

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий
Кафедра алгебраических и информационных систем

КУРСОВАЯ РАБОТА
по предмету
«Проектирование информационных систем»

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ
ДЛЯ ПРОВЕДЕНИЯ АУКЦИОНА

Студент 3 курса очного отделения
Группа 02371–ДБ
Спиваченко Михаил Витальевич

Руководитель:
к.ф.-м.н., доцент Рябец Л.В.

Иркутск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
Глава 1. Описание предметной области	5
1.1. Анализ предметной области	5
1.2. Требования к разрабатываемому приложению	6
Глава 2. Обзор технологий разработки	8
2.1. Spring Boot	8
2.2. Thymeleaf	8
2.3. Bootstrap	8
2.4. JPA и Hibernate	9
2.5. MySQL	9
Глава 3. Описании реализации приложения	11
3.1. Хранимые сущности и проектирование структуры классов	11
3.2. Разработка серверной части веб-приложения	16
3.3. Разработка клиентской части веб-приложения	19
3.4. Реализованная функциональность	19
ЗАКЛЮЧЕНИЕ	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	23
ПРИЛОЖЕНИЕ 1. UML-диаграмма классов основного модуля приложения	24
ПРИЛОЖЕНИЕ 2. Примеры контроллеров	25
ПРИЛОЖЕНИЕ 3. Примеры шаблонов	30

ПРИЛОЖЕНИЕ 4. Пример создания аукциона	38
ПРИЛОЖЕНИЕ 5. Пример присоединения к аукциону	39
ПРИЛОЖЕНИЕ 6. Пример получение администратором списка транзакций пользователя	40
ПРИЛОЖЕНИЕ 7. Пример повышения ставки пользователем	41

ВВЕДЕНИЕ

В современном мире информационных технологий разработка веб-приложений стала основополагающим элементом, предоставляя пользователям доступ к разнообразным сервисам и функциональным возможностям. В рамках данного проекта были тщательно изучены современные технологии, необходимые для создания инновационного веб-приложения для управления аукционами. Это включает как клиентскую, так и серверную части приложения, а также инструменты для эффективного взаимодействия с базами данных.

Разработка веб-приложений сейчас является неотъемлемой частью современной информационной инфраструктуры, обеспечивая гибкость и масштабируемость для создания различных сервисов. Востребованность в сфере веб-разработки обусловлена не только широкими возможностями, которые она предоставляет, но и рядом других факторов.

Во-первых, веб-приложения обеспечивают доступ к данным и сервисам в режиме реального времени, что делает их незаменимыми для широкого круга пользователей. Во-вторых, разработка веб-приложений поддерживает концепцию кросс-платформенности, что позволяет использовать приложения на различных устройствах и операционных системах. В-третьих, современные технологии веб-разработки обеспечивают высокую степень интерактивности и персонализации, что значительно улучшает пользовательский опыт.

Глава 1. Описание предметной области

1.1. Анализ предметной области

Проект по созданию веб-приложения для управления аукционами требует тщательного анализа предметной области. Необходимо разобраться в ключевых аспектах работы аукционов, их разновидностях и основных потребностях пользователей и администраторов.

Изучим типы аукционов рассмотренных в работе [5]

Типы аукционов

- **Английский аукцион:** Самый распространенный тип аукциона, где ставки повышаются до тех пор, пока не останется только один участник, готовый заплатить наибольшую цену.
- **Голландский аукцион:** Начальная цена очень высокая, и постепенно снижается до тех пор, пока кто-то не согласится купить товар по текущей цене.
- **Закрытый аукцион:** Участники подают свои предложения в запечатанных конвертах, и победителем становится тот, кто предложил наивысшую цену.

Участники аукциона

- **Аукционеры:** Пользователи, размещающие свои товары или услуги на аукционе. Они устанавливают начальную цену и возможные условия продажи.
- **Аукционисты:** Пользователи, делающие ставки на аукционные лоты. Их цель - выиграть аукцион, предложив наивысшую цену.
- **Администраторы:** Пользователи, ответственные за управление и модерацию аукционов, включая проверку достоверности информации о лотах

и участников, а также разрешение споров.

Процесс аукциона

- **Создание аукциона:** Продавец регистрирует новый лот, указывая его описание, начальную цену, время окончания аукциона и другие параметры.
- **Подача ставок:** Покупатели делают ставки на лоты. В зависимости от типа аукциона, ставки могут повышаться или понижаться.
- **Завершение аукциона:** Аукцион завершается либо по истечении времени, либо при достижении определенной цены. Победитель определяется в зависимости от правил аукциона.

Анализ предметной области показал, что создание веб-приложения для управления аукционами требует учета множества факторов и требований. В ходе анализа было принято решение реализовать проект используя Английский тип аукциона.

1.2. Требования к разрабатываемому приложению

Функциональные требования

Для продавцов

- Возможность создавать и редактировать аукционы.
- Управление лотами (добавление описаний, фотографий и т.д.).

Для покупателей

- Возможность поиска и фильтрации аукционов по различным параметрам.
- Участие в аукционах путем подачи ставок.
- Уведомления о перебитых ставках и завершении аукционов.

Для администраторов

- Модерация аукционов и пользователей.
- Управление тематиками и категориями аукционов.

Нефункциональные требования

Производительность

- Система должна поддерживать одновременное участие большого числа пользователей без значительного ухудшения производительности.

Безопасность

- Защита данных пользователей и обеспечение конфиденциальности.
- Механизмы аутентификации и авторизации.

Удобство использования

- Интуитивно понятный интерфейс для всех типов пользователей.
- Поддержка различных устройств и браузеров.

Глава 2. Обзор технологий разработки

2.1. Spring Boot

Spring Boot – это фреймворк для создания производительных, готовых к использованию приложений на платформе Java [1]. Он предоставляет:

- **Упрощенную конфигурацию и развертывание:** Spring Boot позволяет быстро настроить проект с помощью автоматической конфигурации и предустановленных шаблонов.
- **Интеграция с различными компонентами Spring:** Включая Spring Data, Spring Security и другие модули, что позволяет создавать гибкие и масштабируемые приложения.

2.2. Thymeleaf

Thymeleaf – это серверный шаблонизатор для Java, который используется для создания динамических веб-страниц [2]. Основные преимущества:

- **Интуитивно понятный синтаксис:** Позволяет легко интегрировать статические шаблоны HTML с динамическими данными.
- **Поддержка модульности и переиспользования кода:** Возможность создания фрагментов шаблонов, что улучшает структуру и поддержку кода.
- **Совместимость с Spring:** Простая интеграция с Spring MVC, что делает его идеальным выбором для нашего проекта.

2.3. Bootstrap

Bootstrap – это популярный фреймворк для разработки адаптивных веб-интерфейсов [3]. Он предоставляет:

- **Сеточную систему:** Упрощает создание адаптивного дизайна, который корректно отображается на различных устройствах.
- **Компоненты пользовательского интерфейса:** Большой набор готовых компонентов (формы, кнопки, навигационные панели и т.д.), что ускоряет процесс разработки.
- **Кросс-браузерная совместимость:** Гарантирует корректное отображение и функционирование интерфейса во всех современных браузерах.

