**Nome: Fernando Pereira da Silva Data: 14/01/2025**

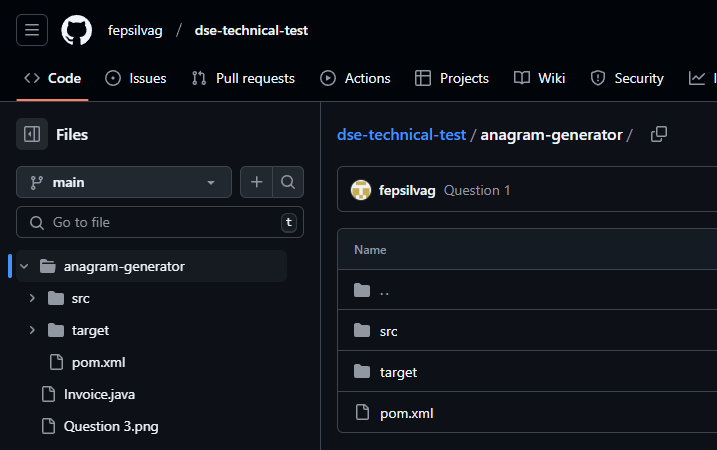
*Responda em português ou inglês.*

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.**



<https://github.com/fepsilvag/dse-technical-test/tree/main/anagram-generator>

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.**

Geralmente você sobrescreve "equals()" para realizar uma comparação customizada, isso só se torna interessante quando é necessário reaproveitar esse conjunto de "ifs";

Exemplo: Nos sistemas de logística emitimos notas fiscais, a SEFAZ gera um código único (accessKey) que contém 44 dígitos, na nota fiscal também temos vários outros campos, em nossos sistemas temos sobrescrita do "equals()" da seguinte maneira



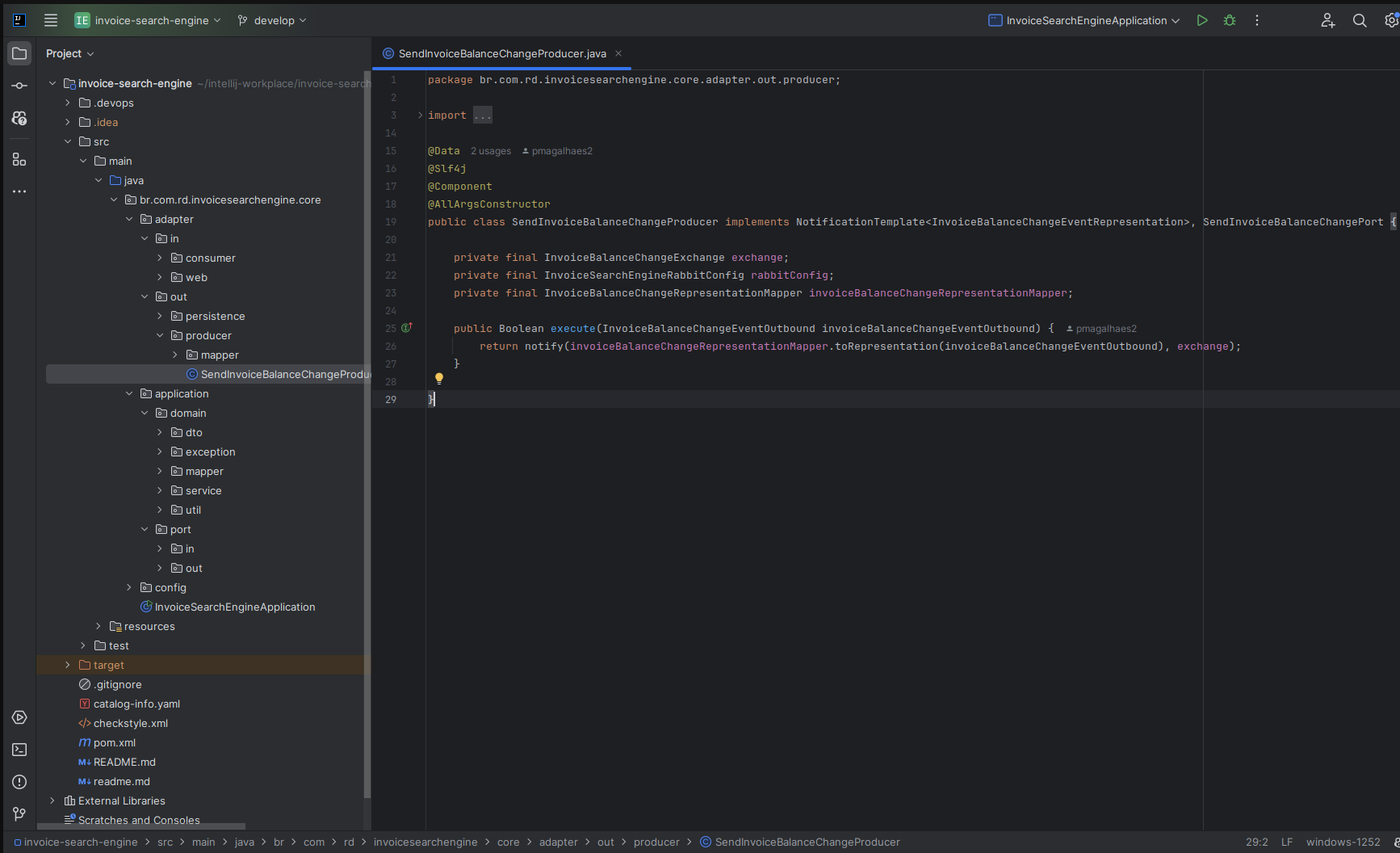
<https://github.com/fepsilvag/dse-technical-test/blob/main/Invoice.java>

