить поддержку безопасности в вашем веб-приложении. Обычно эту аннотацию размещают на классе конфигурации Spring, который настраивает безопасность в приложении.

Аннотация **@EnableGlobalMethodSecurity (prePostEnabled = true)** добавляется в конфигурацию Spring Security, для активации поддержки аннотаций безопасности на уровне методов в вашем приложении.

Когда устанавливается значение **prePostEnabled** равным **true**, это означает, что вы разрешаете использование аннотаций безопасности **@PreAuthorize** и **@PostAuthorize** для определения правил доступа к методам вашего приложения.

Рис. 11,12 - Реализация доступа к страницам приложения

**PasswordEncoder** — это интерфейс в фреймворке Spring Security, используемый для шифрования и проверки паролей пользователей.

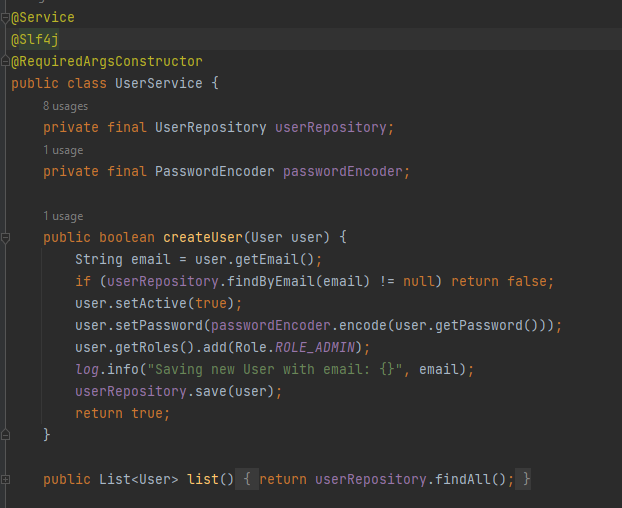


Рис.13 - Сервис класс



Рис. 14 - метод для бана юзеров и смены ролей.

**@Service** используется для аннотированных классов, которые содержат бизнес-логику.

Аналогичны реализованы классы для работы с базой данных.



Рис. 15 - Класс-модель для базы данных объявлений

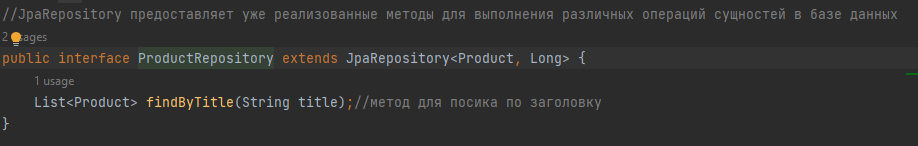


Рис. 16 - Репозиторий для хранения товаров

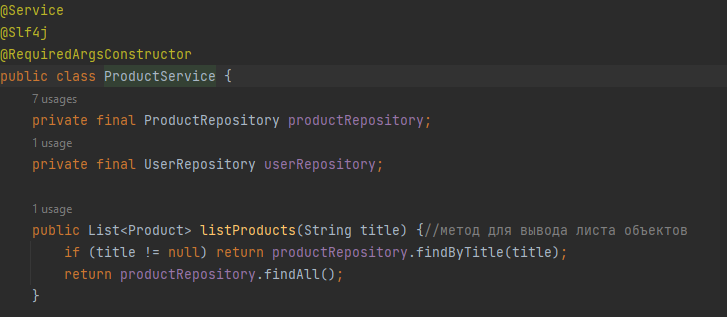


Рис. 17 - Сервис класс

Также в базе данных будут храниться картинки. Связанные с товарами, которые будут добавлены пользователями.

