МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

# ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №17 - 22

**по дисциплине**

**«Шаблоны программных платформ языка Java**

**Вариант 21**

| | | |
|---|---|---|
| Выполнил студент группы ИКБО-20-19 | | Николаев-Аксенов И. С. |
| Принял<br>*Ассистент* | | Батанов А. О. |
| Практические работы выполнены | «___»_______2021 г. | (подпись студента) |
| «Зачтено» | «___»_______2021 г. | (подпись руководителя) |

Москва 2021

## Содержание

# Практическая работа №17

## *Цель работы*

Тема: Знакомство с Criteria API в Hibernate.

Постановка задачи: Добавить возможность фильтрации по всем полям всех классов с использованием Criteria API в Hibernate для программы из предыдущего задания. Добавить эндпоинты для каждой фильтрации.

## *Листинг программы*

*Application.java (в следующих работах тоже присутствует, но не изменяется)*

```java
package app.Application;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

*User.java*

```java
package app.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;
    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {

    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

*Post.java*

```java
package app.Application.model;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;
import org.springframework.format.annotation.DateTimeFormat;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.Date;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }
}
```

*UserService.java*

```java
package app.Application.service;

import app.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;
    private CriteriaBuilder builder;
    private CriteriaQuery<User> userCriteriaQuery;
    private Root<User> root;

    public UserService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
        builder = session.getCriteriaBuilder();
        userCriteriaQuery = builder.createQuery(User.class);
        root = userCriteriaQuery.from(User.class);
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addUser(User user) {
        session.beginTransaction();
        session.saveOrUpdate(user);
        session.getTransaction().commit();
    }

    public List<User> getUsers() {
        return session.createQuery("select u from User u", User.class).list();
    }

    public User getUser(UUID id) {
        return session.createQuery("select u from User u where u.id = p.id = '" + id +
"'", User.class).getSingleResult();
    }

    public void deleteUser(UUID id) {
        session.beginTransaction();

        User t = session.load(User.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }

    public List<User> getByFirstName() {
        userCriteriaQuery.select(root).orderBy(builder.asc(root.get("firstName")));
        Query<User> query = session.createQuery(userCriteriaQuery);
        return query.getResultList();
    }

    public List<User> getByLastName() {
        userCriteriaQuery.select(root).orderBy(builder.asc(root.get("lastName")));
        Query<User> query = session.createQuery(userCriteriaQuery);
        return query.getResultList();
    }
}
```

*PostService.java*

```
package app.Application.service;

import app.Application.model.Post;
import app.Application.model.User;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final SessionFactory sessionFactory;

    private Session session;
    private CriteriaBuilder builder;
    private CriteriaQuery<Post> criteriaQuery;
    private Root<Post> root;

    public PostService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    @PostConstruct
    public void init() {
        session = sessionFactory.openSession();
        builder = session.getCriteriaBuilder();
        criteriaQuery = builder.createQuery(Post.class);
        root = criteriaQuery.from(Post.class);
    }

    @PreDestroy
    public void unSession() {
        session.close();
    }

    public void addPost(Post post) {
        session.beginTransaction();
        session.saveOrUpdate(post);
        session.getTransaction().commit();
    }

    public List<Post> getPosts() {
        return session.createQuery("select p from Post p", Post.class).list();
    }

    public User getUser(UUID id) {
        return session.createQuery("from Post where id = :id",
Post.class).setParameter("id",id).getSingleResult().getUser();
    }

    public void deletePosts(Post post) {
        session.beginTransaction();

        List<Post> query = session.createQuery("select p from Post p where p.id = '" +
post.getId() + "'", Post.class).list();
        for (Post p : query) {
            session.delete(p);
        }

        session.getTransaction().commit();
    }

    public void deletePost(UUID id) {
        session.beginTransaction();

        Post t = session.load(Post.class, id);
        session.delete(t);

        session.getTransaction().commit();
    }

    public List<Post> getByText() {
        criteriaQuery.select(root).orderBy(builder.asc(root.get("text")));
        Query<Post> query = session.createQuery(criteriaQuery);
        return query.getResultList();
    }

    public List<Post> getByCreationDate() {
        criteriaQuery.select(root).orderBy(builder.asc(root.get("creationDate")));
        Query<Post> query = session.createQuery(criteriaQuery);
        return query.getResultList();
    }
}
```

*UserController.java*

```java
package app.Application.controller;

import app.Application.model.Post;
import app.Application.model.User;
import app.Application.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getUsers() {
        return userService.getUsers();
    }

    @GetMapping("/users/{id}")
    public User getUser(@PathVariable UUID id) {
        return userService.getUser(id);
    }

    @DeleteMapping("/users/{id}")
    public void deleteUser(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getByFirstName")
    public List<User> getByFirstName() {
        return userService.getByFirstName();
    }

    @GetMapping("/getByLastName")
    public List<User> getByLastName() {
        return userService.getByLastName();
    }
}
```

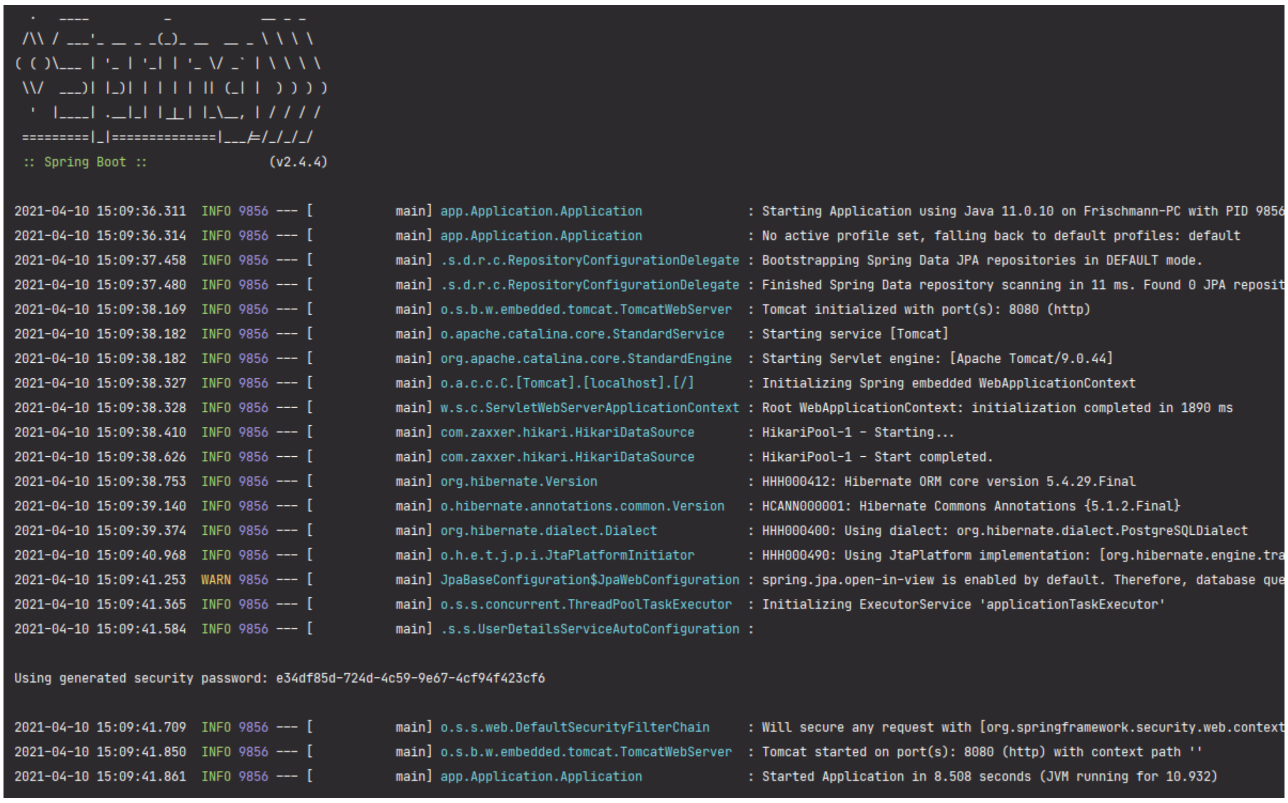
*PostController.java*

```java
package app.Application.controller;

import app.Application.model.Post;
import app.Application.model.User;
import app.Application.service.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/post")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getPosts() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void deletePost(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping("/getByText")
    public List<Post> getByText() {
        return postService.getByText();
    }

    @GetMapping("/getByCreationDate")
    public List<Post> getByCreationDate() {
        return postService.getByCreationDate();
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getUser(@PathVariable("id") UUID id) {
        return postService.getUser(id);
    }
}
```

*Config.java*

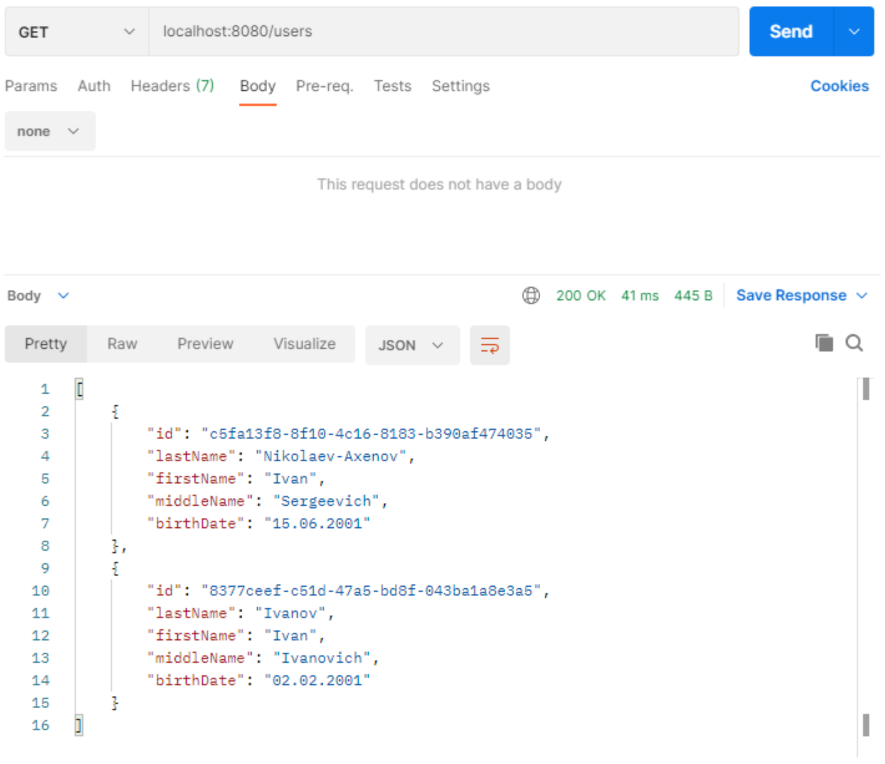
```
package app.Application.config;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.PlatformTransactionManager;

import javax.sql.DataSource;
import java.util.Properties;

@Configuration
public class Config {
    @Bean
    public HikariDataSource dataSource(){
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:postgresql://localhost:5432/pr17db");
        config.setUsername("postgres");
        config.setPassword("secret");
        config.setDriverClassName("org.postgresql.Driver");
        return new HikariDataSource(config);
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory(DataSource dataSource){
        LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();
        factoryBean.setDataSource(dataSource);
        factoryBean.setPackagesToScan("app.Application");
        Properties properties = new Properties();
        properties.setProperty("hibernate.dialect",
"org.hibernate.dialect.PostgreSQLDialect");
        factoryBean.setHibernateProperties(properties);
        return factoryBean;
    }

    @Bean
    public PlatformTransactionManager platformTransactionManager(LocalSessionFactoryBean
factoryBean){
        HibernateTransactionManager transactionManager = new
HibernateTransactionManager();
        transactionManager.setSessionFactory(factoryBean.getObject());
        return transactionManager;
    }
}
```

***Результат выполнения программы***



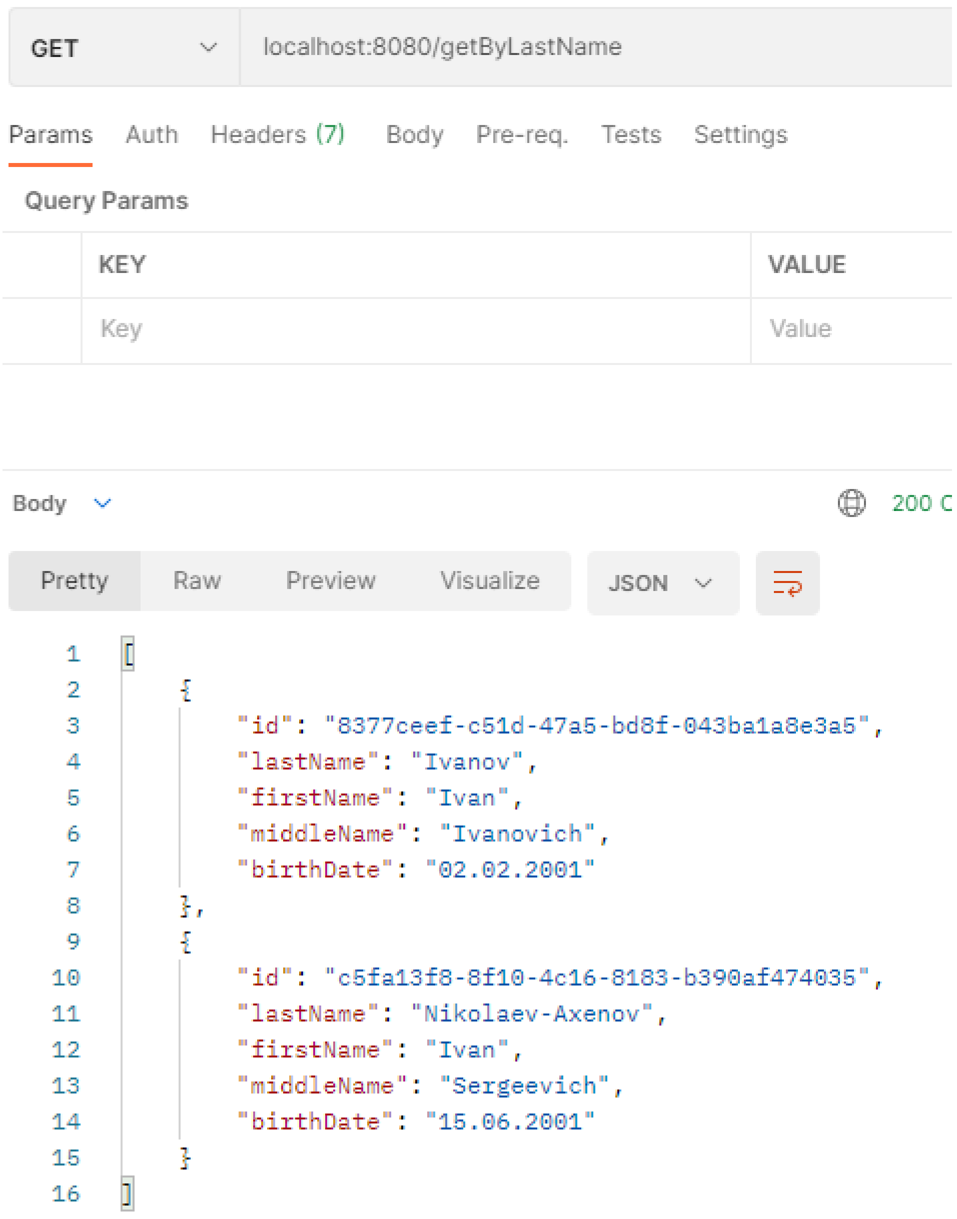
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