2.4. JPA и Hibernate

JPA – это спецификация Java для управления постоянными данными. Hibernate – это одна из реализаций JPA, которая предоставляет ORM (Object-Relational Mapping) [4] . функциональность:

- **Объектно-реляционное отображение:** Упрощает работу с базой данных, позволяя разработчикам работать с данными в виде объектов.
- **Автоматическое создание и управление таблицами в базе данных:** Упрощает процессы миграции и управления схемами базы данных.
- **Поддержка запросов на уровне объектов:** Позволяет писать запросы к базе данных на языке HQL (Hibernate Query Language), что улучшает читаемость и поддержку кода.

2.5. MySQL

MySQL – это реляционная система управления базами данных, которая использовалась для хранения данных нашего приложения:

- **Надежность и производительность:** Широко используется благодаря своей стабильности и эффективности.

- **Интеграция с Hibernate:** Позволяет легко настроить и управлять базой данных через Spring Boot и Hibernate.
- **Поддержка транзакций и сложных запросов:** Обеспечивает целостность данных и возможность выполнения сложных аналитических запросов.

Глава 3. Описании реализации приложения

Взаимодействие классов между собой, представлено в диаграмме классов в приложении 1

3.1. Хранимые сущности и проектирование структуры классов

Хранимые сущности

- **Пользователь (User):** Содержит основную информацию о зарегистрированных пользователях приложения. Пример в листинге 1

Листинг 1. User

```
public class User implements UserDetails{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Integer id;
    @Size(min = 4, message = "Логин не может быть меньше 4 символов")
    String login;
    @Size(min = 4, message = "Пароль не может быть меньше 4 символов")
    String password;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "user_data_id", referencedColumnName = "id")
    UserData userData;
    @ManyToMany(fetch = FetchType.EAGER)
    private Set<Role> roles;
```

- **Данные пользоватлея (UserData):** Содержит дополнительную информацию о зарегистрированных пользователях приложения. Пример в листинге 2

Листинг 2. UserData

```
public class UserData {
```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Integer id;
String email;
@Column(name="number_phone")
String numberPhone;
Integer deposit;

}

```

- **Роль (Role):** Роли пользователей. Пример в листинге 3

Листинг 3. Role

```

public class Role implements GrantedAuthority{
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Integer id;

String name;

@Override
public String getAuthority() {
    return getName();
}

}

```

- **Аукцион (Auction):** Содержит основную информацию о аукционах приложения. Пример в листинге 4

Листинг 4. Role

```

public class Auction {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

Integer id;

@ManyToOne
@JoinColumn(name = "host_user_id", referencedColumnName = "id")
User userHost;

@Size(min = 1, message = "Имя не может быть пустым")
@NotNull(message = "Имя не может быть пустым")
String name;

@Valid
@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "auction_date_id", referencedColumnName = "id")
AuctionData auctionData;
}

```

- **Данные аукциона** Содержит дополнительную информацию о аукционах приложения. Пример в листинге 5

Листинг 5. Role

```

public class AuctionData {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Integer id;

@Size(min = 1, message = "Описание не может быть пустым")
@NotNull(message = "Описание не может быть пустым")
String description;

String Photo;

Integer startCost;
}

```

```

@NotNull(message = "Выберите тематику")
@ManyToOne
@JoinColumn(name = "thematics_id", referencedColumnName = "id")
Thematics thematics;

@DateTimeFormat(pattern = "YYYY-MM-DD HH:MM:SS")
@Column(name = "start_date")
LocalDateTime startDate;

@ManyToMany
List <User> members;

Integer timeEndDelay;

@Column(name = "is_active")
boolean Active;

@Column(name = "sand_msg")
int sandMsg = 0;

}

```

- **Оповещения (Notification):** Содержит сообщение, дату отправки, получателя и прочитано ли оповещение. Пример в листинге 6

Листинг 6. Role

```

public class Notification {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Integer id;

@ManyToOne
@JoinColumn(name = "recipient_user_id", referencedColumnName = "id")

```

```

User recipient;
String message;
@DateTimeFormat(pattern = "YYYY-MM-DD HH:MM:SS")
@Column(name = "post_date")
LocalDateTime postDate;
@Column(name = "readed")
boolean Readed;

}

```

- **Тематика (Thematics):** Тематики аукционов в приложении. Пример в листинге 7

Листинг 7. Role

```

public class Thematics {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Integer id;
String name;
}

```

Дополнительные классы

- **Поиск аукциона (AuctionSearch):** Содержит информацию о параметрах для нахождения аукционов. Пример в листинге 8

Листинг 8. Role

```

public class AuctionSearch {
Integer maxCost = null;
Integer minCost = null;
Thematics thematics = null;
String name;
}

```

3.2. Разработка серверной части веб-приложения

Контроллеры

В проекте реализованы 6 контроллеров

- **AdminController** – обрабатывает запросы для администратора.
- **AuctionController** – обрабатывает запросы связанные с аукционами.
- **AuthController** – обрабатывает запросы связанные с аутентификацией пользователей.
- **HomeController** – обрабатывает домашнюю страницу.
- **NotificationController** – обрабатывает запросы связанные с оповещениями.
- **ThematicsController** – обрабатывает запросы связанные с тематиками.
- **UserController** – обрабатывает запросы связанные с пользователями.

Примеры контроллеров расположены в приложении 2

Конфигураторы

Реализованы 3 конфигуратора

- **AdditionalResourceWebConfiguration** – Настраивает папки для хранения дополнительных ресурсов. Пример в листинге 6.

Листинг 9. AdditionalResourceWebConfiguration

```
@Configuration
```

```
public class AdditionalResourceWebConfiguration implements
```

```
    WebMvcConfigurer {
```

```
        @Override
```

```
        public void addResourceHandlers(final ResourceHandlerRegistry
```

```
registry) {
```

```
    registry.addResourceHandler("/Aimages/**").addResourceLocations("file:Aimages/")
```



```
    }  
}
```

- **SchedulingConfiguration** – Настраивает выполнение асинхронных задач на сервере. Пример в листинге 6.

Листинг 10. SchedulingConfiguration

