МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Звіт

до лабораторної роботи №2

з курсу «Аналіз та рефакторінг коду програмного забезпечення»

На тему

«Створення серверної частини»

Перевірив:

Сокорчук І. П.

Виконав:

ст. гр. ПІ-14-4

Доротенко О.В

ХНУРЕ 2017

Основными понятиями предметной области есть такие элементы как пользователь, заказ, комментарий и т.д. Сама система содержит в себе записи о всех заказах, пользователях, их комментариях как к заказам, так и к другим пользователям, а также теги для характеризации заказов.

База данных отображает структуру системы и содержит в себе такие документы с информацией:

Документ users:

1) id – уникальный номер пользователя;

2) username – логин пользователя;

3) password – уникальный код доступа пользователя;

4) role – содержит запись для идентификации пользователя в системе;

5) fio – ФИО пользователя;

6) registerDate – дата регистрации;

7) bornDate ­– дата рождения;

8) rating – рейтинг пользователя;

9) commentaries ­– комментарии адресованые этому пользователю;

10) authorCommentaries – комментарии от этого пользователя другим;

11) orderCommentaries – комментарии от этого пользователя к заказам.

Документ orders:

1. id – уникальный номер заказа;
2. name – название заказа;
3. description – описание;
4. createDate – дата создания;
5. tags – теги данного заказа;
6. author – автор данного заказа;
7. isCompleted – завершен ли;
8. executer – пользователь, который выполняет данный заказ;
9. commentaries – комментарии к этому заказу;
10. count – цена этого заказа;

Документ tags:

1. id – уникальный номер тега;
2. name – название тега.

Документ orderCommentaries:

1. id ­– уникальный номер комментария к заказу;
2. text – текст комментария;
3. createDate – дата создания комментария;
4. author – автор комментария;
5. rating – рейтинг комментария;
6. order – цель(заказ) комментария.

Документ userCommentaries:

1. id – уникальный номер комментария к пользователю;
2. text – текст комментария;
3. createDate – дата создания комментария;
4. author – автор комментария;
5. rating – рейтинг комментария;
6. user – цель(пользователь) комментария.

На рисунке 1 приведена схема базы данных.

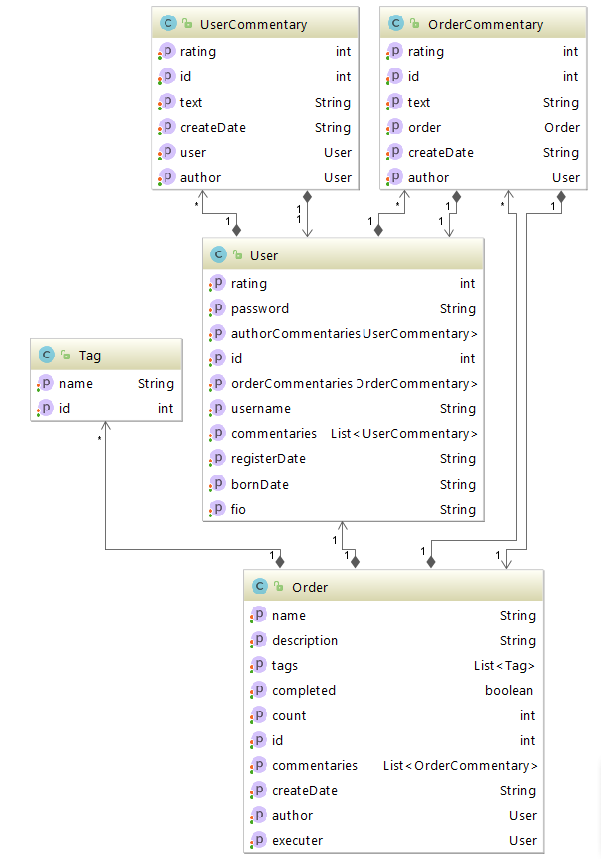


Рисунок 1 – Схема базы данных(NoSql)

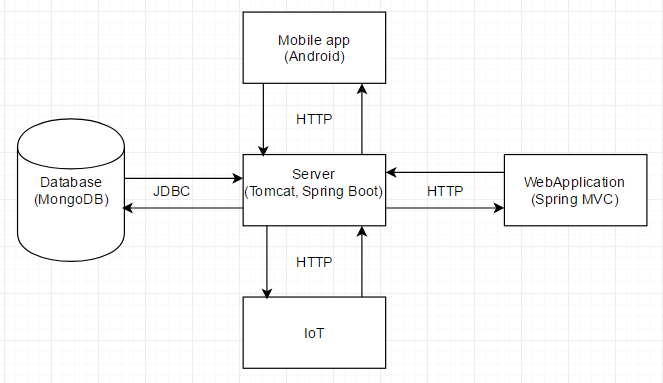
Система состоит из четырех модулей: веб-приложение, бекенд-сервер, IoT-модуль и мобильное приложение. Общая схема взаимодействия компонентов представлена на рисунке 2.

