API - Spring Boot Versão 2.3.2

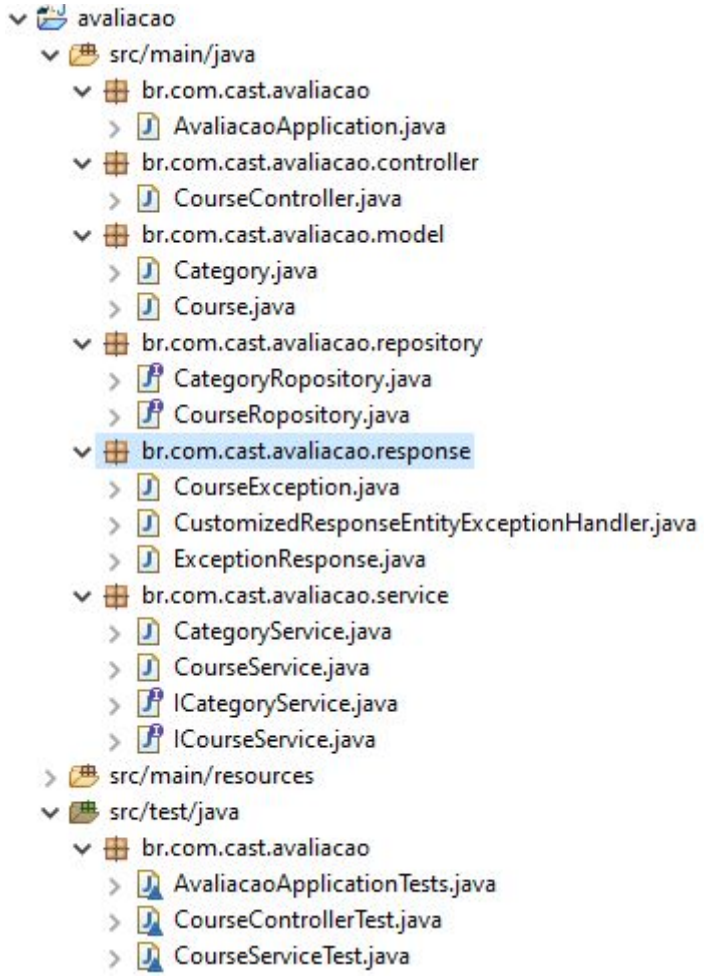
API Rest desenvolvida com Spring Boot, com as seguintes dependências pelo Gradle:

MySQL Driver: connector responsável por comunicação com o banco de dados MySQL;

Junit Júpiter 5.3.1: framework utilizado para testes;

MockMVC: classe pertencente ao Spring MVC para realizar casos de teste Junit nos métodos da Controller REST.

Estrutura atual da API obedecendo os padrões da avaliação técnica :



CourseController
classe responsável por todas as requisições relacionadas aos cursos.

```java
package br.com.cast.avaliacao.controller;

import java.text.SimpleDateFormat;

@RestController
@RequestMapping("/v1/api")
public class CourseController {

    @Autowired
    private ICourseService courseService;

    @GetMapping("/courses")
    ResponseEntity<List<Course>> getAllCourses() {
        return new ResponseEntity<List<Course>>(courseService.findAll(), HttpStatus.OK);
    }

    @PostMapping("/course")
    public ResponseEntity<Course> create(@RequestBody Course course){

        SimpleDateFormat dmyFormat = new SimpleDateFormat("yyyy-MM-dd");

        //Regra de negócio 1
        //Verifica se existe(m) curso(s) cadastrado dentro do período informado
        var courses = courseService.findCourseByStartAndEndDate(
               dmyFormat.format(course.getStartDate()),
               dmyFormat.format(course.getEndDate()));

        if(courses.size() >= 1)
            throw new CourseException("Existe(m) curso(s) planejados(s) dentro do período informado");

       return new ResponseEntity<Course>(courseService.save(course), HttpStatus.CREATED);
    }

    @DeleteMapping("/courses/{id}")
    public void delete(@PathVariable long id) {
        courseService.deleteById(id);
    }
}
```

Exceção CourseException para mostrar mensagem de erro apropriada.

Api base URL: http://localhost:8080/v1/api

Endpoints:

GET
/courses
retorna uma lista de cursos

POST
/course
cadastra um curso

DELETE
/courses/{id}
deleta um curso pelo id informado

Models

Modelos com as entidades mapeadas conforme a avaliação.

```java
import java.util.Date;

@Entity
@Table(name = "courses")
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="course_id")
    private Long id;

    @Column(name="description_subject")
    private String descriptionSubject;

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

    @Column(name="end_date")
    private Date endDate;

    @Column(name="student_amount_per_class")
    private Integer studentAmountPerClass;

    @ManyToOne
    @JoinColumn(name ="category_id")
    private Category category;

    public Course() {}

    public Course(String descriptionSubject, Date startDate, Date endDate, Integer studentAmountPerClass,
            Category category) {
        this.descriptionSubject = descriptionSubject;
        this.startDate = startDate;
        this.endDate = endDate;
        this.studentAmountPerClass = studentAmountPerClass;
        this.category = category;
    }


    @Entity
    @Table(name = "categories")
    public class Category {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name="category_id")
        private Long id;

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

        public Category() {};

        public Category(String description) {
            this.description = description;
        }

        public Long getId() {
            return id;
        }
        public void setId(Long id) {
            this.id = id;
        }
        public String getDescription() {
            return description;
        }
        public void setDescription(String description) {
            this.description = description;
        }
```

