МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**«Технологии Программирования»**

Выполнил студент группы M32081

Найман Егор Борисович

Санкт-Петербург

2022

**Работа № 2 cats-and-owners (hibernate)**

**Задание:**

Нужно написать сервис по учету котиков и их владельцев.

*Существующая информация о котиках:*

- Имя

- Дата рождения

- Порода

- Цвет (один из заранее заданных вариантов)

- Хозяин

- Список котиков, с которыми дружит этот котик (из представленных в базе)

*Существующая информация о хозяевах:*

- Имя

- Дата рождения

- Список котиков

Сервис должен реализовывать архитектуру controller-service-dao.

Вся информация хранится в БД PostgreSQL. Для связи с БД должен использоваться Hibernate.

Проект должен собираться с помощью Maven или Gradle (на выбор студента). Слой доступа к данным и сервисный слой должны являться двумя разными модулями Maven/Gradle. При этом проект должен полностью собираться одной командой.

При тестировании рекомендуется использовать Mockito, чтобы избежать подключения к реальным базам данных. Фреймворк для тестирования рекомендуется Junit 5.

В данной лабораторной нельзя использовать Spring или подобные ему фреймворки.

**Решение:**

С помощью hibernate.cfg.xml и HibernateSessionFactoryUtil выполняем подключение к базе данных.

Для работы с базой данных создаём 3 таблицы: owners, cats и friends.

CREATE TABLE cats  
(  
 cat\_id SERIAL PRIMARY KEY,  
 owner\_id INTEGER,  
 cat\_colour CHARACTER VARYING(50),  
 cat\_name CHARACTER VARYING(50),  
 cat\_dob DATE  
);  
  
CREATE TABLE owners  
(  
 owner\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER,  
 owner\_name CHARACTER VARYING(50),  
 owner\_dob DATE  
);  
  
CREATE TABLE friends  
(  
 friends\_id SERIAL PRIMARY KEY,  
 friend\_first INTEGER,  
 friend\_second INTEGER  
);

ALTER TABLE cats  
 ADD CONSTRAINT UK\_l329ab0g4c1t78onljnxmbnp6  
 UNIQUE (cat\_id);  
  
ALTER TABLE cats  
 ADD CONSTRAINT FKmw13yfsjypiiq0i1osdkaeqpg  
 FOREIGN KEY (owner\_id) REFERENCES owners ON DELETE CASCADE;  
  
ALTER TABLE friends  
 ADD CONSTRAINT UK\_l329ab0g4c1t78onljnxmbnp1  
 UNIQUE (friends\_id);  
  
ALTER TABLE friends  
 ADD CONSTRAINT FKmw10yfsjypiiq0i1osdkaeqpi  
 FOREIGN KEY (friend\_first) REFERENCES cats ON DELETE CASCADE;  
  
ALTER TABLE friends  
 ADD CONSTRAINT FKmw11yfsjypiiq0i1osdkaeqpo  
 FOREIGN KEY (friend\_second) REFERENCES cats ON DELETE CASCADE;

Класс OwnerService и CatServise служат управляющей системой, классы CatDAO, FriendsDAO, OwnersDAO, служат интерфейсами для классов реализаций (с приставкой I) выполняющих работу с базой данных, классы Cats, Friends, Owners реализуют сущности базы данных.

Тесты содержат в себе:

- Добавление владельца с последующим удалением

- Добавление котика с последующим удалением

- Добавление друзей с последующим удалением

Проверка построенных связей на уровне дб, посредством каскадного удаления

**Работа № 3 - 5 cats-and-owners**

**Задание:**

**(Spring)**

К созданному в прошлой лабораторной сервису добавляется Spring.

Сервис должен предоставлять http интерфейс (REST API) для получения информации о конкретных котиках и владельцах и для получения фильтрованной информации (например, получить всех рыжих котиков)

Внимание: недопустимо отдавать через HTTP интерфейс сущности JPA. Рекомендуется создать отдельные оберточные классы.

Сервисы и dao должны превратиться в Spring Bean’ы с использованием Dependency Injection (Autowired). Dao при этом наследуют JpaRepository и имеет шаблонные Spring Data JPA методы: https://www.baeldung.com/spring-data-repositories#repositories

При сдаче лабораторной нужно будет показать работоспособность endpoint’ов через http запросы (рекомендуется Postman).