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.**

Uma boa forma de trabalhar com isso é a arquitetura hexagonal, ela funciona muito bem utilizando interfaces e respeitando o SOLID, gerando código bem desacoplado

O problema dessa arquitetura é que ela gera muitas classes, em um CRUD simples talvez houvesse 3 classes (1 controller, 1 service e 1 repository), mas usando essa arquitetura haveria em torno de 18 classes (1 controller, 4 ports in, 4 implementações de usecase, 4 ports out, 4 implementações de port out e 1 repository, além dos DTOs e seus mapeamentos)

Exemplo: Há um tempo tivemos uma mudança relacionada ao Outlook, ele parou de funcionar via autenticação simples e foi necessário integrar via Microsoft Graph para enviar e-mails, foi necessário apenas alterar o EmailAdapter, não houve necessidade de alterar mais nada graças ao desacoplamento

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%203.png>

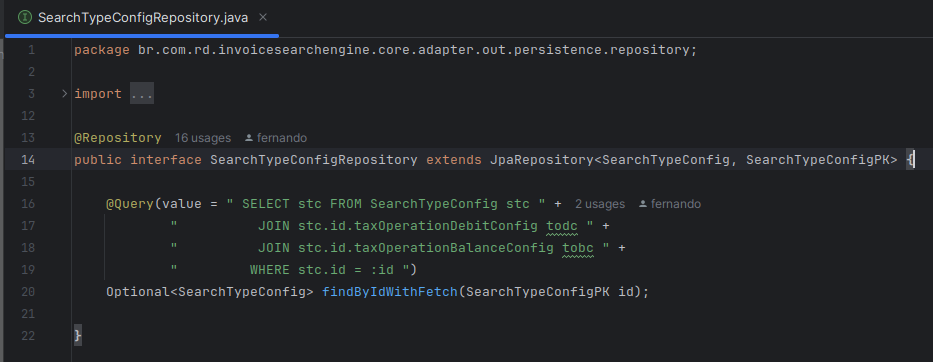
1. **Describe your experience with Angular, including its core features and use cases. Provide an example of a practical application where you used Angular and include a code snippet demonstrating a key feature, such as component communication, data binding, or service integration.**

