

FIAP - FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA

JAVA ADVANCED

- João Carlos Lima e Silva

TURMA:

- 2TDSPZ

ALUNOS:

- Felipe Torlai RM 550263
- Felipe Pinheiro RM 550244
- Gabriel Girami RM 98017
- Gustavo Vinhola RM 98826
- Jean Carlos RM 550430

EMPRESA:

- NextGen

PROJETO:

- Sistema de Gestão de Experiência do Cliente

OBJETIVO:

- Mostrar as alterações feitas da Sprint 1 para a Sprint 2

SUMÁRIO

1. CLASSES NOVAS.....	4
2. NOVO DICIONÁRIO (ENUM)	8
3. RELACIONAMENTO	9
4. BUILDER.....	9
5. PAGINAÇÃO	11
6. SISTEMA DE CACHE	12
7. SPRINGDOC	13

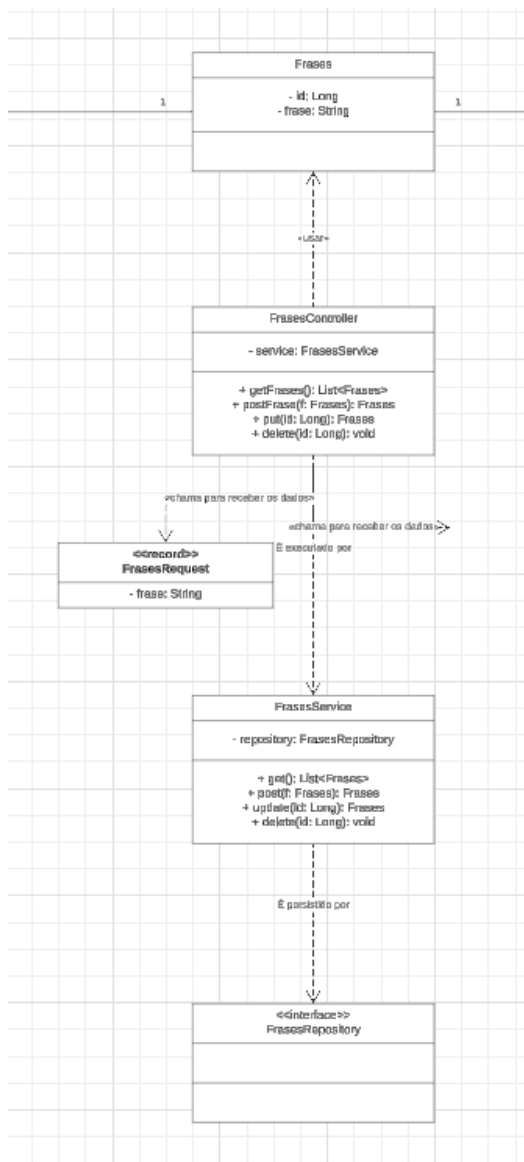
1. CLASSES NOVAS

A primeira e mais crucial mudança foi no diagrama de classes, no qual foi adicionada mais uma entidade chamada “frases”, cujo objetivo é guardar as frases que o usuário colocou em seu feedback. Essa entidade vai possuir dois atributos:

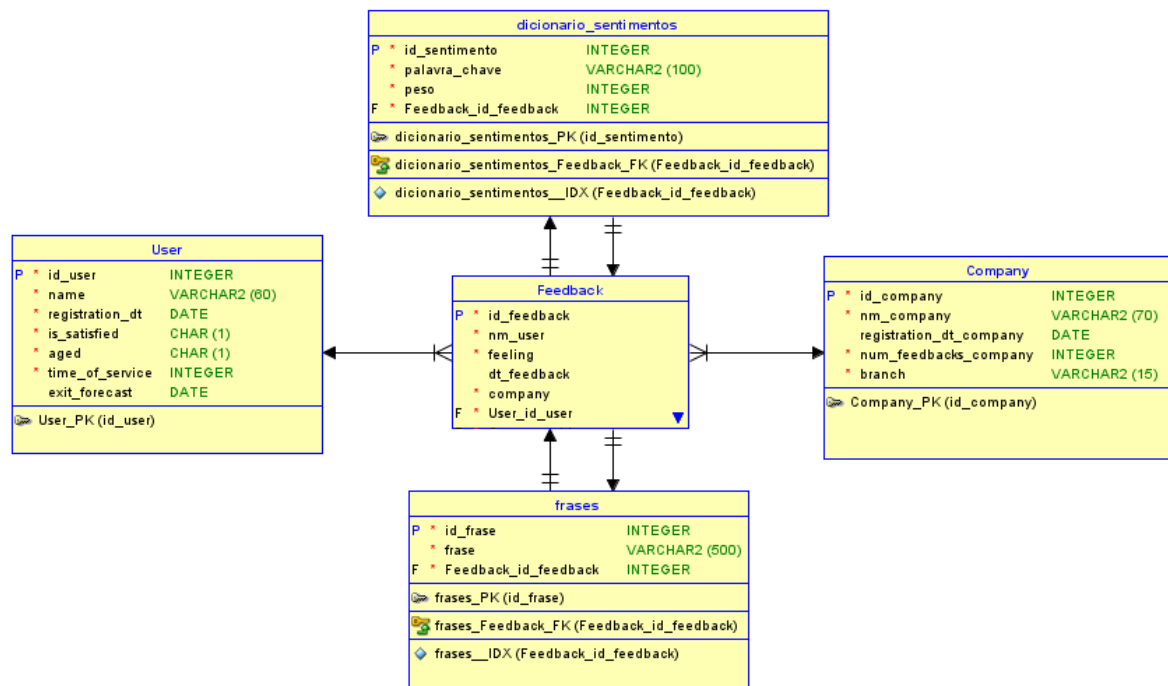
- ✓ Id: atributo Long com o objetivo de identificar a frase
- ✓ Frase: atributo String que irá guardar as frases escritas pelo usuário

Já que vamos fornecer as requisições HTTP para essa entidade, as classes de controle, repositório e serviço também foram adicionadas. Essa entidade foi adicionada de acordo com a sugestão do professor de banco de dados:

Mudança no Diagrama de Classes:



Mudança no Diagrama Entidade-Relacionamento:



Entidade adicionada:

```

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Frases {

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

    @NotBlank
    private String frase;
}
  
```

FrasesRequest adicionado:

```
1 + package com.fiap.nextgen.DTO;
2 +
3 + import jakarta.persistence.GeneratedValue;
4 + import jakarta.persistence.GenerationType;
5 + import jakarta.persistence.Id;
6 + import jakarta.validation.constraints.NotBlank;
7 +
8 + public record FrasesRequest(
9 +
10 +     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
11 +     Long id,
12 +
13 +     @NotBlank
14 +     String frase
15 +
16 + ) {}
```

Controller adicionado:

```
26 + @RestController
27 + @Slf4j
28 + @RequestMapping("frases")
29 + public class FrasesController {
30 +
31 +     @Autowired
32 +     FrasesService frasesService;
33 +
34 +     @GetMapping()
35 +     public List<Frases> getFrases() {
36 +         log.info("Pegando todas as frases");
37 +         return frasesService.getAll();
38 +     }
39 +
40 +     @PostMapping()
41 +     @ResponseStatus(CREATED)
42 +     public Frases postFrases(@RequestBody FrasesRequest fraseRequest) {
43 +         log.info("Postando um texto");
44 +         return frasesService.create(fraseRequest);
45 +     }
46 +
47 +     @PutMapping("{id}")
48 +     public Frases put(@PathVariable Long id, @RequestBody FrasesRequest frasesRequest) {
49 +         log.info("Atualizando as frases");
50 +         return frasesService.atualizar(id, frasesRequest);
51 +     }
52 +
53 +     @DeleteMapping("{id}")
54 +     @ResponseStatus(NO_CONTENT)
55 +     public void delete(@PathVariable Long id) {
56 +         log.info("Apagando as frases criadas");
57 +         frasesService.delete(id);
58 +     }
59 + }
```