**(Security)**

Владельцы недовольны, что информацию о котиках может получить кто угодно. В этой лабораторной мы добавим авторизацию к сервису.

Добавляется роль администратора. Он имеет доступ ко всем методам и может создавать новых пользователей. Пользователь связан с владельцем в соотношении 1:1.

Методы по получению информации и котиках и владельцах должны быть защищены Spring Security. Доступ к соответствующим endpoint’ам имеют только владельцы котиков и администраторы. Доступ к методам для фильтрации имеют все авторизованные пользователи, но на выходе получают только данные о своих котиках.

Внимание: эндпоинты, созданные на предыдущем этапе, не должны быть удалены.

**(Microservices)**

Бизнес прочитал статью о том, что микросервисы это круто и попросил нас разбить программу на микросервисы.

Из созданного приложения выделяются три микросервиса:

* Микросервис доступа к котикам
* Микросервис доступа к владельцам
* Микросервис с внешними интерфейсами.

Все они являются разными приложениями.

Все созданные ранее эндпоинты и авторизация переезжает на третий микросервис.

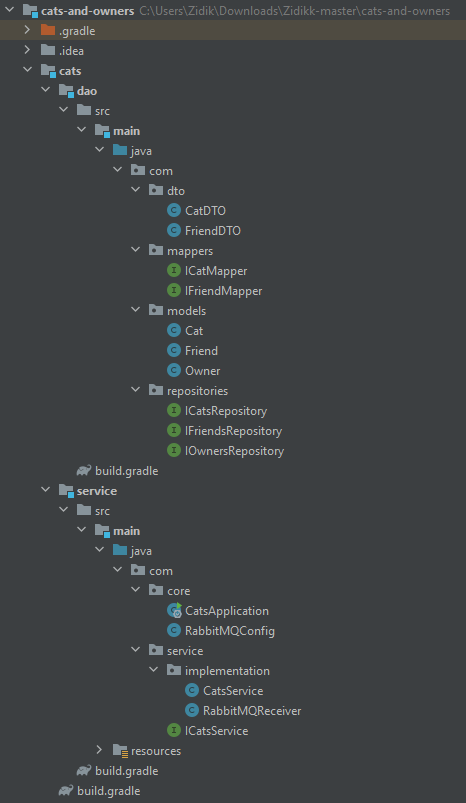
Общение между микросервисами происходит посредством RabbitMQ/Kafka (на выбор студента).

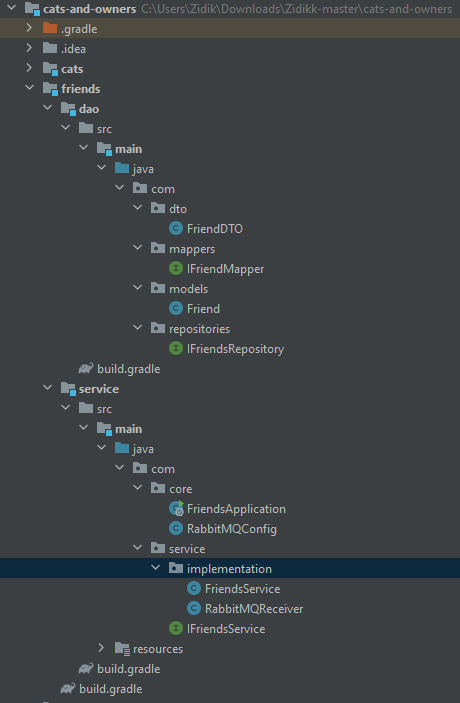
Сервисы доступа к котикам и доступа к владельцам могут либо быть подключены к одной БД, либо иметь разные БД. Во втором случае недопустимо делать один запрос на получение данных из двух БД, запроса должно быть два (по одному в каждую).

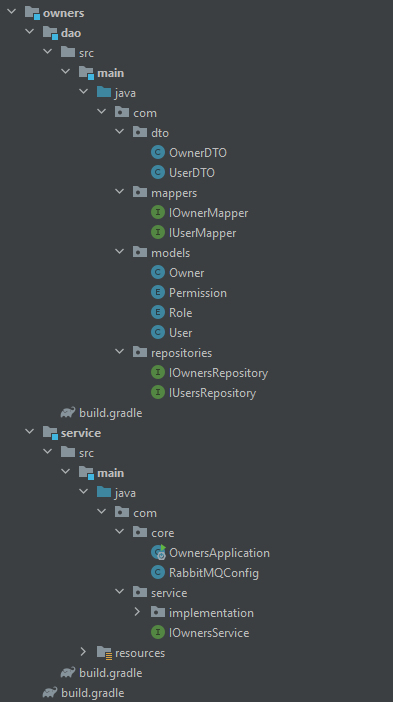
Внимание: недопустимо передавать через RabbitMQ/Kafka JPA сущности. Рекомендуется создать отдельные оберточные классы.

