



ФГБОУ ВО Уральский государственный горный
университет

Инженерно-экономический факультет
Кафедра информатики

Курсовой проект

По дисциплине «Технологии программирования»

На тему «Разработка приложения для оценки
покупателей «Яндекс.Маркет»

Работу выполнил:
Студент группы ИНФ-20-2
Варанкин Денис

Проверила:
ст. преп. каф. информатики
Волкова Е.А.

Екатеринбург. 2023г

Оглавление

1. Постановка задачи	3
1.1 Характеристика объекта	3
1.2 Потенциал внедрения	3
1.3 Краткое техническое задание.....	3
1.3.1 Назначение и цели создания системы.....	3
1.3.1.1 Назначение системы	3
1.3.1.2 Цели создания системы	3
1.3.2 Требования к системе	4
1.3.2.1 Требования к системе в целом.....	4
1.3.2.1.1 Требования к структуре и функционированию системы.....	4
2. Проектирование системы	4
2.1 Моделирование системы.....	4
2.2 Проектирование БД	8
2.3 Проектирование интерфейса	10
3. Разработка системы	11
3.1 Выбор средств реализации	11
3.2 Структура проекта	11
3.3.1 Листинг и описание классов	13
3.3.2 Интерфейс приложения	18

1. Постановка задачи

Целью этого курсового проекта является разработка WEB-приложения для оценки покупателей «Яндекс.Маркет»

Приложение должно иметь три уровня: уровень базы данных, бизнес=логики приложения и web-интерфейс, а также иметь разграничения доступа (модератор, пользователь)

1.1 Характеристика объекта

Объектом информатизации является WEB-приложение, которое связано с БД.

1.2 Потенциал внедрения

Данное приложение разрабатывается для курьеров, которые доставляют заказы из «Яндекс.Маркета», чтобы поделиться отзывами о клиенте

Приложение будет очень полезным, так как пользователи могут оставлять и просматривать подробную информацию о клиенте и адресе доставки, что позволит быстрее доставлять заказы

1.3 Краткое техническое задание

Согласно ГОСТ 34.602-89

1.3.1 Назначение и цели создания системы

1.3.1.1 Назначение системы

Данное приложение предназначена для создания и просмотра отзывов о клиенте

1.3.1.2 Цели создания системы

- Предоставление возможности оставлять отзыв о клиенте с подробной информацией о нём
- Предоставление возможности просматривать отзывы, чтобы получить подробную информацию о клиенте

1.3.2 Требования к системе

1.3.2.1 Требования к системе в целом

1.3.2.1.1 Требования к структуре и функционированию системы

Функциональные требования:

2. Авторизация: система должна предоставлять форму с обязательными полями: электронная почта и пароль от учётной записи Яндекс
3. Создание отзыва о клиенте: система должна предоставлять интерфейс с формой для создания отзыва о клиенте с добавлением информации (адрес доставки, имя клиента, комментарий, оценка пользователя)
4. Просмотр отзыва о клиенте: система должна показывать информацию о просматриваемом отзыве клиента (адрес доставки, имя клиента, комментарий, оценка пользователя)

Требования к интерфейсу:

- 1) Страница для авторизации должна иметь поля для ввода с названием электронной почты и пароля
- 2) Страница для создания отзыва о клиенте должна иметь поля для ввода с названием адреса доставки, имени клиента, комментария и количества звёзд из 5, для оценки клиента
- 3) Страница просмотра отзыва о клиенте должна иметь информацию о адресе доставки, имени клиента, комментарии и оценки клиента

Требования безопасности:

- 1) Обеспечение защиты конфиденциальных данных пользователей;

Требования к производительности:

- 1) Авторизация пользователей не должна превышать более 10 секунд
- 2) Сохранение созданного и отредактированного отзыва не должна превышать 10 секунд
- 3) Работа сервера должна быть постоянной и не иметь сбоев

Требования к сопровождению:

- 1) Система должна иметь возможность обновления для добавления новых функций и улучшения существующих
- 2) Система должна иметь поддержку, для технического обслуживания системы при возможных проблемах в работоспособности

2. Проектирование системы

2.1 Моделирование системы

Ниже приведена UML-диаграмма для Use Case (диаграмма вариантов использования)

