**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №4

по дисциплине «**Технологии программирования**»

Автор: Юрпалов Сергей Николаевич

Факультет: ИТиП

Группа: М32051



Санкт-Петербург 2022

Описание:

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.

Исходный код:

Изменен файл kotiki-java/database-access/src/main/java/dao/CatDao.java:

package dao;

import models.Cat;

+import models.Owner;

import models.enums.CatColor;

import org.springframework.data.jpa.repository.JpaRepository;

@@ -8,8 +9,14 @@

import java.util.List;

public interface CatDao extends JpaRepository<Cat, Integer> {

+

+ List<Cat> findByOwner(Owner owner);

List<Cat> findByName(String name);

+ List<Cat> findByNameAndOwner(String name, Owner owner);

List<Cat> findByBirthDate(LocalDate birthDate);

+ List<Cat> findByBirthDateAndOwner(LocalDate birthDate, Owner owner);

List<Cat> findByBreed(String breed);

+ List<Cat> findByBreedAndOwner(String breed, Owner owner);

List<Cat> findByColor(CatColor color);

+ List<Cat> findByColorAndOwner(CatColor color, Owner owner);

}

Изменен файл kotiki-java/database-access/src/main/java/dao/OwnerDao.java:

import java.util.List;

public interface OwnerDao extends JpaRepository<Owner, Integer> {

+

List<Owner> findByName(String name);

+ List<Owner> findByNameAndId(String name, int id);

List<Owner> findByBirthDate(LocalDate birthDate);

+ List<Owner> findByBirthDateAndId(LocalDate birthDate, int id);

}

Добавлен файл kotiki-java/database-access/src/main/java/dao/UserDao.java:

package dao;

import models.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserDao extends JpaRepository<User, Integer> {

User getByUsername(String username);

}

Удален файл kotiki-java/database-access/src/main/java/dao/interfaces/AbstractDao.java

Изменен файл kotiki-java/database-access/src/main/java/models/Cat.java:

package models;

import models.enums.CatColor;

+import tools.CatsException;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.\*;

@Entity

-@Table(name = "cats", schema = "public", catalog = "postgres")

+@Table(name = "cats")

public class Cat {

@GeneratedValue(strategy = GenerationType.IDENTITY)

public void addFriend(Cat cat) throws CatsException {

if (equals(cat)) {

throw new CatsException("Cat cannot make friends with himself");

}

-

+

friendOf.add(cat);

friends.add(cat);

cat.getFriends().add(this);

Удален файл kotiki-java/database-access/src/main/java/models/CatColors.java

Изменен файл kotiki-java/database-access/src/main/java/models/Owner.java:

import java.util.Set;

@Entity

-@Table(name = "owners", schema = "public", catalog = "postgres")

+@Table(name = "owners")

public class Owner {

@GeneratedValue(strategy = GenerationType.IDENTITY)

public class Owner {

private int id;

@Basic

- @Column(name = "name")

+ @Column(name = "name", nullable = false)

private String name;

@Basic

- @Column(name = "birth\_date")

+ @Column(name = "birth\_date", nullable = false)

private LocalDate birthDate;

@OneToMany(mappedBy = "owner", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

private Set<Cat> cats = new HashSet<>();

+ @OneToOne(mappedBy = "owner", cascade = CascadeType.MERGE)

+ private User user;

+

public Owner() {}

public Owner(String name, LocalDate birthDate) {

Добавлен файл kotiki-java/database-access/src/main/java/models/User.java:

package models;

import models.enums.Authority;

import javax.persistence.\*;

@Entity

@Table(name = "users")

public class User {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id", nullable = false)

private int id;

@Basic

@Column(name = "username", unique = true, nullable = false)

private String username;

@Basic

@Column(name = "password", nullable = false)

private String password;

@Enumerated(EnumType.STRING)

@Column(name = "role", nullable = false)

private Authority role;

@OneToOne(cascade = CascadeType.MERGE)