2021-04-10 15:09:36.311  INFO 9856 --- [           main] app.Application.Application              : Starting Application using Java 11.0.10 on Frischmann-PC with PID 9856
2021-04-10 15:09:36.314  INFO 9856 --- [           main] app.Application.Application              : No active profile set, falling back to default profiles: default
2021-04-10 15:09:37.458  INFO 9856 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-04-10 15:09:37.480  INFO 9856 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 11 ms. Found 0 JPA reposit
2021-04-10 15:09:38.169  INFO 9856 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-04-10 15:09:38.182  INFO 9856 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-04-10 15:09:38.182  INFO 9856 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-04-10 15:09:38.327  INFO 9856 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-04-10 15:09:38.328  INFO 9856 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1890 ms
2021-04-10 15:09:38.410  INFO 9856 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-04-10 15:09:38.626  INFO 9856 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-04-10 15:09:38.753  INFO 9856 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.29.Final
2021-04-10 15:09:39.140  INFO 9856 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-04-10 15:09:39.374  INFO 9856 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2021-04-10 15:09:40.968  INFO 9856 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.tra
2021-04-10 15:09:41.253  WARN 9856 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database que
2021-04-10 15:09:41.365  INFO 9856 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-04-10 15:09:41.584  INFO 9856 --- [           main] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: e34df85d-724d-4c59-9e67-4cf94f423cf6

2021-04-10 15:09:41.709  INFO 9856 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.web.context
2021-04-10 15:09:41.850  INFO 9856 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-04-10 15:09:41.861  INFO 9856 --- [           main] app.Application.Application              : Started Application in 8.508 seconds (JVM running for 10.932)
```

Рисунок 17.1 – Демонстрация работы программы



```
[
    {
        "id": "c5fa13f8-8f10-4c16-8183-b390af474035",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "8377ceef-c51d-47a5-bd8f-043ba1a8e3a5",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 17.2 – Демонстрация работы программы



```
[
    {
        "id": "8377ceef-c51d-47a5-bd8f-043ba1a8e3a5",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    },
    {
        "id": "c5fa13f8-8f10-4c16-8183-b390af474035",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    }
]
```

Рисунок 17.3 – Демонстрация работы программы

# Практическая работа №18

## *Цель работы*

Тема: Знакомство с репозиториями и сервисами, реализация в проекте. Взаимодействие с Spring Data JPA.

Постановка задачи: Переписать код предыдущего задания с использованием сервисов и отделения логики контроллера от логики сервиса и репозитория. В программе всё взаимодействие с базой данных должно быть реализовано через репозитории Spring Data Jpa.

## *Листинг программы*

*User.java*

```java
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {

    }
    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

*Post.java*

```java
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;
    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }
}
```

*UserRepository.java*

```
package app.Application.Interfaces;

import app.Application.Classes.User;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("UserRepository")
public interface UserRepository extends JpaRepository<User,Long> {
    List<User> findAllByFirstName(String firstName);
    List<User> findAllByLastName(String lastName);

    @NotNull List<User> findAll();
    void deleteById(UUID id);
}
```

*PostRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.Post;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("PostRepository")
public interface PostRepository extends JpaRepository<Post,Long> {
    Post findById(UUID id);

    @NotNull List<Post> findAll();
    void deleteById(UUID id);
}
```

*UserService.java*

```java
package app.Application.Services;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
public class UserService {
    @Autowired
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void addUser(User user) {
        userRepository.save(user);
    }

    public List<User> getUsers() {
        return userRepository.findAll();
    }

    public void deleteUser(UUID id) {
        userRepository.deleteById(id);
    }
    public List<User> getByFirstName(String firstName) {
        return userRepository.findAllByFirstName(firstName);
    }

    public List<User> getByLastName(String lastName) {
        return userRepository.findAllByLastName(lastName);
    }
}
```

*PostService.java*

```
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Interfaces.PostRepository;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
public class PostService {
    @Autowired
    private final PostRepository postRepository;

    public PostService(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    public void addPost(Post post) {
        postRepository.save(post);
    }

    public List<Post> getPosts() {
        return postRepository.findAll();
    }

    public void deletePost(UUID id) {
        postRepository.deleteById(id);
    }

    public User getUserByPost(UUID id) {
        return postRepository.findById(id).getUser();
    }
}
```

*UserController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.User;
import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getAll() {
        return userService.getUsers();
    }

    @DeleteMapping("/user/{id}")
    public void delete(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getUserByFirstName/{firstName}")
    public List<User> getByFirstName(@PathVariable String firstName){
        return userService.getByFirstName(firstName);
    }

    @GetMapping("/getUserByLastName/{lastName}")
    public List<User> getByLastName(@PathVariable String lastName){
        return userService.getByLastName(lastName);
    }
}
```

*PostController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Services.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;
    @PostMapping("/posts")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void delete(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getGame(@PathVariable("id") UUID id) {
        return postService.getUserByPost(id);
    }
}
```

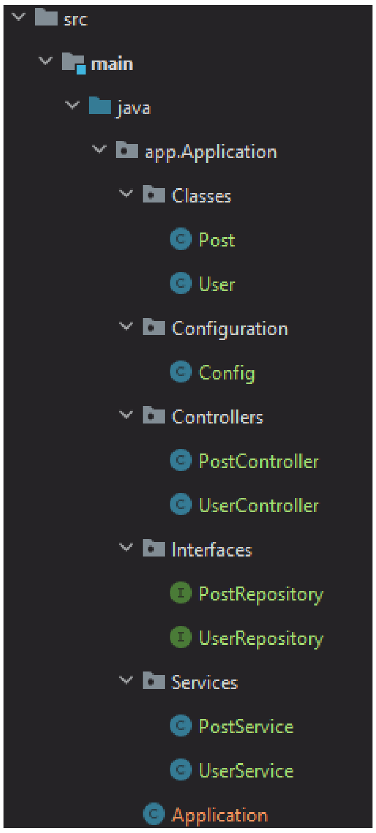
*Config.java*

```
package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@Configuration
@EnableJpaRepositories(basePackages = {"app.Application"})
public class Config {
}
```

*application.yml (в следующих работах тоже присутствует, но меняется только ссылка на базу данных)*

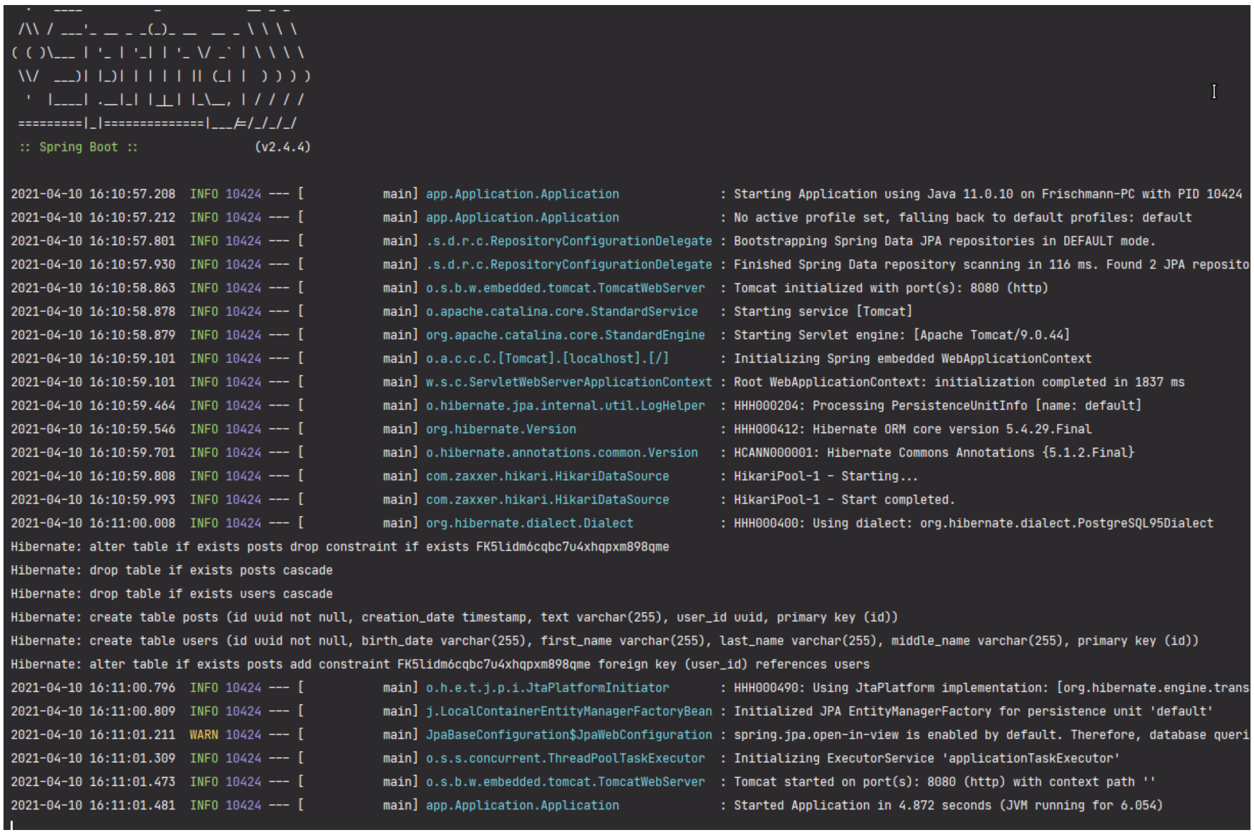
```
spring:
  jpa:
    database: POSTGRESQL
    show-sql: true
    hibernate:
      ddl-auto: create-drop
  datasource:
    platform: postgres
    url: jdbc:postgresql://localhost:5432/pr18db
    username: postgres
    password: secret
    driverClassName: org.postgresql.Driver
```

***Результат выполнения программы***



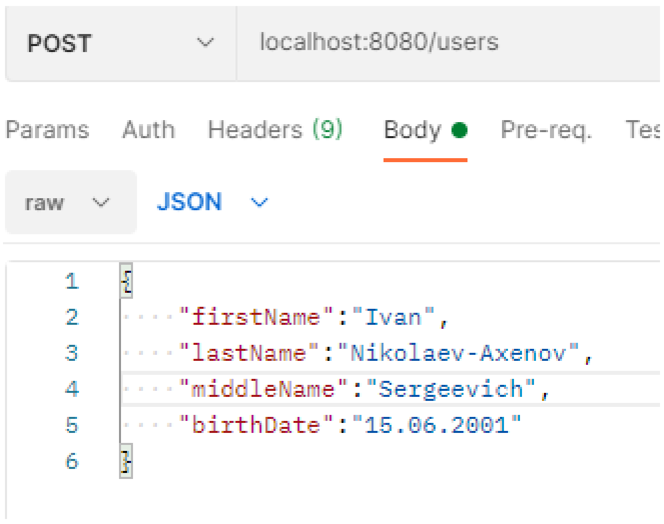
```
src
  main
    java
      app.Application
        Classes
          Post
          User
        Configuration
          Config
        Controllers
          PostController
          UserController
        Interfaces
          PostRepository
          UserRepository
        Services
          PostService
          UserService
        Application
```

Рисунок 18.1 – Демонстрация работы программы



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

2021-04-10 16:10:57.208  INFO 10424 --- [           main] app.Application.Application              : Starting Application using Java 11.0.10 on Frischmann-PC with PID 10424
2021-04-10 16:10:57.212  INFO 10424 --- [           main] app.Application.Application              : No active profile set, falling back to default profiles: default
2021-04-10 16:10:57.801  INFO 10424 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-04-10 16:10:57.930  INFO 10424 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 116 ms. Found 2 JPA reposito
2021-04-10 16:10:58.863  INFO 10424 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-04-10 16:10:58.878  INFO 10424 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-04-10 16:10:58.879  INFO 10424 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-04-10 16:10:59.101  INFO 10424 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-04-10 16:10:59.101  INFO 10424 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1837 ms
2021-04-10 16:10:59.464  INFO 10424 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-04-10 16:10:59.546  INFO 10424 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.29.Final
2021-04-10 16:10:59.701  INFO 10424 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-04-10 16:10:59.808  INFO 10424 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-04-10 16:10:59.993  INFO 10424 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-04-10 16:11:00.008  INFO 10424 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQL95Dialect
Hibernate: alter table if exists posts drop constraint if exists FK5lidm6cqbc7u4xhqpxm898qme
Hibernate: drop table if exists posts cascade
Hibernate: drop table if exists users cascade
Hibernate: create table posts (id uuid not null, creation_date timestamp, text varchar(255), user_id uuid, primary key (id))
Hibernate: create table users (id uuid not null, birth_date varchar(255), first_name varchar(255), last_name varchar(255), middle_name varchar(255), primary key (id))
Hibernate: alter table if exists posts add constraint FK5lidm6cqbc7u4xhqpxm898qme foreign key (user_id) references users
2021-04-10 16:11:00.796  INFO 10424 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.trans
2021-04-10 16:11:00.809  INFO 10424 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-04-10 16:11:01.211  WARN 10424 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queri
2021-04-10 16:11:01.309  INFO 10424 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-04-10 16:11:01.473  INFO 10424 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-04-10 16:11:01.481  INFO 10424 --- [           main] app.Application.Application              : Started Application in 4.872 seconds (JVM running for 6.054)
```

Рисунок 18.2 – Демонстрация работы программы



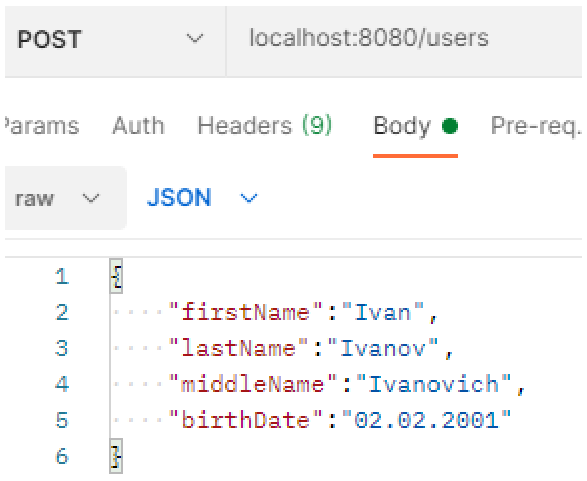
POST localhost:8080/users

Params Auth Headers (9) Body Pre-req. Tes

raw JSON