Service adicionado:

```
15 + @Service
16 + public class FrasesService {
17 +
18 +     @Autowired
19 +     FrasesRepository frasesRepository;
20 +
21 +     public List<Frases> getAll() {
22 +         return frasesRepository.findAll();
23 +     }
24 +
25 +     public Frases create(FrasesRequest fraseRequest) {
26 +         Frases frase = constructFrases(fraseRequest);
27 +         return frasesRepository.save(frase);
28 +     }
29 +
30 +     public Frases atualizar(Long id, FrasesRequest fraseRequest) {
31 +         Frases frases = constructFrases(fraseRequest);
32 +         checkExistence(id);
33 +         frases.setId(id);
34 +         return frasesRepository.save(frases);
35 +     }
36 +
37 +     public void delete(Long id) {
38 +         frasesRepository.deleteById(id);
39 +     }
40 +
41 +
42 +     public Frases checkExistence(Long id) {
43 +         return frasesRepository
44 +             .findById(id)
45 +             .orElseThrow(() -> new RuntimeException(NOT_FOUND, "Não existe um usuário com este ID"));
46 +     }
47 +
48 +     public Frases constructFrases(FrasesRequest frasesRequest) {
49 +         return new Frases(
50 +             frasesRequest.id(),
51 +             frasesRequest.frase()
52 +         );
53 +     }
54 +
55 + }
```

Repository adicionado:

```
1 + package com.fiap.nextgen.Repository;
2 +
3 + import org.springframework.data.jpa.repository.JpaRepository;
4 +
5 + import com.fiap.nextgen.Model.Frases;
6 +
7 + public interface FrasesRepository extends JpaRepository<Frases, Long> {
8 +
9 + }
```

2. NOVO DICIONÁRIO (ENUM)

Também pela sugestão do professor do banco de dados, um novo enum vai ser criado, chamado “Dicionário de Sentimentos”, cujo objetivo é ser realmente um dicionário com sentimentos padrão que o usuário vai ter com a empresa, contendo 5 sentimentos, dentre eles:

- ✓ Terrible
- ✓ Bad
- ✓ Regular
- ✓ Good
- ✓ Awesome

Foi criado um atributo, dois construtores e um método GET, veja as alterações no código:

```
1 + package com.fiap.nextgen.Model;  
2 +  
3 + public enum DicionarioSentimentos {  
4 +  
5 +     TERRIBLE(1),  
6 +     BAD(2),  
7 +     REGULAR(3),  
8 +     GOOD(4),  
9 +     AWESOME(5);  
10 +  
11 +     private int feeling;  
12 +  
13 +     DicionarioSentimentos(int feeling) {  
14 +         this.feeling = feeling;  
15 +     }  
16 +  
17 +     DicionarioSentimentos() {  
18 +  
19 +     }  
20 +  
21 +     public int getFeeling() {  
22 +         return feeling;  
23 +     }  
24 + }
```


3. RELACIONAMENTO

A terceira alteração feita foi nos relacionamentos. A classe model “Feedback” tem um atributo chamado “Company” que é uma Chave Estrangeira (Foreign Key). Para a entrega da Sprint 2, adicionamos esse relacionamento com as anotações adequadas:

```
6      6      import jakarta.persistence.GeneratedValue;
7      7      import jakarta.persistence.GenerationType;
8      8      import jakarta.persistence.Id;
9      9      + import jakarta.persistence.ManyToOne;
10     10     import jakarta.validation.constraints.NotNull;
11     11     import jakarta.validation.constraints.PastOrPresent;
12     12     import lombok.AllArgsConstructor;
13     13
14     14     @@ -32,5 +33,6 @@ public class Feedback {
15     15
16     16     public LocalDate date;
17     17
18     18     @NotNull
19     19     public String company;
20     20
21     21     + @ManyToOne
22     22     + public Company company;
```

