Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО».

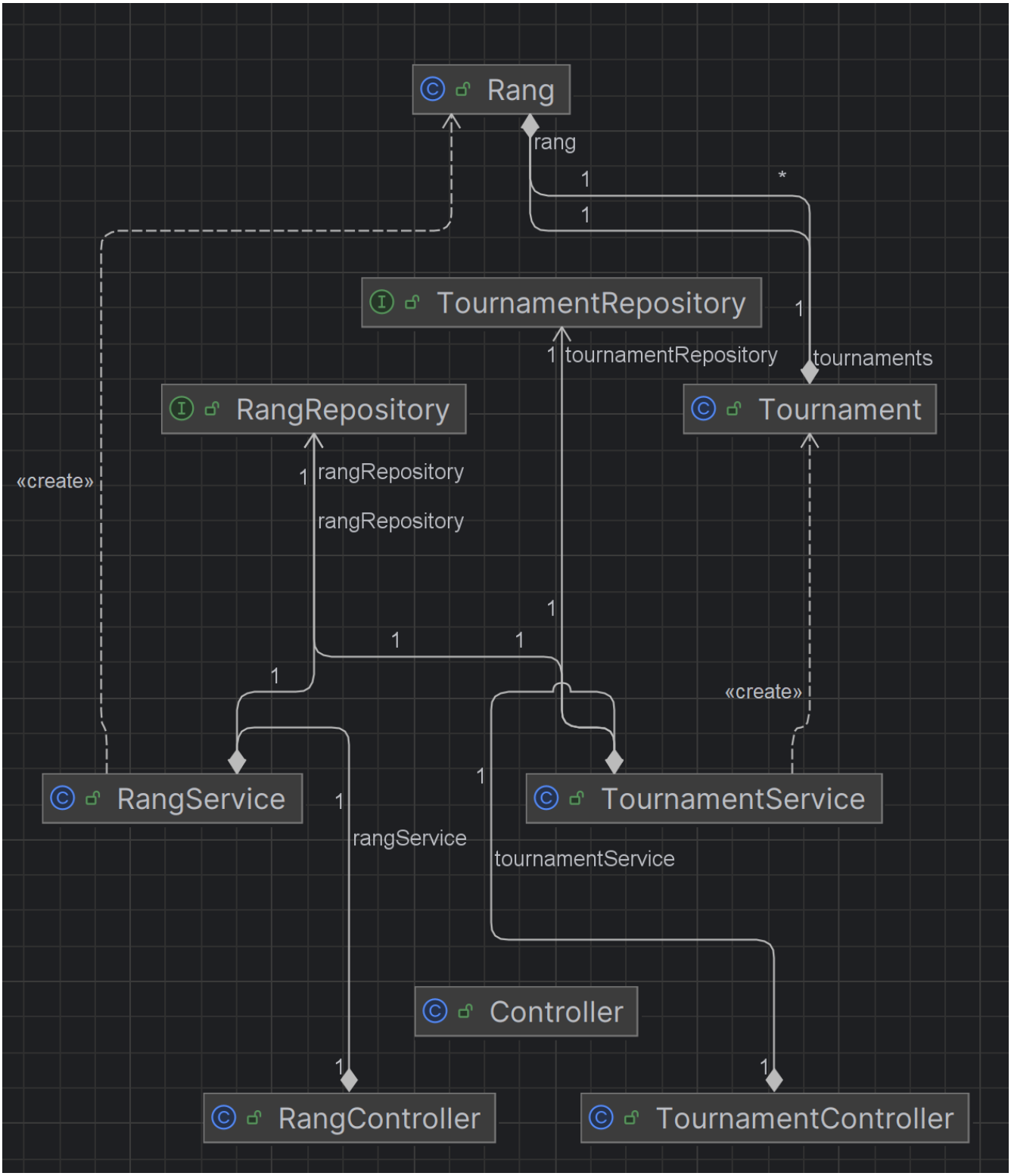
Факультет программной инженерии и компьютерной техники