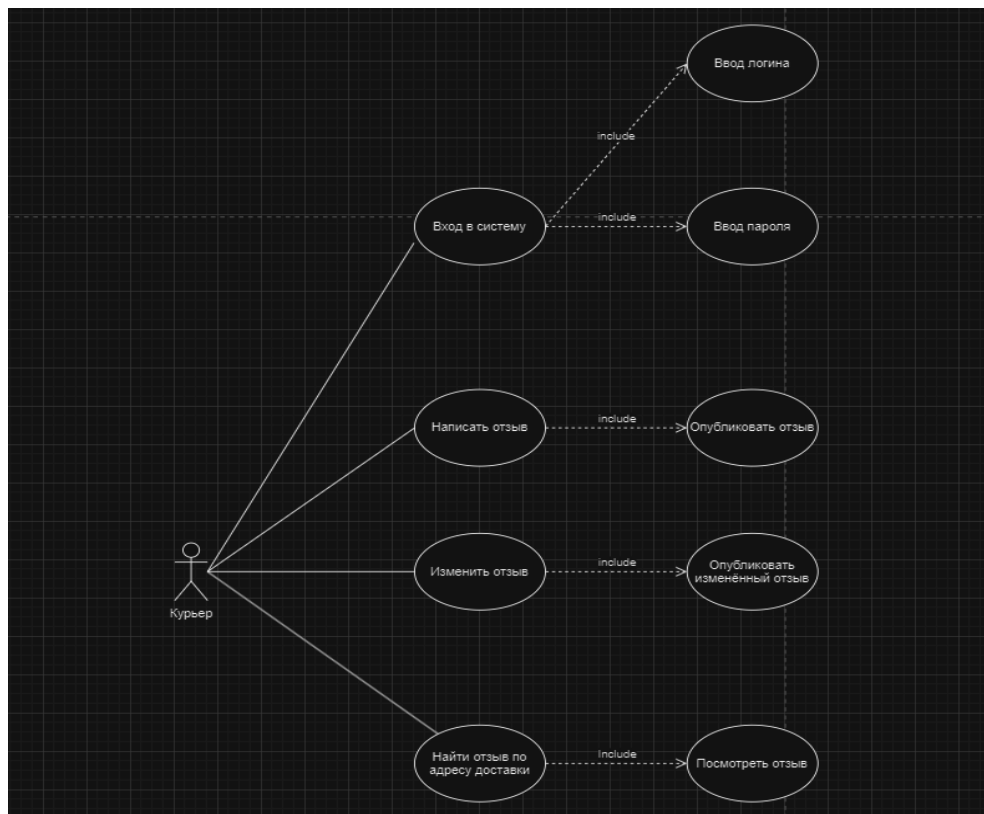


Рисунок 1. Use Case

Ниже приведена контекстная (концептуальная) диаграмма приложения

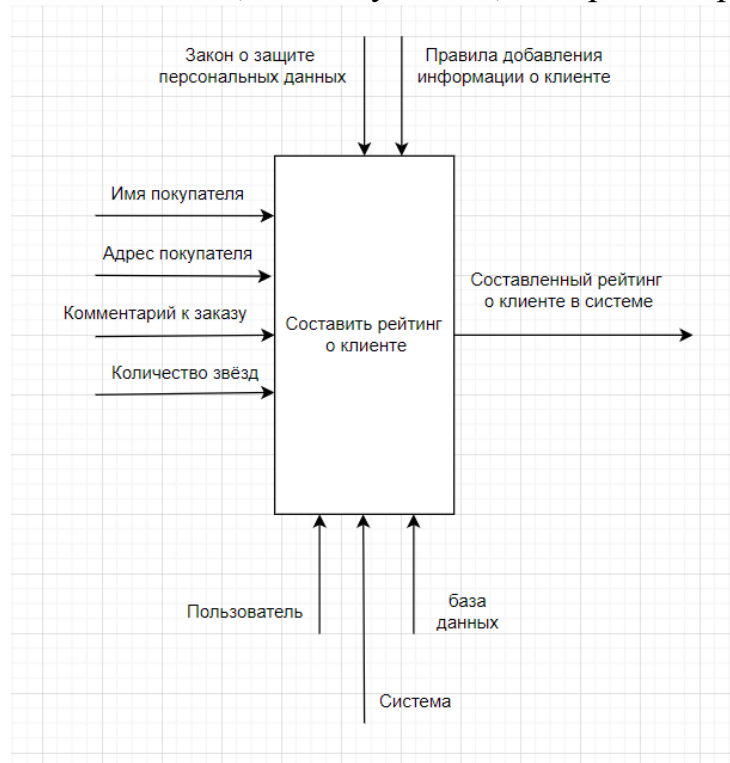


Рисунок 2. Контекстная диаграмма

Ниже приведена диаграмма IDEF0, которая показывает процесс создания отзыва о клиенте