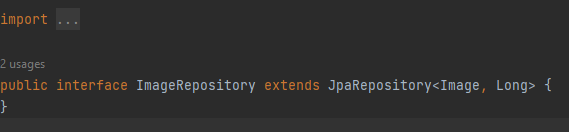


Рис. 18 - Репозиторий для хранения изображений

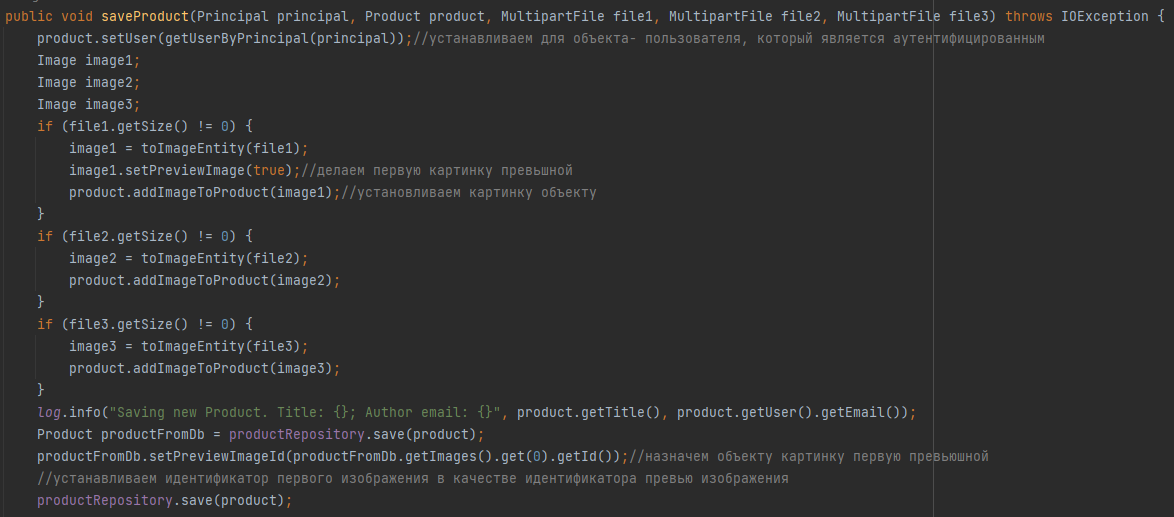


Рис. 19 - Метод класса ProductService

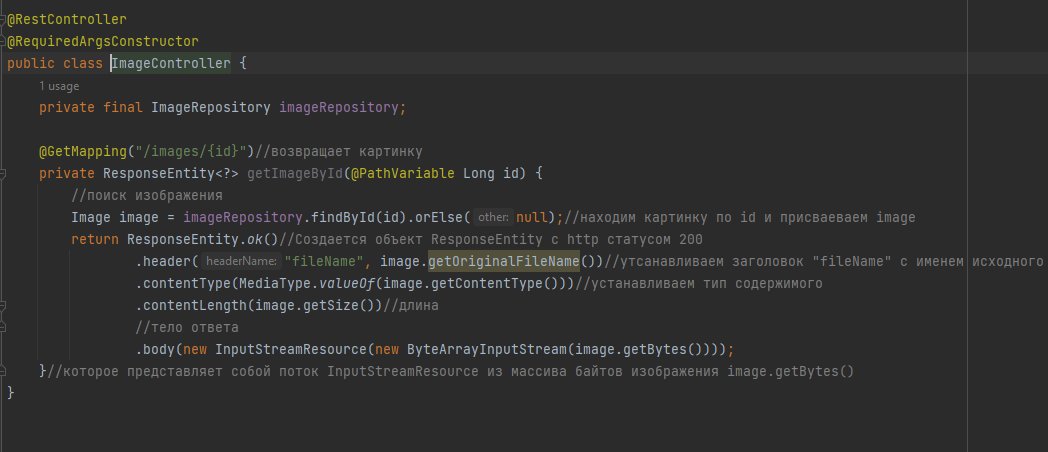


Рис. 20 - Контроллер для обработки запросов пользователей, связанных с картинками для объявлений

## 2.6. Тестирование.

Для того, чтобы проверить наше приложение его нужно протестировать.

Правильно протестированный программный продукт обеспечивает надежность, безопасность и высокую производительность, что в дальнейшем приводит к экономии времени, денег и удовлетворенности клиентов.

Когда данные готовы, мы заходим в web-часть нашего приложения и проходим авторизацию.

Запуская приложение, мы попадаем на главную страницу, доступ к которой реализован для всех пользователей.

