

Especificação de requisitos

Um time de futebol está com um novo sistema de contrato com seus jogadores e precisa de um sistema para gerenciar seus salários. A ideia para pagamento dos salários é a seguinte:

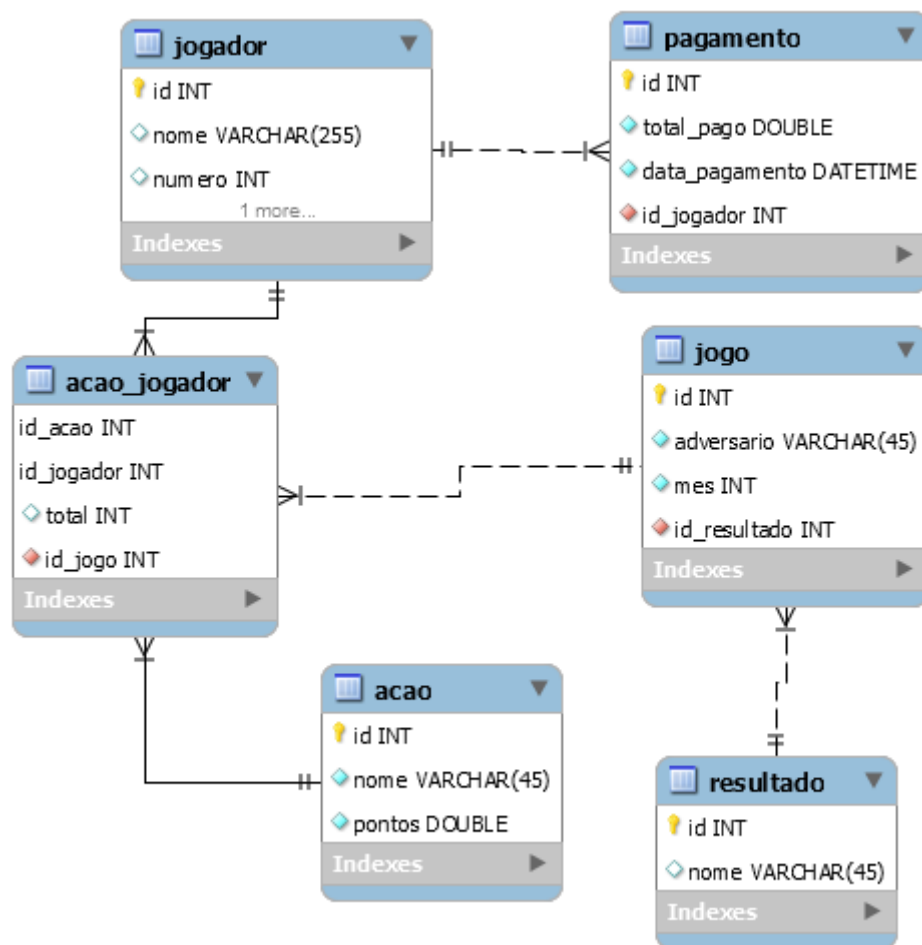
Todos jogadores terão um salário base e este pode ser aumentado ou abaixado de acordo com seu desempenho. Tendo seu mínimo sendo 10 mil reais e o teto 100 mil.

As pontuações são as seguintes

Gol	1 ponto
Assistência	0.5
Pênalti	0.7
Pênalti cometido	-0.7
Falta sofrida	0.3
Falta cometida	-0,3
Impedimento	-0,2
Passe errado	-0,1
Cartão amarelo	-0,5
Cartão vermelho	-1
Roubada de bola	0,2
Gol contra	-2
Vitoria	2 pontos para todos
Empate	0,5 para todos
Derrota	-2 para todos

A soma das pontuações será a porcentagem de acréscimo ou decréscimo no salário fixo do jogador.

Modelo de dados



Resultados de testes

Teste unitários com postman

- Verificar o tratamento de ids nulas ao tentar obter uma entidade do banco.

