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

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)

Потенциал внедерения

- 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

Краткое ТЗ

Назначение системы:

- Данное приложение предназначена для создания и просмотра отзывов о клиенте

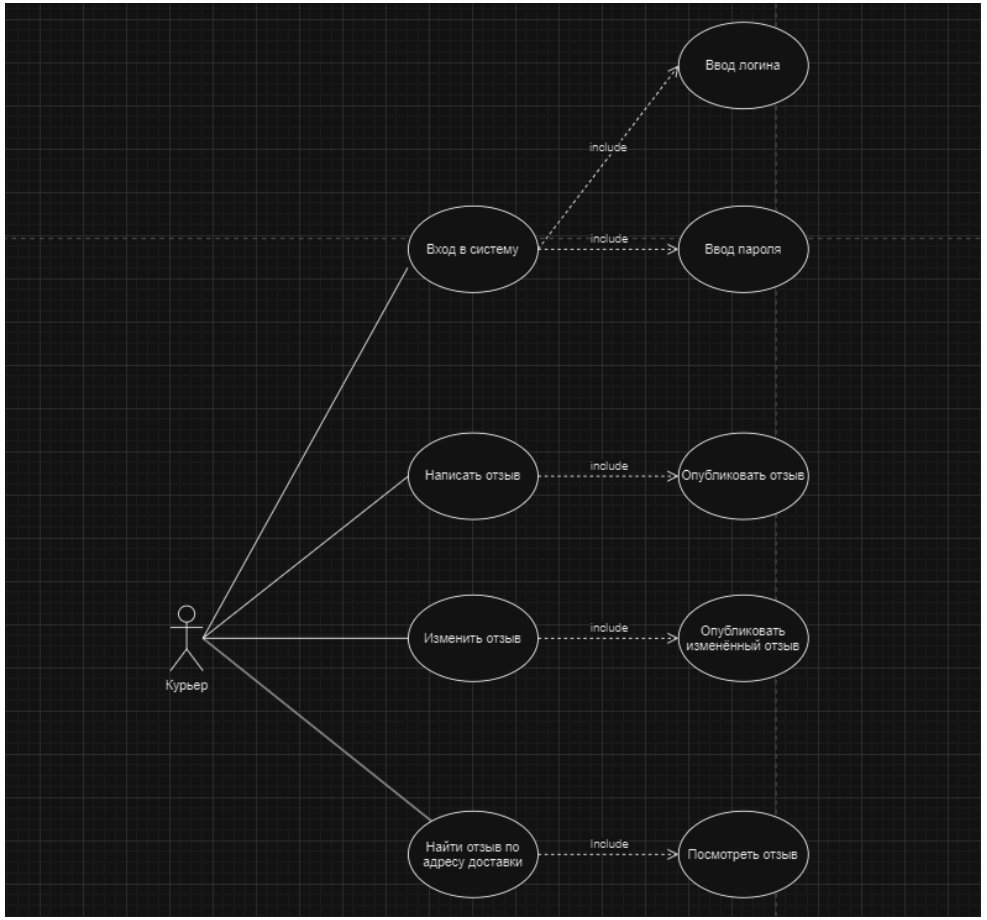
Цели создания проекта:

- Предоставление возможности оставлять отзыв о клиенте с подробной информацией о нём