```
@Configuration  
@EnableScheduling  
public class SchedulingConfiguration {  
  
    public ThreadPoolTaskScheduler threadPoolTaskScheduler() {  
        ThreadPoolTaskScheduler threadPoolTaskScheduler = new  
        ThreadPoolTaskScheduler();  
        threadPoolTaskScheduler.setPoolSize(10);  
  
        threadPoolTaskScheduler.setThreadNamePrefix("ThreadPoolTaskScheduler");  
        return threadPoolTaskScheduler;  
    }  
}
```

- **SecurityConfig** – Настраивает аутентификацию пользователя.

Компоненты

Реализованы 2 компонента

- **DatabaseInitializer** – Добавляет данные в таблицу при запуске проекта.
- **ScheduledTasks1** – Задача, которая асинхронно выполняется на сервере каждые 5 секунд. Нужна для отправки оповещений о начале и конце аукционов.

Ошибки

Реализованы 5 классов ошибок.

- **ActualDateExeption** – Ошибка, указывающая, что дата выбранная пользователем при создании аукциона, уже не является актуальной (меньше чем текущая дата).
- **DateIsNull** – Ошибка о том, что дата является пустой.
- **RoleRegisterExeption** – Ошибка о том, что роль с данным названием уже зарегистрирована.
- **ThematicsSaveExeption** – Ошибка что тематика уже зарегистрирована.
- **UserRegisterExeption** – Ошибка что пользователь уже зарегистрирован.

Сервисы

Реализованы 5 сервисов.

- **AuctionService** – Сервис обрабатывающий действия с аукционами.
- **NotificationService** – Сервис обрабатывающий действия с оповещениями
- **RoleService** – Сервис обрабатывающий действия с ролями.
- **ThematicsService** – Сервис обрабатывающий действия с тематиками.
- **UserService** – Сервис обрабатывающий действия с пользователями.

Репозитории

Реализованы 6 репозиториев.

- **AuctionRepository** – Репозиторий для взаимодействия с аукционами
- **NotificationRepository** – Репозиторий для взаимодействия с оповещениями
- **RoleRepository** – Репозиторий для взаимодействия с ролями
- **TransactionRepository** – Репозиторий для взаимодействия с транзакциями
- **ThematicsRepository** – Репозиторий для взаимодействия с тематиками

- **UserRepository** – Репозиторий для взаимодействия с пользователями

3.3. Разработка клиентской части веб-приложения

В проекте реализованы 16 шаблонов

- **header** – Шапка для страниц
- **home** – Домашняя страница
- **login** – Страница для аутентификации пользователя
- **myauction** – Страница с аукционами пользователя
- **register** – Страница для регистрации пользователя
- **listTransactions** – Страница для отображения списка транзакций
- **userlist** – Страница для отображения списка пользователей
- **auctionChange** – Страница для изменения аукциона
- **auctionCteate** – Страница для создания аукциона
- **auctionInfo** – Страница для отображения информации об аукционе
- **notifications** – Страница для отображения оповещений
- **create** – Страница для создания тематики
- **list** – Страница для отображения тематик
- **addMoney** – Страница для добавления денег
- **userChange** – Страница для изменения данных пользователя
- **userInfo** – Страница для отображения данных пользователя

Примеры шаблонов расположены в приложении 3

3.4. Реализованная функциональность

В web-приложении были реализованны:

Создание аукциона

1. Пользователь нажимает на ссылку "Создать аукцион"
2. Система выводит форму для заполнения данных аукциона
3. Пользователь заполняет форму и нажимает "Создать"
4. Если поля заполнены верно, аукцион будет создан, иначе система вернет форму и укажет на неверно заполненные поля

Пример реализации расположен в приложении 4

Присоединение к аукциону

1. Пользователь вводит критерии для поиска аукциона и нажимает кнопку поиск
2. Система выводит список аукционов по заданным критериям
3. Пользователь выбирает аукцион и нажимает на его название
4. Система выводит данные аукциона
5. Пользователь нажимает кнопку "Присоединиться"

Пример реализации расположен в приложении 5

Поднятие ставки

1. Пользователь нажимает на ссылку "Мои аукционы"
2. Система выводит список аукционов в которых участвует пользователь
3. Пользователь выбирает аукцион
4. Система выводит данные аукциона
5. Вводит сумму ставки и нажимает поднять ставку
6. Если ставка корректна, система создает транзакцию с этой ставкой.

Пример реализации расположен в приложении 7

Администратор просматривает транзакции пользователя

1. Администратор нажимает на ссылку "Список пользователей"
2. Система выводит список пользователей
3. Администратор выбирает пользователя
4. Администратор нажимает на кнопку "Список транзакций"
5. Система выводит список транзакций пользователя

Пример реализации расположен в приложении 6

Список дополнительной функциональности

- Регистрация и аутентификация
- Изменения данных аукциона
- Добавление денег
- Добавление тематик администратором
- Изменение данных пользователя

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы были получены следующие результаты:

- Изучена предметная область по проведению аукционов ;
- Реализовано приложение для проведения Английского типа аукционов;
- Реализованы серверная и клиентская части приложения;
- Реализовано хранения данных приложения в базе данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Machine Learning (vc.ru). Spring Boot 101: Введение в создание веб-приложений. — 2023. — URL: <https://vc.ru/u/1389654-machine-learning/586955-spring-boot-101-vvedenie-v-sozdanie-veb-prilozheniy> (дата обр. 28.05.2024).
2. pilot911. Spring Boot: современный подход к разработке Java-приложений. — 2018. — URL: <https://habr.com/ru/articles/350864/> (дата обр. 28.05.2024).
3. SkillFactory. Bootstrap: что это такое и зачем нужен. — 2023. — URL: <https://blog.skillfactory.ru/glossary/bootstrap/> (дата обр. 28.05.2024).
4. vedenin1980. Шпаргалка Java программиста 1: JPA и Hibernate в вопросах и ответах. — 2015. — URL: <https://habr.com/ru/articles/265061/> (дата обр. 28.05.2024).
5. Национальный исследовательский университет "Высшая школа экономики". Четыре основных типа аукционов. — 2012. — URL: https://www.hse.ru/data/2012/01/26/1264433944/lecture_06_12.pdf (дата обр. 28.05.2024).

ПРИЛОЖЕНИЕ 1.

UML-диаграмма классов основного модуля приложения

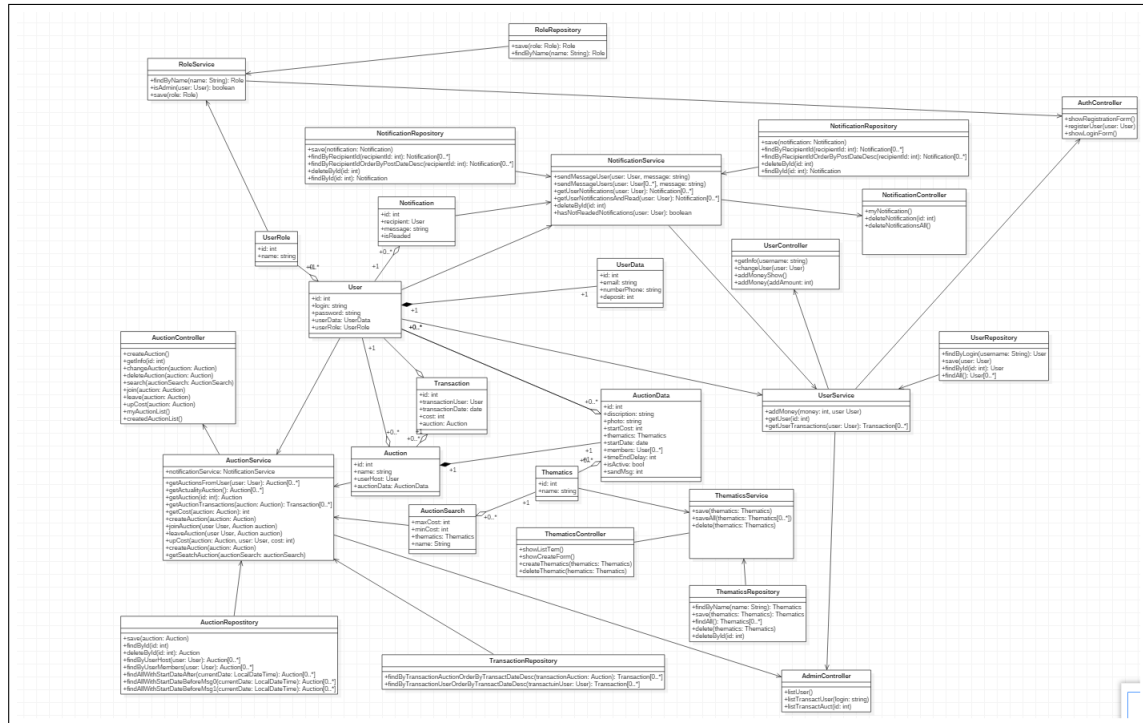


Рисунок 1. UML-диаграмма классов основного модуля приложения

ПРИЛОЖЕНИЕ 2.

Примеры контроллеров

Листинг 11. AdminController