4. BUILDER

A quarta alteração foi a implementação do Design Pattern Builder, que nos permite a criação de dados automaticamente quando rodamos nosso aplicativo Spring. Para essa implementação, foi adicionado nas 3 classes model (Users, Feedback, Company) a notação Builder:

```
11     11     import jakarta.validation.constraints.PositiveOrZero;
12     12     import jakarta.validation.constraints.Size;
13     13     import lombok.AllArgsConstructor;
14     14     + import lombok.Builder;
15     15     import lombok.Data;
16     16     import lombok.NoArgsConstructor;
17     17
18     18     @Entity
19     19     @Data
20     20     + @Builder
21     21     @AllArgsConstructor
22     22     @NoArgsConstructor
```

Após, foi adicionado uma classe chamada DatabaseSeeder cujo objetivo é automatizar a criação dos dados no runtime:

1. Configuração e Injeção de Dependências:

```
21 @Configuration
22 public class DatabaseSeeder implements CommandLineRunner {
23
24
25     @Autowired
26     UserRepository userRepository;
27
28     @Autowired
29     CompanyRepository companyRepository;
30
31     @Autowired
32     FeedbackRepository feedbackRepository;
33
34     @Autowired
35     FrasesRepository frasesRepository;
36
```

2. Builders de cada entidade:

```
@Override
public void run(String... args) throws Exception {
    userRepository.saveAll(
        List.of(
            Users.builder().id(id:1L).name(name:"Felipe").registrationDate(LocalDate.now().minusYears(yearsToSubtract:2))
                .isSatisfied(isSatisfied:true).gender(gender:"Masculine").aged(aged:false).timeOfService(BigDecimal.valueOf(val:12)).exitForecast(LocalDate.now().plusYears(yearsToAdd:2)).build(),
            Users.builder().id(id:2L).name(name:"Gabriel").registrationDate(LocalDate.now().minusMonths(monthsToSubtract:3))
                .isSatisfied(isSatisfied:false).gender(gender:"Masculine").aged(aged:false).timeOfService(BigDecimal.valueOf(val:15)).exitForecast(LocalDate.now().plusYears(yearsToAdd:3)).build(),
            Users.builder().id(id:3L).name(name:"Gustavo").registrationDate(LocalDate.now().minusWeeks(weeksToSubtract:3))
                .isSatisfied(isSatisfied:true).gender(gender:"Masculine").aged(aged:false).timeOfService(BigDecimal.valueOf(val:17)).exitForecast(LocalDate.now().plusYears(yearsToAdd:4)).build()
        )
    );

    companyRepository.saveAll(
        List.of(
            Company.builder().id(id:1L).name(name:"Apple").registrationDate(LocalDate.of(year:1985, month:4, dayOfMonth:15))
                .numberOfFeedbacks(BigDecimal.ZERO).branch(branch:"Omni CRM").build(),
            Company.builder().id(id:2L).name(name:"Microsoft").registrationDate(LocalDate.of(year:1996, month:9, dayOfMonth:2))
                .numberOfFeedbacks(BigDecimal.ONE).branch(branch:"Social").build(),
            Company.builder().id(id:3L).name(name:"Samsung").registrationDate(LocalDate.of(year:2005, month:7, dayOfMonth:6))
                .numberOfFeedbacks(BigDecimal.valueOf(val:2)).branch(branch:"AI").build()
        )
    );

    feedbackRepository.saveAll(
        List.of(
            Feedback.builder().id(id:1L).feeling(DicionarioSentimentos.TERRIBLE).date(LocalDate.of(year:2010, month:12, dayOfMonth:25)).company(companyRepository.findById(id:1L).get()).build(),
            Feedback.builder().id(id:2L).feeling(DicionarioSentimentos.REGULAR).date(LocalDate.of(year:2015, month:1, dayOfMonth:2)).company(companyRepository.findById(id:2L).get()).build(),
            Feedback.builder().id(id:3L).feeling(DicionarioSentimentos.AWESOME).date(LocalDate.of(year:2020, month:5, dayOfMonth:15)).company(companyRepository.findById(id:3L).get()).build(),
            Feedback.builder().id(id:4L).feeling(DicionarioSentimentos.BAD).date(LocalDate.of(year:2005, month:7, dayOfMonth:12)).company(companyRepository.findById(id:1L).get()).build(),
            Feedback.builder().id(id:5L).feeling(DicionarioSentimentos.GOOD).date(LocalDate.of(year:2012, month:3, dayOfMonth:8)).company(companyRepository.findById(id:2L).get()).build(),
            Feedback.builder().id(id:6L).feeling(DicionarioSentimentos.TERRIBLE).date(LocalDate.of(year:2019, month:9, dayOfMonth:20)).company(companyRepository.findById(id:3L).get()).build()
        )
    );

    frasesRepository.saveAll(
        List.of(
            Frases.builder().id(id:1L).frase(frase:"Brabo demais").build(),
            Frases.builder().id(id:2L).frase(frase:"Mdio").build(),
            Frases.builder().id(id:3L).frase(frase:"Esse atendimento é precário").build()
        )
    );
}
```