GET http://localhost:8080/jogador/obtem

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
id	1	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 500 Internal Server Error Time: 657ms Size: 394 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "timestamp": "2019-10-30T22:42:33.268+0000",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "A id procurada está nula",
6   "path": "/jogador/obtem"
7 }
```

Implementação do método

```
8 @CrossOrigin(origins = "http://localhost:4200")
9 @RestController
10 @RequestMapping("/jogador")
11 public class JogadorControlador {
12
13     @Autowired
14     private IJogadorServico jogadorServico;
15
16     @PostMapping(path = "/cria")
17     public JogadorDto cria(@RequestBody JogadorDto jogadorDto) {
18
19         return jogadorServico.cria(jogador, null);
20     }
21
22     @GetMapping(path = "/obtem")
23     public JogadorDto obtem(Long id) { return jogadorServico.obtem(id); }
24
25 }
26
27
```

```
@Override
public JogadorDto obtem(Long id) {

    if (ObjectUtils.isEmpty(id)) {
        throw new IllegalArgumentException("A id procurada está nula");
    }

    return JogadorDto.paradto(jogadorJpaRepository.findById(id.longValue()));
}
```

- Verificar o tratamento de datas de pagamento invalidas.



POST http://localhost:8080/pagamento/cria?idjogador=1&data=20-3-5 Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	idjogador	1			
<input checked="" type="checkbox"/>	data	20-3-5			
	Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 500 Internal Server Error Time: 1018ms Size: 385 B Save Response

Pretty Raw Preview Visualize BETA JSON  

```
1 {
2   "timestamp": "2019-10-31T10:49:17.042+0000",
3   "status": 500,
4   "error": "Internal Server Error",
5   "message": "Data inválida.",
6   "path": "/pagamento/cria"
7 }
```

Implementação do método

```
40 @Override
41 public PagamentoDto cria(Long idJogador, String data) {
42
43     Jogador jogador = JogadorDto.doDto(jogadorServico.obtem(idJogador));
44
45     if(ObjectUtils.isEmpty(jogador)){
46         throw new IllegalArgumentException("O jogador indicado é inválido.");
47     }
48     Pagamento pagamento = new Pagamento();
49     pagamento.setJogador(jogador);
50
51     try{
52         pagamento.setDataPagamento(ZonedDateTime.of(LocalDate.parse(data), LocalTime.now(), ZoneId.of("America/Sao_Paulo")));
53     } catch (Exception e){
54         throw new IllegalArgumentException("Data inválida.");
55     }
56     return PagamentoDto.parado(calculaPagamento(pagamento));
57 }
```

- Método de buscar ações por jogador no mês para pagamento.

GET Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION	**
<input checked="" type="checkbox"/>	idJogador	1		
<input checked="" type="checkbox"/>	mes	12		
<input checked="" type="checkbox"/>	ano	2019		
	Key	Value	Description	

Body Cookies Headers (6) Test Results Status: 200 OK Time: 33ms Size: 1.03 KB Save

Pretty Raw Preview Visualize BETA JSON 🔗

```

1  [
2    {
3      "id": 4,
4      "total": 3,
5      "jogador": {
6        "id": 1,
7        "nome": "jogador teste",
8        "numero": 18,
9        "listaAcaoJogador": null
10     },
11     "acao": {
12       "id": 1,
13       "nome": "Gol",
14       "pontos": null,
15       "listaAcaoJogador": null
16     },
17     "jogo": {
18       "id": 2,
19       "adversario": "America MG",
20       "data": "2019-12-02T22:00:00-02:00",
21       "resultado": {
22         "id": 1,
23         "nome": "Vitória"
24       }
25     }
26   }
27 ]

```

Implementação método

