**Nome:** Anderson Rodrigues Gomes **Data:** 22 de maio de 2025

*Responda em português ou inglês.*

1. **Write a Java program to solve the following problem:**

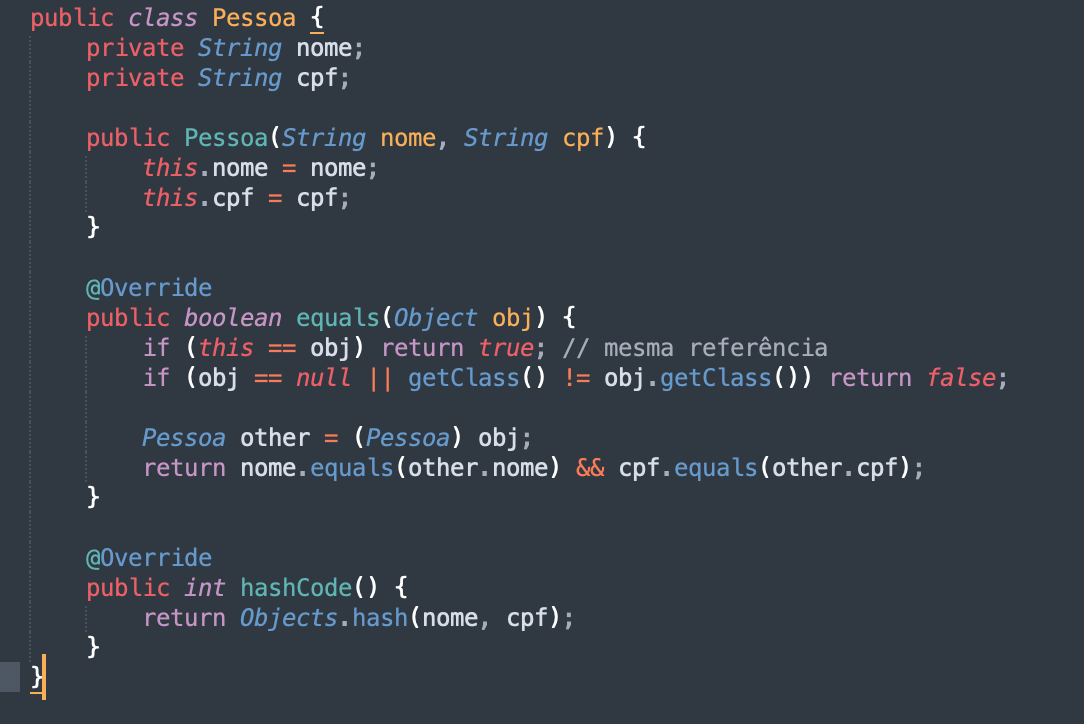
**You are tasked with creating a utility function for a text-processing application. The function must generate all possible anagrams from a given group of distinct letters. For example, the input {a, b, c} should return the output: abc, acb, bac, bca, cab, cba.**

**Additional Requirements:**

1. **The program should accept any group of distinct letters as input and produce the correct result.**
2. **Optimize for readability and clarity.**
3. **Include basic validation (e.g., ensure the input is non-empty and contains only letters).**
4. **Write unit tests to validate your function with at least three different test cases, including edge cases (e.g., input with a single letter or empty input).**
5. **Document your code clearly, explaining the logic for generating anagrams.**

1. **Provide an example scenario where overriding the equals() method is necessary in Java. Explain the key considerations when implementing this method, such as ensuring it aligns with the hashCode() method. Include code examples if possible.**

Precisamos sobrescrever o método equals, quando precisamos comparar o seu conteúdo. Para saber se os objetos são exatamente iguais, preciso sobrescrever o método hashCode também

Suponhamos que temos uma classe Pessoa, e temos como atributo nome, idade. Para realizar uma comparação da classe Pessoa, necessitaremos sobrescrever o método equals e o hashCode:  


1. **Explain how you would use a design pattern to decouple your code from a third-party library that might be replaced in the future. Describe the advantages and limitations of your chosen approach, and provide a small code snippet illustrating its application.**

Usaria o padrão adapter , pois permite desacoplar a responsabilidade direta de uma classe que utiliza uma biblioteca externa. Isso respeita os princípios de SOLID, e ajuda na realização de testes unitários.

1. **Describe your experience with Angular, including its core features and use cases. Provide an example of a practical application where you used Angular and include a code snippet demonstrating a key feature, such as component communication, data binding, or service integration.**

Atuei na manutenção e evolução de um sistema Angular utilizado por gerentes do Bradesco, desenvolvido com uma arquitetura de micro-frontends. Essa abordagem permitia separar os domínios de negócio de forma modular e escalável, promovendo maior organização e independência entre as equipes.

