1. Цель проекта

Разработка RESTful API для управления академическими оценками студентов с использованием современных технологий:

- Spring Boot 3
- Kotlin
- H2 Database
- OpenAPI 3.0

2. Функциональные требования

1. Управление оценками:
   - Создание новых оценок
   - Просмотр всех оценок

Сущности данных:

```
Grade {
  id: Long (автоинкремент)
  studentId: Long
  courseId: Long
  value: Int (1-5)
  date: LocalDate
}
```

3. Технические требования

1. **Архитектура:**
   - REST API
   - 3-слойная архитектура (Controller-Service-Repository)
1. **Технологический стек:**
   - Язык: Kotlin
   - Фреймворк: Spring Boot 3.2+
   - База данных: H2 (in-memory)
   - Сборка: Maven
   - Документация: Swagger/OpenAPI 3.0
1. **Эндпоинты:**

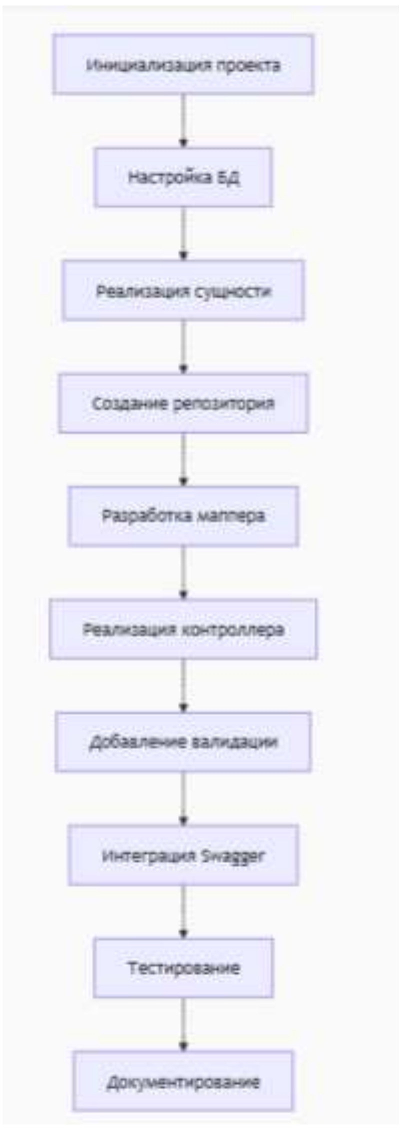
Метод	**Путь**	**Описание**
POST	/api/grades	Создать новую оценку

Метод	**Путь**	**Описание**
GET	/api/grades	Получить все оценки

4. Требования к качеству

1. Валидация входных данных
2. Логирование операций
3. Обработка ошибок
4. Полная документация API
5. 100% покрытие основного функционала тестами

5. Этапы реализации



Реализованный функционал

1. Подключение к БД (application.properties)

```
# application.properties
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

2. Сущность Grade

```
@Entity
data class Grade(
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   val id: Long = 0,
   val studentId: Long,
   val courseId: Long,
   val gradeValue: Int,
   val date: LocalDate? = null
)
```

3. Репозиторий

```
interface GradeRepository : JpaRepository<Grade, Long>
```

4. Маппер преобразования

```
object GradeMapper {
   fun toDto(grade: Grade): GradeDto = ...
   fun toEntity(dto: GradeDto): Grade = ...
}
```

5. Контроллер с эндпоинтами

```
@RestController
@RequestMapping("/api/grades")
class GradeController {
   @PostMapping
   fun createGrade(): ResponseEntity<GradeDto> { ... }


   @GetMapping
   fun getAllGrades(): ResponseEntity<List<GradeDto>> { ... }
}
```

6. Логгирование

```
private val logger = LoggerFactory.getLogger(javaClass)
logger.info("Creating grade: $dto")
```

7. Документирование через Swagger

```
@Tag(name = "Grades", description = "API for managing student grades")
@Operation(summary = "Create a new grade")
@ApiResponse(responseCode = "201", description = "Grade created successfully")
```

8. Тестирование через Swagger UI

Доступно по адресу: http://localhost:8080/swagger-ui/index.html

1. Главный класс приложения

Файл: src/main/kotlin/com/example/gradeapi/GradeApiApplication.kt

```
package com.example.gradeapi


import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication


@SpringBootApplication
```

```
class GradeApiApplication

fun main(args: Array<String>) {
    runApplication<GradeApiApplication>(*args)
}
```

2. Конфигурация Swagger

Файл: src/main/kotlin/com/example/gradeapi/config/SwaggerConfig.kt