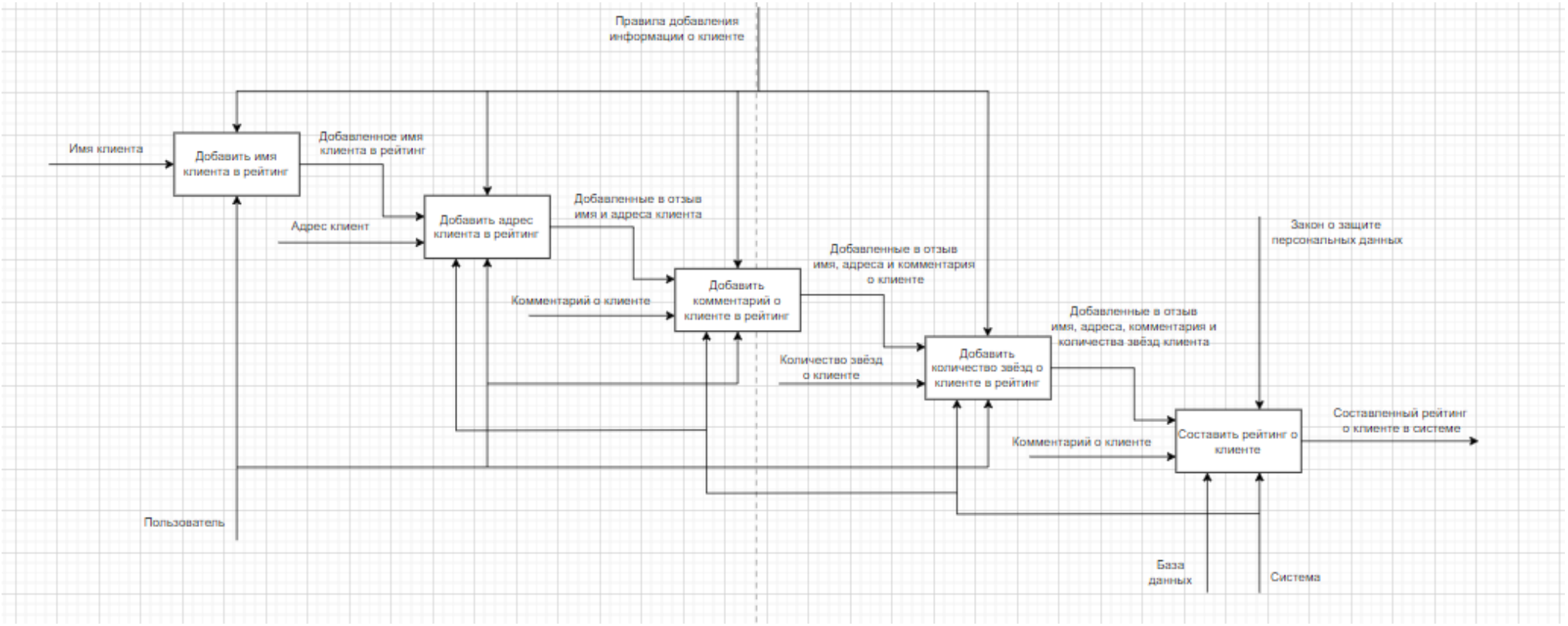
- Предоставление возможности просматривать отзывы, чтобы получить подробную информацию о клиенте