```
{
    "firstName":"Ivan",
    "lastName":"Nikolaev-Axenov",
    "middleName":"Sergeevich",
    "birthDate":"15.06.2001"
}
```

Рисунок 18.3 – Демонстрация работы программы



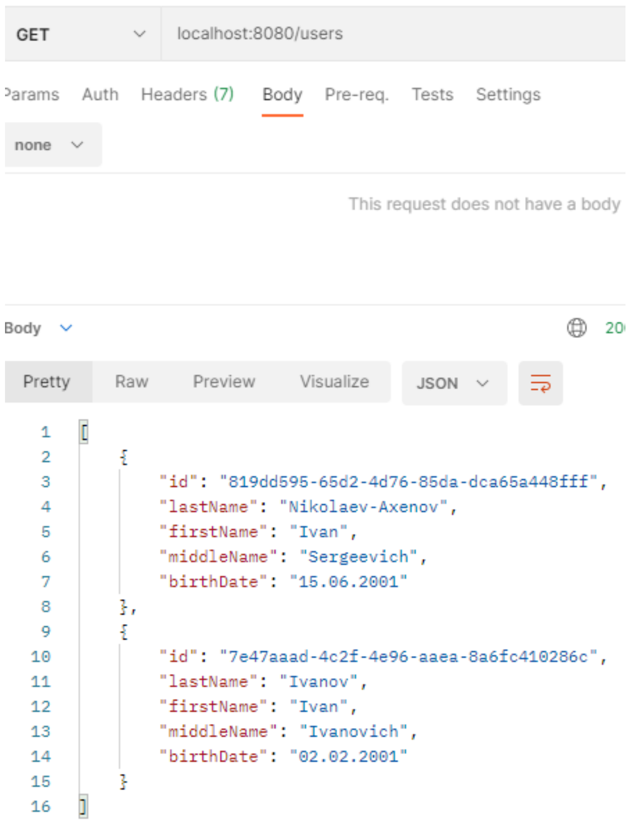
POST localhost:8080/users

Params Auth Headers (9) Body Pre-req.

raw JSON

```
{
    "firstName":"Ivan",
    "lastName":"Ivanov",
    "middleName":"Ivanovich",
    "birthDate":"02.02.2001"
}
```

Рисунок 18.4 – Демонстрация работы программы

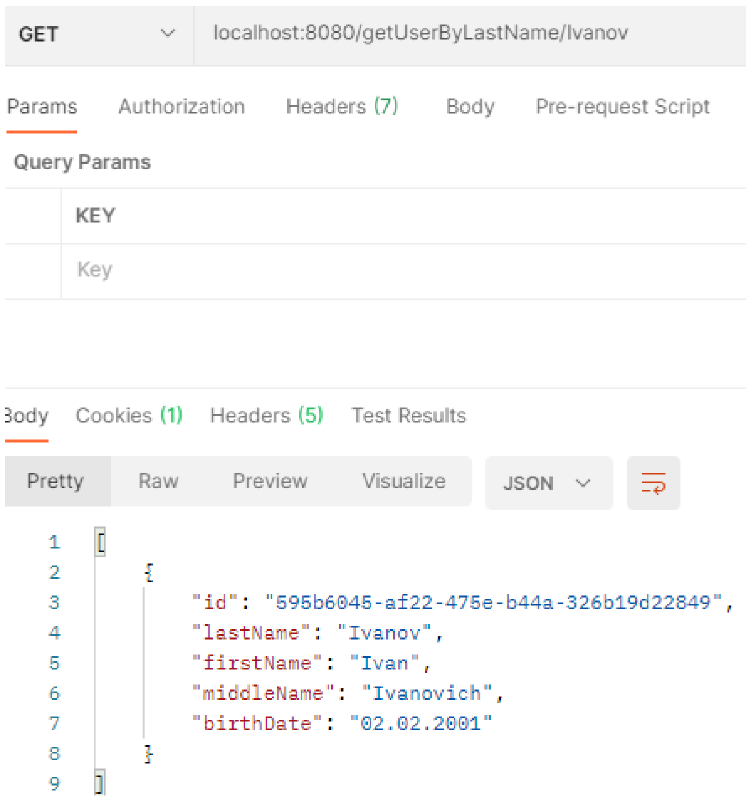


localhost:8080/users

This request does not have a body

```
[
    {
        "id": "819dd595-65d2-4d76-85da-dca65a448fff",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "7e47aaad-4c2f-4e96-aaea-8a6fc410286c",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 18.5 – Демонстрация работы программы



localhost:8080/getUserByLastName/Ivanov

```
[
    {
        "id": "595b6045-af22-475e-b44a-326b19d22849",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 18.6 – Демонстрация работы программы

## Практическая работа №19

### *Цель работы*

Тема: Знакомство с логированием с использованием Logback в Spring.

Постановка задачи: Создать файл logback.xml, добавить логирование во все методы классов-сервисов.

### *Листинг программы*

*User.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {

    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

*Post.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;
    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }
}
```

*UserRepository.java*

```
package app.Application.Interfaces;

import app.Application.Classes.User;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("UserRepository")
public interface UserRepository extends JpaRepository<User,Long> {
    List<User> findAllByFirstName(String firstName);
    List<User> findAllByLastName(String lastName);

    @NotNull List<User> findAll();
    void deleteById(UUID id);
}
```

*PostRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.Post;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("PostRepository")
public interface PostRepository extends JpaRepository<Post,Long> {
    Post findById(UUID id);

    @NotNull List<Post> findAll();
    void deleteById(UUID id);
}
```

*UserService.java*

```java
package app.Application.Services;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class UserService {
    @Autowired
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void addUser(User user) {
        userRepository.save(user);
    }

    public List<User> getUsers() {
        return userRepository.findAll();
    }

    public void deleteUser(UUID id) {
        userRepository.deleteById(id);
    }

    public List<User> getByFirstName(String firstName) {
        return userRepository.findAllByFirstName(firstName);
    }

    public List<User> getByLastName(String lastName) {
        return userRepository.findAllByLastName(lastName);
    }
}
```

*PostService.java*

```java
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Interfaces.PostRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class PostService {
    @Autowired
    private final PostRepository postRepository;

    public PostService(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    public void addPost(Post post) {
        postRepository.save(post);
    }

    public List<Post> getPosts() {
        return postRepository.findAll();
    }

    public void deletePost(UUID id) {
        postRepository.deleteById(id);
    }

    public User getUserByPost(UUID id) {
        return postRepository.findById(id).getUser();
    }
}
```

*UserController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.User;
import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getAll() {
        return userService.getUsers();
    }

    @DeleteMapping("/user/{id}")
    public void delete(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getUserByFirstName/{firstName}")
    public List<User> getByFirstName(@PathVariable String firstName){
        return userService.getByFirstName(firstName);
    }

    @GetMapping("/getUserByLastName/{lastName}")
    public List<User> getByLastName(@PathVariable String lastName){
        return userService.getByLastName(lastName);
    }
}
```

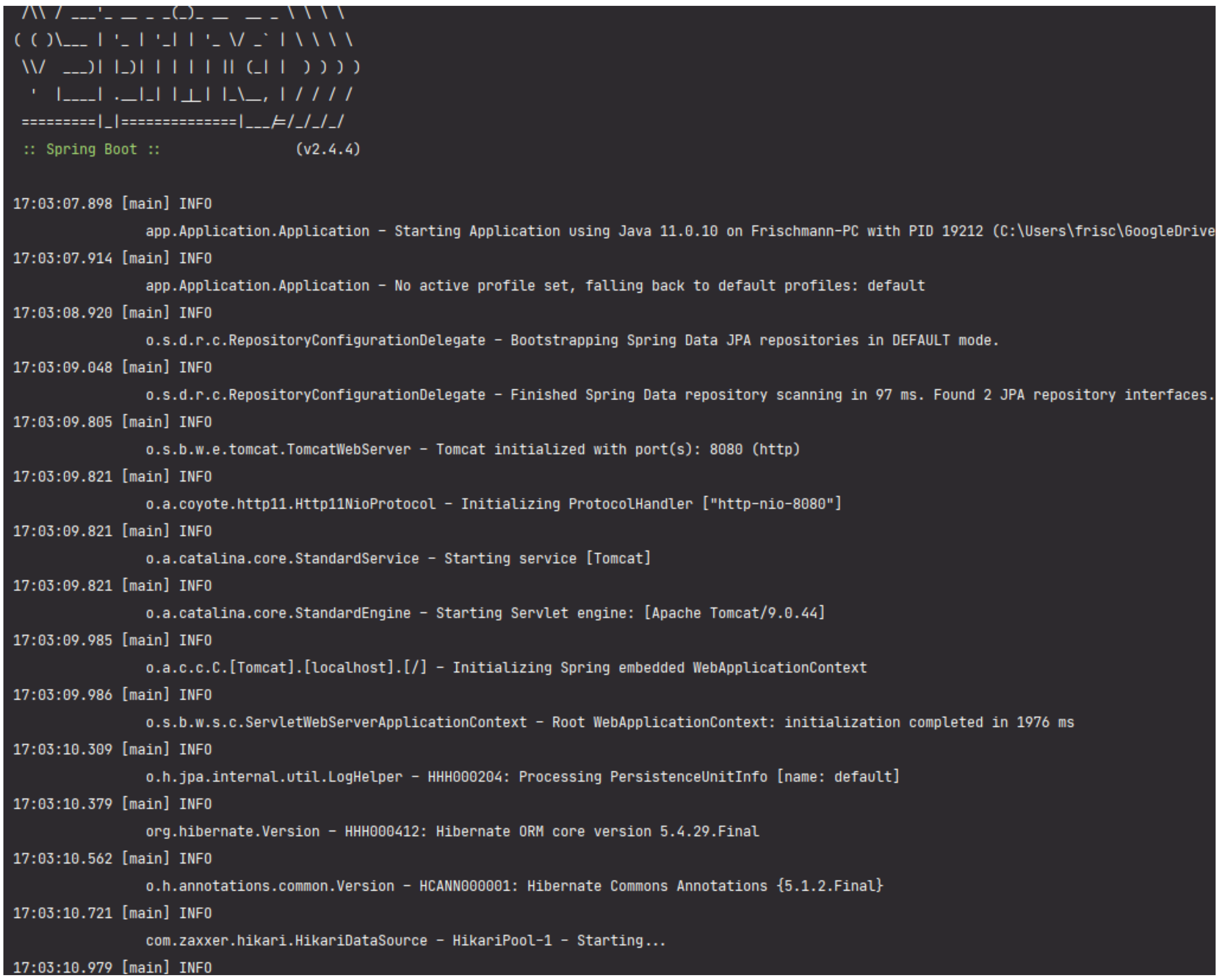
*PostController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Services.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;
    @PostMapping("/posts")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void delete(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getGame(@PathVariable("id") UUID id) {
        return postService.getUserByPost(id);
    }
}
```

*Config.java*

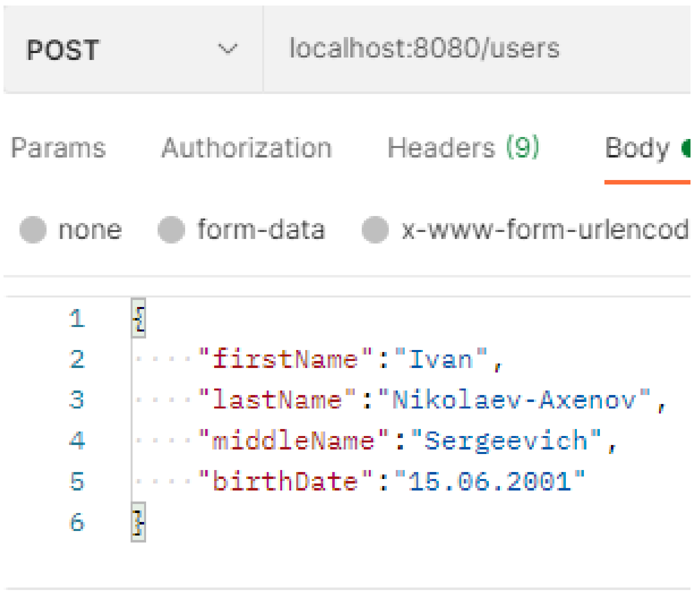
```
package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@Configuration
@EnableJpaRepositories(basePackages = {"app.Application"})
public class Config {
}
```

***Результат выполнения программы***