```
@Controller
@RequestMapping("/adm")
public class AdminController {
    @Autowired
    UserService userService;
    @Autowired
    RoleService roleService;
    @Autowired
    AuctionService auctionService;
    @ModelAttribute("getUser")
    public User currentUser()
    {
        return userService.getCurrentUser();
    }
    @ModelAttribute("isAdmin")
    public boolean isAdmin()
    {
        return roleService.isAdmin();
    }
    @Autowired
    NotificationService notificationService;
    @ModelAttribute("hasNotification")
    public boolean hasNotification()
    {
        return notificationService.hasNotReadedNotifications(currentUser());
    }
    @GetMapping("/listUser")
    public String CreateAuction(Model model)
```

```

    {
        model.addAttribute("users",userService.findAll());
        return "adm/userlist";
    }
    @GetMapping("/listTransactUser/{login}")
    public String listTansactUser(@PathVariable("login") String login, Model
model)
    {
        User user = userService.findByUsername(login);
        model.addAttribute("transactions",
auctionService.getTransactionsUser(user));
        return "adm/listTransactions";
    }

    @GetMapping("/listTransactAuction/{id}")
    public String listTansactAuct(@PathVariable("id") int id, Model model)
    {
        Auction auction = auctionService.findById(id);
        model.addAttribute("transactions",
auctionService.getTransactions(auction));
        return "adm/listTransactions";
    }
}

```

Листинг 12. AuthController

```

@Controller
public class AuthController {
    @Autowired
    private UserService userService;
    @Autowired
    private RoleService roleService;

```

```

@GetMapping("/register")
public String showRegistrationForm(Model model) {
    model.addAttribute("user", new User());
    return "register";
}

@PostMapping("/register")
public String registerUser(@Valid @ModelAttribute("user") User user,
BindingResult result, Model model, RedirectAttributes redirectAttributes) {
    if(result.hasErrors()){
        return "register";
    }
    try{
        Role roleUser = roleService.findByName("ROLE_USER");
        Set <Role> roles = new HashSet<Role>();
        roles.add(roleUser);
        user.setRoles(roles);
        user.getUserData().setDeposit(0);
        userService.registerUser(user);
    }
    catch (UserRegisterExeption e)
    {
        model.addAttribute("ErrorMessage",e.getMessage());
        return "register";
    }
    redirectAttributes.addFlashAttribute("succesfull", "Успешно зарегестриров
ан");
    return "redirect:/login";
}

@GetMapping("/login")
public String showLoginForm(@RequestParam(name = "error", required = false)
String error, Model model) {
    if ("true".equals(error)) {

```

```

        model.addAttribute("ErrorMessage", "Неверный логин или пароль");
    }
    return "login";
}
}

```

Листинг 13. NotificationController

```

@Controller
@RequestMapping("/mess")
public class NotificationController {
    @Autowired
    UserService userService;
    @Autowired
    RoleService RoleService;
    @ModelAttribute("getUser")
    public User currentUser()
    {
        return userService.getCurrentUser();
    }
    @ModelAttribute("isAdmin")
    public boolean isAdmin()
    {
        return RoleService.isAdmin();
    }
    @Autowired
    NotificationService notificationService;
    @ModelAttribute("hasNotification")
    public boolean hasNotification()
    {
        return notificationService.hasNotReadedNotifications(currentUser());
    }
    @GetMapping("/my")

```

```

public String GetMapping(Model model)
{
    model.addAttribute("notifications",
notificationService.getUserNotificationsAndRead(currentUser()));
    return "notification/notifications";
}

@Transactional
@PostMapping("/delete/{id}")
public String deleteNotification(@PathVariable("id") int id) {
    Notification notification = notificationService.findById(id);
    if (notification.getRecipient().equals(currentUser()))
        notificationService.deleteById(id);
    return "redirect:/mess/my";
}

@Transactional
@PostMapping("/deleteall")
public String deleteNotificationsAll() {
    List<Notification> notifications =
notificationService.getUserNotifications(currentUser());
    notificationService.deleteAll(notifications);
    return "redirect:/mess/my";
}

}

```

ПРИЛОЖЕНИЕ 3.

Примеры шаблонов

Листинг 14. auctionCreate

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml"
  xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8"/>
  <title>Создать аукцион</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"

  integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMHjY6hW+ALEwIH"
  crossorigin="anonymous"/>
</head>
<body>
<div th:replace="~{header :: header}"></div>
<div class="container">
  <div class="row justify-content-center mt-5">
    <div class="col-lg-8">
      <div class="card">
        <div class="card-header">
          <h2 class="card-title">Создать аукцион</h2>
        </div>
        <div class="card-body">
          <form th:action="@{/auctions/create}" th:object="${auction}"
method="post">

            <div class="mb-3">
              <label for="name"
```

```

class="form-label">Название:</label>

      <input type="text" id="name" class="form-control"
maxlength="100" th:field="*{name}"/>

      <span th:if="${#fields.hasErrors('name')}}"
th:errors="*{name}" class="text-danger"></span>

    </div>

    <div class="mb-3">

      <label for="description"
class="form-label">Описание:</label>

      <textarea id="description" maxlength="1000"
class="form-control" th:field="*{auctionData.description}"/></textarea>

      <span
th:if="${#fields.hasErrors('auctionData.description')}}"
th:errors="*{auctionData.description}" class="text-danger"></span>

    </div>

    <div class="mb-3">

      <label for="thematics"
class="form-label">Тематика:</label>

      <select id="thematics" class="form-select"
th:field="*{auctionData.thematics}">

        <option value="">Выберите тематику</option>
        <option th:each="thematic : ${listThematics}"
th:value="${thematic.id}" th:text="${thematic.name}"></option>

      </select>

      <span
th:if="${#fields.hasErrors('auctionData.thematics')}}"
th:errors="*{auctionData.thematics}" class="text-danger"></span>

    </div>

    <div class="mb-3">

      <label for="startCost" class="form-label">Начальная с
тоимость:</label>

      <input type="number" min="1" max="1000000000"
id="startCost" class="form-control" th:field="*{auctionData.startCost}"/>