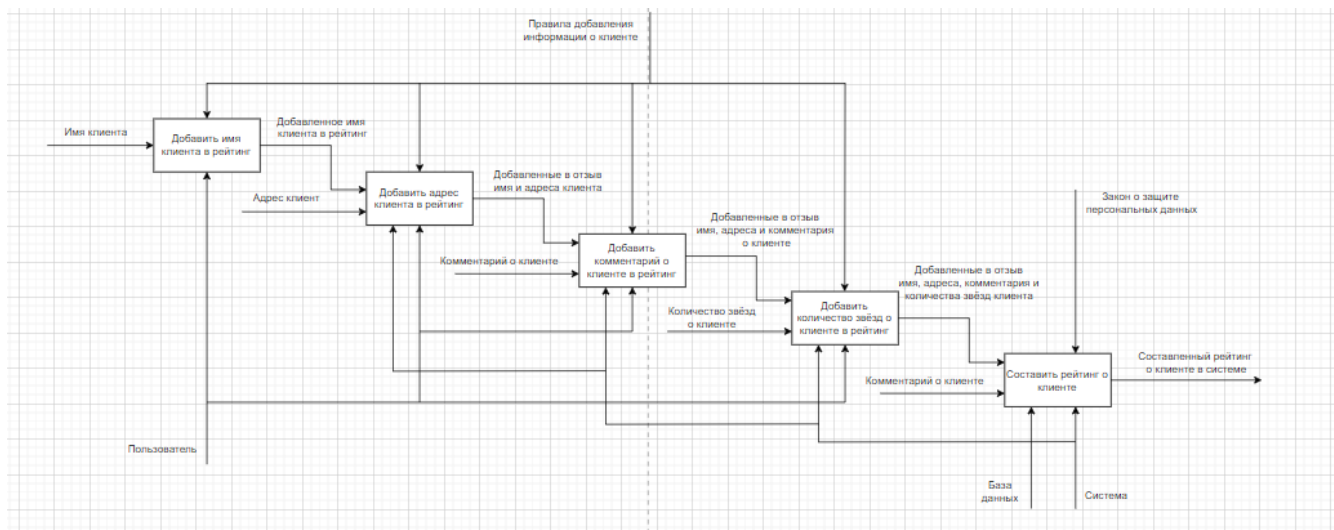


Рисунок 3. IDEF0

Ниже приведена диаграмма потоковых данных DFD

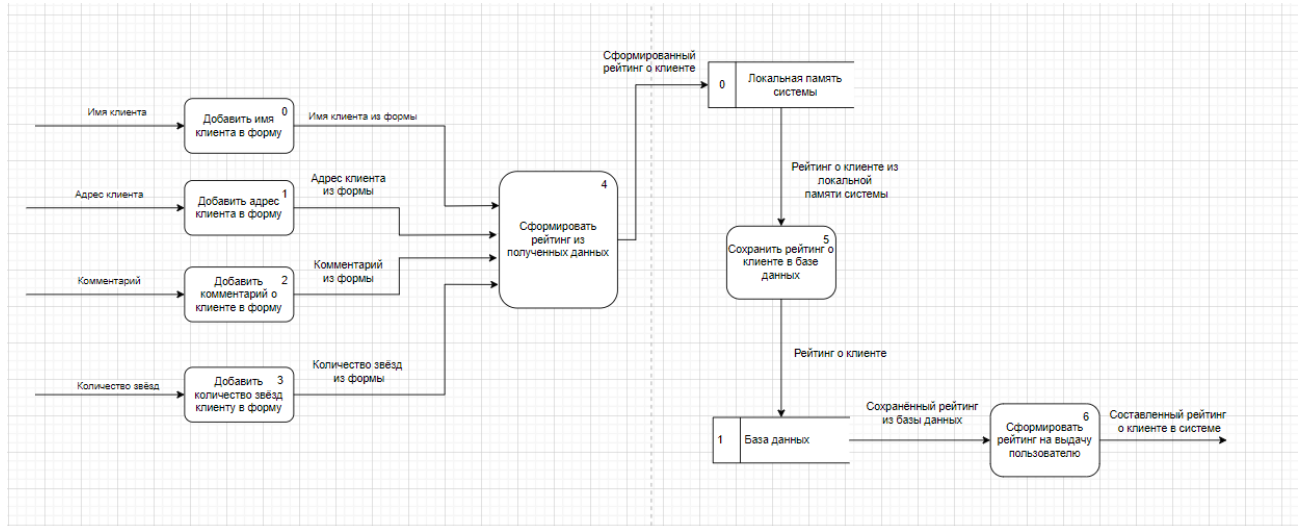


Рисунок 4. DFD

Для реализации данного web-приложение используется архитектурный паттерн MVC (Model-View-Controller), который разделяет архитектуру приложения на три модуля: модель, представление и контроллер. Это позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки web-приложения