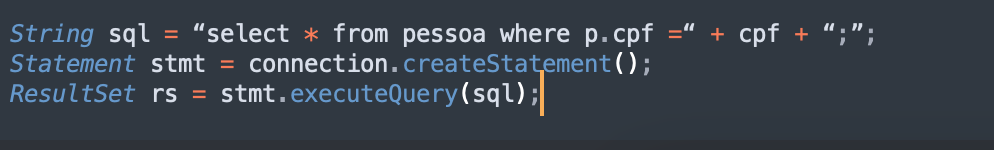
Minha principal responsabilidade foi no módulo de Perfil de Investidor (PFIN), que apresentava uma série de perguntas ao cliente para classificá-lo como conservador, moderado ou agressivo, com base nas respostas fornecidas. Embora meu foco principal fosse no backend, também contribuía ativamente no front-end sempre que necessário, especialmente por se tratar de uma equipe enxuta.

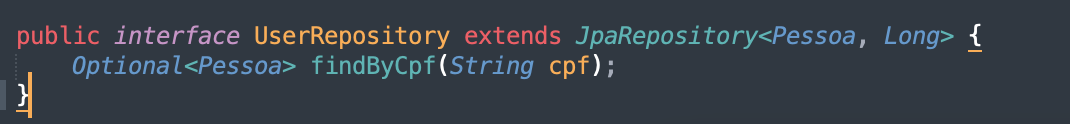
As perguntas e alternativas eram armazenadas em um banco de dados SQL Server, e a aplicação utilizava Spring Integration para buscar essas informações. As requisições eram orquestradas por uma API reativa em Spring WebFlux, que então repassava os dados a um BFF em Node.js, até chegarem ao módulo Angular.

No front-end, usávamos conceitos avançados do Angular como injeção de dependência com @Injectable, comunicação entre componentes com @Input e @Output, e também criávamos snippets customizados para reaproveitamento de lógica de interface. Havia uma tela padrão dinâmica que montava automaticamente as perguntas e, ao final do processo, exibia um gráfico com o perfil do cliente, usando os dados coletados.

1. **Discuss the techniques you use to prevent SQL injection attacks in web applications. Provide examples of code showing secure implementations, such as using parameterized queries or ORMs. Mention any additional measures you take to secure the database layer.**

Para prevenção de SQL injection, é necessário seguir uma abordagem de multicamadas incluindo consultas parametrizadas, frameworks ORM como Spring Data ou Hibernate, validação de entrada e acessos ao banco de dados com o mínimo de privilégios.

Evite utilizar consultas como:

Spring Data JPA

1. **Describe the steps you would take to diagnose and improve the performance of a batch process that interacts with a database and an FTP server. Explain how you would identify bottlenecks, optimize database queries, improve logic execution, and enhance file transfer efficiency. Provide examples of tools or techniques you would use during the analysis.**

Primeiro devemos analizar os gargalos para identificar os pontos mais lentos ou custosos para o processo

**Ferramentas:**

JProfile para analize de memória e CPU

Wireshark para analizar o desempenho de rede

Para banco de dados temos o pg\_stat\_statements para o postgresql para analizar as execuções do servidor. Caso for oracle usar o EXPLAIN PLAN