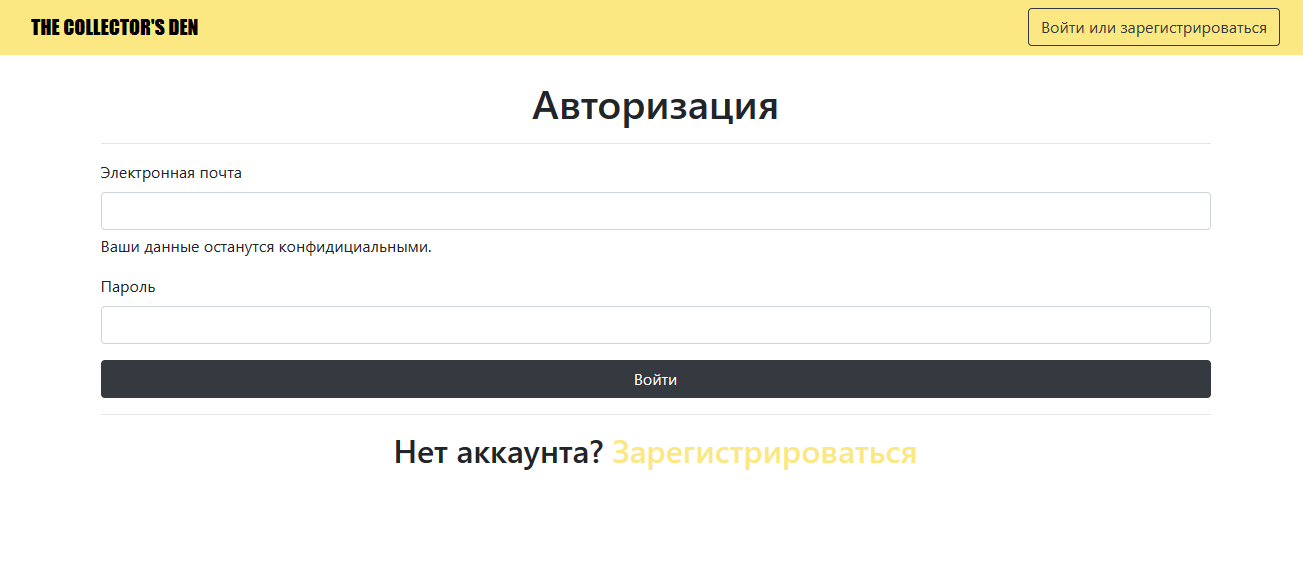


Рис. 21 - Процесс авторизации

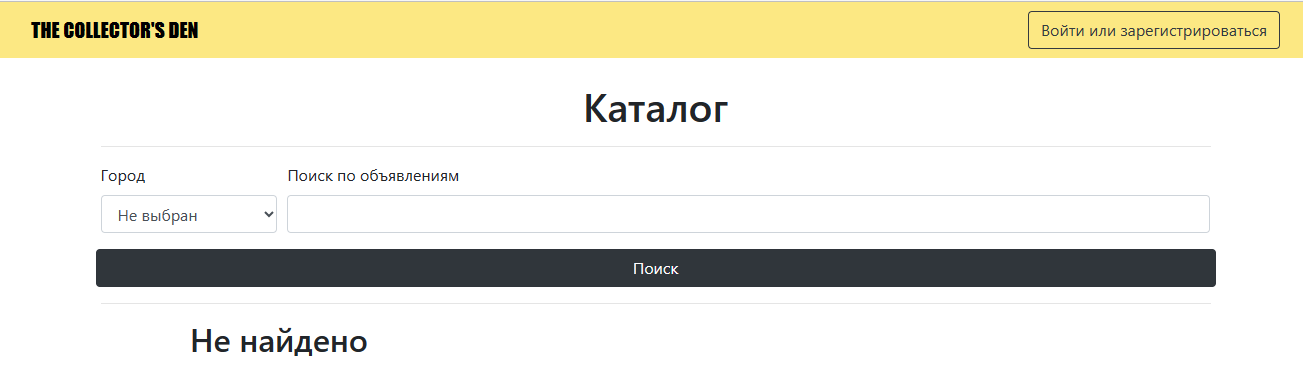


Рис. 22 - Главная страница

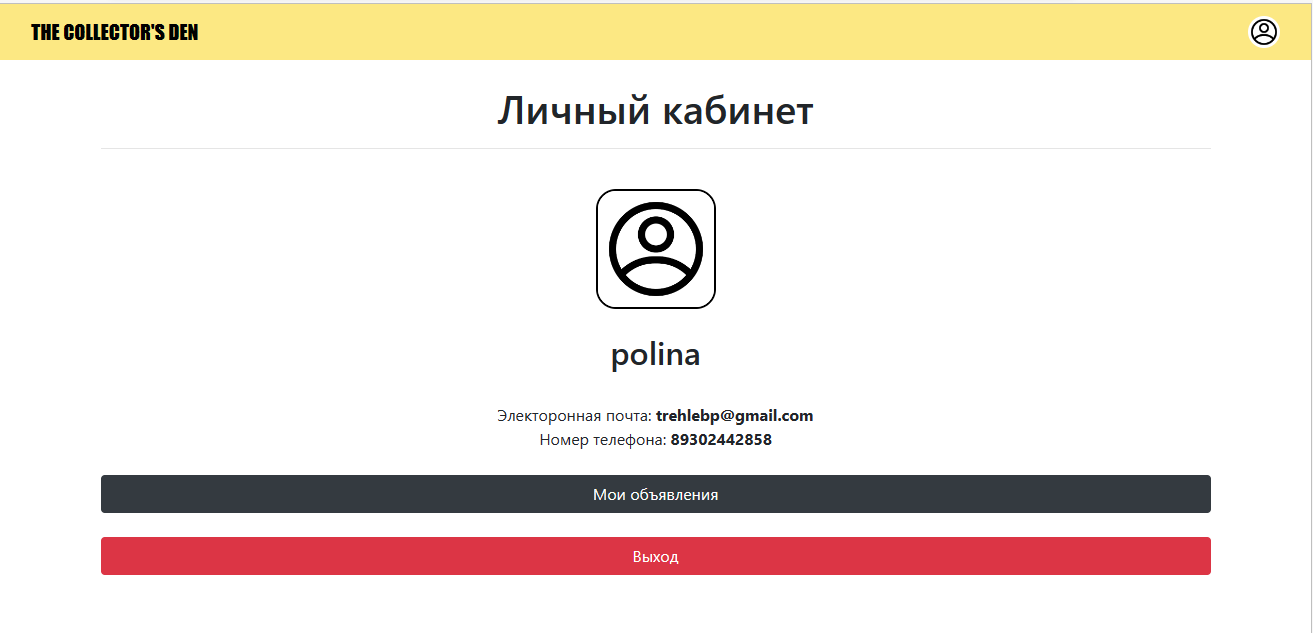


Рис. 23 - Личный кабинет пользователя

При нажатии на кнопки “Мои объявления” мы попадаем на страницу с созданием, удалением объявлений. Данная функция доступна только для авторизированных пользователей.

Нажав на кнопку “Выход” производится выход из аккаунта пользователя.