```



```

    xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"/>
    <title>Уведомления</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"

    integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
    crossorigin="anonymous"/>
</head>
<body>
<div th:replace="~{header :: header}"></div>
<div class="container">
    <div class="row justify-content-center mt-5">
        <div class="col-lg-8">
            <div class="card">
                <div class="card-header">
                    <h2 class="card-title">Уведомления</h2>
                </div>
                <div class="card-body">
                    <table class="table">
                        <thead>
                            <tr>
                                <th>Сообщение</th>
                                <th>Дата</th>
                                <th></th> <!-- Добавляем заголовок для кнопки уда
ления -->
                            </tr>
                        </thead>
                        <tbody>
                            <tr th:each="notification : ${notifications}">
                                <td th:text="${notification.message}"></td>

```

```

        <td
th:text="${#temporals.format(notification.postDate, 'HH:mm:ss
dd-MM-yyyy')}"></td>

        <td>
            <form th:action="@{'/mess/delete/' +
${notification.id}}" method="post">
                <button type="submit" class="btn
btn-danger">Удалить</button>
            </form>
        </td>
    </tr>
</tbody>
</table>
<div class="mt-3">
    <form th:action="@{/mess/deleteall}" method="post">
        <button type="submit" class="btn btn-danger">Удалить
все</button>
    </form>
</div>
<div th:if="${ErrorMessage}" class="alert alert-danger mt-3">
    <span th:text="${ErrorMessage}"></span>
</div>
</div>
</div>
</div>
</div>
</div>
</body>
</html>

```

Листинг 16. userInfo

```
<!DOCTYPE html>
```

```

<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title th:text="${user.login}">Пользователь</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    <div th:replace="~{header :: header}"></div>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card">
                    <div class="card-header">
                        <h2 class="card-title">Информация</h2>
                    </div>
                    <div class="card-body">
                        <p><strong>Логин:</strong> <span
th:text="${user.login}"></span></p>
                        <p><strong>email:</strong> <span
th:text="${user.userData.email}"></span></p>
                        <p><strong>номер телефона:</strong> <span
th:text="${user.userData.numberPhone}"></span></p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>

```

Листинг 17. listTransactions

```

<!DOCTYPE html>
<html lang="en"
  xmlns = "http://www.w3.org/1999/xhtml"
  xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8"/>
  <title>Транзакции</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJY6hW+ALEwIH"
crossorigin="anonymous"/>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</head>
<body>
<div th:replace="~{header :: header}"></div>
  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-8">
        <div class="card">
          <div class="card-header">
            <h2 class="card-title">Транзакции</h2>
          </div>
          <div class="card-body">
            <table class="table">
              <thead>
                <tr>
                  <th>Пользователь</th>
                  <th>Аукцион</th>

```

```

        <th>Цена</th>
        <th>Дара</th>
    </tr>
</thead>
<tbody>
    <tr th:each="transaction : ${transactions}">
        <td><a
th:href="@{/user/{id}(id=${transaction.transactionUser.Login})}"> <span
th:text="${transaction.transactionUser.Login}"></span> </a></td>
        <td><a
th:href="@{/auctions/{id}(id=${transaction.transactionAuction.id})}"> <span
th:text="${transaction.transactionAuction.name}"></span> </a></td>
        <td><span
th:text="${transaction.cost}"></span></td>
        <td><span
th:text="${#temporals.format(transaction.transactDate, 'dd-MM-yyyy
HH:mm:ss')}"></span></td>
    </tr>
</tbody>
</table>
</div>
</div>
</div>
</div>
</div>
</body>
</html>

```

ПРИЛОЖЕНИЕ 4.