```
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

17:03:07.898 [main] INFO
                app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 19212 (C:\Users\frisc\GoogleDrive
17:03:07.914 [main] INFO
                app.Application.Application - No active profile set, falling back to default profiles: default
17:03:08.920 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
17:03:09.048 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 97 ms. Found 2 JPA repository interfaces.
17:03:09.805 [main] INFO
                o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
17:03:09.821 [main] INFO
                o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
17:03:09.821 [main] INFO
                o.a.catalina.core.StandardService - Starting service [Tomcat]
17:03:09.821 [main] INFO
                o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
17:03:09.985 [main] INFO
                o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
17:03:09.986 [main] INFO
                o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1976 ms
17:03:10.309 [main] INFO
                o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
17:03:10.379 [main] INFO
                org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final
17:03:10.562 [main] INFO
                o.h.annotations.common.Version - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
17:03:10.721 [main] INFO
                com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
17:03:10.979 [main] INFO
```

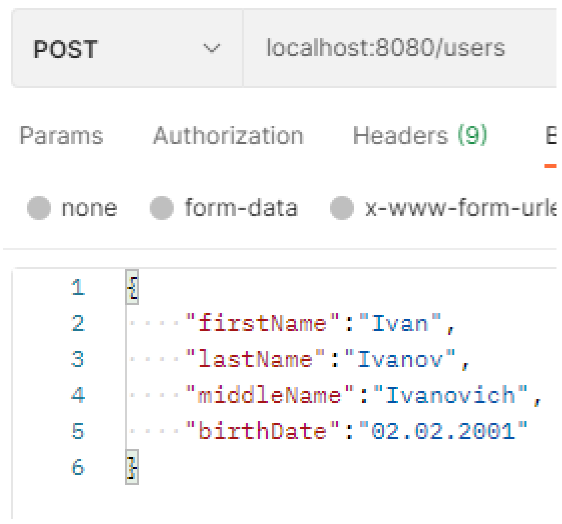
Рисунок 19.1 – Демонстрация работы программы



POST localhost:8080/users

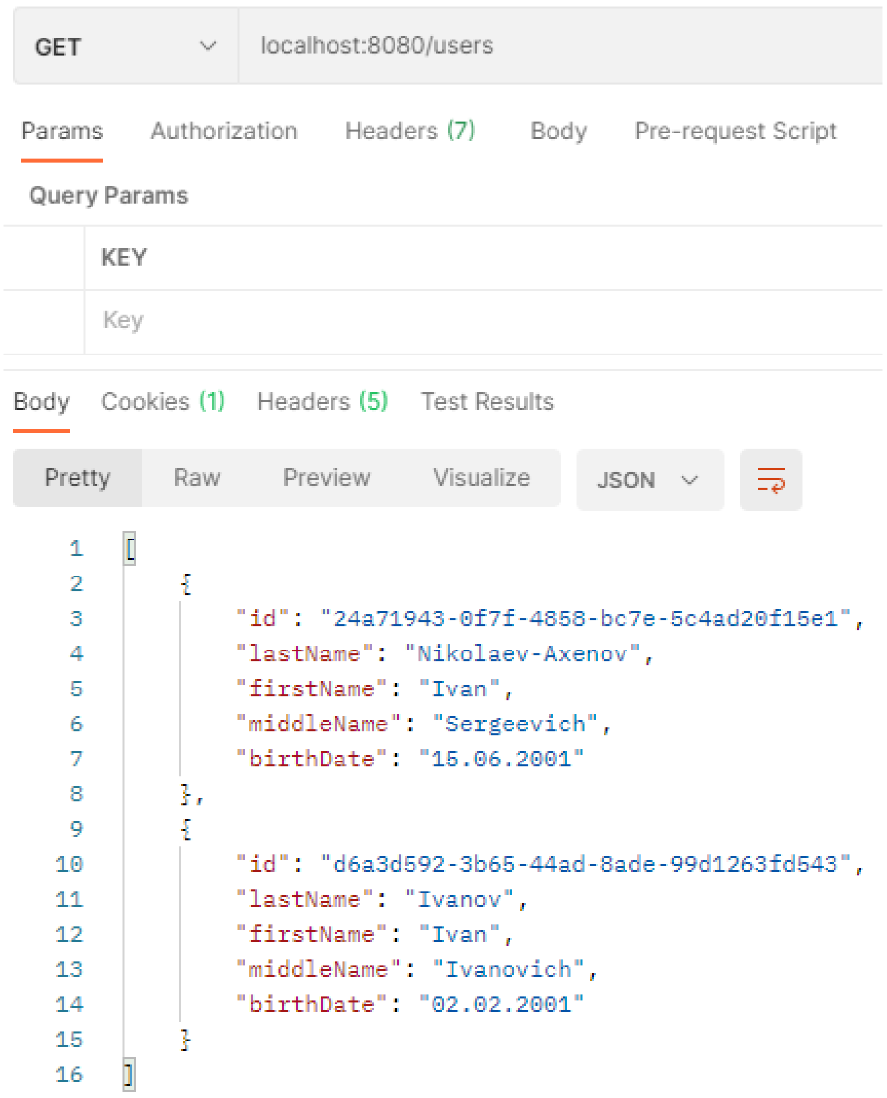
```
{
    "firstName":"Ivan",
    "lastName":"Nikolaev-Axenov",
    "middleName":"Sergeevich",
    "birthDate":"15.06.2001"
}
```

Рисунок 19.2 – Демонстрация работы программы



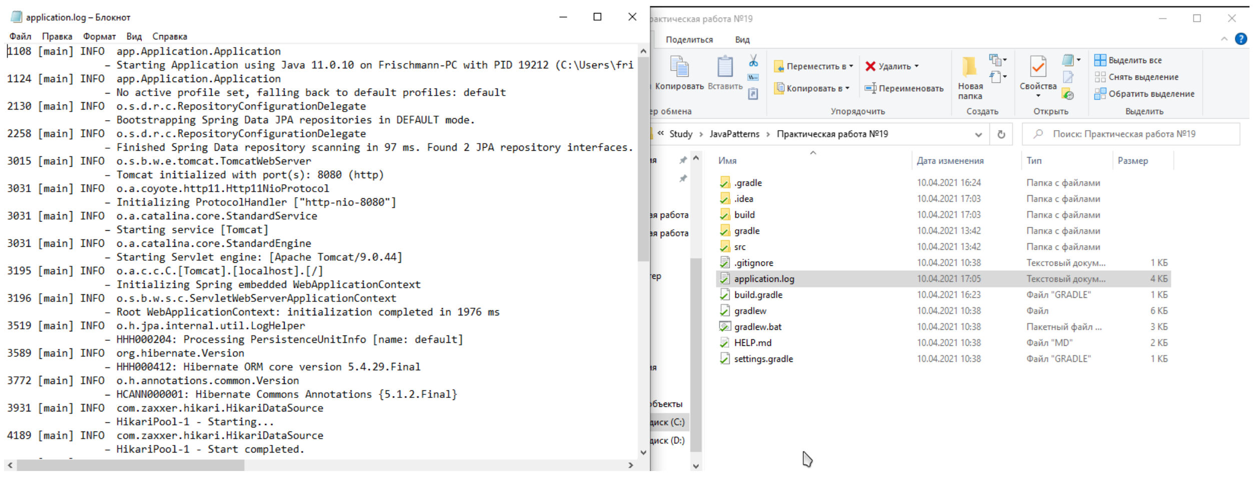
```
{
    "firstName":"Ivan",
    "lastName":"Ivanov",
    "middleName":"Ivanovich",
    "birthDate":"02.02.2001"
}
```

Рисунок 19.3 – Демонстрация работы программы



```
[
    {
        "id": "24a71943-0f7f-4858-bc7e-5c4ad20f15e1",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "d6a3d592-3b65-44ad-8ade-99d1263fd543",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 19.4 – Демонстрация работы программы


application.log – Блокнот
Файл Правка Формат Вид Справка
Поделиться Вид
Копировать Вставить
Переместить в Удалить
Копировать в Переименовать
Новая папка
Свойства
Выделить все
Снять выделение
Обратить выделение
Упорядочить Создать Открыть Выделить
Study › JavaPatterns › Практическая работа №19
Поиск: Практическая работа №19

```
1108 [main] INFO  app.Application.Application
                - Starting Application using Java 11.0.10 on Frischmann-PC with PID 19212 (C:\Users\fri
1124 [main] INFO  app.Application.Application
                - No active profile set, falling back to default profiles: default
2130 [main] INFO  o.s.d.r.c.RepositoryConfigurationDelegate
                - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2258 [main] INFO  o.s.d.r.c.RepositoryConfigurationDelegate
                - Finished Spring Data repository scanning in 97 ms. Found 2 JPA repository interfaces.
3015 [main] INFO  o.s.b.w.e.tomcat.TomcatWebServer
                - Tomcat initialized with port(s): 8080 (http)
3031 [main] INFO  o.a.coyote.http11.Http11NioProtocol
                - Initializing ProtocolHandler ["http-nio-8080"]
3031 [main] INFO  o.a.catalina.core.StandardService
                - Starting service [Tomcat]
3031 [main] INFO  o.a.catalina.core.StandardEngine
                - Starting Servlet engine: [Apache Tomcat/9.0.44]
3195 [main] INFO  o.a.c.c.C.[Tomcat].[localhost].[/]
                - Initializing Spring embedded WebApplicationContext
3196 [main] INFO  o.s.b.w.s.c.ServletWebServerApplicationContext
                - Root WebApplicationContext: initialization completed in 1976 ms
3519 [main] INFO  o.h.jpa.internal.util.LogHelper
                - HHH000204: Processing PersistenceUnitInfo [name: default]
3589 [main] INFO  org.hibernate.Version
                - HHH000412: Hibernate ORM core version 5.4.29.Final
3772 [main] INFO  o.h.annotations.common.Version
                - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
3931 [main] INFO  com.zaxxer.hikari.HikariDataSource
                - HikariPool-1 - Starting...
4189 [main] INFO  com.zaxxer.hikari.HikariDataSource
                - HikariPool-1 - Start completed.
```

| Имя | Дата изменения | Тип | Размер |
|---|---|---|---|
| .gradle | 10.04.2021 16:24 | Папка с файлами | |
| .idea | 10.04.2021 17:03 | Папка с файлами | |
| build | 10.04.2021 17:03 | Папка с файлами | |
| gradle | 10.04.2021 13:42 | Папка с файлами | |
| src | 10.04.2021 13:42 | Папка с файлами | |
| .gitignore | 10.04.2021 10:38 | Текстовый докум... | 1 КБ |
| application.log | 10.04.2021 17:05 | Текстовый докум... | 4 КБ |
| build.gradle | 10.04.2021 16:23 | Файл "GRADLE" | 1 КБ |
| gradlew | 10.04.2021 10:38 | Файл | 6 КБ |
| gradlew.bat | 10.04.2021 10:38 | Пакетный файл ... | 3 КБ |
| HELP.md | 10.04.2021 10:38 | Файл "MD" | 2 КБ |
| settings.gradle | 10.04.2021 10:38 | Файл "GRADLE" | 1 КБ |

Рисунок 19.5 – Демонстрация работы программы

## Практическая работа №20

### *Цель работы*

Тема: Использование Spring AOP. Pointcut, JoinPoint. Advice.

Постановка задачи: Для приложения из предыдущего задания добавить логирование времени выполнения каждого метода сервиса с использованием Spring AOP.

### *Листинг программы*

*User.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {

    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

*Post.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }
}
```

*UserRepository.java*

```
package app.Application.Interfaces;

import app.Application.Classes.User;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("UserRepository")
public interface UserRepository extends JpaRepository<User,Long> {
    List<User> findAllByFirstName(String firstName);
    List<User> findAllByLastName(String lastName);

    @NotNull List<User> findAll();
    void deleteById(UUID id);
}
```

*PostRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.Post;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("PostRepository")
public interface PostRepository extends JpaRepository<Post,Long> {
    Post findById(UUID id);

    @NotNull List<Post> findAll();
    void deleteById(UUID id);
}
```

*UserService.java*

```java
package app.Application.Services;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class UserService {
    @Autowired
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void addUser(User user) {
        userRepository.save(user);
    }

    public List<User> getUsers() {
        return userRepository.findAll();
    }

    public void deleteUser(UUID id) {
        userRepository.deleteById(id);
    }

    public List<User> getByFirstName(String firstName) {
        return userRepository.findAllByFirstName(firstName);
    }

    public List<User> getByLastName(String lastName) {
        return userRepository.findAllByLastName(lastName);
    }
}
```

*PostService.java*

```java
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Interfaces.PostRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class PostService {
    @Autowired
    private final PostRepository postRepository;

    public PostService(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    public void addPost(Post post) {
        postRepository.save(post);
    }

    public List<Post> getPosts() {
        return postRepository.findAll();
    }

    public void deletePost(UUID id) {
        postRepository.deleteById(id);
    }

    public User getUserByPost(UUID id) {
        return postRepository.findById(id).getUser();
    }
}
```

*UserController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.User;
import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getAll() {
        return userService.getUsers();
    }

    @DeleteMapping("/user/{id}")
    public void delete(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getUserByFirstName/{firstName}")
    public List<User> getByFirstName(@PathVariable String firstName){
        return userService.getByFirstName(firstName);
    }

    @GetMapping("/getUserByLastName/{lastName}")
    public List<User> getByLastName(@PathVariable String lastName){
        return userService.getByLastName(lastName);
    }
}
```

*PostController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Services.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/posts")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void delete(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getGame(@PathVariable("id") UUID id) {
        return postService.getUserByPost(id);
    }
}
```

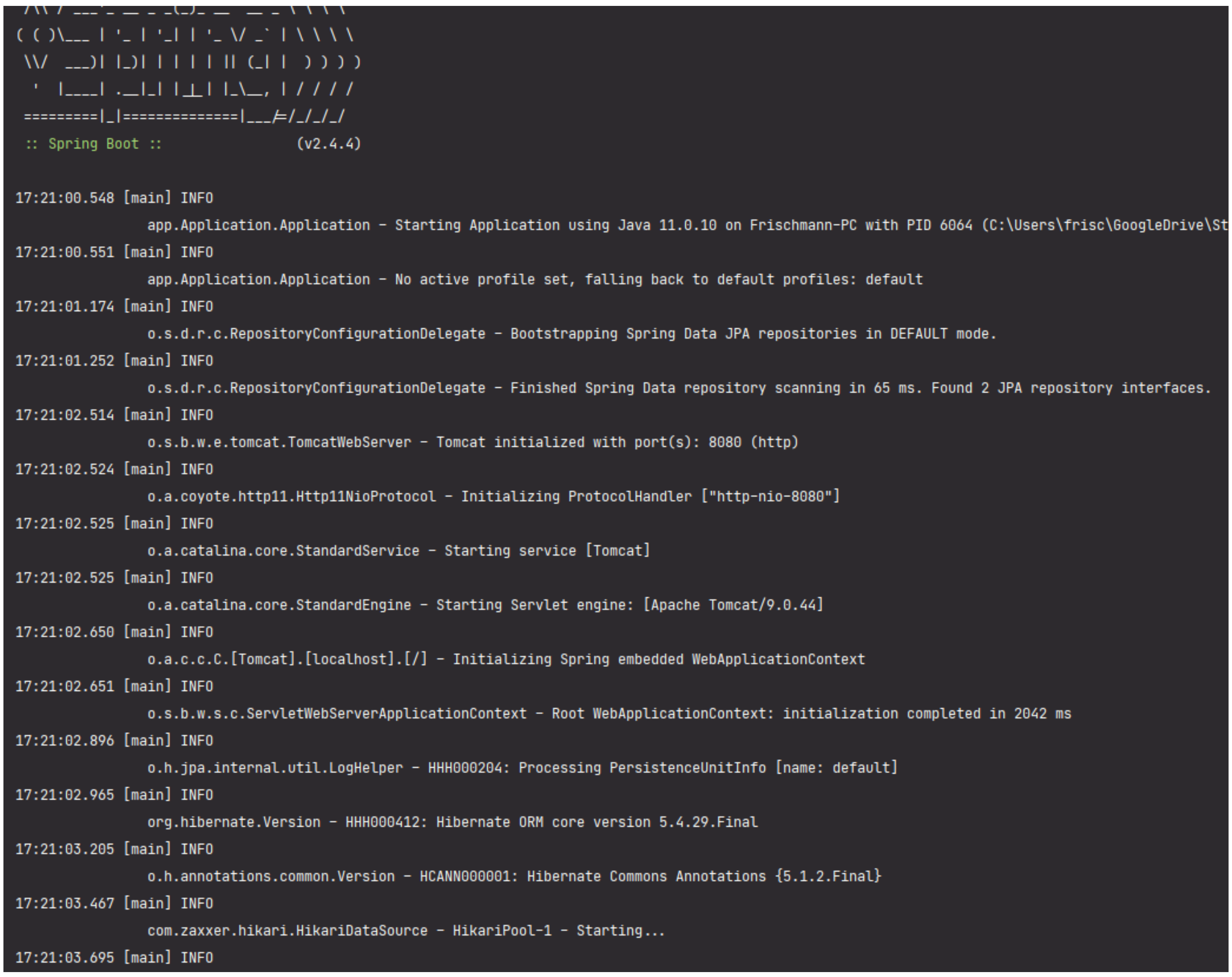
*Config.java*

```java
package app.Application.Configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@Configuration
@EnableAspectJAutoProxy
@EnableJpaRepositories(basePackages = {"app.Application"})
public class Config {
}
```