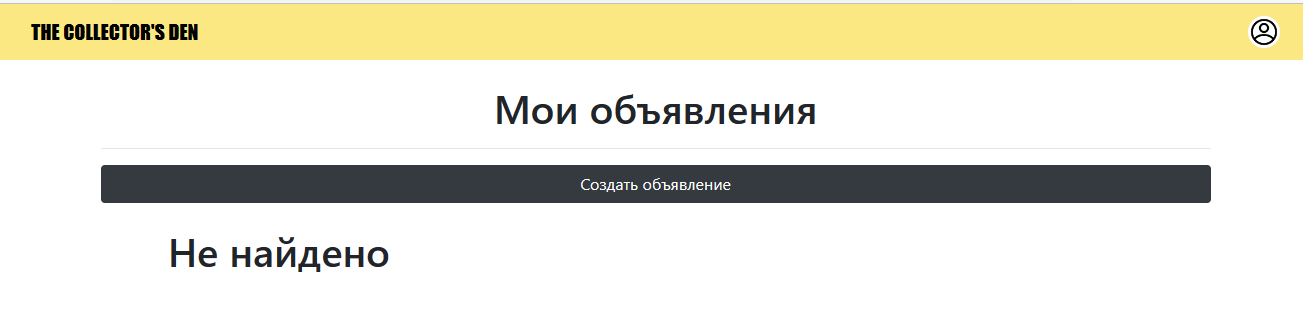


Рис. 24 - страница для создания объявлений

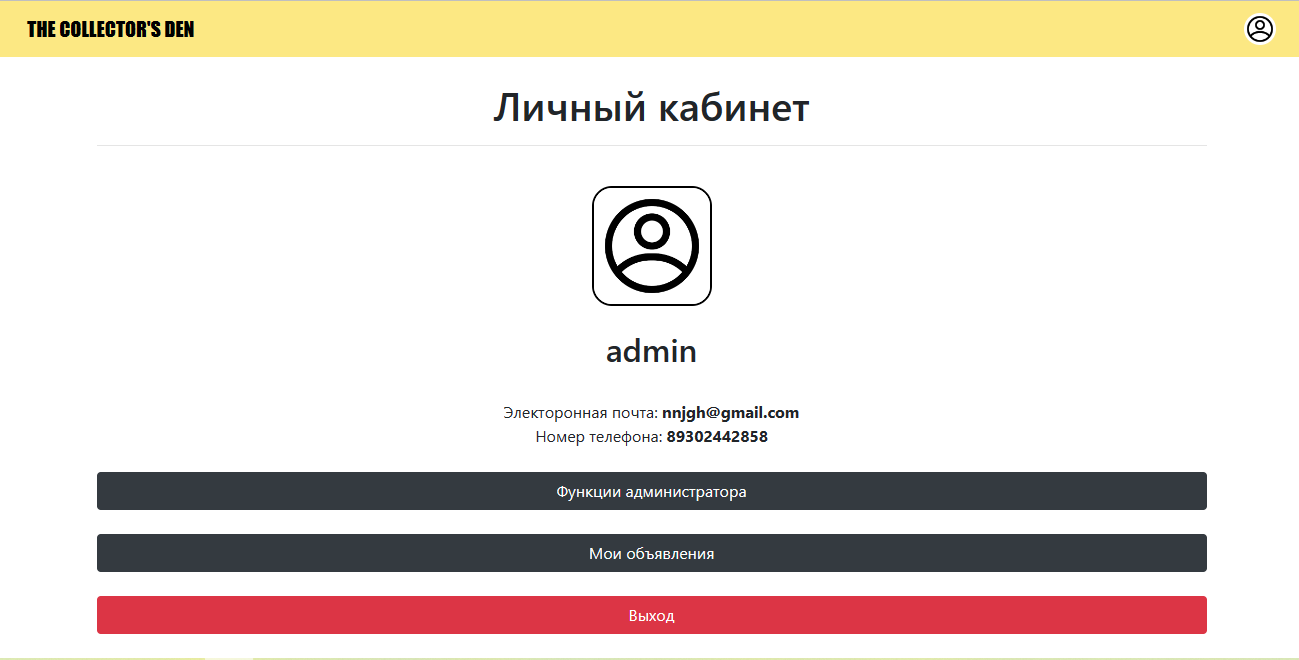
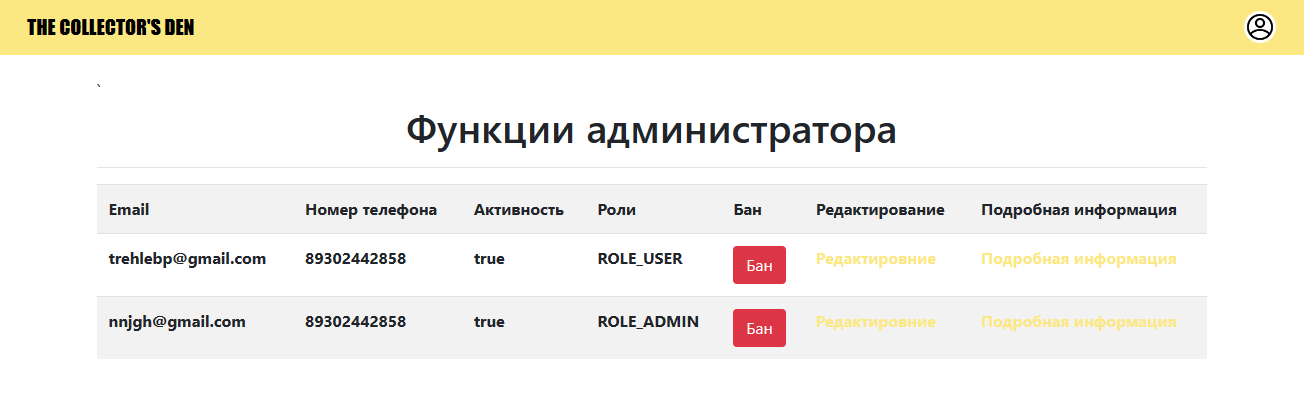