4.1 Проектирование БД

Для реализации хранения данных была выбрана база данных MySQL, на сервере MySQL

Ниже приведена ER-диаграмма хранения данных

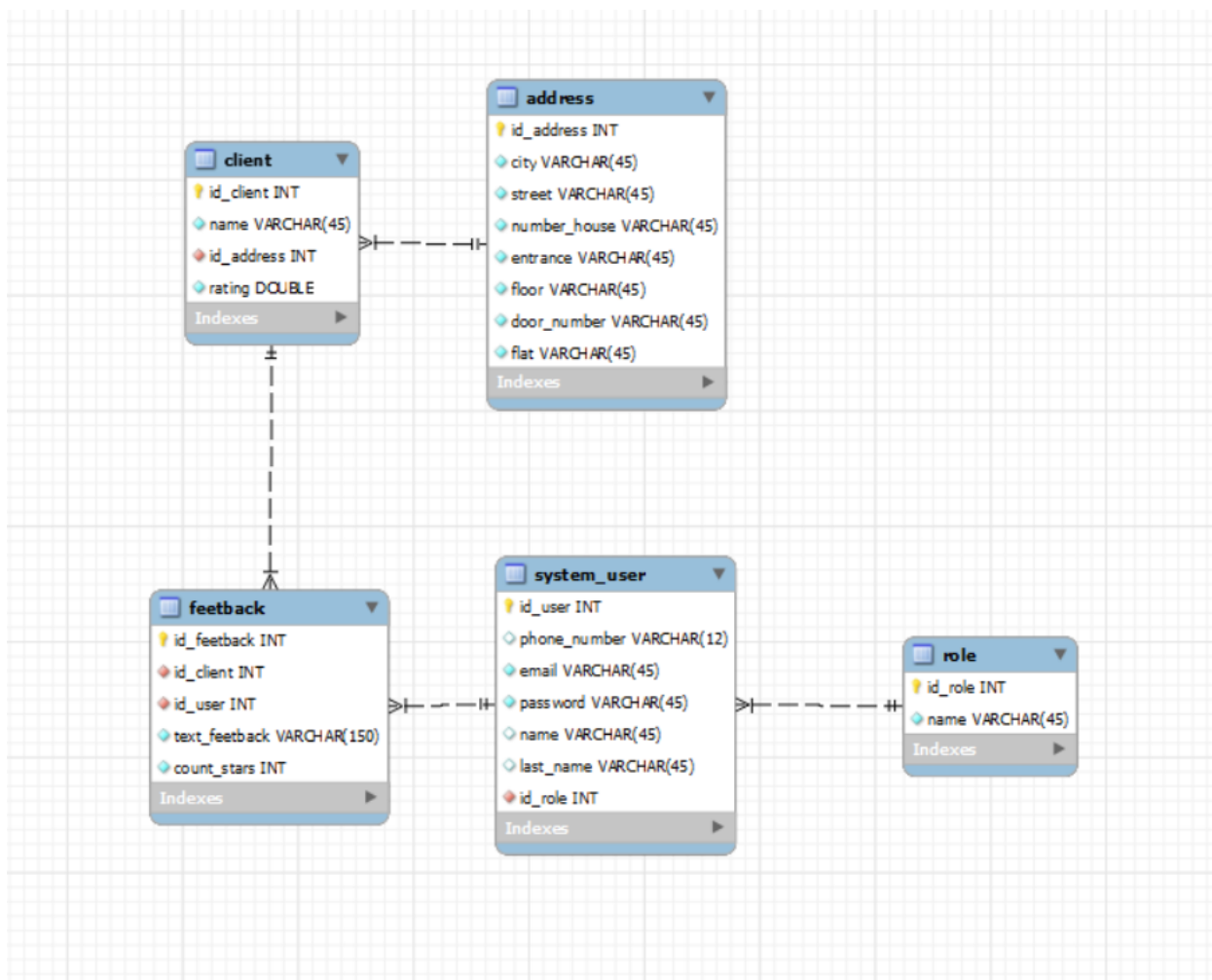


Рисунок 5. ER-диаграмма

В таблице «address» хранится информация об адресе доставки, в которую включены такие поля, как id, город, улица, номер дома, подъезд, этаж, домофон, номер квартиры

В таблице «client» хранится информация об клиенте, в которую включены такие поля, как id, имя клиента, его адрес и рейтинг

В таблице «feedback» хранится информация об отзыве, в которую включены такие поля как id, id клиента, id пользователя, который оставил отзыв, комментарий и количество звёзд.

В таблице «system-user» хранится информация о пользователе, в которую включены такие поля, как его id, номер телефона, электронная почта, пароль, имя, фамилия и id роли.

В таблице «role» хранится информация о ролях, в которую включены такие поля, как id роли и название роли.

4.2 Проектирование интерфейса

Ниже приведены скетчи интерфейса приложения

The image displays five wireframe sketches of application screens, each with a title and a set of form elements:

- Авторизация**
