**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](#_Toc165712396)

[2. NOVO DICIONÁRIO (ENUM) 8](#_Toc165712397)

[3. RELACIONAMENTO 9](#_Toc165712398)

[4. BUILDER 9](#_Toc165712399)

[5. PAGINAÇÃO 11](#_Toc165712400)

[6. SISTEMA DE CACHE 12](#_Toc165712401)

[7. SPRINGDOC 13](#_Toc165712402)

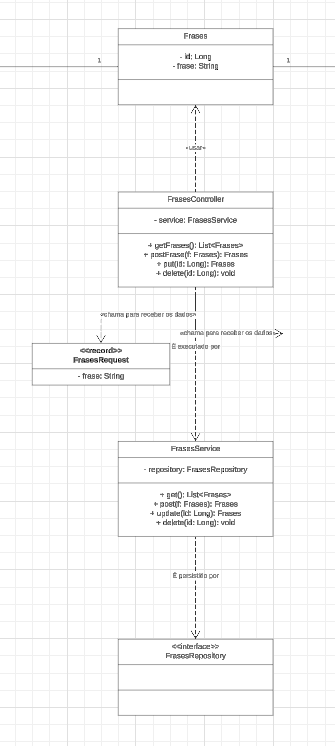
1. CLASSES NOVAS

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:**

Diagrama

Descrição gerada automaticamente

Entidade adicionada:

Texto

Descrição gerada automaticamente

FrasesRequest adicionado:

Texto

Descrição gerada automaticamente

Controller adicionado:

Texto

Descrição gerada automaticamente

Service adicionado:

Texto

Descrição gerada automaticamente

Repository adicionado:

Texto

Descrição gerada automaticamente

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:

Texto

Descrição gerada automaticamente

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:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

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:

Tela de computador com fundo verde

Descrição gerada automaticamente

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

1. Configuração e Injeção de Dependências:

Texto

Descrição gerada automaticamente

1. Builders de cada entidade:

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

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:**

Texto

Descrição gerada automaticamente

**Alteração no FeedbackService:**

Texto

Descrição gerada automaticamente

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):**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

1. **Classe CompanyController**

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

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:

Texto

Descrição gerada automaticamente

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

Texto

Descrição gerada automaticamente

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

* Método GET:

Texto

Descrição gerada automaticamente

* Método POST:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* Método PUT:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* Método DELETE:

Texto

Descrição gerada automaticamente

* Método Get by ID:

Texto

Descrição gerada automaticamente

8. OBSERVABILIDADE

Para verificar a saúde do projeto e poder ter maior observabilidade, também foi adicionado dependências, um projeto de server e a transformação do projeto atual em um projeto client:

* Dependências:
  + Actuator:

Texto

Descrição gerada automaticamente

* + Spring Client

Texto

Descrição gerada automaticamente

* + Gerenciamento da dependência

Texto

Descrição gerada automaticamente

* Exposição do sistema e portas para o Spring boot admin:

Texto

Descrição gerada automaticamente

* [Novo projeto de server para o NextGen](https://github.com/FelipeSPinheiro06/NextGen-Admin)
  + Dependências do novo projeto:

Texto

Descrição gerada automaticamente

* + Classe principal com a Annotation @EnableAdminServer

Texto

Descrição gerada automaticamente

* Configuração de envio de e-mail