Пример создания аукциона

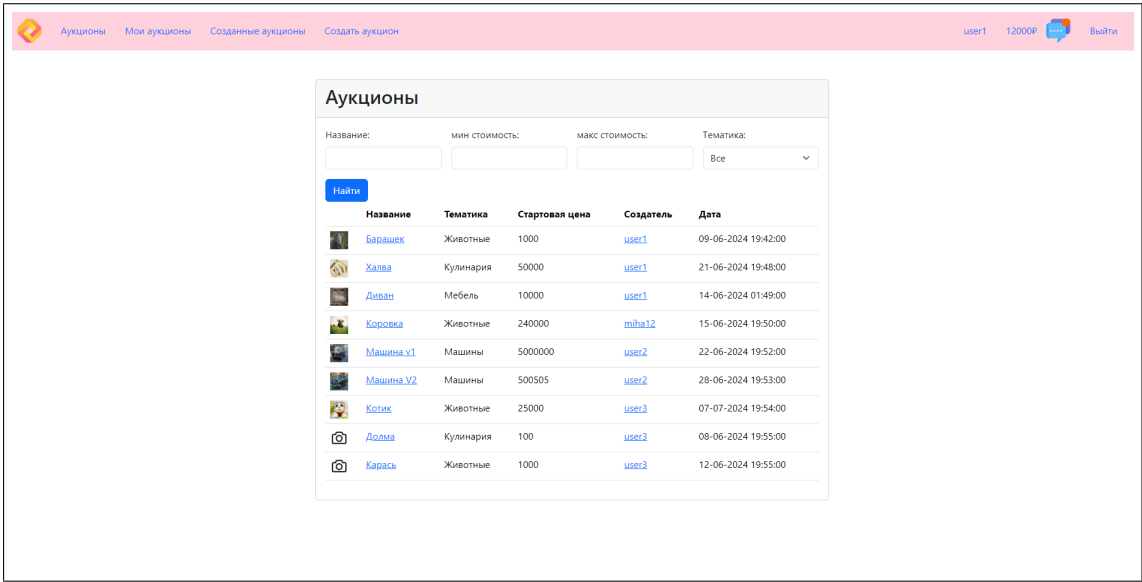


Рисунок 2. Список аукционов

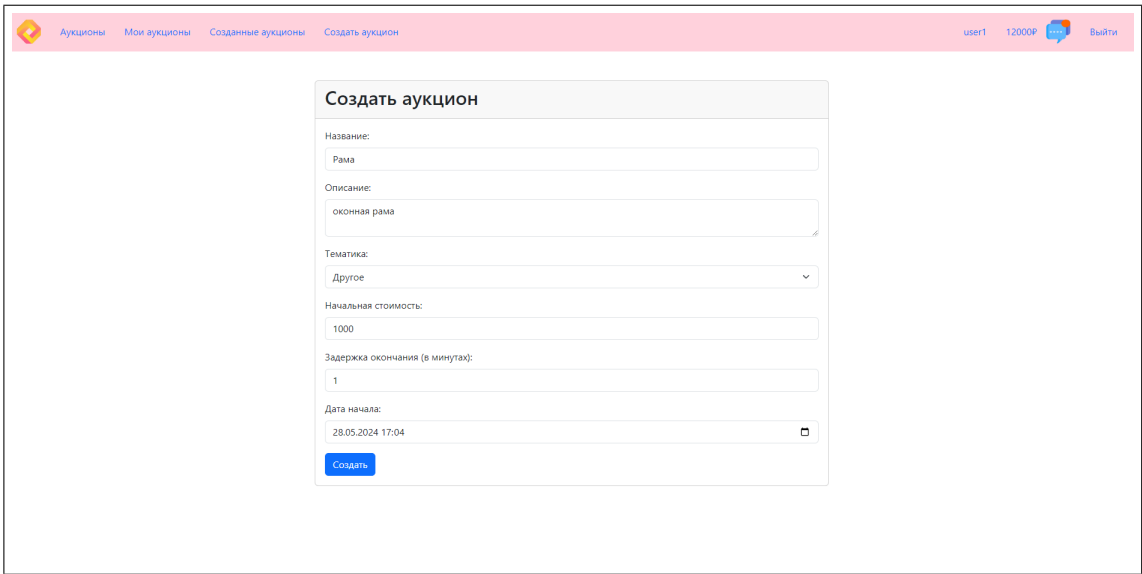


Рисунок 3. Создание аукциона

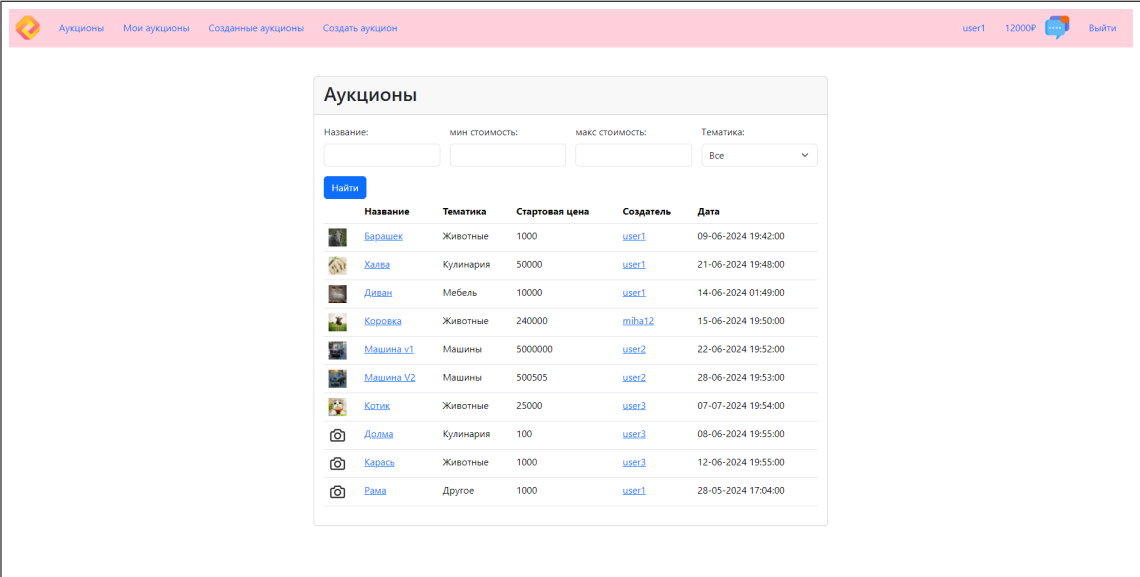


Рисунок 4. Список с созданным аукционом

ПРИЛОЖЕНИЕ 5.