**Vamos verificar:**

Leitura do banco de dados

Transferência FTP

Parsing de arquivos

Processamento em Memória

Verificar ciclos de CPU, consumo de memória, I/O

Podemos, por fim cria métricas de observability usando Micrometer + Prometheus

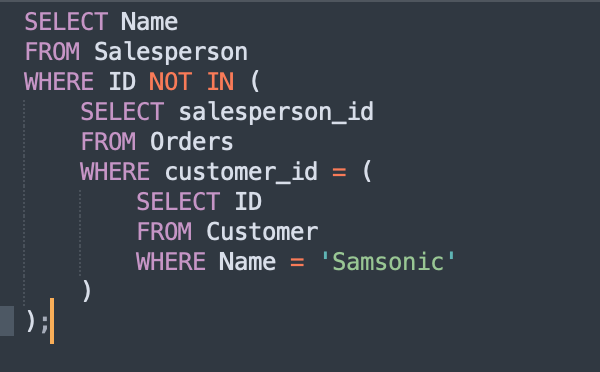
Criar alertas no Grafana

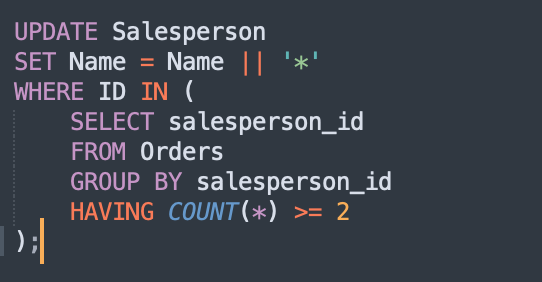
Logs estruturados como o ELK

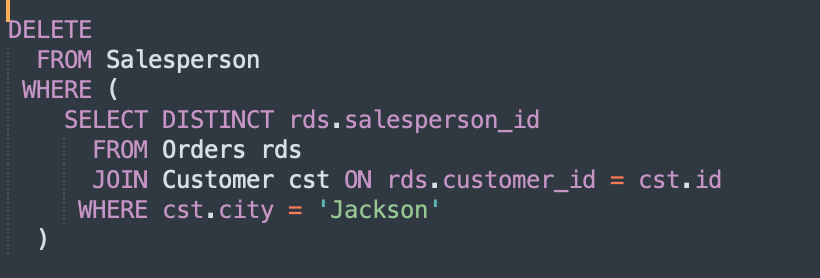
Configurar tracing com OpenTelemetry

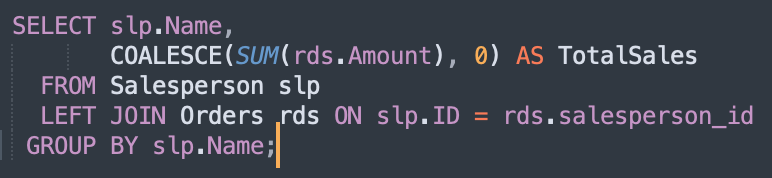
|  |  |
| --- | --- |
| **Salesperson** | **Customer** |
| **ID Name Age Salary**  1 Abe 61 140000  2 Bob 34 44000  5 Chris 34 40000  7 Dan 41 52000  8 Ken 57 115000  11 Joe 38 38000 | **ID Name City Industry Type**  4 Samsonic Pleasant J  6 Panasung Oaktown J  7 Samony Jackson B  9 Orange Jackson B |
|  |  |

|  |
| --- |
| **Orders** |
| **ID order\_date customer\_id salesperson\_id Amount**  10 8/2/96 4 2 540  20 1/30/99 4 8 1800  30 7/14/95 9 1 460  40 1/29/98 7 2 2400  50 2/3/98 6 7 600  60 3/2/98 6 7 720  70 5/6/98 9 7 150 |

1. **Given the tables above, write the SQL query that:   
     
   a. Returns the names of all Salesperson that don’t have any order with Samsonic.**

**b. Updates the names of Salesperson that have 2 or more orders. It’s necessary to add an ‘\*’ in the end of the name.**

**c. Deletes all Ssalesperson that placed orders to the city of Jackson.**

**d. The total sales amount for each Salesperson. If the salesperson hasn’t sold anything, show zero.**

1. **The customer has a system called XYZ and intends to start updates split into 3 phases. The requirements for the first phase are as follows:**
2. **Enable new data entries in the system, which will serve as input for the second phase.**
3. **Implement functionality to create, update, delete, and search plants.**
   * **Plants should have the following attributes:**
     + **Code: Numeric only, mandatory, and unique.**
     + **Description: Alphanumeric, up to 10 characters, optional.**
   * **Only admin users can delete plants.**
4. **Ensure that the system prevents duplication of plant codes.**

**Task:  
Based on the above information:**

1. **Write a use case or user story for this scenario, ensuring that it clearly addresses the requirements.**
2. **Highlight any business rules or assumptions relevant to the solution.**
3. **Describe any validations or security measures you would implement in the system.**
4. **Suggest how you would test this functionality, including examples of edge cases.**

**Use Case / Use Story**

**Titulo:** gerenciar o cadastro de plantas no Sistema XYZ - Fase - 1

**Como** usuário do sistema XYZ,

**Eu quero** poder criar, atualizar, remover o registro (quando autorizado) e pesquisar as plantas

**Fluxo Principal**

* + O usuário entra no sistema e acessa a funcionalidade de gerenciamento de plantas.
  + O usuário pode inserir uma nova planta com o código e descrição.
  + O sistema valida o código (numérico, único e obrigatório) e descrição com 10 com até 10 caracteres.
  + O usuário pode buscar plantas com o código e a descrição.
  + O usuário pode alterar dados de uma planta existente.
  + Se o usuário form admin, pode remover as plantas; caso o contrário, essa opção é desabilitada.
  + O sistema impede a criação de uma planta com o código já existente.