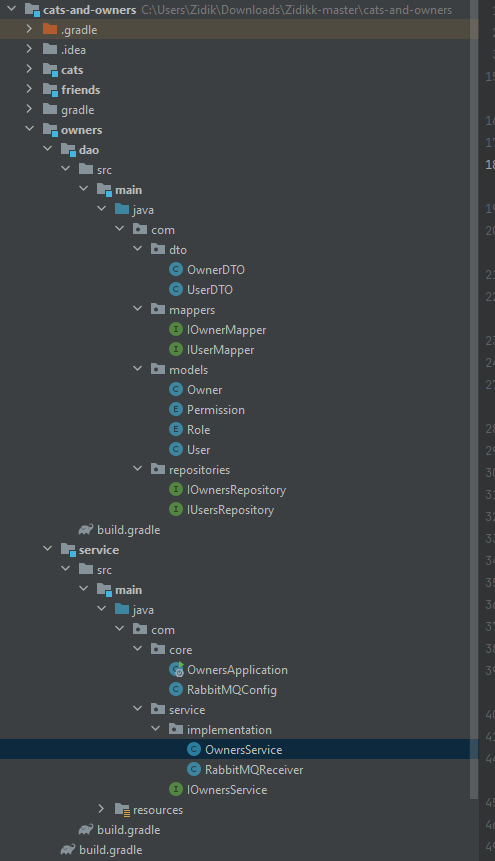
Рекомендуется выделить модуль с JPA сущностями в отдельный подключаемый модуль.

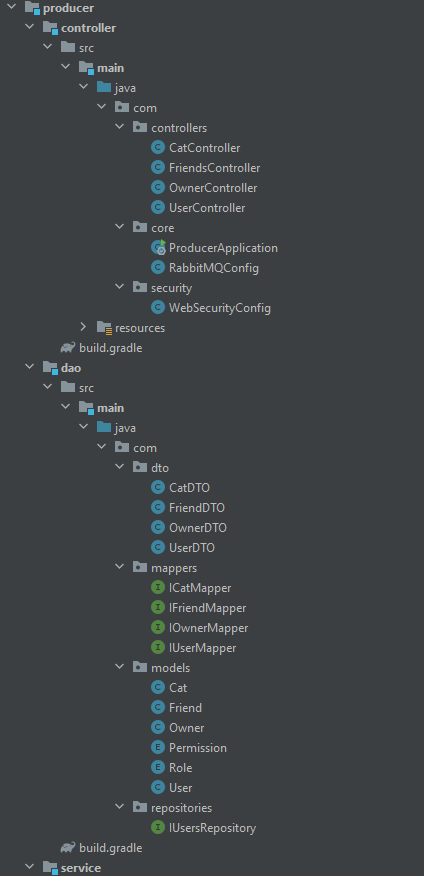
**Структура кода программы:**

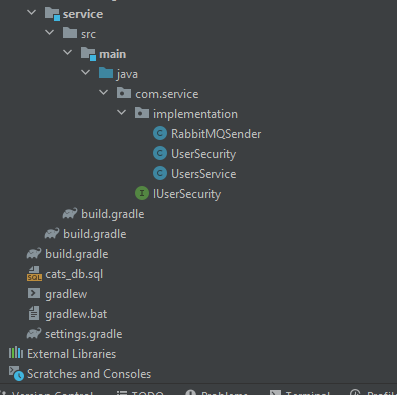
****

****

****

****





**Решение:**

Для работы с базой данных создаём 4 таблицы: owners, cats и friends, users.