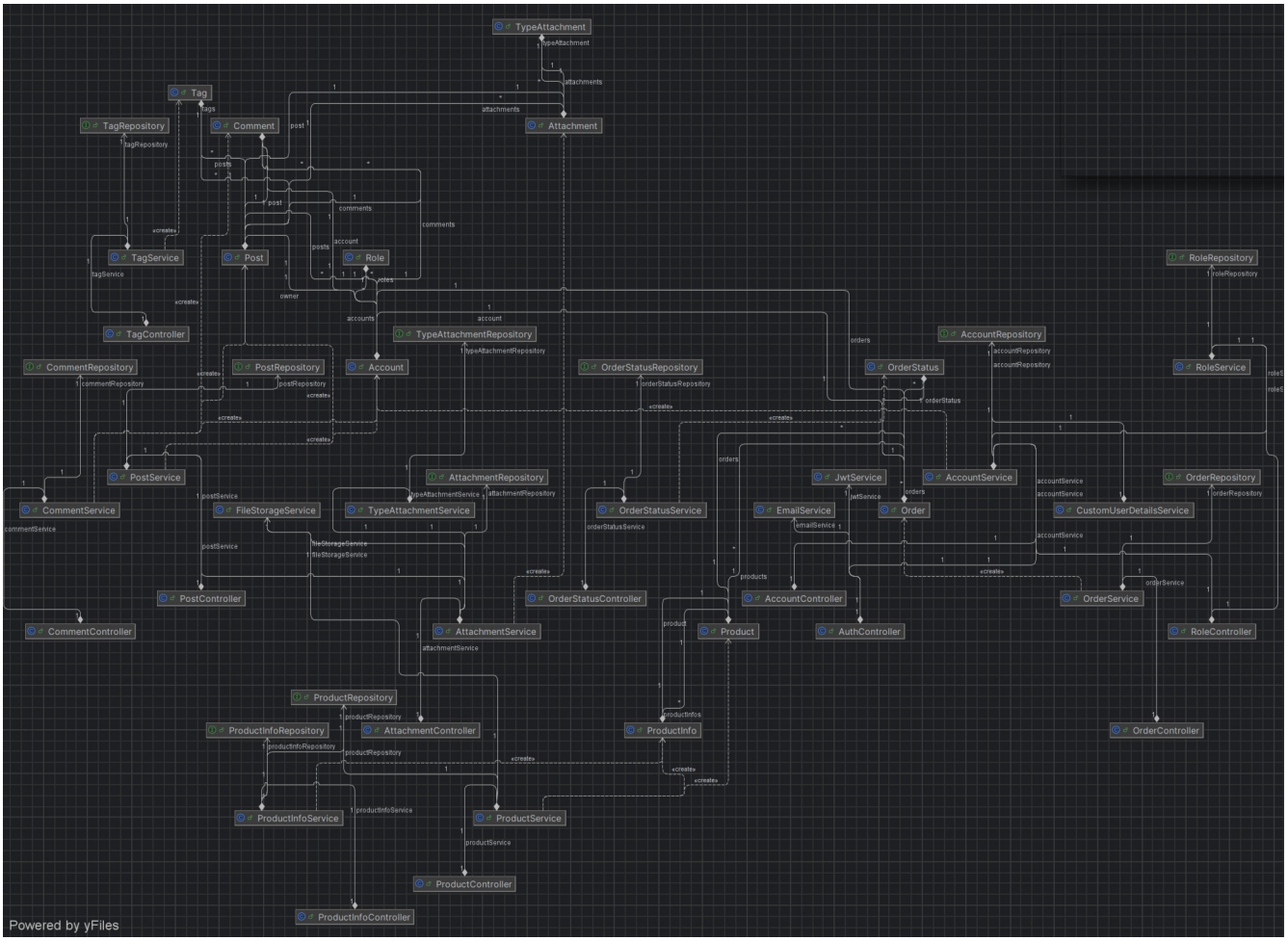
Курсовая работа
Часть №3

Выполнил
Путинцев Данил Денисович
Группа P3307
Проверил(а)
Преподаватель: Харитонова А.Е.

Санкт-Петербург 2025 год

Диаграмма классов, представляющую общую архитектуру системы





Реализовать уровень хранения информационной системы на основе разработанной на предыдущем этапе базы данных.

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDateTime;
import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Account")
public class Account {

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Integer id;
```

```
 @Column(name = "name", nullable = false)
 private String name;

 @Column(name = "email", nullable = false, unique = true)
 private String email;

   @Column(name = "password", nullable = false)
   private String password;

   @Column(name = "enabled")
   private boolean enabled = false;

   @Column(name = "verification_code")
   private String verificationCode;

   @Column(name = "verification_code_expiry")
   private LocalDateTime verificationCodeExpiry;

   @Column(name = "reset_password_token")
   private String resetPasswordToken;

   @Column(name = "reset_password_token_expiry")
   private LocalDateTime resetPasswordTokenExpiry;

   @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL)
   private List<Post> posts;

   @OneToMany(mappedBy = "account", cascade = CascadeType.ALL)
   private List<Comment> comments;

   @ManyToMany(fetch = FetchType.EAGER)
   @JoinTable(
         name = "AccountRole",
         joinColumns = @JoinColumn(name = "account_id"),
         inverseJoinColumns = @JoinColumn(name = "role_id")
   )
   private List<Role> roles;