@JoinColumn(name = "owner\_id")

private Owner owner;

public User(String username, String password, Authority role, Owner owner) {

this.username = username;

this.password = password;

this.role = role;

this.owner = owner;

}

public User() {}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Authority getRole() {

return role;

}

public void setRole(Authority role) {

this.role = role;

}

public Owner getOwner() {

return owner;

}

public void setOwner(Owner owner) {

this.owner = owner;

}

}

Добавлен файл kotiki-java/database-access/src/main/java/models/enums/Authority.java:

package models.enums;

public enum Authority {

ROLE\_USER,

ROLE\_ADMIN

}

Изменен файл kotiki-java/database-access/src/main/java/models/enums/CatColor.java:

package models.enums;

public enum CatColor {

+

WHITE,

BLACK,

RED,

Удален файл kotiki-java/database-access/src/main/java/utils/HibernateSessionFactoryUtil.java

Удален файл kotiki-java/database-access/src/main/resources/hibernate.cfg.xml

Изменен файл kotiki-java/pom.xml:

<version>2.6.5</version>

<scope>test</scope>

</dependency>

+ <dependency>

+ <groupId>org.springframework.boot</groupId>

+ <artifactId>spring-boot-starter-security</artifactId>

+ <version>2.6.5</version>

+ </dependency>

</dependencies>

<properties>

@@ -63,4 +68,5 @@

</plugin>

</plugins>

</build>

+

</project>

Удален файл kotiki-java/src/main/java/controllers/CatsController.java

Удален файл kotiki-java/src/main/java/services/CatsService.java

Удален файл kotiki-java/src/test/java/tests/CatsTests.java

Изменен файл kotiki-java/user-service/src/main/java/controllers/CatController.java:

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

+import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.server.ResponseStatusException;

import services.CatService;

-import java.time.LocalDate;

+import javax.persistence.EntityNotFoundException;

import java.util.List;

@RestController

public CatController(CatService catService, CatToDtoMapper catToDtoMapper, DtoTo

@GetMapping("/get/{id}")

public ResponseEntity<CatDto> getById(@PathVariable int id) {

- Cat cat = catService.getById(id);

+ try {

+ Cat cat = catService.getById(id);

+ return ResponseEntity.ok(catToDtoMapper.convert(cat));

- if (cat == null) {

+ } catch (EntityNotFoundException | NullPointerException e) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

-

- return ResponseEntity.ok(catToDtoMapper.convert(cat));

}

@PostMapping("/save")

+ @PreAuthorize("hasRole('ADMIN')")

public void save(@RequestBody CatDto catDto) {

catService.save(dtoToCatMapper.convert(catDto));

}

public void deleteById(@PathVariable int id) {

try {

catService.deleteById(id);

}

- catch (Exception \_\_) {

+ catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

public List<CatDto> findByName(@PathVariable String name) {

@GetMapping("/find-by-birthDate/{birthDate}")

public List<CatDto> findByBirthDate(@PathVariable String birthDate) {

- LocalDate date = LocalDate.parse(birthDate);

- return catService.findByBirthDate(date).stream().map(catToDtoMapper::convert).toList();

+ return catService.findByBirthDate(birthDate).stream().map(catToDtoMapper::convert).toList();

}

@GetMapping("/find-by-breed/{breed}")

Изменен файл kotiki-java/user-service/src/main/java/controllers/OwnerController.java:

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

+import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.server.ResponseStatusException;

import services.OwnerService;

-import java.time.LocalDate;

+import javax.persistence.EntityNotFoundException;

import java.util.List;

@RestController

public OwnerController(OwnerService ownerService, OwnerToDtoMapper ownerToDtoMap

@GetMapping("/get/{id}")

public ResponseEntity<OwnerDto> getById(@PathVariable int id) {

- Owner owner = ownerService.getById(id);

-

- if (owner == null) {

+ try {

+ Owner owner = ownerService.getById(id);

+ return ResponseEntity.ok(ownerToDtoMapper.convert(owner));

+ } catch (EntityNotFoundException | NullPointerException e) {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