Пример присоединения к аукциону

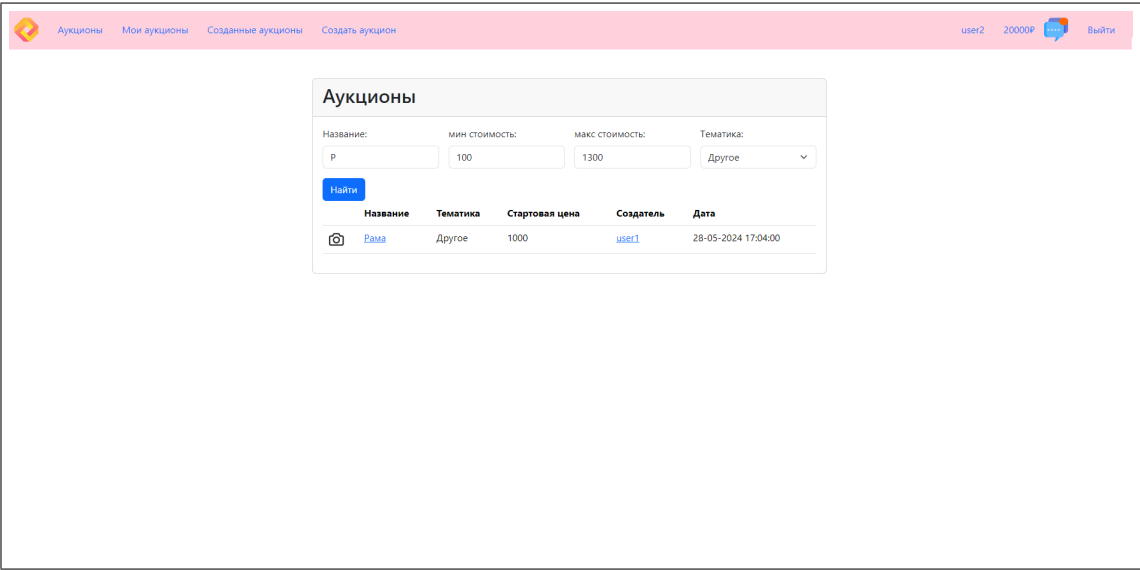


Рисунок 5. Аукционы пользователя

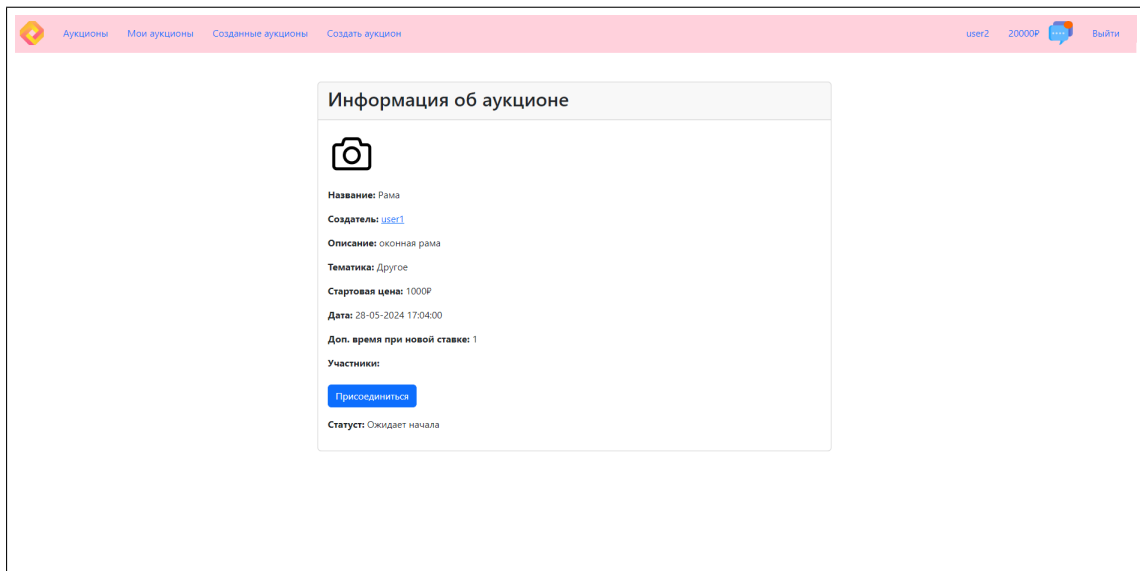


Рисунок 6. Информация об аукционе

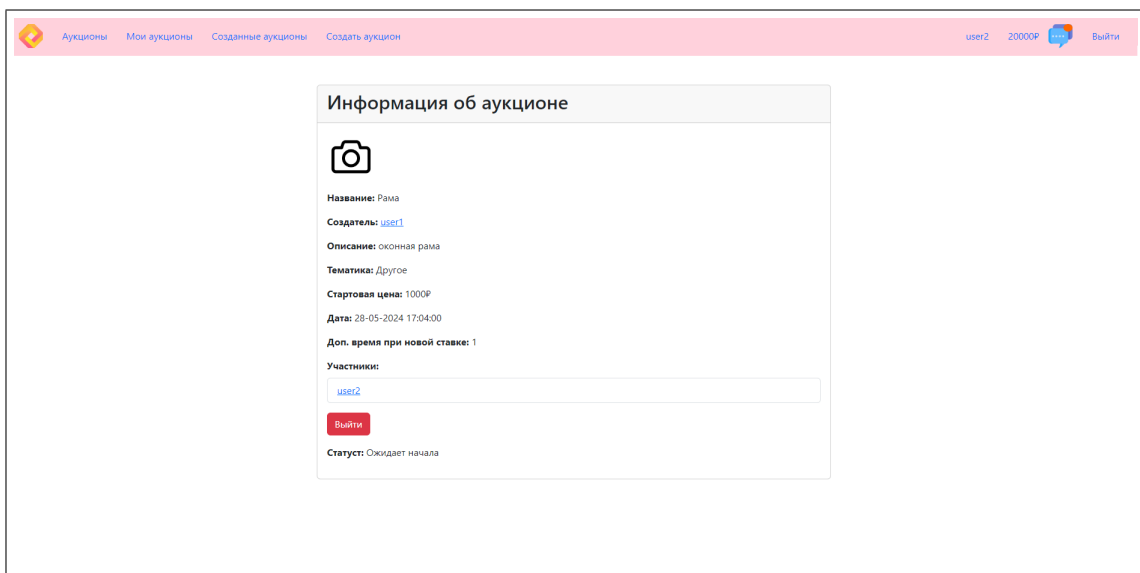


Рисунок 7. Информация об аукционе после присоединения

ПРИЛОЖЕНИЕ 6.

Пример получение администратором списка транзакций пользователя

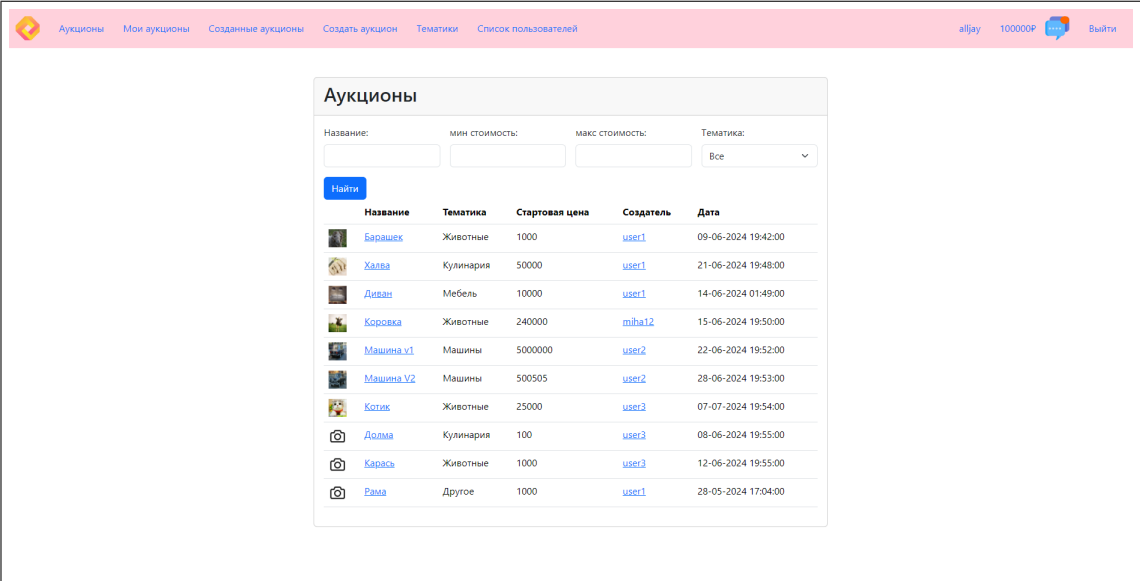


Рисунок 8. Список аукционов (администратор)