*Aspect.java*

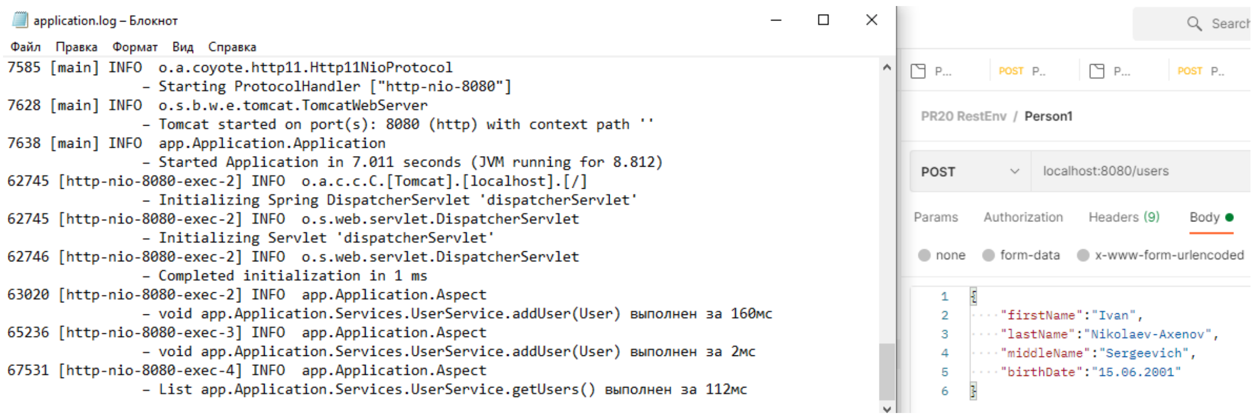
```java
package app.Application;

import lombok.extern.slf4j.Slf4j;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j
@Component
@org.aspectj.lang.annotation.Aspect
public class Aspect {
    private Logger log = Logger.getLogger(Aspect.class.getName());

    @Around("allServiceMethods()")
    public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        long executionTime = System.currentTimeMillis() - start;
        log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");
        return proceed;
    }

    @Pointcut("within(app.Application.Services.*)")
    public void allServiceMethods() {}
}
```

## *Результат выполнения программы*



```
  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
 ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
   '  |____| .__|_| |_|_| |_\__, | / / / /
  =========|_|==============|___/=/_/_/_/
  :: Spring Boot ::                (v2.4.4)

17:21:00.548 [main] INFO
                app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 6064 (C:\Users\frisc\GoogleDrive\St
17:21:00.551 [main] INFO
                app.Application.Application - No active profile set, falling back to default profiles: default
17:21:01.174 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
17:21:01.252 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 65 ms. Found 2 JPA repository interfaces.
17:21:02.514 [main] INFO
                o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
17:21:02.524 [main] INFO
                o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
17:21:02.525 [main] INFO
                o.a.catalina.core.StandardService - Starting service [Tomcat]
17:21:02.525 [main] INFO
                o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
17:21:02.650 [main] INFO
                o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
17:21:02.651 [main] INFO
                o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 2042 ms
17:21:02.896 [main] INFO
                o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
17:21:02.965 [main] INFO
                org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final
17:21:03.205 [main] INFO
                o.h.annotations.common.Version - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
17:21:03.467 [main] INFO
                com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
17:21:03.695 [main] INFO
```

Рисунок 20.1 – Демонстрация работы программы

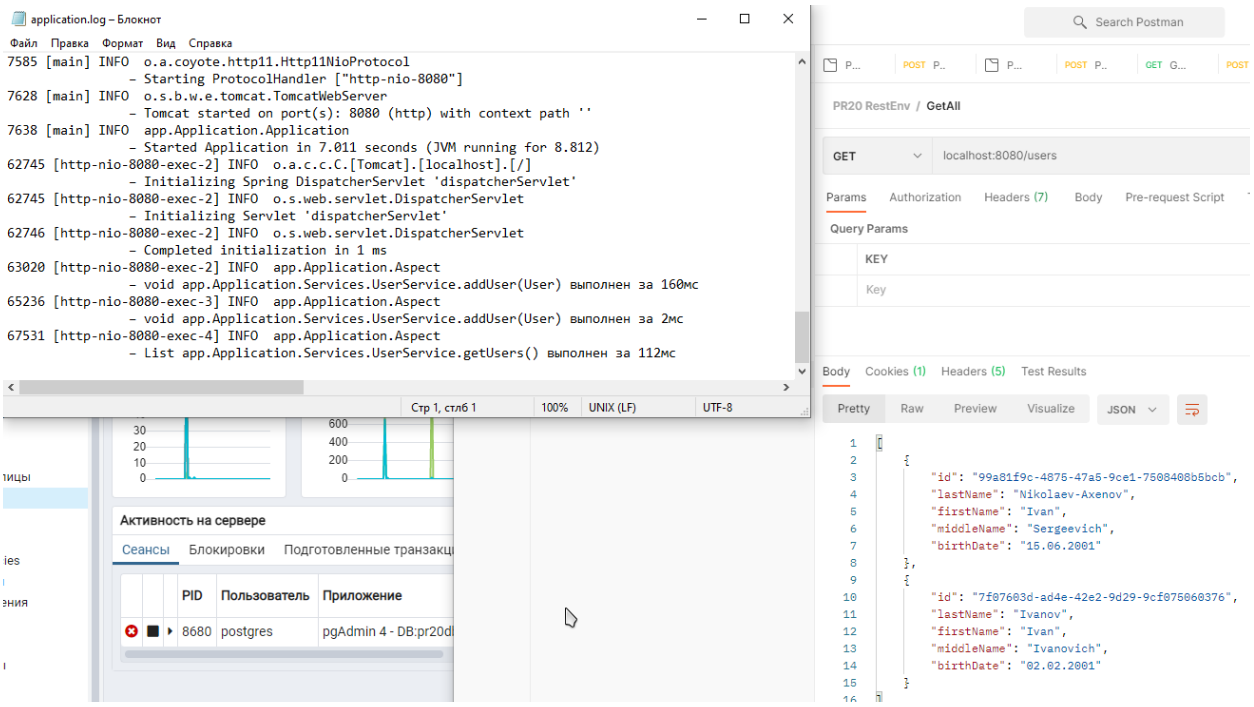


```
7585 [main] INFO  o.a.coyote.http11.Http11NioProtocol
                  - Starting ProtocolHandler ["http-nio-8080"]
7628 [main] INFO  o.s.b.w.e.tomcat.TomcatWebServer
                  - Tomcat started on port(s): 8080 (http) with context path ''
7638 [main] INFO  app.Application.Application
                  - Started Application in 7.011 seconds (JVM running for 8.812)
62745 [http-nio-8080-exec-2] INFO  o.a.c.c.C.[Tomcat].[localhost].[/]
                  - Initializing Spring DispatcherServlet 'dispatcherServlet'
62745 [http-nio-8080-exec-2] INFO  o.s.web.servlet.DispatcherServlet
                  - Initializing Servlet 'dispatcherServlet'
62746 [http-nio-8080-exec-2] INFO  o.s.web.servlet.DispatcherServlet
                  - Completed initialization in 1 ms
63020 [http-nio-8080-exec-2] INFO  app.Application.Aspect
                  - void app.Application.Services.UserService.addUser(User) выполнен за 160мс
65236 [http-nio-8080-exec-3] INFO  app.Application.Aspect
                  - void app.Application.Services.UserService.addUser(User) выполнен за 2мс
67531 [http-nio-8080-exec-4] INFO  app.Application.Aspect
                  - List app.Application.Services.UserService.getUsers() выполнен за 112мс
```

POST localhost:8080/users

```
{
    "firstName":"Ivan",
    "lastName":"Nikolaev-Axenov",
    "middleName":"Sergeevich",
    "birthDate":"15.06.2001"
}
```

Рисунок 20.2 – Демонстрация работы программы



```
7585 [main] INFO  o.a.coyote.http11.Http11NioProtocol
            - Starting ProtocolHandler ["http-nio-8080"]
7628 [main] INFO  o.s.b.w.e.tomcat.TomcatWebServer
            - Tomcat started on port(s): 8080 (http) with context path ''
7638 [main] INFO  app.Application.Application
            - Started Application in 7.011 seconds (JVM running for 8.812)
62745 [http-nio-8080-exec-2] INFO  o.a.c.c.C.[Tomcat].[localhost].[/]
            - Initializing Spring DispatcherServlet 'dispatcherServlet'
62745 [http-nio-8080-exec-2] INFO  o.s.web.servlet.DispatcherServlet
            - Initializing Servlet 'dispatcherServlet'
62746 [http-nio-8080-exec-2] INFO  o.s.web.servlet.DispatcherServlet
            - Completed initialization in 1 ms
63020 [http-nio-8080-exec-2] INFO  app.Application.Aspect
            - void app.Application.Services.UserService.addUser(User) выполнен за 160мс
65236 [http-nio-8080-exec-3] INFO  app.Application.Aspect
            - void app.Application.Services.UserService.addUser(User) выполнен за 2мс
67531 [http-nio-8080-exec-4] INFO  app.Application.Aspect
            - List app.Application.Services.UserService.getUsers() выполнен за 112мс
```

PR20 RestEnv / GetAll

GET localhost:8080/users

| PID | Пользователь | Приложение |
|---|---|---|
| 8680 | postgres | pgAdmin 4 - DB:pr20d |

```
[
    {
        "id": "99a81f9c-4875-47a5-9ce1-7508408b5bcb",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "7f07603d-ad4e-42e2-9d29-9cf075060376",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 20.3 – Демонстрация работы программы

**Практическая работа №21**

***Цель работы***

Тема: Проксирование. Аннотация Transactional. Аннотация Async.

Постановка задачи: Для приложения из предыдущего задания пометить все классы сервисов, в которых происходит взаимодействие с базой данных, как Transactional. Добавить отправку информации о сохранении каждого объекта по электронной почте, создав отдельный класс EmailService с асинхронными методами отправки сообщений. Для асинхронности методов используйте аннотацию Async.

***Листинг программы***

*User.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {
    }

    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "Пользователь #" + id + " " + lastName + " " + firstName + " " +
middleName + ", день рождения: " + birthDate;
    }
}
```

*Post.java*

```
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
```

```java
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(String text) {
        this.text = text;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }
}
```

*UserRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.User;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("UserRepository")
public interface UserRepository extends JpaRepository<User,Long> {
    List<User> findAllByFirstName(String firstName);
    List<User> findAllByLastName(String lastName);

    @NotNull List<User> findAll();
    void deleteById(UUID id);
}
```

*PostRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.Post;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("PostRepository")
public interface PostRepository extends JpaRepository<Post,Long> {
    Post findById(UUID id);

    @NotNull List<Post> findAll();
    void deleteById(UUID id);
}
```

*UserService.java*

```java
package app.Application.Services;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class UserService {
    @Autowired
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void addUser(User user) {
        userRepository.save(user);
    }

    public List<User> getUsers() {
        return userRepository.findAll();
    }

    public void deleteUser(UUID id) {
        userRepository.deleteById(id);
    }

    public List<User> getByFirstName(String firstName) {
        return userRepository.findAllByFirstName(firstName);
    }

    public List<User> getByLastName(String lastName) {
        return userRepository.findAllByLastName(lastName);
    }
}
```

*PostService.java*

```java
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Interfaces.PostRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class PostService {
    @Autowired
    private final PostRepository postRepository;

    public PostService(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    public void addPost(Post post) {
        postRepository.save(post);
    }

    public List<Post> getPosts() {
        return postRepository.findAll();
    }

    public void deletePost(UUID id) {
        postRepository.deleteById(id);
    }

    public User getUserByPost(UUID id) {
        return postRepository.findById(id).getUser();
    }
}
```

*UserController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.User;
import app.Application.EmailService;
import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @Autowired
    private EmailService emailService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getAll() {
        emailService.SendEmail();
        return userService.getUsers();
    }

    @DeleteMapping("/user/{id}")
    public void delete(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getUserByFirstName/{firstName}")
    public List<User> getByFirstName(@PathVariable String firstName){
        return userService.getByFirstName(firstName);
    }

    @GetMapping("/getUserByLastName/{lastName}")
    public List<User> getByLastName(@PathVariable String lastName){
        return userService.getByLastName(lastName);
    }
}
```

*PostController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Services.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/posts")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void delete(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getGame(@PathVariable("id") UUID id) {
        return postService.getUserByPost(id);
    }
}
```

*Config.java*

```java
package app.Application.Configuration;

import app.Application.EmailService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.scheduling.annotation.EnableAsync;

import java.util.Properties;

@Configuration
@EnableAspectJAutoProxy
@EnableJpaRepositories(basePackages = {"app.Application"})
@EnableAsync
public class Config {
    @Bean
    public JavaMailSender getJavaMailSender() {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
        mailSender.setHost("smtp.mail.ru");
        mailSender.setPort(465);

        mailSender.setUsername("loremail@bk.ru");
        mailSender.setPassword("secret");

        Properties props = mailSender.getJavaMailProperties();
        props.put("mail.transport.protocol", "smtps");
        props.put("mail.smtp.auth", "true");
        props.put("smtp.ssl.enable", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.debug", "true");

        return mailSender;
    }
    @Bean
    public EmailService getEmailService(){
        return new EmailService();
    }
}
```