 - Form fields: электронная почта, Пароль
 - Buttons: Войти, Регистрация
- Регистрация**
 - Form fields: электронная почта, Пароль
 - Button: Зарегистрироваться
- Личный кабинет**
 - Button: Редактировать профиль
 - Buttons: Создать отзыв, Написать отзыв, Мои отзывы
- Создание отзыва**
 - Form fields: Имя клиента, Адрес клиента, Комментарий
 - Rating: ☆ ☆ ☆ ☆ ☆
 - Button: Создать
- Мой отзыв**
 - Form fields: Имя клиента, Адрес клиента, Комментарий
 - Rating: ☆ ☆ ☆ ☆ ☆
 - Buttons: Редактировать, Удалить отзыв, Назад

5. Разработка системы

5.1 Выбор средств реализации

В качестве языка программирования был выбран java, использующий Spring-boot. Данный фреймворк был выбран потому, что он удобен для создания web-приложений и для создания API для связи приложения с базой данных.

В качестве базы данных была выбрана MySQL, так как она имеет графический интерфейс, что позволяет быстрее и удобнее работать с базой данных

Для оформления HTML страниц были выбраны собственные css стили, чтобы иметь возможность более гибко настраивать оформление страниц

Для оформления HTML страниц были выбраны собственные js-скрипты, чтобы более гибко настраивать отображение страниц