CREATE TABLE cats  
(  
 cat\_id SERIAL PRIMARY KEY,  
 owner\_id INTEGER,  
 cat\_colour CHARACTER VARYING(50),  
 cat\_name CHARACTER VARYING(50),  
 cat\_dob DATE  
);  
  
CREATE TABLE owners  
(  
 owner\_id SERIAL PRIMARY KEY,  
 user\_id INTEGER,  
 owner\_name CHARACTER VARYING(50),  
 owner\_dob DATE  
);  
  
CREATE TABLE friends  
(  
 friends\_id SERIAL PRIMARY KEY,  
 friend\_first INTEGER,  
 friend\_second INTEGER  
);  
  
CREATE TABLE users  
(  
 user\_id SERIAL PRIMARY KEY,  
 owner\_id INTEGER,  
 user\_password CHARACTER VARYING(50),  
 user\_name CHARACTER VARYING(50),  
 user\_role CHARACTER VARYING(50)  
);  
  
ALTER TABLE cats  
 ADD CONSTRAINT UK\_l329ab0g4c1t78onljnxmbnp6  
 UNIQUE (cat\_id);  
  
ALTER TABLE cats  
 ADD CONSTRAINT FKmw13yfsjypiiq0i1osdkaeqpg  
 FOREIGN KEY (owner\_id) REFERENCES owners ON DELETE CASCADE;  
  
ALTER TABLE friends  
 ADD CONSTRAINT UK\_l329ab0g4c1t78onljnxmbnp1  
 UNIQUE (friends\_id);  
  
ALTER TABLE friends  
 ADD CONSTRAINT FKmw10yfsjypiiq0i1osdkaeqpi  
 FOREIGN KEY (friend\_first) REFERENCES cats ON DELETE CASCADE;  
  
ALTER TABLE friends  
 ADD CONSTRAINT FKmw11yfsjypiiq0i1osdkaeqpo  
 FOREIGN KEY (friend\_second) REFERENCES cats ON DELETE CASCADE;  
  
  
ALTER TABLE users  
 ADD CONSTRAINT FKeotuev8ja8v0sdh29dynqj05p  
 FOREIGN KEY (owner\_id) REFERENCES owners on DELETE CASCADE;

Проект реализует микросервисную архитектуру, в которой общение происходит посредством работы брокера сообщений RabbitMQ, каждый микросервис представляет из себя многомодульное приложение, в проекте присутствует 4 микросервиса: cats, friends, owners и producer.

***producer:***

producer является главным микросервисом генерирующим и обрабатывающим запросы, после обработки запроса producer передает сообщение на сервер, где его перехватывает другой микросервис, обрабатывает и возвращает результат обратно producer-у, продюсер выводит полученный результат пользователю.

producer включает в себя три модуля: controller, service, dao

* ***controller***

Здесь хранятся контроллеры, предназначенный для обработки запросов от клиента и возвращения результатов.

Также здесь находиться ядро приложения, и две конфигурации для настройки безопасности приложения и настройки брокера сообщений.

* ***service***

Здесь расположены сервисы для отправки сообщений, для работы с безопасностью и для взаимодействия с базой данных пользователей.

* ***dao***

Здесь находятся сущности баз данных, объекты передачи данных, маппер и интерфейс для доступа к данным.

***owners:***

owners является consumer-ом, он получает сообщение от producer-a, обрабатывает его и возвращает результат producer-y.

owners включает в себя три модуля: service, dao

* ***service***

Здесь расположены сервисы для получения и обработки сообщений, для взаимодействия с базой данных пользователей и владельцев котов. Также здесь расположено ядро приложения и конфигурация для настройки брокера сообщений.

* ***dao***

Здесь находятся сущности баз данных, объекты передачи данных, маппер и интерфейсы для доступа к данным

***cats:***

cats является consumer-ом, он получает сообщение от producer-a, обрабатывает его и возвращает результат producer-y.

cats включает в себя три модуля: service, dao

* ***service***

Здесь расположены сервисы для получения и обработки сообщений, для взаимодействия с базой данных котов. Также здесь расположено ядро приложения и конфигурация для настройки брокера сообщений.

* ***dao***

Здесь находятся сущности баз данных, объекты передачи данных, маппер и интерфейс для доступа к данным

***friends:***

friends является consumer-ом, он получает сообщение от producer-a, обрабатывает его и возвращает результат producer-y.

friends включает в себя три модуля: service, dao

* ***service***

Здесь расположены сервисы для получения и обработки сообщений, для взаимодействия с базой данных друзей. Также здесь расположено ядро приложения и конфигурация для настройки брокера сообщений.

* ***dao***

Здесь находятся сущности баз данных, объекты передачи данных, маппер и интерфейс для доступа к данным