-

- return ResponseEntity.ok(ownerToDtoMapper.convert(owner));

}

- @PostMapping("/update")

+ @PostMapping("/save")

+ @PreAuthorize("hasRole('ADMIN')")

public void save(@RequestBody OwnerDto ownerDto) {

ownerService.save(dtoToOwnerMapper.convert(ownerDto));

}

@PostMapping("/delete/{id}")

+ @PreAuthorize("hasRole('ADMIN')")

public void deleteById(@PathVariable int id) {

try {

ownerService.deleteById(id);

}

- catch (Exception \_\_) {

+ catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

@GetMapping ("/find-all")

+ @PreAuthorize("hasRole('ADMIN')")

public List<OwnerDto> findAll() {

return ownerService.findAll().stream().map(ownerToDtoMapper::convert).toList();

}

public List<OwnerDto> findByName(@PathVariable String name) {

@GetMapping ("/find-by-birthDate/{birthDate}")

public List<OwnerDto> findByBirthDate(@PathVariable String birthDate) {

- LocalDate date = LocalDate.parse(birthDate);

- return ownerService.findByBirthDate(date).stream().map(ownerToDtoMapper::convert).toList();

+ return ownerService.findByBirthDate(birthDate).stream().map(ownerToDtoMapper::convert).toList();

}

}

Добавлен файл kotiki-java/user-service/src/main/java/controllers/UserController.java:

package controllers;

import dto.UserDto;

import dto.mappers.DtoToUserMapper;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.server.ResponseStatusException;

import services.UserService;

@RestController

@RequestMapping("/users")

public class UserController {

private final UserService userService;

private final DtoToUserMapper dtoToUserMapper;

@Autowired

public UserController(UserService userService, DtoToUserMapper dtoToUserMapper) {

this.userService = userService;

this.dtoToUserMapper = dtoToUserMapper;

}

@PostMapping("/save")

@PreAuthorize("hasRole('ADMIN')")

public void save(@RequestBody UserDto user) {

userService.save(dtoToUserMapper.convert(user));

}

@PostMapping("/delete/{id}")

@PreAuthorize("hasRole('ADMIN')")

public void deleteById(@PathVariable int id) {

try {

userService.deleteById(id);

}

catch (Exception \_\_) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST);

}

}

}

Изменен файл kotiki-java/user-service/src/main/java/dto/CatDto.java:

package dto;

-import models.Owner;

import models.enums.CatColor;

import java.time.LocalDate;

Добавлен файл kotiki-java/user-service/src/main/java/dto/UserDto.java:

package dto;

import models.enums.Authority;

public class UserDto {

private int id;

private String username;

private String password;

private Authority role;

private OwnerDto owner;

public UserDto(int id, String username, String password, Authority role, OwnerDto owner) {

this.id = id;

this.username = username;

this.password = password;

this.role = role;

this.owner = owner;

}

public UserDto() {}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Authority getRole() {

return role;

}

public void setRole(Authority role) {

this.role = role;

}

public OwnerDto getOwner() {

return owner;

}

public void setOwner(OwnerDto owner) {

this.owner = owner;

}

}

Изменен файл kotiki-java/user-service/src/main/java/dto/mappers/DtoToOwnerMapper.java:

package dto.mappers;

-import dto.CatDto;

import dto.OwnerDto;

import models.Cat;

import models.Owner;

public DtoToOwnerMapper(CatService catService) {

public Owner convert(OwnerDto ownerDto) {

Owner owner = new Owner();

owner.setId(ownerDto.getId());

- owner.setName(owner.getName());

- owner.setBirthDate(owner.getBirthDate());

+ owner.setName(ownerDto.getName());

+ owner.setBirthDate(ownerDto.getBirthDate());

Set<Cat> cats = new HashSet<>();

if (ownerDto.getCats() != null) {

cats = ownerDto.getCats().stream().map(catService::getById).collect(Collectors.toSet());

Добавлен файл kotiki-java/user-service/src/main/java/dto/mappers/DtoToUserMapper.java:

package dto.mappers;

import dto.UserDto;

import models.User;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Component;