Рис. 25 - Функции администратора

При выборе вкладки “Функции администратора” мы переходим на страницу, с содержанием информации о всех зарегистрированных пользователей. А также можем осуществить функцию бана и редактирования роли.

Рис. 26 - Таблица всех пользователей системы

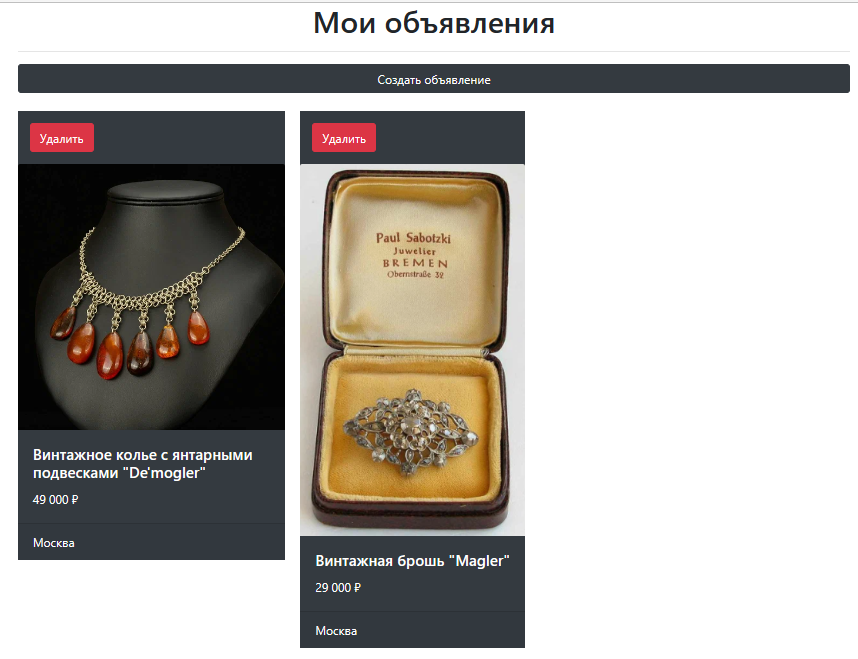
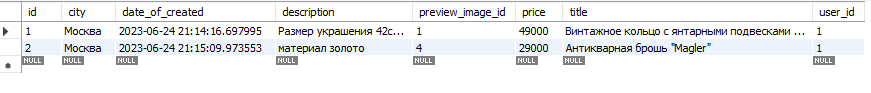
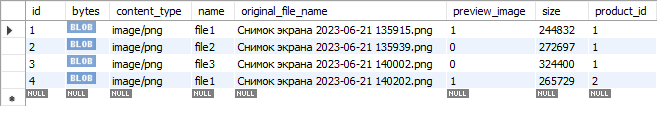
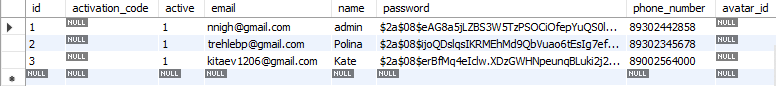