   @OneToMany(mappedBy = "account", cascade = CascadeType.ALL)
   private List<Order> orders;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "Attachment")
public class Attachment {

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Integer id;
```

```
 @Column(name = "name", nullable = false)
 private String name;

 @Column(name = "path", nullable = false)
 private String path;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "post_id", nullable = false)
    private Post post;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "type_id", nullable = false)
    private TypeAttachment typeAttachment;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDateTime;

@Getter
@Setter
@Entity
@Table(name = "Comment")
public class Comment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "created_at", nullable = false)
    private LocalDateTime createdAt;

    @Column(name = "user_comment", columnDefinition = "TEXT")
    private String userComment;

    @ManyToOne()
    @JoinColumn(name = "post_id", nullable = false)
    private Post post;

    @ManyToOne()
    @JoinColumn(name = "account_id", nullable = false)
    private Account account;

}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
```

```java
@Getter
@Setter
@Entity
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "address", nullable = false)
    private String address;

    @Column(name = "created_at", nullable = false)
    private LocalDateTime createdAt;

    @Column(name = "total_amount", nullable = false)
    private Integer totalAmount;

    @Column(name = "phone")
    private String phone;

    @Column(name = "email")
    private String email;

    @Column(name = "customer_name")
    private String customerName;

    @Column(name = "delivery_method")
    private String deliveryMethod;

    @Column(name = "payment_method")
    private String paymentMethod;

    @Column(name = "postal_code")
    private String postalCode;

    @Column(name = "notes")
    private String notes;

    @Column(name = "order_items_json", columnDefinition = "TEXT")
    private String orderItemsJson;

    @ManyToOne
    @JoinColumn(name = "account_id", nullable = false)
    private Account account;

    @ManyToOne
    @JoinColumn(name = "status_id", nullable = false)
    private OrderStatus orderStatus;

    @ManyToMany
    @JoinTable(
            name = "OrderProduct",
            joinColumns = @JoinColumn(name = "order_id"),
            inverseJoinColumns = @JoinColumn(name = "product_id")
    )
```

```
    private List<Product> products = new ArrayList<>();
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "OrderStatus")
public class OrderStatus {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "description")
    private String description;

    @OneToMany(mappedBy = "orderStatus", cascade = CascadeType.ALL)
    private List<Order> orders;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDateTime;
import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Post")
public class Post {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "title", nullable = false)
    private String title;

    @Column(name = "text", columnDefinition = "TEXT")
    private String text;

    @Column(name = "created_at", nullable = false)
    private LocalDateTime createdAt;
```

```
    @Column(name = "count_like")
    private Integer countLike;

    @OneToMany(mappedBy = "post", cascade = CascadeType.ALL)
    private List<Attachment> attachments;

    @ManyToMany
    @JoinTable(
            name = "TagPost",
            joinColumns = @JoinColumn(name = "post_id"),
            inverseJoinColumns = @JoinColumn(name = "tag_id")
    )
    private List<Tag> tags;

    @ManyToOne()
    @JoinColumn(name = "owner_id", nullable = false)
    private Account owner;

    @OneToMany(mappedBy = "post", cascade = CascadeType.ALL)
    private List<Comment> comments;

}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Product")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "description")
    private String description;

    @Column(name = "category")
    private String category;

    @Column(name = "base_price")
    private Integer basePrice;

    @Column(name = "popularity")
    private Integer popularity = 0;

    @Column(name = "images", columnDefinition = "TEXT")
```

```
    private String images;

    @ManyToMany(mappedBy = "products")
    private List<Order> orders;

    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL)
    private List<ProductInfo> productInfos;

    public List<String> getImagesList() {
        if (images == null || images.isEmpty()) {
            return List.of();
        }
        return List.of(images.split(","));
    }
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "ProductInfo")
public class ProductInfo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne
    @JoinColumn(name = "product_id", nullable = false)
    private Product product;

    @Column(name = "size_name", nullable = false)
    private String sizeName;

    @Column(name = "count_items", nullable = false)
    private Integer countItems;

    @Column(name = "price")
    private Integer price;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Rang")
```

```
public class Rang {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false, unique = true)
    private String name;

    @Column(name = "description")
    private String description;

    @OneToMany(mappedBy = "rang", cascade = CascadeType.ALL)
    private List<Tournament> tournaments;

}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Role")
public class Role {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false, unique = true)
    private String name;

    @Column
    private String description;

    @ManyToMany(mappedBy = "roles")
    private List<Account> accounts;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "Tag")
public class Tag {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false, unique = true)
    private String name;

    @Column(name = "description")
    private String description;

    @ManyToMany(mappedBy = "tags")
    private List<Post> posts;