```

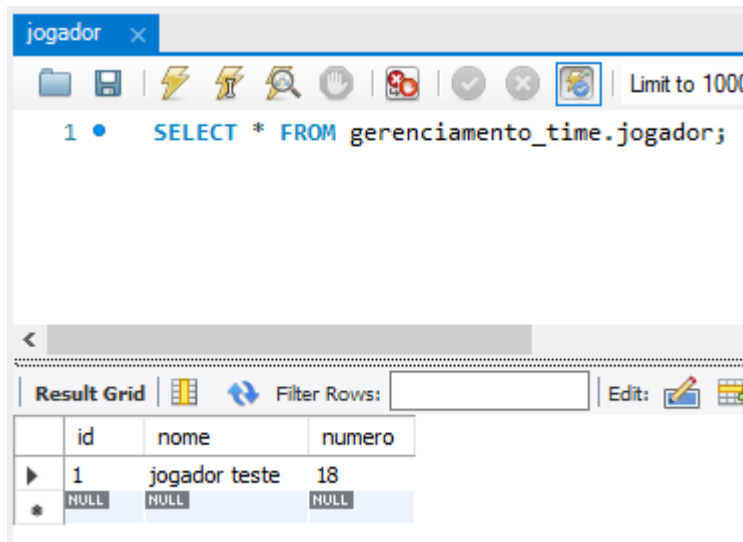
16
17 @Autowired
18 private JPAQueryFactory jpaQueryFactory;
19
20 @Override
21 public List<AcaoJogador> listaPorJogadorEMesDoAno(Long idJogador, Integer mes, Integer ano){
22
23     QAcaoJogador acaoJogador = QAcaoJogador.acaoJogador;
24
25     JPQLQuery<AcaoJogador> query = jpaQueryFactory.selectFrom(acaoJogador);
26
27     BooleanExpression predicado = acaoJogador.id.isNotNull();
28
29     if(!ObjectUtils.isEmpty(idJogador) && !ObjectUtils.isEmpty(mes) && !ObjectUtils.isEmpty(ano)){
30
31         predicado = predicado.and(acaoJogador.jogador.id.eq(idJogador));
32
33         predicado = predicado.and(acaoJogador.jogo.data.month().eq(mes.intValue()));
34
35         predicado = predicado.and(acaoJogador.jogo.data.year().eq(ano.intValue()));
36     }
37
38     query.where(predicado);
39
40     return query.fetch();
41 }
42

```

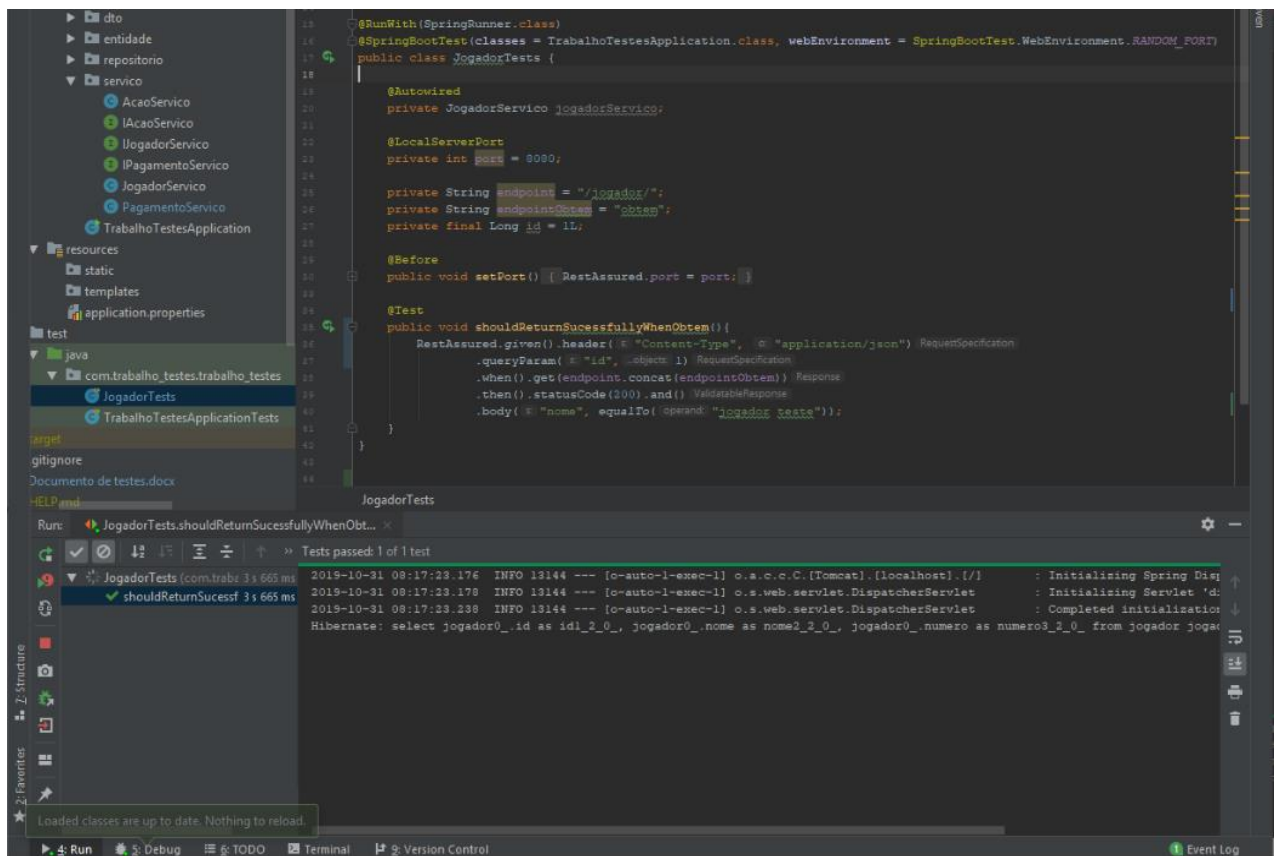
Testes de integração com Framework Rest Assured-Junit

- Teste obter jogador do banco

Jogador de teste criado manualmente no banco



Classe e método de teste de integração efetuando teste



Serviço da classe jogador