Рис. 27 - Добавление объявлений







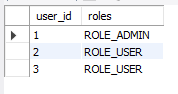


Рис. 28,29,30,31 - Данные в базе данных

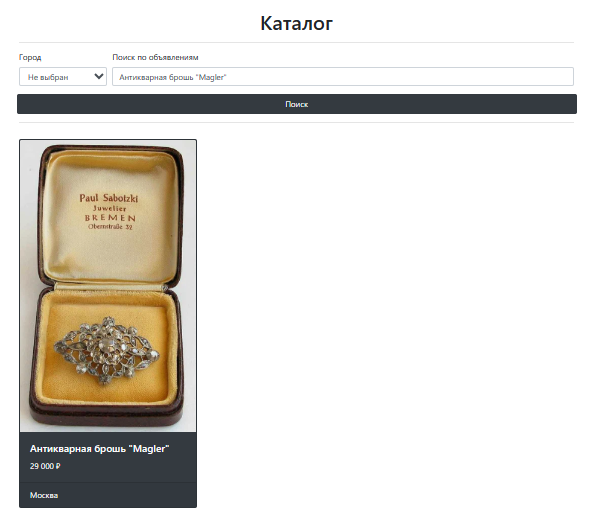


Рис. 32 - Функция поиска

В итоге мы видим, что в процессе работы приложения его функционал работает исправно.

# Заключение

В результате этого курсового проекта все цели и задачи были выполнены. Было успешно разработано веб-приложение с архитектурой MVC (Model-View-Controller) для продажи предметов искусства. Проект предоставляет удобный и надежный интерфейс для пользователей, позволяющий просматривать доступные предметы искусства, добавлять собственные товары.

Веб-приложение MVC для продажи предметов искусства включает следующие ключевые функции и возможности:

1. Регистрация и аутентификация пользователей: Пользователи могут создать учетные записи, а затем войти в систему с использованием своих учетных данных. Это обеспечивает безопасность и индивидуальные профили для каждого пользователя.
2. Просмотр предметов искусства: Пользователи могут просматривать список доступных предметов искусства. Каждый предмет представлен с информацией о его названии, авторе, описании и цене.
3. Поиск и фильтрация: Пользователи могут использовать функцию поиска для нахождения конкретных предметов искусства или применять фильтры для уточнения результатов поиска по автору, жанру, цене и другим характеристикам.
4. Управление администратором: Администратор имеет специальные привилегии, такие как редактирование роли пользователей, а также функции бана.
5. Профиль пользователя: Каждый пользователь имеет свой профиль, где они могут просматривать свои данные, списки своих товаров.

В ходе практической работы были изучены фреймворк Java Spring, набор инструментов для создания интерфейса приложения, а также получен навык работы с СУБД MySQL.

В целом, разработанное веб-приложение MVC для продажи предметов искусства обладает интуитивно понятным интерфейсом, обеспечивает безопасность и удобство использования для пользователей, а также предоставляет управление и администрирование для администратора.

# Список литературы

1. А. А. Байдыбеков, к.воен.н. P. Г. Гильванов, магистр И. А. Молодкин Современные фреймворки для разработки web-приложений. - 2020. - No 4 - С.23-28.
2. Ефремов В. М. РАЗРАБОТКА ВЕБ-САЙТА С ИСПОЛЬЗОВАНИЕМ BOOTSTRAP// ИННОВАЦИОННАЯ НАУКА - 2020. - 6 - С.40-44.
3. Cилкэ Лауренциу С72 Spring быстро. — СПб.: Питер, 2023. — 448 с.: ил. ISBN 978-5-4461-1969-1
4. Уоллс К. Spring в действии. 6-е изд./ пер. с англ. А. Н. Киселева. – М.: ДМКПресс, 2022. – 544 с.: ил.
5. LAURENŢIU SPILCĂ Spring Security in Action [Книга] / LAURENŢIU SPILCĂ — ISBN 9781617297731. — Shelter Island, NY 11964: MANNING SHELTER ISLAND, 2020–562 c.
6. Marco Behler Spring MVC: Подробное руководство / Marco Behler [Электронный ресурс] // javascopes.com: [сайт]. — URL: https://javascopes.com/spring-mvc-in-depth-guide-4adf-4626f9b2/?ysclid=lj5qtnvf3u43088404 (дата обращения: 18.06.2023).
7. Руководство Spring Boot и FreeMarker / [Электронный ресурс] // betacode : [сайт]. — URL: https://betacode.net/11547/spring-boot-and-freemarker (дата обращения: 18.06.2023).