5.2 Структура проекта

Ниже приведено дерево проекта

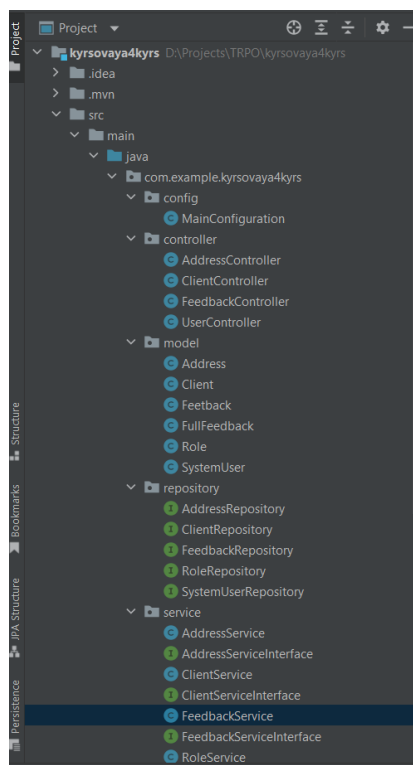


Рисунок 6. Дерево проекта

Ниже приведена UML-диаграмма классов

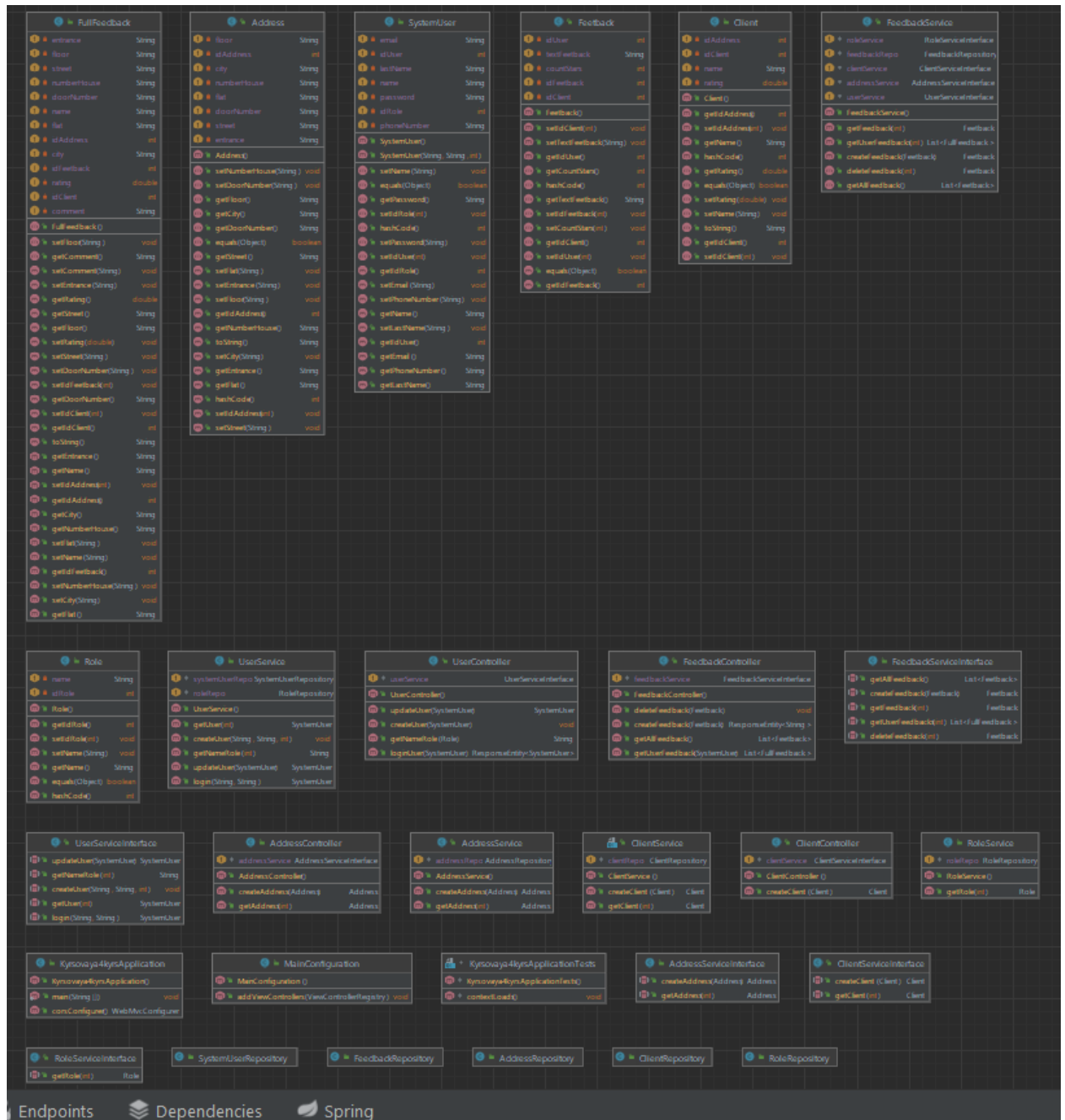


Рисунок 7. UML-диаграмма классов

3.3.1 Листинг и описание классов

Разрабатываемое приложение имеет 4 контроллера:

- 1) «AddressController» - отвечает за все действия, связанные с адресами клиентов