Моделирование системы

IDEF0

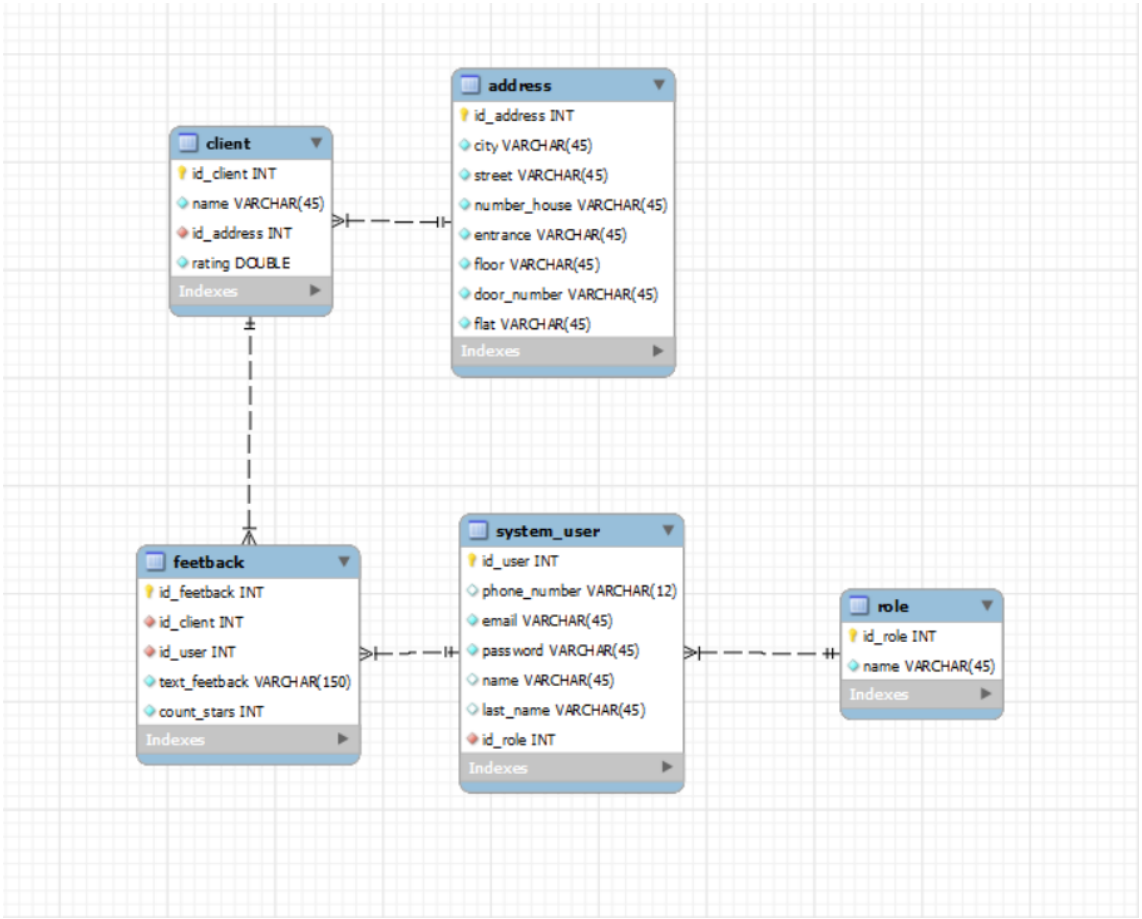


IDEF0


Правила добавления информации о клиенте
Имя клиента
Добавить имя клиента в рейтинг
Добавленное имя клиента в рейтинг
Адрес клиент
Добавить адрес клиента в рейтинг
Добавленные в отзыв имя и адреса клиента
Комментарий о клиенте
Добавить комментарий о клиенте в рейтинг
Добавленные в отзыв имя, адреса и комментария о клиенте
Количество звёзд о клиенте
Добавить количество звёзд о клиенте в рейтинг
Добавленные в отзыв имя, адреса, комментария и количества звёзд клиента
Закон о защите персональных данных
Комментарий о клиенте
Составить рейтинг о клиенте
Составленный рейтинг о клиенте в системе
Пользователь
База данных
Система