}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDateTime;

@Getter
@Setter
@Entity
@Table(name = "Tournament")
public class Tournament {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "start_date")
    private LocalDateTime startDate;

    @Column(name = "finish_date")
    private LocalDateTime finishDate;

    @Column(name = "address")
    private String address;

    @Column(name = "link")
    private String link;

    @Column(name = "archived", nullable = false)
    private Boolean archived = false;

    @Column(name = "minimal_age")
    private Integer minimalAge;

    @ManyToOne(optional = false)
    @JoinColumn(name = "rang_id", nullable = false)
```

```
    private Rang rang;
}
```

```
package com.danp1t.backend.model;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
@Entity
@Table(name = "TypeAttachment")
public class TypeAttachment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "name", nullable = false, unique = true)
    private String name;

    @Column(name = "description")
    private String description;

    @OneToMany(mappedBy = "typeAttachment", cascade = CascadeType.ALL)
    private List<Attachment> attachments;

}
```

При реализации уровня хранения должны использоваться функции/процедуры, созданные на втором этапе с помощью pl/pgsql. Нельзя замещать их использование альтернативной реализацией аналогичных запросов на уровне хранения информационной системы.

```
@Repository
public interface RangRepository extends JpaRepository<Rang, Integer> {
 boolean existsByName(String name);

 @Query(value = "SELECT insert_rang(:name, :description)",
 nativeQuery = true)
 Integer callInsertRangFunction(@Param("name") String name,
 @Param("description") String description);
}
```

```
package com.danp1t.backend.repository;

import com.danp1t.backend.model.Account;
```

```java
import jakarta.transaction.Transactional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

@Repository
public interface AccountRepository extends JpaRepository<Account, Integer> {
    Optional<Account> findByEmail(String email);
    boolean existsByEmail(String email);

    @Query("SELECT a FROM Account a LEFT JOIN FETCH a.roles WHERE a.id = :id")
    Optional<Account> findByIdWithRoles(@Param("id") Integer id);

    Optional<Account> findByResetPasswordToken(String resetPasswordToken);

    @Query("SELECT a FROM Account a LEFT JOIN FETCH a.posts WHERE a.id = :id")
    Optional<Account> findByIdWithPosts(@Param("id") Integer id);

    @Query("SELECT a FROM Account a LEFT JOIN FETCH a.comments WHERE a.id = :id")
    Optional<Account> findByIdWithComments(@Param("id") Integer id);

    @Query("SELECT COUNT(a) FROM Account a JOIN a.roles r WHERE r.id = :roleId")
    Long countByRoleId(@Param("roleId") Integer roleId);

    @Query("SELECT a FROM Account a JOIN a.roles r WHERE r.id = :roleId")
    List<Account> findByRoleId(@Param("roleId") Integer roleId);

    @Modifying
    @Query(value = "CALL ban_user(:accountId)", nativeQuery = true)
    void banUser(@Param("accountId") Integer accountId);
}
```

```java
package com.danp1t.backend.repository;

import com.danp1t.backend.dto.CommentDTO;
import com.danp1t.backend.model.Comment;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import java.util.List;
import java.util.Optional;

@Repository
public interface CommentRepository extends JpaRepository<Comment, Integer> {
    List<Comment> findByAccountId(Integer accountId);

    @Query("SELECT c FROM Comment c LEFT JOIN FETCH c.post LEFT JOIN FETCH c.account WHERE c.id = :id")
    Optional<Comment> findByIdWithDetails(@Param("id") Integer id);

    @Query("SELECT c FROM Comment c LEFT JOIN FETCH c.account WHERE c.post.id = :postId ORDER BY
c.createdAt DESC")
```

```
    List<Comment> findByPostIdWithAccount(@Param("postId") Integer postId);

    @Query(value = "SELECT * FROM create_comment(:userComment, :postId, :accountId)",
            nativeQuery = true)
    Integer createComment(
            @Param("userComment") String userComment,
            @Param("postId") Integer postId,
            @Param("accountId") Integer accountId
    );
}
```

На основе описания бизнес-процессов из первого этапа и построенного уровня хранения реализовать уровень бизнес-логики информационной системы.

Реализация доступна по ссылке:
https://github.com/danp1t/ITMO/tree/main/course3/information_system/course_work/part3/backend

Вывод

В данной части курсовой работы была реализован уровень бизнес-логики информационной системы. Я использовал Controller, Service и Repository для структуризации кода. Помимо этого я познакомился со Spring Security для организации работы информационной системы с различными ролями пользователя. Также был реализован компонент FileStorageService, который отвечает за хранение файлов на жестком диске, для того чтобы не хранить медиа-файлы в базах данных.