Листинг 1.

```
package com.example.kyrsovaya4kyrs.controller;

import com.example.kyrsovaya4kyrs.model.Address;
import com.example.kyrsovaya4kyrs.service.AddressServiceInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class AddressController {

    @Autowired
    AddressServiceInterface addressService;

    @ResponseBody
    @PostMapping(value = "/createAddress")
    public Address createAddress(@RequestBody Address address) {
        return addressService.createAddress(address);
    }

    @ResponseBody
    @PostMapping(value = "/getAddress")
    public Address getAddress(@RequestBody int idAddress) {
        return addressService.getAddress(idAddress);
    }
}
```

- 2) «ClientController» - отвечает за все действия, связанные с адресами клиентов

Листинг 2

```
package com.example.kyrsovaya4kyrs.controller;

import com.example.kyrsovaya4kyrs.model.Client;
import com.example.kyrsovaya4kyrs.service.ClientServiceInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class ClientController {

    @Autowired
    ClientServiceInterface clientService;

    @ResponseBody
```

```

    @PostMapping(value = "/createClient")
    public Client createClient(@RequestBody Client client){
        return clientService.createClient(client);
    }
}

```

3) «FeedbackController» - отвечает за все действия, связанные с адресами клиентов

Листинг 3

```

package com.example.kyrsovaya4kyrs.controller;

import com.example.kyrsovaya4kyrs.model.Feedback;
import com.example.kyrsovaya4kyrs.model.FullFeedback;
import com.example.kyrsovaya4kyrs.model.SystemUser;
import com.example.kyrsovaya4kyrs.service.FeedbackServiceInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.List;

@Controller
public class FeedbackController {

    @Autowired
    FeedbackServiceInterface feedbackService;

    @ResponseBody
    @PostMapping(value = "/createFeedback")
    public ResponseEntity<String> createFeedback(@RequestBody Feedback
feedback) {
        if (feedbackService.createFeedback(feedback) != null) {
            return new ResponseEntity<> (HttpStatus.OK);
        }
        else {
            return new ResponseEntity<> (HttpStatus.NOT_FOUND);
        }
    }

    @ResponseBody
    @PostMapping(value = "/getAllFeedbacks")
    public List<Feedback> getAllFeedback() {
        return feedbackService.getAllFeedback();
    }

    @ResponseBody
    @PostMapping(value = "/getUserFeedback")
    public List<FullFeedback> getUserFeedback(@RequestBody SystemUser
user) {
        return feedbackService.getUserFeedbacks (user.getIdUser());
    }

    @ResponseBody
    @PostMapping(value = "/deleteFeedback")

```

```

        public void deleteFeedback(@RequestBody Feedback feedback) {
            feedbackService.deleteFeedback(feedback.getIdFeedback());
        }
    }
}

```

4) «UserController» - отвечает за все действия, связанные с адресами клиентов

Листинг 4

```

package com.example.kyrsovaya4kyrs.controller;

import com.example.kyrsovaya4kyrs.model.Role;
import com.example.kyrsovaya4kyrs.model.SystemUser;
import com.example.kyrsovaya4kyrs.service.UserService;
import com.example.kyrsovaya4kyrs.service.UserServiceInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class UserController {

    @Autowired
    UserServiceInterface userService;

    @ResponseBody
    @PostMapping(value = "/createUser", produces =
        MediaType.APPLICATION_JSON_VALUE)
    public void createUser(@RequestBody SystemUser systemUser){
        userService.createUser(systemUser.getEmail(),
            systemUser.getPassword(), systemUser.getIdRole());
    }

    @ResponseBody
    @PostMapping(value = "/loginUser", consumes =
        MediaType.APPLICATION_JSON_VALUE, produces =
        MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<SystemUser> loginUser(@RequestBody SystemUser
        user){
        SystemUser responseUser = userService.login(user.getEmail(),
            user.getPassword());
        if(responseUser != null){
            return new ResponseEntity<>(responseUser, HttpStatus.OK);
        }
        else{
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @ResponseBody
    @PostMapping(value = "/getNameRole")
    public String getNameRole(@RequestBody Role role){
        return userService.getNameRole(role.getIdRole());
    }
}

```

```

        @ResponseBody
        @PostMapping(value = "/updateUser")
        public SystemUser updateUser(@RequestBody SystemUser user) {
            return userService.updateUser(user);
        }
    }
}