```
12  @Service
13  @Transactional
14  public class JogadorServico implements IJogadorServico {
15
16      @Autowired
17      private JogadorJpaRepository jogadorJpaRepository;
18
19      @Override
20      public JogadorDto cria(Jogador jogador) {
21
22          if (ObjectUtils.isEmpty(jogador)) {
23              throw new IllegalArgumentException("O jogador está nulo.");
24          }
25          return JogadorDto.parado(jogadorJpaRepository.save(jogador));
26      }
27
28      @Override
29      public JogadorDto obtem(Long id) {
30
31          if (ObjectUtils.isEmpty(id)) {
32              throw new IllegalArgumentException("A id procurada está nula");
33          }
34          return JogadorDto.parado(jogadorJpaRepository.findById(id.longValue()));
35      }
36  }
```

- Verificar o método de cálculo de salário

Método de testes

```

11
12 @Autowired
13 private IPagamentoService pagamentoService;
14
15 @LocalServerPort
16 private int port = 8080;
17
18 private String endpoint = "/pagamento/";
19 private String endPointCria = "cria";
20 private final Long idJogador = 1L;
21 private final String data = "2019-11-03";
22
23 @Before
24 public void setPort() { RestAssured.port = port; }
25
26 @Test
27 public void shouldReturnSucessfullyWhenCria() {
28     RestAssured.given().header("Content-Type", "application/json")
29         .queryParams("idJogador", idJogador)
30         .body("{\"data\": \"" + data + "\"}")
31         .when().post(endpoint.concat(endPointCria))
32         .then().statusCode(200).and();
33 }
34
35
36
37

```

Run: PagamentoTests.shouldReturnSucessfullyWhenCria... 8 s 935 ms

Tests passed: 1 of 1 test - 8 s 935 ms

SLF4J: Class path contains multiple SLF4J bindings.
 SLF4J: Found binding in [jar:file:/C:/Users/junior.DESKTOP-LE3H6DC/.m2/repository/ch/qos/logback/logback-classic/1.2.3/logback-classic-1.2.3.jar]
 SLF4J: Found binding in [jar:file:/C:/Users/junior.DESKTOP-LE3H6DC/.m2/repository/org/slf4j/slf4j-api/1.7.32/slf4j-api-1.7.32.jar]
 SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
 SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]

Implementação método