@Component

public class DtoToUserMapper {

private final DtoToOwnerMapper dtoToOwnerMapper;

private final PasswordEncoder passwordEncoder;

@Autowired

public DtoToUserMapper(DtoToOwnerMapper dtoToOwnerMapper, PasswordEncoder passwordEncoder) {

this.dtoToOwnerMapper = dtoToOwnerMapper;

this.passwordEncoder = passwordEncoder;

}

public User convert(UserDto userDto) {

User user = new User();

user.setId(userDto.getId());

user.setUsername(userDto.getUsername());

user.setPassword(passwordEncoder.encode(userDto.getPassword()));

user.setRole(userDto.getRole());

user.setOwner(dtoToOwnerMapper.convert(userDto.getOwner()));

return user;

}

}

Изменен файл kotiki-java/user-service/src/main/java/services/CatService.java:

import dao.CatDao;

import models.Cat;

import models.Owner;

+import models.User;

+import models.enums.Authority;

import models.enums.CatColor;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@@ -14,18 +16,39 @@

public class CatService {

private final CatDao catDao;

+ private final UserService userService;

@Autowired

- public CatService(CatDao catDao) {

+ public CatService(CatDao catDao, UserService userService) {

this.catDao = catDao;

+ this.userService = userService;

}

public Cat getById(int id) {

- return catDao.getById(id);

+ Cat cat = catDao.getById(id);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return cat;

+ }

+

+ Owner owner = user.getOwner();

+ if (owner == cat.getOwner()) {

+ return cat;

+ }

+

+ return null;

}

public void save(Cat cat) {

- catDao.save(cat);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ catDao.save(cat);

+ }

+

+ Owner owner = user.getOwner();

+ if (owner.getId() == cat.getOwner().getId()) {

+ catDao.save(cat);

+ }

}

public void deleteById(int id) {

public void deleteById(int id) {

}

public List<Cat> findAll() {

- return catDao.findAll();

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return catDao.findAll();

+ }

+

+ return catDao.findByOwner(user.getOwner());

}

public List<Cat> findByName(String name) {

- return catDao.findByName(name);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return catDao.findByName(name);

+ }

+

+ return catDao.findByNameAndOwner(name, user.getOwner());

+

}

- public List<Cat> findByBirthDate(LocalDate birthDate) {

- return catDao.findByBirthDate(birthDate);

+ public List<Cat> findByBirthDate(String birthDate) {

+ LocalDate date = LocalDate.parse(birthDate);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return catDao.findByBirthDate(date);

+ }

+

+ return catDao.findByBirthDateAndOwner(date, user.getOwner());

+

}

public List<Cat> findByBreed(String breed) {

- return catDao.findByBreed(breed);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return catDao.findByBreed(breed);

+ }

+

+ return catDao.findByBreedAndOwner(breed, user.getOwner());

}

public List<Cat> findByColor(CatColor color) {

- return catDao.findByColor(color);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return catDao.findByColor(color);

+ }

+

+ return catDao.findByColorAndOwner(color, user.getOwner());

}

}

Изменен файл kotiki-java/user-service/src/main/java/services/OwnerService.java:

import dao.OwnerDao;

import models.Owner;

+import models.User;

+import models.enums.Authority;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@@ -12,14 +14,26 @@

public class OwnerService {

private final OwnerDao ownerDao;

+ private final UserService userService;

@Autowired

- public OwnerService(OwnerDao ownerDao) {

+ public OwnerService(OwnerDao ownerDao, UserService userService) {

this.ownerDao = ownerDao;

+ this.userService = userService;

}

public Owner getById(int id) {

- return ownerDao.getById(id);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return ownerDao.getById(id);

+ }

+

+ Owner owner = user.getOwner();

+ if (owner.getId() == id) {

+ return owner;

+ }

+

+ return null;

}

public void save(Owner owner) {

public void deleteById(int id) {

}

public List<Owner> findAll() {

- return ownerDao.findAll();