```

В приложении находятся 5 классов-сервисов

1) «FeedbackService»

Листинг 5

```

package com.example.kyrsovaya4kyrs.service;

import com.example.kyrsovaya4kyrs.model.*;
import com.example.kyrsovaya4kyrs.repository.FeedbackRepository;
import org.apache.catalina.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class FeedbackService implements FeedbackServiceInterface{

    @Autowired
    FeedbackRepository feedbackRepo;

    @Autowired
    ClientServiceInterface clientService;

    @Autowired
    AddressServiceInterface addressService;

    @Autowired
    UserServiceInterface userService;

    @Autowired
    RoleServiceInterface roleService;

    @Override
    public Feedback createFeedback(Feedback feedback) {
        feedbackRepo.save(feedback);
        return getFeedback(feedback.getIdFeedback());
    }

    @Override
    public Feedback getFeedback(int idFeedback) {
        return feedbackRepo.findById(idFeedback).get();
    }

    @Override
    public List<Feedback> getAllFeedback() {
        return feedbackRepo.findAll();
    }

    @Override
    public List<FullFeedback> getUserFeedbacks(int idUser) {

```



```

List<FullFeedback> userFeedback = new ArrayList<>();
List<Feedback> allFeedback = feedbackRepo.findAll();

SystemUser user = userService.getUser(idUser);
Role role = roleService.getRole(user.getIdRole());

    if(role.getName().equals("Модератор")){
        for (Feedback feedback: allFeedback) {
            Client client =
clientService.getClient(feedback.getIdClient());
            Address address =
addressService.getAddress(client.getIdAddress());
            FullFeedback fullFeedback = new FullFeedback();
            fullFeedback.setIdFeedback(feedback.getIdFeedback());
            fullFeedback.setCity(address.getCity());
            fullFeedback.setStreet(address.getStreet());
            fullFeedback.setNumberHouse(address.getNumberHouse());
            fullFeedback.setEntrance(address.getEntrance());
            fullFeedback.setFloor(address.getFloor());
            fullFeedback.setDoorNumber(address.getDoorNumber());
            fullFeedback.setFlat(address.getFlat());
            fullFeedback.setName(client.getName());
            fullFeedback.setRating(client.getRating());
            fullFeedback.setComment(feedback.getTextFeedback());
            fullFeedback.setIdAddress(address.getIdAddress());
            fullFeedback.setIdClient(client.getIdClient());

            userFeedback.add(fullFeedback);
        }
    }else{
        for (Feedback feedback: allFeedback) {
            if(feedback.getIdUser() == idUser){
                Client client =
clientService.getClient(feedback.getIdClient());
                Address address =
addressService.getAddress(client.getIdAddress());
                FullFeedback fullFeedback = new FullFeedback();
                fullFeedback.setIdFeedback(feedback.getIdFeedback());
                fullFeedback.setCity(address.getCity());
                fullFeedback.setStreet(address.getStreet());
                fullFeedback.setNumberHouse(address.getNumberHouse());
                fullFeedback.setEntrance(address.getEntrance());
                fullFeedback.setFloor(address.getFloor());
                fullFeedback.setDoorNumber(address.getDoorNumber());
                fullFeedback.setFlat(address.getFlat());
                fullFeedback.setName(client.getName());
                fullFeedback.setRating(client.getRating());
                fullFeedback.setComment(feedback.getTextFeedback());
                fullFeedback.setIdAddress(address.getIdAddress());
                fullFeedback.setIdClient(client.getIdClient());

                userFeedback.add(fullFeedback);
            }
        }
    }

    return userFeedback;
}

@Override
public Feedback deleteFeedback(int idFeedback) {
    feedbackRepo.delete(getFeedback(idFeedback));
}

```

```
    return getFeedback(idFeedback) ;  
  }  
}
```

3.3.2 Интерфейс приложения

Ниже приведен интерфейс страницы профиля пользователя, на которой мы можем выбрать дальнейшие действия



Рисунок 8. страница Profile

Ниже приведен интерфейс страница создания отзыва, на которой мы можем заполнить поля с информацией о адресе доставки, комментария и поставить кол-во звёзд

Создание отзыва

Имя клиента

Адрес клиента

Подъезд

Этаж

Домофон

Номер квартиры

Комментарий




Рисунок 9. страница createFeedback