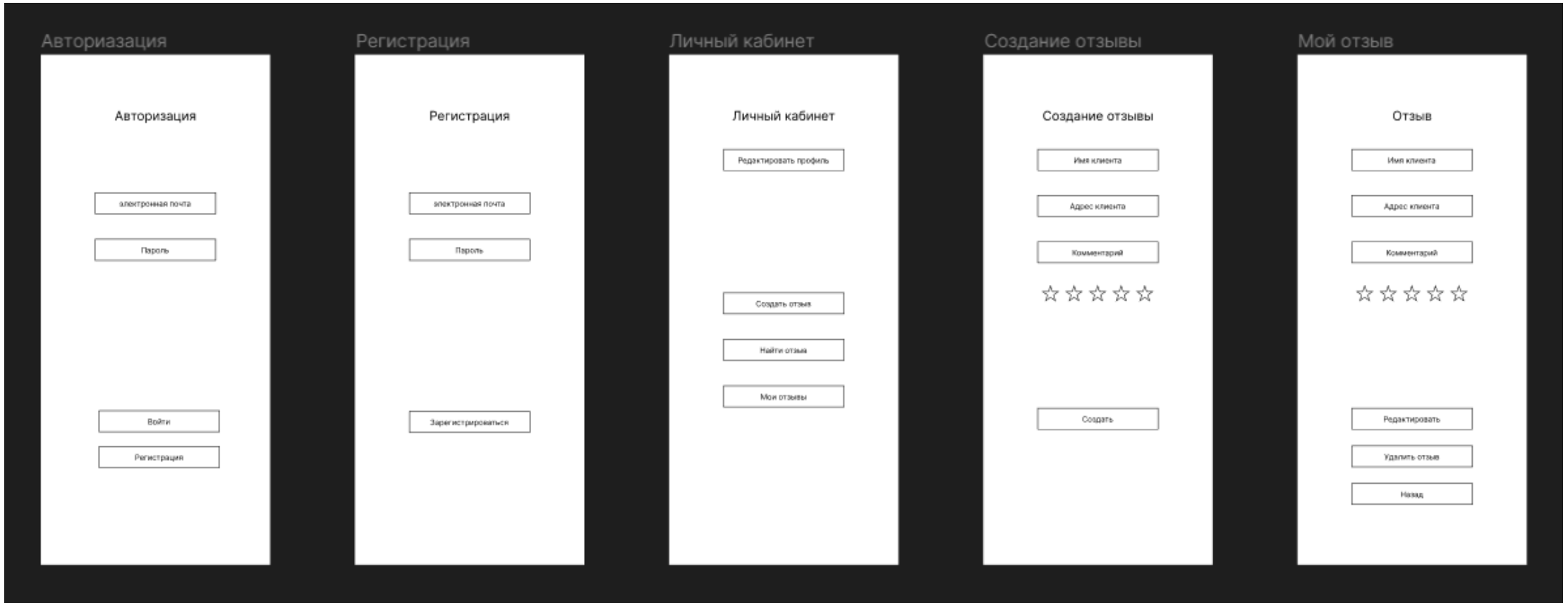
Проектирование БД

ER-диаграмма



Проектирование интерфейса

Скетчи интерфейса



Выбор средств реализации

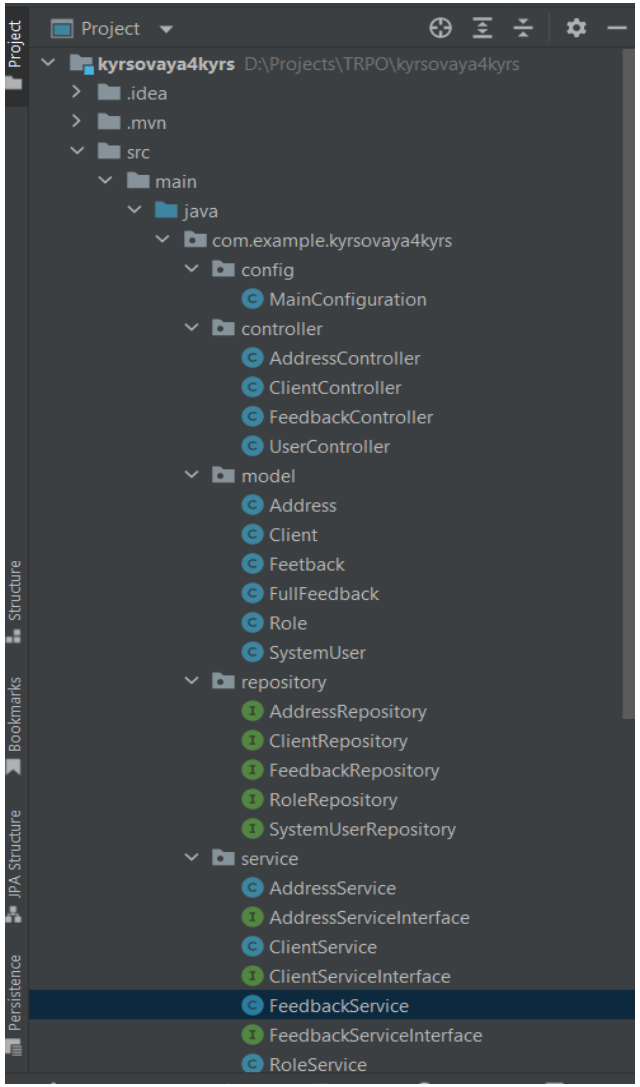
- В качестве языка программирования был выбран java, использующий Sping-boot. Данный фреймворк был выбран потому, что он удобен для создания web-приложений и для создания API для связи приложения с базой данных.

- В качестве базы данных выла выбрана MySQL, так как она имеет графический интерфейс, что позволяет быстрее и удобнее 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

Дерево проекта



UML-диаграмма классов

Листинг «FeedbackController»

```java
@Controller
public class FeedbackController {

    @Autowired
    FeedbackServiceInterface feedbackService;

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

    @ResponseBody
    @PostMapping(value = "/getAllFeedbacks")
    public List<Feetback> 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 Feetback feetback) {
        feedbackService.deleteFeedback(feetback.getIdFeetback());
    }
}
```

Листинг «FeedbackService»

```java
@Service
public class FeedbackService implements FeedbackServiceInterface{

    @Autowired
    FeedbackRepository feedbackRepo;

    @Autowired
    ClientServiceInterface clientService;

    @Autowired
    AddressServiceInterface addressService;

    @Autowired
    UserServiceInterface userService;

    @Autowired
    RoleServiceInterface roleService;

    @Override
    public Feetback createFeedback(Feetback feetback) {
        feedbackRepo.save(feetback);
        return getFeedback(feetback.getIdFeetback());
    }

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

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

```java
    @Override
    public List<FullFeedback> getUserFeedbacks(int idUser) {
        List<FullFeedback> userFeedback = new ArrayList<>();
        List<Feetback> allFeedback = feedbackRepo.findAll();

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

        if(role.getName().equals("Модератор")){
            for (Feetback feedback: allFeedback) {
                Client client = clientService.getClient(feedback.getIdClient());
                Address address = addressService.getAddress(client.getIdAddress());
                FullFeedback fullFeedback = new FullFeedback();
                fullFeedback.setIdFeetback(feedback.getIdFeetback());
                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.getTextFeetback());
                fullFeedback.setIdAddress(address.getIdAddress());
                fullFeedback.setIdClient(client.getIdClient());

                userFeedback.add(fullFeedback);
            }
        }else{
```

```java
            for (Feetback feedback: allFeedback) {
                if(feedback.getIdUser() == idUser){
                    Client client = clientService.getClient(feedback.getIdClient());
                    Address address = addressService.getAddress(client.getIdAddress());
                    FullFeedback fullFeedback = new FullFeedback();
                    fullFeedback.setIdFeetback(feedback.getIdFeetback());
                    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.getTextFeetback());
                    fullFeedback.setIdAddress(address.getIdAddress());
                    fullFeedback.setIdClient(client.getIdClient());

                    userFeedback.add(fullFeedback);
                }
            }
        }

        return userFeedback;
    }

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

        return getFeedback(idFeedback);
    }
}
```