Interfaces

Interfaces da categoria e do curso com suas respectivas implementações

ICategoryService.java

```java
package br.com.cast.avaliacao.service;

import java.util.List;

public interface ICategoryService {

    public List<Category> findAll();

    public Category findById(Long id);

}
```

CategoryService.java

```java
package br.com.cast.avaliacao.service;

import java.util.List;

@Service
public class CategoryService implements ICategoryService {

    @Autowired
    private CategoryRopository repository;

    @Override
    public List<Category> findAll() {
        return (List<Category>) repository.findAll();
    }

    @Override
    public Category findById(Long id) {
        return  repository.findById(id).get() ;
    }

}
```

ICourseService.java

```java
import java.util.List;

public interface ICourseService {

    public List<Course> findAll();

    public Course save(Course course);

    public void deleteById(Long id);

    public List<Course> findCourseByStartAndEndDate(String startDate, String endDate);
```

CourseService.java

```java
@Service
public class CourseService implements ICourseService {

    @Autowired
    private CourseRopository repository;

    @Override
    public List<Course> findAll() {
            return (List<Course>) repository.findAll();
    }

    @Override
    public Course save(Course course) {
            return repository.save(course);
    }

    @Override
    public void deleteById(Long id) {
        repository.deleteById(id);
    }

    @Override
    public List<Course> findCourseByStartAndEndDate(String startDate, String endDate) {
```

Repositorio

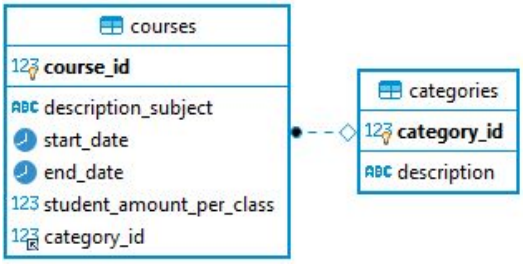
método criado para atender a necessidade da regra de negócio 1

```
@Repository
public interface CourseRopository extends CrudRepository<Course, Long>{

@Query( value = "SELECT * FROM courses c WHERE c.start_date >= ?1 AND c.end_date <= ?2",
        nativeQuery = true  )
List<Course> findCourseByStartAndEndDate(
 String startDate,
 String endDate);
}
```

Banco De Dados

O DB utilizado para armazenar os dados foi o MySQL em um container docker. O script do banco com as tabelas e os dados para inserção, está na raiz do projeto.
Diagrama de relacionamento de entidades entre as tabelas courses e categories.



Junit Test

Teste da CourseController usando mockMvc para fazer as requisições HTTP. O Postman também foi utilizado para auxiliar nos testes.

CourseControllerTest.java

```
38    @Autowired
39    MockMvc mockMvc;
40
41    @Test
42    public void getAllCourses() throws Exception
43    {
44
45        List<Course> courses = new ArrayList<Course>();
46        courses.add(new Course("Teste", new Date(), new Date(), 3, new Category("Programação")));
47        courses.add(new Course("Teste2", new Date(), new Date(), 3, new Category("Qualidade")));
48
49        when(courseService.findAll()).thenReturn(courses);
50
51        mockMvc.perform( MockMvcRequestBuilders
52          .get("/v1/api/courses")
53          .accept(MediaType.APPLICATION_JSON))
54          .andDo(print())
55          .andExpect(status().isOk())
56          .andExpect(MockMvcResultMatchers.jsonPath("$.[*].id").exists());
57    }
58
59    @Test
60    public void createCourse() throws Exception
61    {
62        mockMvc.perform( MockMvcRequestBuilders
63          .post("/v1/api/course")
64          .content(toJsonString(new Course("Teste", new Date(), new Date(), 3, new Category("Programação"))))
65          .contentType(MediaType.APPLICATION_JSON)
66          .characterEncoding("utf-8")
67          .accept(MediaType.APPLICATION_JSON))
68          .andExpect(status().isCreated())
69          .andDo(print());
70    }
71
72    @Test
73    public void deleteCourse() throws Exception
74    {
75        mockMvc.perform( MockMvcRequestBuilders.delete("/v1/api/courses/{id}", 1) )
76            .andExpect(status().isOk());
```

Teste unitário do CourseService.

CourseServiceTest.java

```
16
17 @SpringBootTest
18 class CourseServiceTest {
19
20     @Autowired
21     private CourseRopository courseRepository;
22
23     @Autowired
24     private CourseService courseService;
25
26     @Test
27     void deleteCourse() {
28
29         Category category  = new Category("Programação");
30         category.setId(2L);
31         Course course = new Course("TesteDelete", new Date(), new Date(), 3, category );
32         courseRepository.save(course);
33         courseService.deleteById(course.getId());
34
35         var courses = (List<Course>) courseService.findAll();
36
37         assertEquals(courses.size(), 0 );
38
39     }
40
41     @Test
42     void getAllCourses() {
43
44         Category category  = new Category("Comportamental");
45         category.setId(1L);
46         Course course = new Course("Teste", new Date(), new Date(), 3, category );
47         courseRepository.save(course);
48
49         var firstCourse = ((List<Course>) courseService.findAll()).get(0);
50
51         assertEquals(course.getId(), firstCourse.getId());
52
53     }
54
```