```

58
59 @
60 private Pagamento calculaPagamento(Pagamento pagamento) {
61     SalarioBase salarioBase = salarioBaseJpaRepository.obtemPorJogador(pagamento.getJogador().getId());
62
63     Integer mesPagamento = pagamento.getDataPagamento().getMonthValue();
64
65     Integer anoPagamento = pagamento.getDataPagamento().getYear();
66
67     List<AcaoJogador> lista = acaoJogadorJpaRepository
68         .listaPorJogadorEMesDoAno(pagamento.getJogador().getId(), mesPagamento, anoPagamento);
69
70     BigDecimal pontuacaoMensal = BigDecimal.ZERO;
71
72     for (AcaoJogador acaoJogador : lista) {
73         pontuacaoMensal = pontuacaoMensal.add(acaoJogador.getAcao().getPontos().multiply(BigDecimal.valueOf(acaoJogador.getTotal())));
74     }
75
76     pagamento.setDataPagamento(ZonedDateTime.now());
77
78     BigDecimal pagamentoMes = (pontuacaoMensal.divide(new BigDecimal(100)).add(BigDecimal.ONE))
79         .multiply(salarioBase.getSalario());
80
81     if (pagamentoMes.compareTo(new BigDecimal(100000)) == 1) {
82         pagamentoMes = new BigDecimal(100000);
83     } else if (pagamentoMes.compareTo(new BigDecimal(10000)) == -1) {
84         pagamentoMes = new BigDecimal(10000);
85     } else {
86         pagamento.setTotalPago(pagamentoMes);
87     }
88
89     pagamento = pagamentoJpaRepository.save(pagamento);
90     return pagamento;
91 }
92

```


Teste com Selenium

```
10 public class TestAutomatisado {
11     public static void main(String[] args) throws InterruptedException {
12
13         System.setProperty("webdriver.gecko.driver", "C:\\geckodriver.exe");
14         WebDriver driver = new FirefoxDriver();
15
16         driver.get("http://localhost:4200/consultaJogador");
17
18         WebElement filtro = driver.findElement(By.name("filtro"));
19         Thread.sleep(3000);
20         filtro.click();
21         WebElement adicionaJogador = driver.findElement(By.name("adicionaJogador"));
22         Thread.sleep(3000);
23         adicionaJogador.click();
24
25         driver.findElement(By.name("nomeJogador"))
26             .sendKeys(_keysToSend: "Julimar dos Santos");
27         Thread.sleep(3000);
28         driver.findElement(By.name("numeroJogador")).sendKeys(_keysToSend: "28");
29         Thread.sleep(3000);
30         driver.findElement(By.name("salarioJogador")).sendKeys(_keysToSend: "12000");
31         Thread.sleep(3000);
32         WebElement posicaoJogador = driver.findElement(By.name("posicaoJogador"));
33         posicaoJogador.click();
34
35         WebDriverWait wait = new WebDriverWait(driver, timeoutInSeconds: 5);
36         wait.until(ExpectedConditions
37             .elementToBeClickable(By.xpath("//span[text()='Goleiro']")));
38
39         driver.findElement(By.xpath("//span[text()='Goleiro']")).click();
40         Thread.sleep(3000);
41
42         WebElement botaoCadastrar = driver.findElement(By.name("submit"));
43         botaoCadastrar.click();
44
45         driver.get("http://localhost:4200/consultaJogador");
46
47         Thread.sleep(3000);
48
49         filtro = driver.findElement(By.name("filtro"));
50         filtro.click();
51         Thread.sleep(3000);
52
53         WebElement nome = driver.findElement(By.name("nomeJogador"));
54         nome.sendKeys(_keysToSend: "Julimar");
55         Thread.sleep(3000);
56
57         WebElement buscaJogador = driver.findElement(By.name("buscaJogador"));
58         buscaJogador.click();
59
60
61     }
62 }
```

Grupo

Junior Moreira

Israel Dias

Samara Guimarães

Lucas Duarte