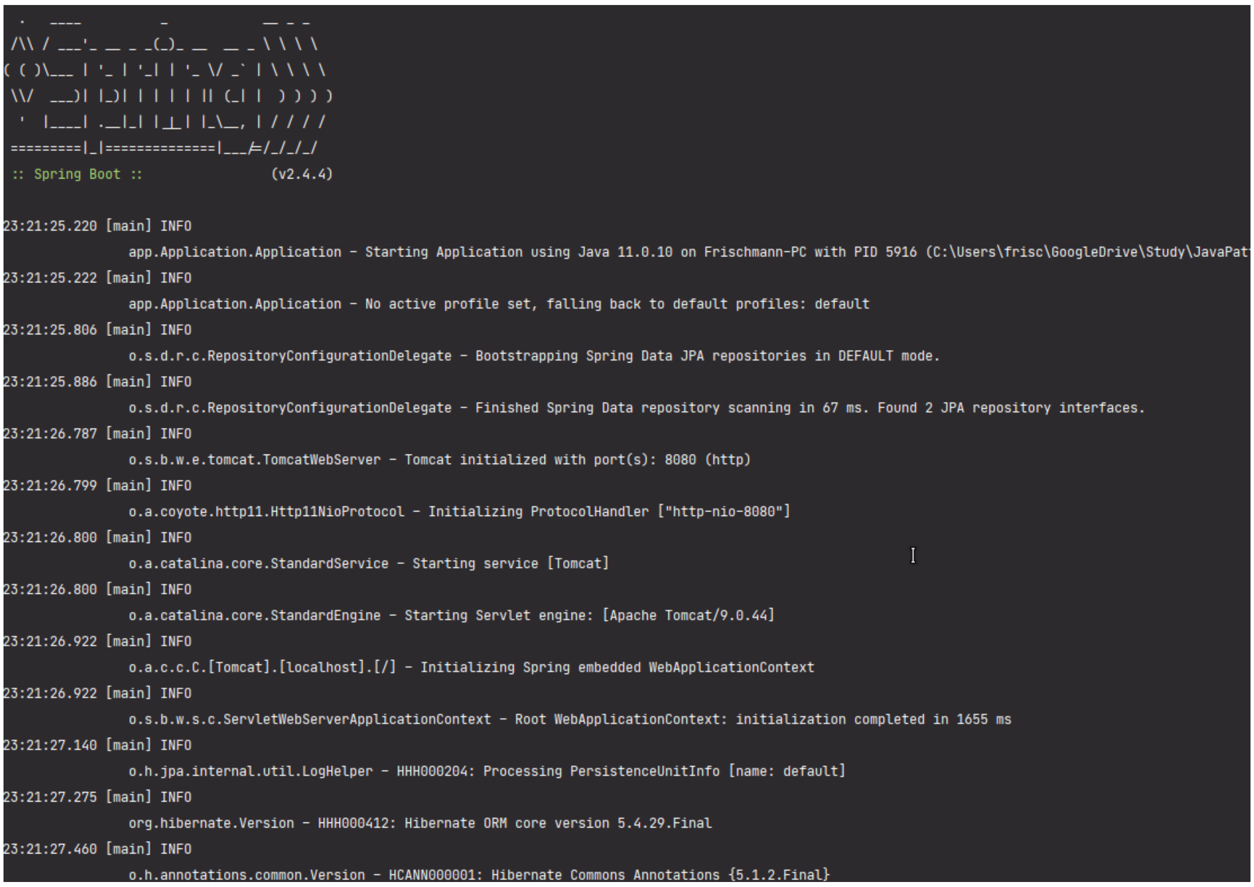
*Aspect.java*

```java
package app.Application;

import lombok.extern.slf4j.Slf4j;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j
@Component
@org.aspectj.lang.annotation.Aspect
public class Aspect {
    private Logger log = Logger.getLogger(Aspect.class.getName());

    @Around("allServiceMethods()")
    public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        long executionTime = System.currentTimeMillis() - start;
        log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");
        return proceed;
    }

    @Pointcut("within(app.Application.Services.*)")
    public void allServiceMethods() {}
}
```

*EmailService.java*

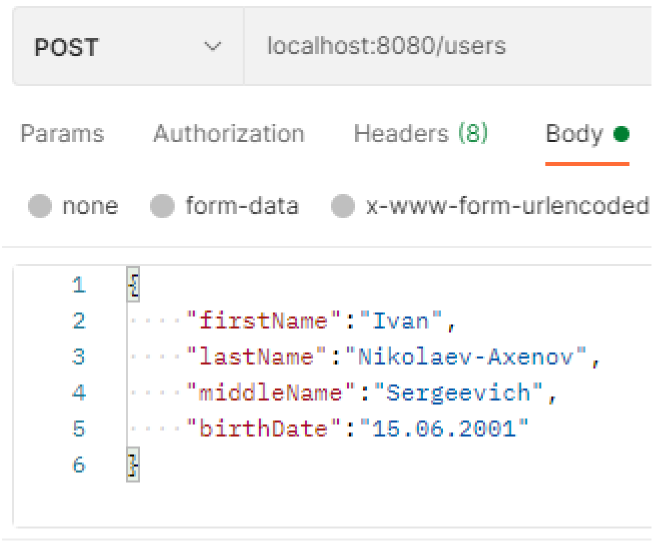
```java
package app.Application;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.scheduling.annotation.Async;

public class EmailService {
    @Autowired
    public JavaMailSender emailSender;

    @Async
    public void SendEmail(){
        SimpleMailMessage message = new SimpleMailMessage();

        message.setFrom("loremeemail@bk.ru");
        message.setTo("ghost777t@ya.ru");
        message.setSubject("Test email message");
        message.setText("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla
feugiat eget sapien sed lacinia.");

        this.emailSender.send(message);
        System.out.println("Email successfully sent!");
    }
}
```

## *Результат выполнения программы*



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

23:21:25.220 [main] INFO
                app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 5916 (C:\Users\frisc\GoogleDrive\Study\JavaPat
23:21:25.222 [main] INFO
                app.Application.Application - No active profile set, falling back to default profiles: default
23:21:25.806 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
23:21:25.886 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 67 ms. Found 2 JPA repository interfaces.
23:21:26.787 [main] INFO
                o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
23:21:26.799 [main] INFO
                o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
23:21:26.800 [main] INFO
                o.a.catalina.core.StandardService - Starting service [Tomcat]
23:21:26.800 [main] INFO
                o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
23:21:26.922 [main] INFO
                o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
23:21:26.922 [main] INFO
                o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1655 ms
23:21:27.140 [main] INFO
                o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
23:21:27.275 [main] INFO
                org.hibernate.Version - HHH000412: Hibernate ORM core version 5.4.29.Final
23:21:27.460 [main] INFO
                o.h.annotations.common.Version - HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
```

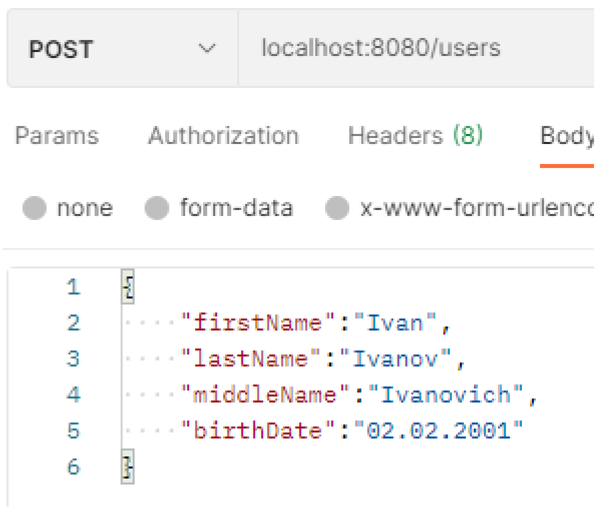
Рисунок 21.1 – Демонстрация работы программы



POST localhost:8080/users

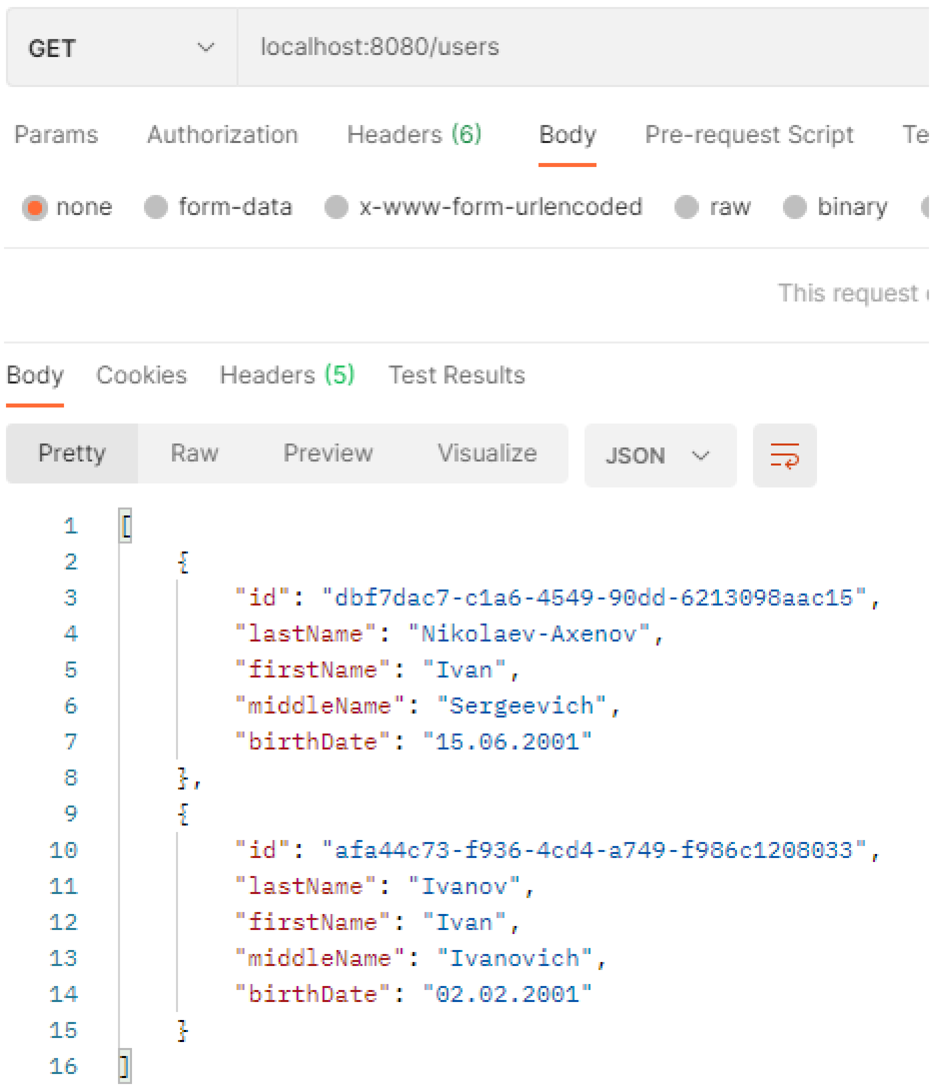
```
{
    "firstName":"Ivan",
    "lastName":"Nikolaev-Axenov",
    "middleName":"Sergeevich",
    "birthDate":"15.06.2001"
}
```

Рисунок 21.2 – Демонстрация работы программы



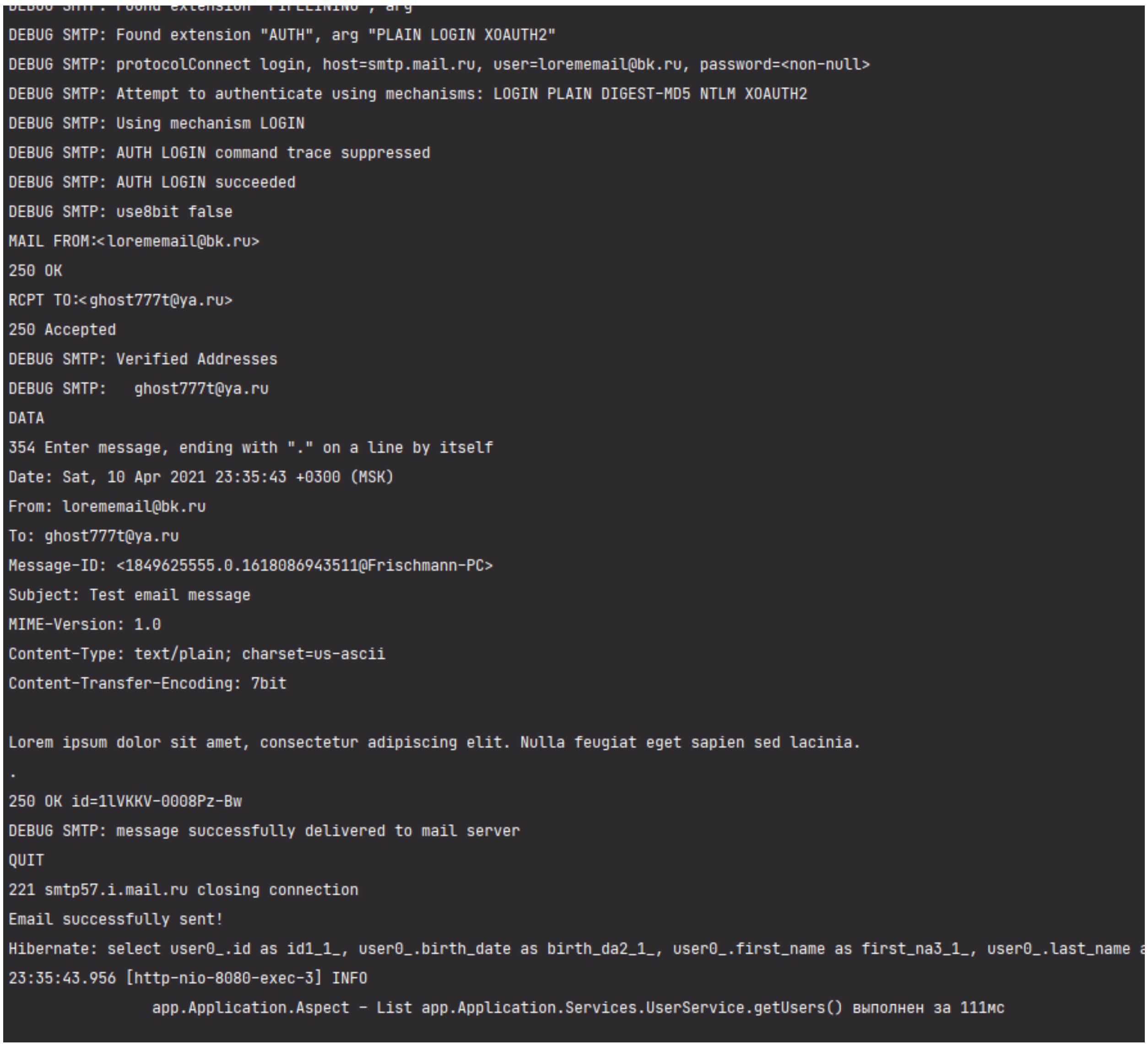
```
{
    "firstName":"Ivan",
    "lastName":"Ivanov",
    "middleName":"Ivanovich",
    "birthDate":"02.02.2001"
}
```

Рисунок 21.3 – Демонстрация работы программы



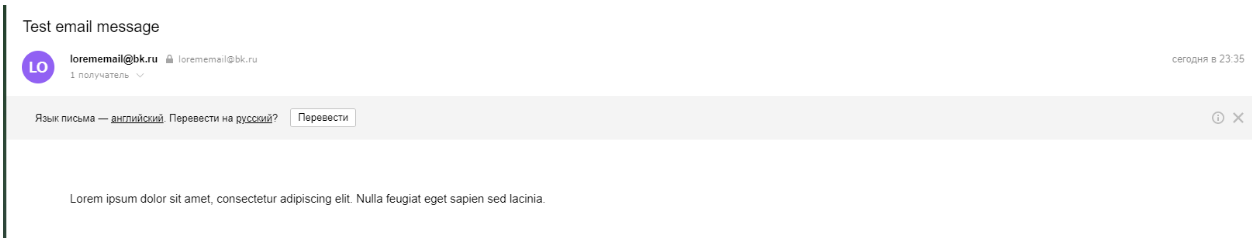
```
[
    {
        "id": "dbf7dac7-c1a6-4549-90dd-6213098aac15",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "afa44c73-f936-4cd4-a749-f986c1208033",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "02.02.2001"
    }
]
```

Рисунок 21.4 – Демонстрация работы программы