```
package com.example.gradeapi.config

import io.swagger.v3.oas.models.OpenAPI
import io.swagger.v3.oas.models.info.Info
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration

@Configuration
class SwaggerConfig {

    @Bean
    fun customOpenAPI(): OpenAPI {
        return OpenAPI()
            .info(Info()
                .title("Grade API")
                .version("1.0.0")
                .description("API for managing student grades"))
    }
}
```

3. Контроллер

Файл: src/main/kotlin/com/example/gradeapi/controller/GradeController.kt

```
package com.example.gradeapi.controller


import com.example.gradeapi.dto.GradeDto
import com.example.gradeapi.mapper.GradeMapper
import com.example.gradeapi.repository.GradeRepository
import io.swagger.v3.oas.annotations.Operation
import io.swagger.v3.oas.annotations.media.Content
import io.swagger.v3.oas.annotations.media.Schema
import io.swagger.v3.oas.annotations.responses.ApiResponse
import io.swagger.v3.oas.annotations.tags.Tag
import jakarta.validation.Valid
import org.slf4j.LoggerFactory
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*


@Tag(name = "Grades", description = "API for managing student grades")
@RestController
@RequestMapping("/api/grades")
class GradeController(
   private val repository: GradeRepository
) {
   private val logger = LoggerFactory.getLogger(javaClass)


   @Operation(summary = "Get all grades")
```

```
@ApiResponse(
   responseCode = "200",
   description = "List of all grades",
   content = [Content(mediaType = "application/json",
          schema = Schema(implementation = GradeDto::class))]
)
@GetMapping
fun getAllGrades(): ResponseEntity<List<GradeDto>> {
   logger.info("Fetching all grades")
   val grades = repository.findAll().map(GradeMapper::toDto)
   return ResponseEntity.ok(grades)
}

@Operation(summary = "Create a new grade")
@ApiResponse(
   responseCode = "201",
   description = "Grade created successfully",
   content = [Content(mediaType = "application/json",
          schema = Schema(implementation = GradeDto::class))]
)
@PostMapping
fun createGrade(
   @Valid @RequestBody dto: GradeDto
): ResponseEntity<GradeDto> {
   logger.info("Creating grade: $dto")
   val entity = GradeMapper.toEntity(dto)
   val saved = repository.save(entity)
```

```
        return ResponseEntity(GradeMapper.toDto(saved), HttpStatus.CREATED)
    }
}
```

4. DTO (Data Transfer Object)

Файл: src/main/kotlin/com/example/gradeapi/dto/GradeDto.kt

```
package com.example.gradeapi.dto

import io.swagger.v3.oas.annotations.media.Schema
import jakarta.validation.constraints.Max
import jakarta.validation.constraints.Min
import jakarta.validation.constraints.NotNull

data class GradeDto(
    @Schema(description = "Unique ID of the grade", example = "1")
    val id: Long? = null,

    @field:NotNull(message = "Student ID is required")
    @field:Min(value = 1, message = "Student ID must be at least 1")
    @Schema(description = "Student ID", example = "123", required = true)
    val studentId: Long,

    @field:NotNull(message = "Course ID is required")
    @field:Min(value = 1, message = "Course ID must be at least 1")
    @Schema(description = "Course ID", example = "456", required = true)
    val courseId: Long,
    @field:NotNull(message = "Grade value is required")
    @field:Min(value = 1, message = "Grade must be at least 1")
    @field:Max(value = 5, message = "Grade cannot be more than 5")
    @Schema(description = "Grade value (1-5)", example = "5", required = true)
    val value: Int,

    @Schema(description = "Date of the grade", example = "2025-06-15")
    val date: String? = null
)
```

5. Сущность (Entity)

Файл: src/main/kotlin/com/example/gradeapi/entity/Grade.kt

```
package com.example.gradeapi.entity

import jakarta.persistence.Entity
import jakarta.persistence.GeneratedValue
import jakarta.persistence.GenerationType
import jakarta.persistence.Id
import java.time.LocalDate

@Entity
data class Grade(
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    val id: Long = 0,
    val studentId: Long,
    val courseId: Long,
    val gradeValue: Int,
    val date: LocalDate? = null
) {
    constructor() : this(0, 0, 0, 0, null)
}
```

6. Маппер

Файл: src/main/kotlin/com/example/gradeapi/mapper/GradeMapper.kt