**Regras de Negócio**

* + O código da planta deve ser numérico exclusivo e obrigatório.
  + A descrição da planta é opcional e limitada a 10 caracteres alfanuméricos.
  + Somente um usuário com perfil de administrador pode remover as plantas.
  + O sistema deve impedir a duplicidade de código da planta.
  + O usuário "comum" pode criar, editar e pesquisar plantas, mas não remove-las.
  + Tem como premissa que há um sistema de autenticação que define o perfil do usuário (admin ou não).

**Validações e Segurança**

* + **Validação de Entrada**
    - Código: Obrigatório
    - Descrição: opcional, tamanho máximo 10 caracteres, aceita letras e números.
  + **Regras de Autorização**
    - Excluir plantas: Somente usuário com permissão de administrador poderá efetuar a exclusão.
  + **Prevenção de duplicidade**
    - Antes de criar uma nova planta, verificar se o código já existe.
  + **Segurança**
    - Autenticação para acessar o módulo.
    - Autorização para controlar a operação de remoção.
    - Sanitização de entradas para evitar ataques.
  + **Tratamento de Erros**
    - Mensagens claras sobre os erros de validação e de autorização

**Testes**

* + **Testes Funcionais** 
    - Criar a planta com o código válido e descrição válida.
    - Criar a planta com o código sem a descrição.
    - Atualizar a planta existente com os novos campos válidos.
    - Buscar a planta existente.
    - Deletar a planta com o usuário possuidor de privilégios elevados.
    - Retornar exceção ao criar uma planta com um código não numérico.
    - Retornar exceção ao criar uma planta com um código duplicado.
    - Retornar exceção ao deletar a planta com um usuário sem privilégios.
    - Retornar exceção ao criar uma planta com descrição maior que 10 caracteres.
  + **Casos de Bordas**
    - Descrição Vazia
    - Buscar textos com letras maiúsculas, minúsculas e caracteres especiais.
    - Código com valor mínimo.
  + **Teste de Segurança**
    - Testar tentativas de injeção de código.
    - Testar operações sem autenticação ou com usuário sem os privilégios necessários.

1. **Consider the following description of a system functionality:  
   User Registration**

* **A screen allows users to insert, delete, or update user information.**
* **Each user has properties: name, email, address, and phone, where name and email are mandatory fields.**
* **Emails must be unique across all users.**
* **Only admin users can delete other users.**

**Task:**

1. **Describe the types of tests you would implement (e.g., unit, integration, or end-to-end tests) and explain the scenarios you would test to ensure the functionality works as expected.**
2. **Provide examples of edge cases and how you would handle them.**
3. **Include an example of a test case in code or pseudocode for one or more scenarios.**

**Tipos de teste e justificativa**

* 1. **Testes unitários**
     + **Objetivo:** Validar as regras de negócio isoladamente.
     + **Exemplo de cenários**
       - Nome e e-mail são obrigatórios.
       - E-mail como formato válido.
       - E-mail único (validação antes de inserir o registro)
       - Apenas usuários com o privilégio de administrador poderá excluir o usuário
     + **Componentes testados**
       - Service, validators e repository (usar mocks).
  2. **Testes de integração**
     + **Objetivo:** Verificar a comunicação entre os componentes (controllers, services, repository e banco de dados).
     + **Exemplo de cenários:**
       - Inserir um novo usuário com dados válidos.
       - Atualizar informações
       - Não cadastrar e-mail duplicado.
       - Restringir a exclusão de usuário.
  3. **Testes de Ponta a Ponta (E2E)**
     + **Objetivo:** Validar fluxo completos via aplicação
     + **Exemplo de cenários:**
       - Usuário irá preencher o formulário, envia, exibe a confirmação.
       - Administrador exclui um usuário, exibe a confirmação.
       - Validação dos erros nos campos obrigatórios.

**Cenários de Testes**

✅ **Caso Positivo:**

* + - Inserir usuário com dados válidos.
    - Atualizar as informações do usuário, menos o nome e o e-mail.
    - Administrador exclui um usuário.

❌ **Caso negativo**

* + - Tentar cadastrar um usuário sem o e-mail.
    - Tentar cadastrar sem um nome.
    - E-mail já existe.
    - Usuário, sem os privilégios necessários, excluir um usuário.

⚠️ **Casos Extremos**

* + - Nome com apenas 1 caractere.
    - Nome com mais de 255 caracteres.
    - E-mail com domínio longo.
    - Endereço ou telefone nulo ou vazio.
    - Teste de injeção SQL/XSS no campo de endereço.
    - Conexão interrompida ao salvar
    - Concorrência: dois usuário tentando cadastrar ao mesmo tempo