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return ownerDao.findAll();

+ }

+

+ return List.of(user.getOwner());

}

public List<Owner> findByName(String name) {

- return ownerDao.findByName(name);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return ownerDao.findByName(name);

+ }

+

+ Owner owner = user.getOwner();

+ return ownerDao.findByNameAndId(name, owner.getId());

}

- public List<Owner> findByBirthDate(LocalDate birthDate) {

- return ownerDao.findByBirthDate(birthDate);

+ public List<Owner> findByBirthDate(String birthDate) {

+ LocalDate date = LocalDate.parse(birthDate);

+ User user = userService.getAuthorizedUser();

+ if (user.getRole() == Authority.ROLE\_ADMIN) {

+ return ownerDao.findByBirthDate(date);

+ }

+

+ Owner owner = user.getOwner();

+ return ownerDao.findByBirthDateAndId(date, owner.getId());

}

}

Добавлен файл kotiki-java/user-service/src/main/java/services/UserService.java:

package services;

import dao.UserDao;

import models.User;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.nio.charset.StandardCharsets;

import java.util.List;

@Service

public class UserService implements UserDetailsService{

private final UserDao userDao;

@Autowired

public UserService(UserDao userDao) {

this.userDao = userDao;

}

@Override

@Transactional

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userDao.getByUsername(username);

if (user == null) {

throw new UsernameNotFoundException("User with username " + username + " was not found");

}

return new org.springframework.security.core.userdetails.User(

username,

user.getPassword(),

List.of(new SimpleGrantedAuthority(user.getRole().toString())));

}

public void save(User user) {

userDao.save(user);

}

public void deleteById(int id) {

userDao.deleteById(id);

}

public User getAuthorizedUser() {

Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

return userDao.getByUsername(

principal instanceof UserDetails userDetails

? userDetails.getUsername() : principal.toString());

}

}

Изменен файл kotiki-java/user-service/src/main/java/startapp/CatsSpringApplication.java:

@SpringBootApplication

@EnableJpaRepositories("dao")

-@ComponentScan({"controllers", "services", "dto"})

+@ComponentScan({"controllers", "services", "dto", "startapp.cfg"})

@EntityScan("models")

public class CatsSpringApplication {

Добавлен файл kotiki-java/user-service/src/main/java/startapp/cfg/SecurityConfig.java:

package startapp.cfg;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import services.UserService;

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private final UserService userService;

@Autowired

public SecurityConfig(UserService userService) {

this.userService = userService;

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.httpBasic()

.and()

.authorizeRequests()

.antMatchers("/\*\*").authenticated()

.and()

.csrf().disable()

.formLogin();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public DaoAuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authenticationProvider = new DaoAuthenticationProvider();

authenticationProvider.setUserDetailsService(userService);

authenticationProvider.setPasswordEncoder(passwordEncoder());

return authenticationProvider;

}

}

Изменен файл kotiki-java/user-service/src/main/resources/application.properties:

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.username=postgres

-spring.datasource.password=123456

+spring.datasource.password=123456

+spring.jpa.hibernate.ddl-auto=update

Изменен файл kotiki-java/user-service/src/test/java/services/CatServiceTest.java:

public class CatServiceTest {

private CatDao catDao;

private CatService catService;

+ private UserService userService;

@BeforeEach

public void setUp() {

catDao = Mockito.mock(CatDao.class);

- catService = new CatService(catDao);

+ userService = Mockito.mock(UserService.class);

+ catService = new CatService(catDao, userService);

}

@Test

Изменен файл kotiki-java/user-service/src/test/java/services/OwnerServiceTest.java:

public class OwnerServiceTest {

private OwnerDao ownerDao;

private OwnerService ownerService;

+ private UserService userService;

@BeforeEach

public void setUp() {

ownerDao = Mockito.mock(OwnerDao.class);

- ownerService = new OwnerService(ownerDao);

+ userService = Mockito.mock(UserService.class);

+ ownerService = new OwnerService(ownerDao, userService);

}

@Test