```
DEBUG SMTP: Found extension "PIPELINING", arg
DEBUG SMTP: Found extension "AUTH", arg "PLAIN LOGIN XOAUTH2"
DEBUG SMTP: protocolConnect login, host=smtp.mail.ru, user=lorememail@bk.ru, password=<non-null>
DEBUG SMTP: Attempt to authenticate using mechanisms: LOGIN PLAIN DIGEST-MD5 NTLM XOAUTH2
DEBUG SMTP: Using mechanism LOGIN
DEBUG SMTP: AUTH LOGIN command trace suppressed
DEBUG SMTP: AUTH LOGIN succeeded
DEBUG SMTP: use8bit false
MAIL FROM:<lorememail@bk.ru>
250 OK
RCPT TO:<ghost777t@ya.ru>
250 Accepted
DEBUG SMTP: Verified Addresses
DEBUG SMTP:   ghost777t@ya.ru
DATA
354 Enter message, ending with "." on a line by itself
Date: Sat, 10 Apr 2021 23:35:43 +0300 (MSK)
From: lorememail@bk.ru
To: ghost777t@ya.ru
Message-ID: <1849625555.0.1618086943511@Frischmann-PC>
Subject: Test email message
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla feugiat eget sapien sed lacinia.
.
250 OK id=1lVKKV-0008Pz-Bw
DEBUG SMTP: message successfully delivered to mail server
QUIT
221 smtp57.i.mail.ru closing connection
Email successfully sent!
Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name a
23:35:43.956 [http-nio-8080-exec-3] INFO
                app.Application.Aspect - List app.Application.Services.UserService.getUsers() выполнен за 111мс
```

Рисунок 21.5 – Демонстрация работы программы



Test email message

LO lorememail@bk.ru lorememail@bk.ru сегодня в 23:35

1 получатель

Язык письма — английский. Перевести на русский? Перевести

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla feugiat eget sapien sed lacinia.

Рисунок 21.6 – Демонстрация работы программы

## Практическая работа №22

### *Цель работы*

Тема: Планирование заданий. Scheduler в Spring.

Постановка задачи: Для приложения из предыдущего задания создать класс-сервис с методом, который будет вызываться каждые 30 минут и очищать определённую директорию, а затем создавать по файлу для каждой из сущностей и загружать туда все данные из базы данных. Также добавить возможность вызывать данный метод с использованием Java Management Extensions (JMX).

### *Листинг программы*

*User.java*

```java
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;
import java.util.*;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "last_name")
    @NotNull
    private String lastName;

    @Column(name = "first_name")
    @NotNull
    private String firstName;

    @Column(name = "middle_name")
    @NotNull
    private String middleName;

    @Column(name = "birth_date")
    @NotNull
    private String birthDate;

    @OneToMany(mappedBy = "user")
    private List<Post> posts = new ArrayList<>();

    public User() {

    }
    public User(String lastName, String firstName, String middleName, String birthDate)
{
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.birthDate = birthDate;
    }

    public UUID getId() {
        return id;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", lastName='" + lastName + '\'' +
                ", firstName='" + firstName + '\'' +
                ", middleName='" + middleName + '\'' +
                ", birthDate='" + birthDate + '\'' +
                '}';
    }
}
```

*Post.java*

```java
package app.Application.Classes;

import com.sun.istack.NotNull;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "posts")
public class Post {
    @Id
    @GeneratedValue(generator = "UUID")
    @GenericGenerator(name = "UUID", strategy = "org.hibernate.id.UUIDGenerator")
    @Column(name = "id", updatable = false, nullable = false)
    private UUID id;

    @Column(name = "text")
    @NotNull
    private String text;

    @CreationTimestamp
    @Column(name = "creation_date")
    private LocalDateTime creationDate;

    @ManyToOne
    private User user;

    public Post() {

    }

    public Post(User user, String text) {
        this.text = text;
        this.user = user;
    }

    public UUID getId() {
        return id;
    }

    public String getText() {
        return text;
    }

    public LocalDateTime getCreationDate() {
        return creationDate;
    }

    public User getUser() {
        return user;
    }

    @Override
    public String toString() {
        return "Post{" +
                "id=" + id +
                ", text='" + text + '\'' +
                ", creationDate=" + creationDate +
                ", user=" + user +
                '}';
    }
}
```

*UserRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.User;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("UserRepository")
public interface UserRepository extends JpaRepository<User,Long> {
    List<User> findAllByFirstName(String firstName);
    List<User> findAllByLastName(String lastName);

    @NotNull List<User> findAll();
    void deleteById(UUID id);
}
```

*PostRepository.java*

```java
package app.Application.Interfaces;

import app.Application.Classes.Post;
import com.sun.istack.NotNull;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository("PostRepository")
public interface PostRepository extends JpaRepository<Post,Long> {
    Post findById(UUID id);

    List<Post> findAll();
    void deleteById(UUID id);
}
```

*UserService.java*

```java
package app.Application.Services;

import app.Application.Classes.User;
import app.Application.Interfaces.UserRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class UserService {
    @Autowired
    private final UserRepository userRepository;

    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public void addUser(User user) {
        userRepository.save(user);
    }

    public List<User> getUsers() {
        return userRepository.findAll();
    }

    public void deleteUser(UUID id) {
        userRepository.deleteById(id);
    }

    public List<User> getByFirstName(String firstName) {
        return userRepository.findAllByFirstName(firstName);
    }

    public List<User> getByLastName(String lastName) {
        return userRepository.findAllByLastName(lastName);
    }
}
```

*PostService.java*

```java
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Interfaces.PostRepository;
import lombok.extern.slf4j.Slf4j;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.util.List;
import java.util.UUID;

@Service
@Slf4j
public class PostService {
    @Autowired
    private final PostRepository postRepository;

    public PostService(PostRepository postRepository) {
        this.postRepository = postRepository;
    }

    public void addPost(Post post) {
        postRepository.save(post);
    }

    public List<Post> getPosts() {
        return postRepository.findAll();
    }

    public void deletePost(UUID id) {
        postRepository.deleteById(id);
    }

    public User getUserByPost(UUID id) {
        return postRepository.findById(id).getUser();
    }
}
```

*UserController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.User;
import app.Application.Services.EmailService;
import app.Application.Services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

    @Autowired
    private EmailService emailService;

    @PostMapping("/users")
    public void addUser(@RequestBody User user) {
        userService.addUser(user);
    }

    @GetMapping("/users")
    public List<User> getAll() {
        //emailService.SendEmail();
        return userService.getUsers();
    }

    @DeleteMapping("/user/{id}")
    public void delete(@PathVariable UUID id) {
        userService.deleteUser(id);
    }

    @GetMapping("/getUserByFirstName/{firstName}")
    public List<User> getByFirstName(@PathVariable String firstName){
        return userService.getByFirstName(firstName);
    }

    @GetMapping("/getUserByLastName/{lastName}")
    public List<User> getByLastName(@PathVariable String lastName){
        return userService.getByLastName(lastName);
    }
}
```

*PostController.java*

```java
package app.Application.Controllers;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Services.PostService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
public class PostController {
    @Autowired
    private PostService postService;

    @PostMapping("/posts")
    public void addPost(@RequestBody Post post) {
        postService.addPost(post);
    }

    @GetMapping("/posts")
    public List<Post> getAll() {
        return postService.getPosts();
    }

    @DeleteMapping("/post/{id}")
    public void delete(@PathVariable UUID id) {
        postService.deletePost(id);
    }

    @GetMapping(value = "/post/{id}/user")
    public @ResponseBody
    User getGame(@PathVariable("id") UUID id) {
        return postService.getUserByPost(id);
    }
}
```

*Config.java*

```java
package app.Application.Configuration;

import app.Application.Services.EmailService;
import app.Application.Services.Schedule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.annotation.EnableScheduling;

import java.util.Properties;

@Configuration
@EnableAspectJAutoProxy
@EnableJpaRepositories(basePackages = {"app.Application"})
```

```
@EnableAsync
@EnableScheduling
public class Config {
    @Bean
    public JavaMailSender getJavaMailSender() {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
        mailSender.setHost("smtp.mail.ru");
        mailSender.setPort(465);

        mailSender.setUsername("lorememail@bk.ru");
        mailSender.setPassword("secret");

        Properties props = mailSender.getJavaMailProperties();
        props.put("mail.transport.protocol", "smtps");
        props.put("mail.smtp.auth", "true");
        props.put("smtp.ssl.enable", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.debug", "true");

        return mailSender;
    }

    @Bean
    public EmailService getEmailService(){
        return new EmailService();
    }
}
```

*Aspect.java*

```
package app.Application;

import lombok.extern.slf4j.Slf4j;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

import java.util.logging.Logger;

@Slf4j
@Component
@org.aspectj.lang.annotation.Aspect
public class Aspect {
    private Logger log = Logger.getLogger(Aspect.class.getName());

    @Around("allServiceMethods()")
    public Object logExecutionTime (ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object proceed = joinPoint.proceed();
        long executionTime = System.currentTimeMillis() - start;
        log.info(joinPoint.getSignature() + " выполнен за " + executionTime + "мс");
        return proceed;
    }

    @Pointcut("within(app.Application.Services.*)")
    public void allServiceMethods() {}
}
```

*EmailService.java*

```java
package app.Application.Services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.scheduling.annotation.Async;

public class EmailService {
    @Autowired
    public JavaMailSender emailSender;

    @Async
    public void SendEmail(){
        SimpleMailMessage message = new SimpleMailMessage();

        message.setFrom("lorememail@bk.ru");
        message.setTo("ghost777t@ya.ru");
        message.setSubject("Test email message");
        message.setText("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla
feugiat eget sapien sed lacinia.");

        this.emailSender.send(message);
        System.out.println("Email successfully sent!");
    }
}
```

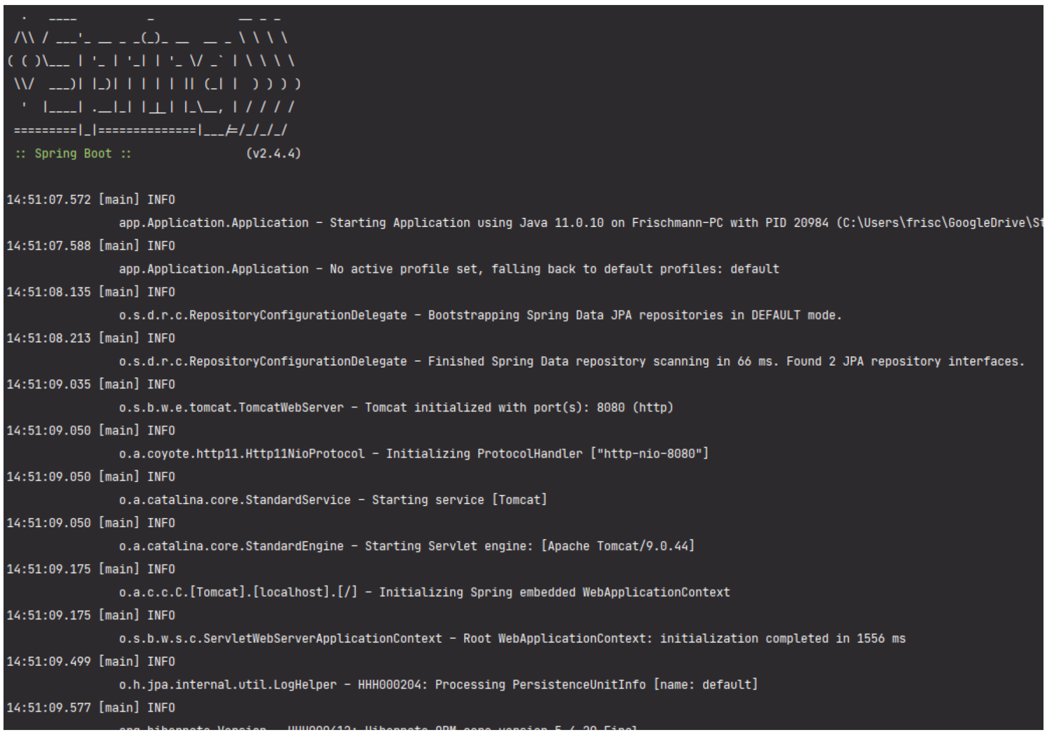
*ScheduleMXBean.java*