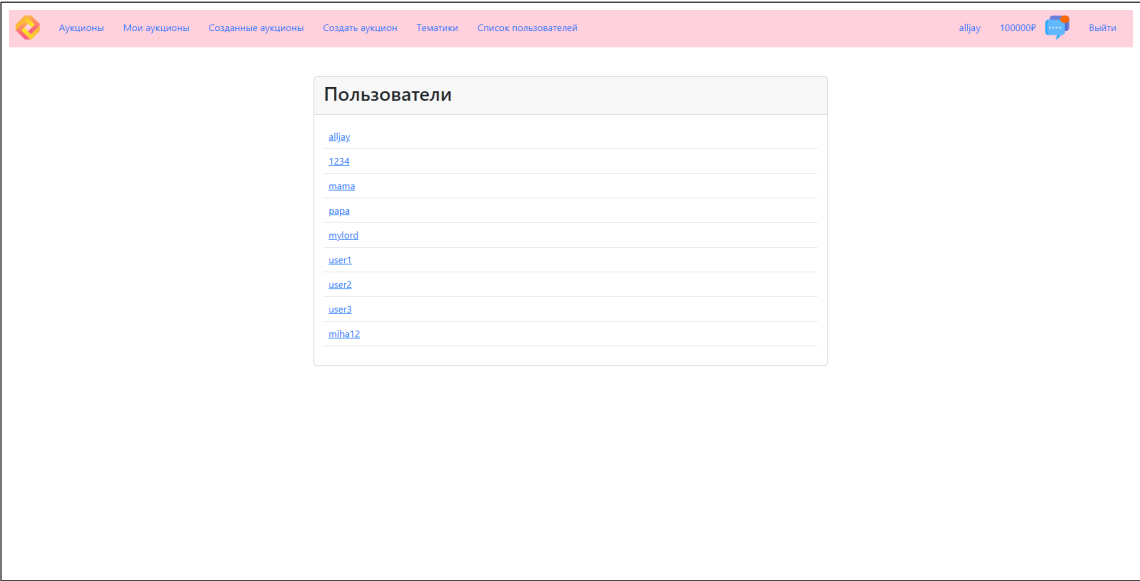


Рисунок 9. Список пользователей

ПРИЛОЖЕНИЕ 7.

Пример повышения ставки пользователем

Аукционы

Мои аукционы

Созданные аукционы

Создать аукцион

Тематики

Список пользователей

alJay

100000₽

Выйти

Информация

Логин:

user2

Пароль:

Email:

Номер телефона:

Изменить данные

Список транзакций

Рисунок 10. Информация о пользователе

Аукционы

Мои аукционы

Созданные аукционы

Создать аукцион

Тематики

Список пользователей

alJay

100000₽

Выйти

Транзакции

Пользователь	Аукцион	Цена	Дата
user2	Изумрудное кольцо	17000	26-05-2024 20:03:48
user2	Сапфировое кольцо	2000	26-05-2024 19:58:55
user2	Сапфировое кольцо	500	26-05-2024 19:58:17

Рисунок 11. Список транзакций пользователя

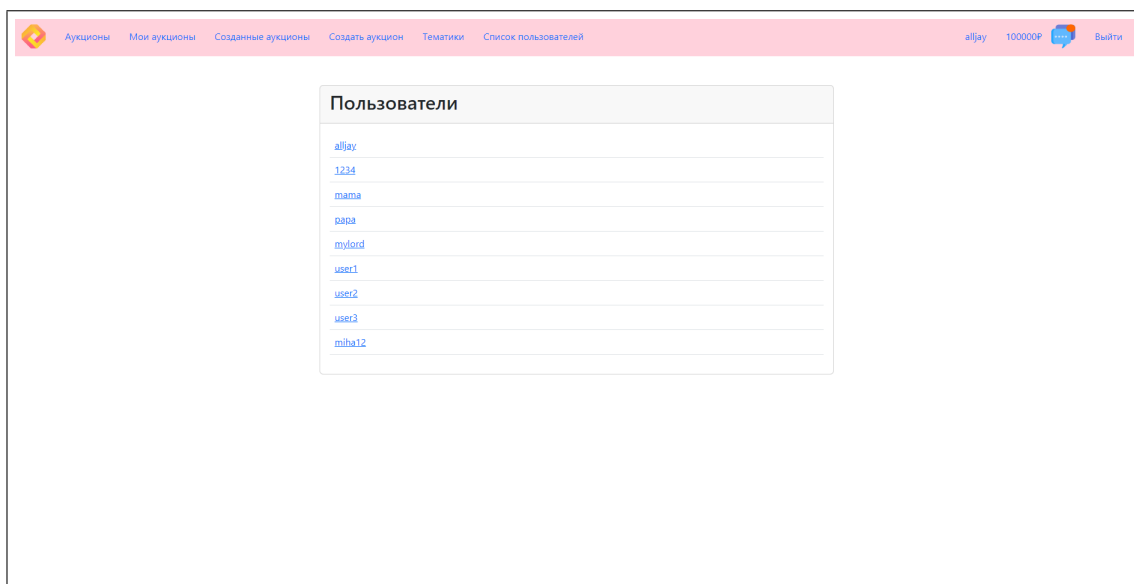


Рисунок 12. Список аукционов пользователя

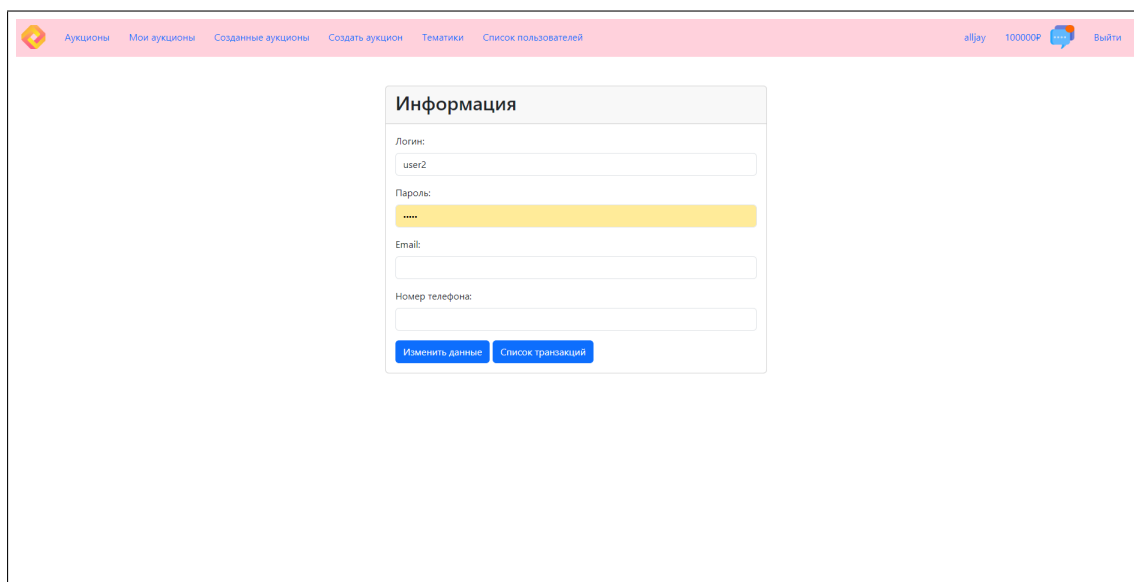


Рисунок 13. Информация об аукционе



	Аукционы	Мои аукционы	Созданные аукционы	Создать аукцион	Тематики	Список пользователей	alljay	100000P		Выйти
Транзакции										
Пользователь	Аукцион	Цена	Дата							
user2	Искусственное кольцо	17000	26-05-2024 20:03:48							
user2	Сапфировое кольцо	2000	26-05-2024 19:58:55							
user2	Сапфировое кольцо	500	26-05-2024 19:58:17							

Рисунок 14. Информация об аукционе после удачной ставки