Obs: Dê um zoom se necessário

5. PAGINAÇÃO

A quinta alteração feita foi a inclusão da paginação no nosso sistema usando o Pageable e suas anotações:

Alteração no FeedbackController:

```
@GetMapping
public List<Feedback> getMethod(
    @RequestParam(required = false) String company,
    @RequestParam(required = false) Integer mes,
    @PageableDefault(size = 5, sort = "date", direction = Direction.DESC) Pageable pageable
) {
    log.info(msg: "Pegando os feedbacks...");
    return feedbackService.getAllFeedbacks(company, mes, pageable);
}
```

Alteração no FeedbackService:

```
public List<Feedback> getAllFeedbacks(
    @RequestParam(required = false) String company,
    @RequestParam(required = false) Integer mes,
    @PageableDefault(size = 5, sort = "date", direction = Direction.DESC) Pageable pageable
) {
    if (mes != null && company != null) {
        return feedRepository.findByCompanyNameAndMes(company, mes, pageable);
    }

    if (mes != null) {
        return feedRepository.findByMes(mes, pageable);
    }

    if (company != null) {
        return feedRepository.findByCompanyName(company, pageable);
    }

    return feedRepository.findAll();
}
```

6. SISTEMA DE CACHE

A sexta alteração foi do sistema de memória cache no sistema, ele foi implementado nas entidades Users e Company, as quais não tem criações com grande frequência. Para tudo funcionar, algumas classes devem ter suas anotações colocadas:

1. Classe main (NextgenApplication.java):

```
2      2
3      3      import org.springframework.boot.SpringApplication;
4      4      import org.springframework.boot.autoconfigure.SpringBootApplication;
5      5 + import org.springframework.cache.annotation.EnableCaching;
6      6
7      7      @SpringBootApplication
8      8 + @EnableCaching
9      9      public class NextgenApplication {
10     10
11     11          public static void main(String[] args) {
12     12              SpringApplication.run(NextgenApplication.class, args);
13     13          }
```

2. Classe CompanyController

```
@RestController
@Slf4j
@RequestMapping(path = "companies")
@CacheConfig(cacheNames = "companies")
public class CompanyController {
```

```
    @PostMapping
    @ResponseStatus(CREATED)
    @CacheEvict(allEntries = true)
    public Company postMethod(@RequestBody @Valid CompanyRequest companyRequest) {
        log.info(msg: "Cadastrando uma empresa...");
        return companyService.createCompany(companyRequest);
    }

    @PutMapping("/{id}")
    @CacheEvict(allEntries = true)
    public Company putMethod(@PathVariable Long id, @RequestBody @Valid CompanyRequest companyRequest) {
        log.info("Atualizando a empresa com o id " + id);
        return companyService.updateCompany(id, companyRequest);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(NO_CONTENT)
    @CacheEvict(allEntries = true)
    public void deleteMethod(@PathVariable @Valid Long id) {
        log.info(msg: "Deletando a empresa");
        companyService.deleteCompany(id);
    }
}
```