```java
package app.Application;

import java.io.IOException;
import app.Application.Interfaces.UserRepository;
import app.Application.Interfaces.PostRepository;

public interface ScheduleMXBean {
    void doScheduledTask() throws IOException;
}
```

*Schedule.java*

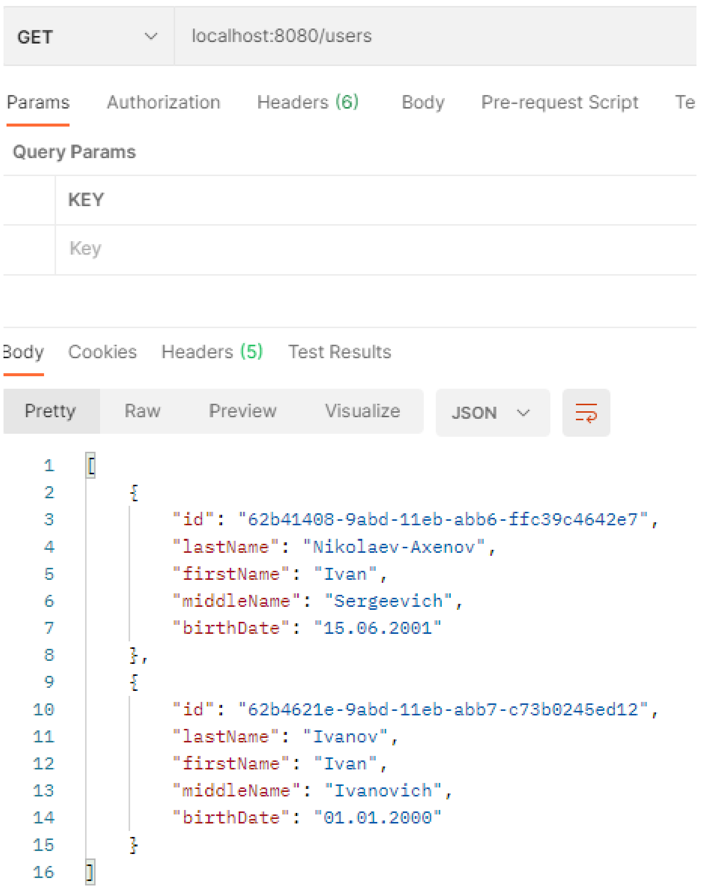
```java
package app.Application.Services;

import app.Application.Classes.Post;
import app.Application.Classes.User;
import app.Application.Controllers.PostController;
import app.Application.Controllers.UserController;
import app.Application.Interfaces.PostRepository;
import app.Application.Interfaces.UserRepository;
import app.Application.ScheduleMXBean;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jmx.export.annotation.ManagedOperation;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Service;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;
import java.util.Objects;

@Service
public class Schedule implements ScheduleMXBean {
    @Autowired
    private final UserRepository userRepository;

    @Autowired
    private final PostRepository postRepository;

    public Schedule(UserRepository userRepository, PostRepository postRepository) {
        this.userRepository = userRepository;
        this.postRepository = postRepository;
    }

    private Boolean isEmpty(final File file) {
        return (file.isDirectory() && (file.list().length > 0));
    }

    @ManagedOperation
    @Scheduled(cron = "0 0/2 * * * *")
    public void doScheduledTask() throws IOException {
        if(isEmpty(new
File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа
№22\\testDirectory"))){
            for (File myFile : new
File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа
№22\\testDirectory").listFiles()) {
                if (myFile.isFile()) myFile.delete();
            }
        }

        List <Post> posts = postRepository.findAll();
        List <User> users = userRepository.findAll();

        for (int i = 0; i < users.size(); i++) {
            File user = new
File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа
№22\\testDirectory\\user_" + i + ".txt");
            FileWriter writer = new FileWriter(user, true);
            System.out.println(users.get(i).toString());
            writer.write(users.get(i).toString());
            writer.close();
        }

        for (int i = 0; i < posts.size(); i++) {
            File post = new
File("C:\\Users\\frisc\\GoogleDrive\\Study\\JavaPatterns\\Практическая работа
№22\\testDirectory\\post_" + i + ".txt");
            FileWriter writer = new FileWriter(post, true);
            writer.write(posts.get(i).toString());
            writer.close();
        }
    }
}
```

***Результат выполнения программы***



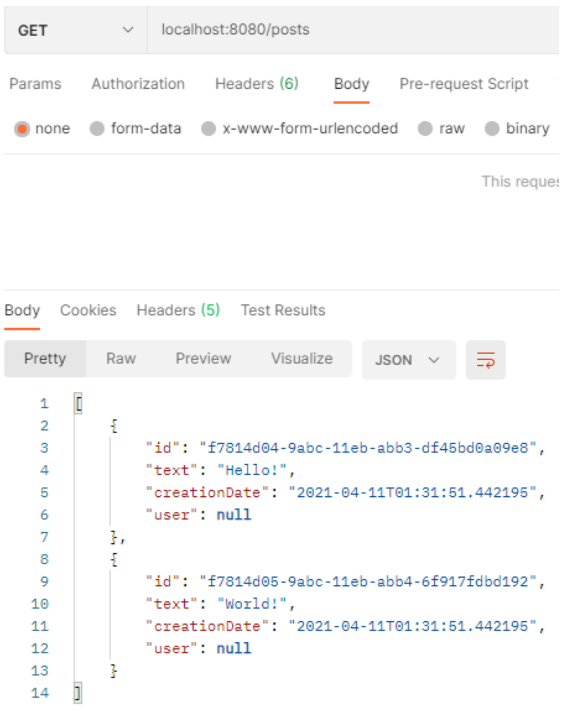
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.4)

14:51:07.572 [main] INFO
                app.Application.Application - Starting Application using Java 11.0.10 on Frischmann-PC with PID 20984 (C:\Users\frisc\GoogleDrive\St
14:51:07.588 [main] INFO
                app.Application.Application - No active profile set, falling back to default profiles: default
14:51:08.135 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Bootstrapping Spring Data JPA repositories in DEFAULT mode.
14:51:08.213 [main] INFO
                o.s.d.r.c.RepositoryConfigurationDelegate - Finished Spring Data repository scanning in 66 ms. Found 2 JPA repository interfaces.
14:51:09.035 [main] INFO
                o.s.b.w.e.tomcat.TomcatWebServer - Tomcat initialized with port(s): 8080 (http)
14:51:09.050 [main] INFO
                o.a.coyote.http11.Http11NioProtocol - Initializing ProtocolHandler ["http-nio-8080"]
14:51:09.050 [main] INFO
                o.a.catalina.core.StandardService - Starting service [Tomcat]
14:51:09.050 [main] INFO
                o.a.catalina.core.StandardEngine - Starting Servlet engine: [Apache Tomcat/9.0.44]
14:51:09.175 [main] INFO
                o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext
14:51:09.175 [main] INFO
                o.s.b.w.s.c.ServletWebServerApplicationContext - Root WebApplicationContext: initialization completed in 1556 ms
14:51:09.499 [main] INFO
                o.h.jpa.internal.util.LogHelper - HHH000204: Processing PersistenceUnitInfo [name: default]
14:51:09.577 [main] INFO
```

Рисунок 22.1 – Демонстрация работы программы



```
[
    {
        "id": "62b41408-9abd-11eb-abb6-ffc39c4642e7",
        "lastName": "Nikolaev-Axenov",
        "firstName": "Ivan",
        "middleName": "Sergeevich",
        "birthDate": "15.06.2001"
    },
    {
        "id": "62b4621e-9abd-11eb-abb7-c73b0245ed12",
        "lastName": "Ivanov",
        "firstName": "Ivan",
        "middleName": "Ivanovich",
        "birthDate": "01.01.2000"
    }
]
```

Рисунок 22.2 – Демонстрация работы программы



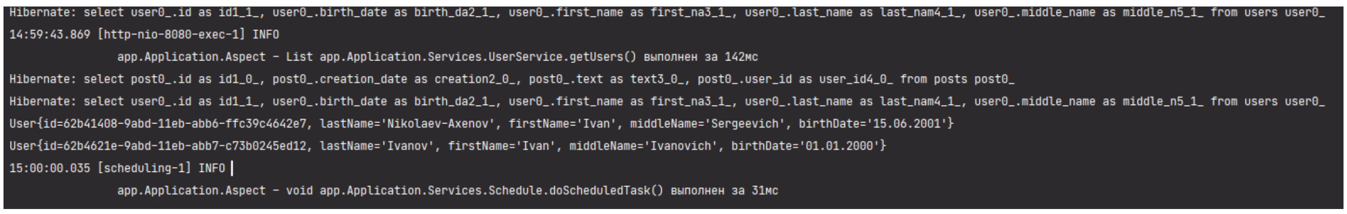
GET localhost:8080/posts

Body: none

This reque

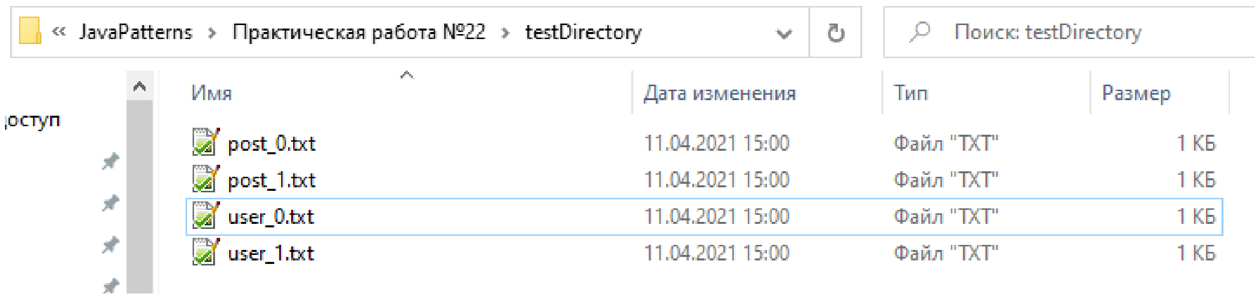
```
[
    {
        "id": "f7814d04-9abc-11eb-abb3-df45bd0a09e8",
        "text": "Hello!",
        "creationDate": "2021-04-11T01:31:51.442195",
        "user": null
    },
    {
        "id": "f7814d05-9abc-11eb-abb4-6f917fdbd192",
        "text": "World!",
        "creationDate": "2021-04-11T01:31:51.442195",
        "user": null
    }
]
```

Рисунок 22.3 – Демонстрация работы программы



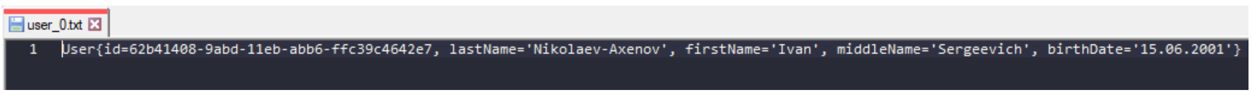
```
Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name as last_nam4_1_, user0_.middle_name as middle_n5_1_ from users user0_
14:59:43.869 [http-nio-8080-exec-1] INFO
                app.Application.Aspect - List app.Application.Services.UserService.getUsers() выполнен за 142мс
Hibernate: select post0_.id as id1_0_, post0_.creation_date as creation2_0_, post0_.text as text3_0_, post0_.user_id as user_id4_0_ from posts post0_
Hibernate: select user0_.id as id1_1_, user0_.birth_date as birth_da2_1_, user0_.first_name as first_na3_1_, user0_.last_name as last_nam4_1_, user0_.middle_name as middle_n5_1_ from users user0_
User{id=62b41408-9abd-11eb-abb6-ffc39c4642e7, lastName='Nikolaev-Axenov', firstName='Ivan', middleName='Sergeevich', birthDate='15.06.2001'}
User{id=62b4621e-9abd-11eb-abb7-c73b0245ed12, lastName='Ivanov', firstName='Ivan', middleName='Ivanovich', birthDate='01.01.2000'}
15:00:00.035 [scheduling-1] INFO
                app.Application.Aspect - void app.Application.Services.Schedule.doScheduledTask() выполнен за 31мс
```

Рисунок 22.4 – Демонстрация работы программы



| Имя | Дата изменения | Тип | Размер |
|---|---|---|---|
| post_0.txt | 11.04.2021 15:00 | Файл "TXT" | 1 КБ |
| post_1.txt | 11.04.2021 15:00 | Файл "TXT" | 1 КБ |
| user_0.txt | 11.04.2021 15:00 | Файл "TXT" | 1 КБ |
| user_1.txt | 11.04.2021 15:00 | Файл "TXT" | 1 КБ |

Рисунок 22.5 – Демонстрация работы программы



```
User{id=62b41408-9abd-11eb-abb6-ffc39c4642e7, lastName='Nikolaev-Axenov', firstName='Ivan', middleName='Sergeevich', birthDate='15.06.2001'}
```

Рисунок 22.6 – Демонстрация работы программы

## Вывод

В ходе выполнения данных практических работ были получены навыки работы с основными технологиями, необходимыми для создания клиент-серверных приложений. Также были получены навыки работы с фреймворком Spring.

## Список использованных источников

1. Стелтинг С., Маасен О. Применение шаблонов Java. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом "Вильяме", 2002. — 576 с.: ил. — Парал. тит. англ.
2. Functional Interfaces in Java: Fundamentals and Examples 1st ed. Edition, Kindle Edition [Электронный ресурс]. URL: https://www.amazon.com/Functional-Interfaces-Java-Fundamentals-Examples-ebook/dp/B07NRHQSCW (дата обращения: 29.01.21). Заголовок с экрана.
3. Hibernate Search 6.0.0.Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html_single/ (дата обращения: 29.01.21). Заголовок с экрана.
4. Паттерны проектирования на Java. Каталог Java-примеров. [Электронный ресурс]. URL: https://refactoring.guru/ru/design-patterns/java (дата обращения: 29.01.21). Заголовок с экрана.
5. Руководство по Spring [Электронный ресурс]. URL: https://proselyte.net/tutorials/spring-tutorial-full-version/ (дата обращения: 29.01.21). Заголовок с экрана.
6. The Reactive Manifesto [Электронный ресурс]. URL: https://www.reactivemanifesto.org/ (дата обращения: 29.01.21). Заголовок с экрана.
7. Spring Framework Documentation [Электронный ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html (дата обращения: 29.01.21). Заголовок с экрана.
8. Hibernate Search 6.0.0. Final: Reference Documentation [Электронный ресурс]. URL: https://docs.jboss.org/hibernate/stable/search/reference/en-US/html_single/ (дата обращения: 29.01.21). Заголовок с экрана.