Рисунок 2 - Схема архитектуры приложения

IoT, Веб-приложение и мобильное приложение будут связываться с сервером посредством HTTP протокола, с помощью класса RestTemplate. IoT устройство будет представлено датчиком открытия и закрытия дверей, для отслеживания заказов на дому пользователя. Мобильное приложение будет реализовано для мобильных устройств Android с помощью Android Studio.

Код системы предоставлен ниже:

Класс для роботы с документом пользователей

@Component  
**public class** UserDao {  
 @Autowired  
 **private** MongoOperations **mongoOperations**;  
  
 **public void** save(User u){  
 **mongoOperations**.save(u);  
 }  
  
 **public** List<User> findAll(){  
 **return mongoOperations**.findAll(User.**class**);  
 }  
  
 **public** User findById(**int** id){  
 **return mongoOperations**.findOne(**new** Query().addCriteria(Criteria.*where*(**"id"**).is(id)), User.**class**);  
 }  
  
 **public void** update(**int** id, User user){  
 Update update = **new** Update();  
 update.set(**"username"**, user.getUsername());  
 update.set(**"password"**, user.getPassword());  
 update.set(**"fio"**, user.getFio());  
 update.set(**"borndate"**, user.getBornDate());  
 update.set(**"rating"**, user.getRating());  
 update.set(**"commentaries"**, user.getCommentaries());  
 update.set(**"userCommentaries"**, user.getAuthorCommentaries());  
 update.set(**"orderCommentaries"**, user.getOrderCommentaries());  
 **mongoOperations**.updateFirst(**new** Query()  
 .addCriteria(Criteria.*where*(**"id"**).is(id)), update, User.**class**);  
 }  
  
}

Главный контроллер сервера:

@RestController  
**public class** HomeController {  
 @Autowired  
 UserDao **userDao**;  
 @Autowired  
 OrderDao **orderDao**;  
 @Autowired  
 TagDao **tagDao**;  
 @Autowired  
 UserCommentaryDao **userCommentaryDao**;  
 @Autowired  
 OrderCommentaryDAO **orderCommentaryDAO**;  
 @Autowired  
 DbEmulator **dbEmulator**;  
 @RequestMapping(**"/emul"**)  
 **public void** emulate(){  
 **dbEmulator**.emulate();  
 }  
 @RequestMapping(**"/getUsers"**)  
 **public** List<User> getAllUsers(){  
 **return userDao**.findAll();  
  
 }  
 @RequestMapping(**"/getOrders"**)  
 **public** List<Order> getAllOrders(){  
 **return orderDao**.findAll();  
 }  
 @RequestMapping(**"/getUserById"**)  
 **public** User getUserById(@RequestParam(**"id"**) **int** id){  
 **return userDao**.findById(id);  
 }  
 @RequestMapping(**"/getOrderById"**)  
 **public** Order getOrderById(@RequestParam(**"id"**) **int** id){  
 **return orderDao**.findById(id);  
 }  
 @RequestMapping(**"/getTags"**)  
 **public** List<Tag> getAllTags(){  
 **return tagDao**.findAll();  
 }  
  
 @RequestMapping(**"/getOrdersByTag"**)  
 **public** List<Order> getOrdersByTag(String tagname){  
 List<Order> result = **new** ArrayList<>();  
 **for**(Order order : **orderDao**.findAll()){  
 **for**(Tag tag : order.getTags()){  
 **if**(tag.getName().equals(tagname)){  
 result.add(order);  
 **break**;  
 }  
 }  
 }  
 **return** result;  
 }  
  
 @RequestMapping(value = **"/saveUser"**, method = RequestMethod.***POST***)  
 **public** Boolean saveUser(@RequestBody User user){  
 **userDao**.save(user);  
 **return true**;  
 }  
  
 @RequestMapping(**"/getUserCommentaries"**)  
 **public** List<UserCommentary> getUserCommentaries(){  
 **return userCommentaryDao**.findAll();  
 }  
  
 @RequestMapping(**"/getOrderCommentaries"**)  
 **public** List<OrderCommentary> getOrderCommentaries(){  
 **return orderCommentaryDAO**.findAll();  
 }  
  
 @RequestMapping(**"/getUserCommentaryByAuthor"**)  
 **public** List<UserCommentary> getUserCommentaryByAuthor(@RequestParam(**"id"**) **int** id){  
 **return** getUserById(id).getAuthorCommentaries();  
 }  
  
 @RequestMapping(**"/getUserCommentaryByTarget"**)  
 **public** List<UserCommentary> getUserCommentaryByTarget(@RequestParam(**"id"**) **int** id){  
 **return** getUserById(id).getCommentaries();  
 }  
  
 @RequestMapping(**"/getOrderCommentaryByAuthor"**)  
 **public** List<OrderCommentary> getOrderCommentaryByAuthor(@RequestParam(**"id"**) **int** id){  
 **return** getUserById(id).getOrderCommentaries();  
 }  
  
 @RequestMapping(**"/getOrderCommentaryByTarget"**)  
 **public** List<OrderCommentary> getOrderCommentaryByTarget(@RequestParam(**"id"**) **int** id){  
 **return** getOrderById(id).getCommentaries();  
 }  
  
  
  
}