```
package com.example.gradeapi.mapper

import com.example.gradeapi.dto.GradeDto
import com.example.gradeapi.entity.Grade
import java.time.LocalDate
import java.time.format.DateTimeFormatter

object GradeMapper {
    fun toDto(grade: Grade): GradeDto = GradeDto(
        id = grade.id,
        studentId = grade.studentId,
        courseId = grade.courseId,
        value = grade.gradeValue,
        date = grade.date?.format(DateTimeFormatter.ISO_DATE)
    )

    fun toEntity(dto: GradeDto): Grade = Grade(
        studentId = dto.studentId,
        courseId = dto.courseId,
        gradeValue = dto.value,
        date = dto.date?.let { LocalDate.parse(it) }
    )
}
```

7. Репозиторий

Файл: src/main/kotlin/com/example/gradeapi/repository/GradeRepository.kt

```
package com.example.gradeapi.repository

import com.example.gradeapi.entity.Grade
import org.springframework.data.jpa.repository.JpaRepository

interface GradeRepository : JpaRepository<Grade, Long>
```

8. Обработчик исключений

Файл: src/main/kotlin/com/example/gradeapi/exception/GlobalExceptionHandler.kt

```
package com.example.gradeapi.exception

import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.MethodArgumentNotValidException
import org.springframework.web.bind.annotation.ControllerAdvice
import org.springframework.web.bind.annotation.ExceptionHandler

@ControllerAdvice
class GlobalExceptionHandler {

    @ExceptionHandler(MethodArgumentNotValidException::class)
    fun handleValidationExceptions(ex: MethodArgumentNotValidException): ResponseEntity<Map<String, String>> {
        val errors = mutableMapOf<String, String>()
        ex.bindingResult.fieldErrors.forEach {
            errors[it.field] = it.defaultMessage ?: "Validation error"
        }
        return ResponseEntity(errors, HttpStatus.BAD_REQUEST)
    }

    @ExceptionHandler(Exception::class)
    fun handleAllExceptions(ex: Exception): ResponseEntity<String> {
        return ResponseEntity("Internal server error: ${ex.message}", HttpStatus.INTERNAL_SERVER_ERROR)
    }
}
```

9. Конфигурация приложения

Файл: src/main/resources/application.properties

```
# Server
server.port=8080
server.servlet.context-path=/api

# Database
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console


# Logging
logging.level.org.springframework=INFO
logging.level.com.example=DEBUG
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n
```

10. Файл зависимостей

Файл: pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.4</version>
        <relativePath/>
    </parent>


    <groupId>com.example</groupId>
    <artifactId>grade-api</artifactId>
    <version>1.0.0</version>
    <name>Grade API</name>
    <description>API for managing student grades</description>
```

```xml
<properties>
   <java.version>17</java.version>
   <kotlin.version>1.9.23</kotlin.version>
   <springdoc.version>2.5.0</springdoc.version>
</properties>

<dependencies>
   <!-- Spring Boot Starters -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
   </dependency>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-validation</artifactId>
   </dependency>

   <!-- Kotlin -->
   <dependency>
      <groupId>org.jetbrains.kotlin</groupId>
      <artifactId>kotlin-reflect</artifactId>
   </dependency>
```

```
    <dependency>
        <groupId>org.jetbrains.kotlin</groupId>
        <artifactId>kotlin-stdlib-jdk8</artifactId>
    </dependency>

    <!-- Database -->
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>

    <!-- Swagger/OpenAPI -->
    <dependency>
        <groupId>org.springdoc</groupId>
        <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
        <version>${springdoc.version}</version>
    </dependency>

    <!-- Test Dependencies -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

```
    <build>
        <sourceDirectory>${project.basedir}/src/main/kotlin</sourceDirectory>

  <testSourceDirectory>${project.basedir}/src/test/kotlin</testSourceDirectory>

        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>

  <mainClass>com.example.gradeapi.GradeApiApplicationKt</mainClass>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.jetbrains.kotlin</groupId>
                <artifactId>kotlin-maven-plugin</artifactId>
                <version>${kotlin.version}</version>
                <executions>
                    <execution>
                        <id>compile</id>
                        <phase>process-sources</phase>
                        <goals>
                            <goal>compile</goal>
                        </goals>
                    </execution>
                    <execution>
```

```
                        <id>test-compile</id>
                        <phase>process-test-sources</phase>
                        <goals>
                            <goal>test-compile</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <jvmTarget>17</jvmTarget>
                    <args>
                        <arg>-Xjsr305=strict</arg>
                    </args>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```