7. SPRINGDOC

A próxima alteração foi a implementação da documentação automática Springdoc com Swagger UI. Para isso, foi feita a instalação da dependência no arquivo pom.xml:

```
54 +         <dependency>
55 +             <groupId>org.springdoc</groupId>
56 +             <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
57 +             <version>2.5.0</version>
58 +         </dependency>
59 +     </dependencies>
```

Depois disso, foi colocado o título da aplicação, descrição, versão e contato no arquivo principal (NextgenApplication.java):

```
13 + @OpenAPIDefinition(
14 +     info = @Info(
15 +         title = "NextGen",
16 +         summary = "API do App NextGen",
17 +         description = "Sistema de Gestão de Experiência do Cliente",
18 +         version = "1.0.0",
19 +         contact = @Contact(
20 +             name = "Felipe Pinheiro",
21 +             email = "fsp12371@gmail.com"
22 +         )
23 +     )
24 + )
```

Logo após, foi colocado as descrições e corpos de resposta para cada método em todos os controllers:

- Método GET:

```
49 + @Operation(
50 +     summary = "Listar Feedbacks",
51 +     description = "Retorna um array com todos os atributos do feedback"
52 + )
53 + @ApiResponses({
54 +     @ApiResponse(responseCode = "200", description = "Feedback retornado com sucesso!"),
55 +     @ApiResponse(responseCode = "401", description = "Feedback não autorizado. Realize a autenticação em /login")
56 + })
```

- Método POST:

```

68 + @Operation(
69 +     summary = "Cadastrar feedback",
70 +     description = "Cadastro de um feedback com o corpo de uma requisição"
71 + )
72 + @ApiResponses({
73 +     @ApiResponse(responseCode = "200", description = "Feedback cadastrado com sucesso!"),
74 +     @ApiResponse(responseCode = "400", description = "Validação falhou. Verifique os dados enviados no corpo da requisição"),
75 +     @ApiResponse(responseCode = "401", description = "Não autorizado. Realize a autenticação em /login")
76 + })

```

- Método PUT:

```

78 + @Operation(
79 +     summary = "Atualizar empresa",
80 +     description = "Atualiza os dados da empresa com o id informado na path"
81 + )
82 + @ApiResponses({
83 +     @ApiResponse(responseCode = "200", description = "Empresa atualizada com sucesso!"),
84 +     @ApiResponse(responseCode = "400", description = "Validação falhou. Verifique os dados enviados no corpo da requisição"),
85 +     @ApiResponse(responseCode = "401", description = "Não autorizado. Realize a autenticação em /login"),
86 +     @ApiResponse(responseCode = "404", description = "Não existe empresa com o 'id' informado")
87 + })

```

- Método DELETE:

```

88 + @Operation(
89 +     summary = "Apagar bloco de frases",
90 +     description = "Apaga a frase com o id informado no parâmetro de path"
91 + )
92 + @ApiResponses({
93 +     @ApiResponse(responseCode = "204", description = "Frase apagada com sucesso!"),
94 +     @ApiResponse(responseCode = "401", description = "Não autorizado. Realize a autenticação em /login")
95 + })

```

- Método Get by ID:

```

110 + @Operation(
111 +     summary = "Pegar empresa pelo id",
112 +     description = "Retorna os dados da empresa com o id informado no parâmetro de path"
113 + )
114 + @ApiResponses({
115 +     @ApiResponse(responseCode = "200", description = "Empresa retornada com sucesso!"),
116 +     @ApiResponse(responseCode = "401", description = "Não autorizado. Realize a autenticação em /login"),
117 +     @ApiResponse(responseCode = "404", description = "Não existe empresa com o 'id' informado")
118 + })

```