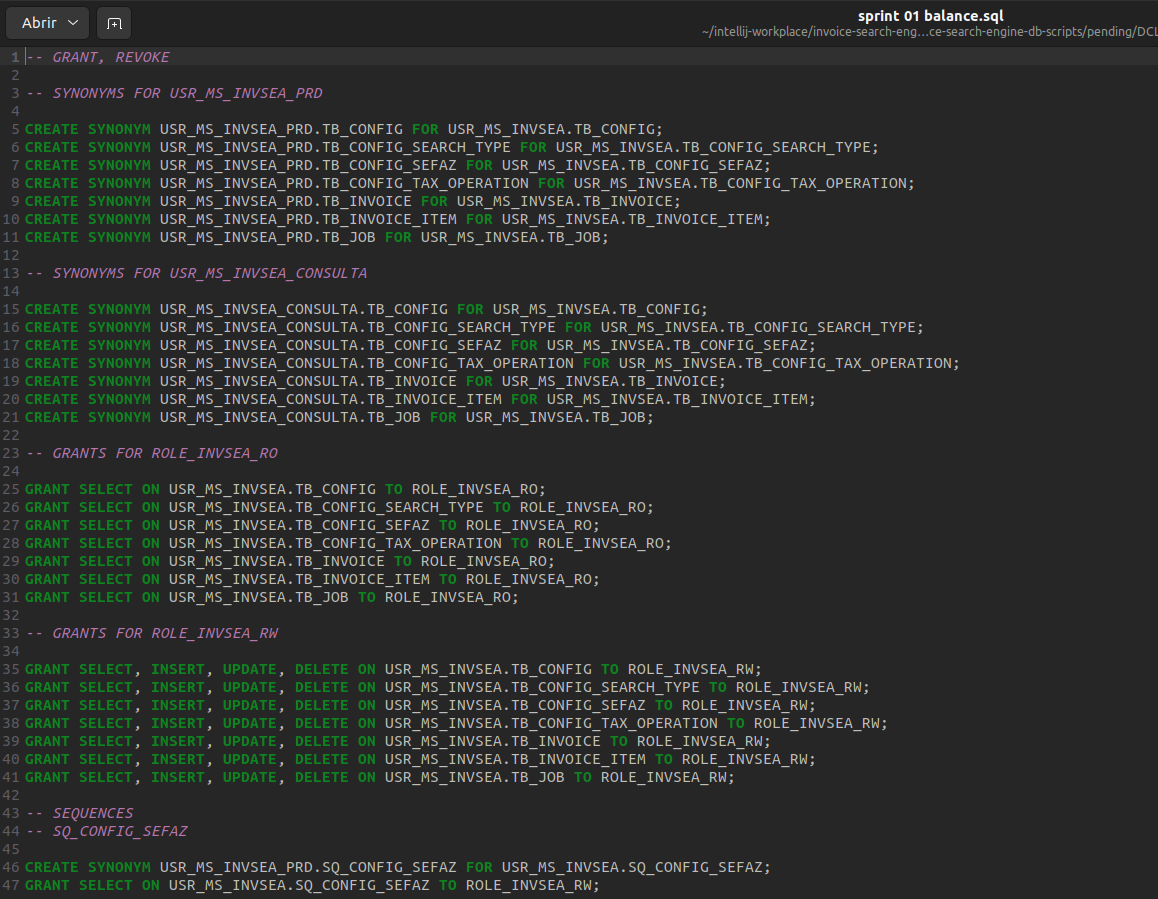
Há cerca de 4 anos a RD usava Angular, hoje em dia usam oficialmente React com o framework Next.Js e Typescript, inclusive teve alguns projetos que foram migrados para React, assim que comecei a trabalhar com microserviços trabalhei um pouco com Angular, acredito que por 1 a 2 anos, realizamos o projeto do volume-balance em Angular.

A arquitetura do Angular é muito centrada em components (o mesmo que widget no Flutter), outro benefício é a injeção de dependências, ele já é nativo da linguagem, o que facilita muito, hoje na RD temos que usar bibliotecas para facilitar essa injeção, o que tornou o processo mais complicado, Next.js torna o React mais parecido com o Angular nesse sentido

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.**

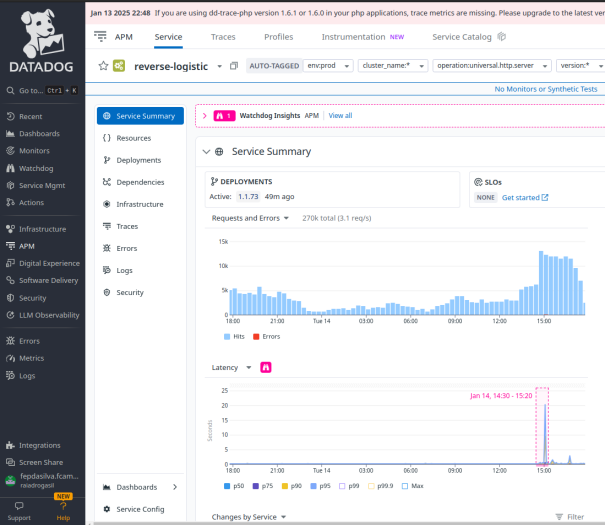
Todas as nossas consultas são realizadas via query params, contamos também com 3 usuários do banco de dados, um usuário OWNER responsável pela criação das tabelas, um usuário PRD responsável pela modificação dos dados (usuário da aplicação) e um usuário CONSULTA responsável pelas consultas, geralmente utilizadas em aplicativos de relatórios

  
**<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%205-1.png>**

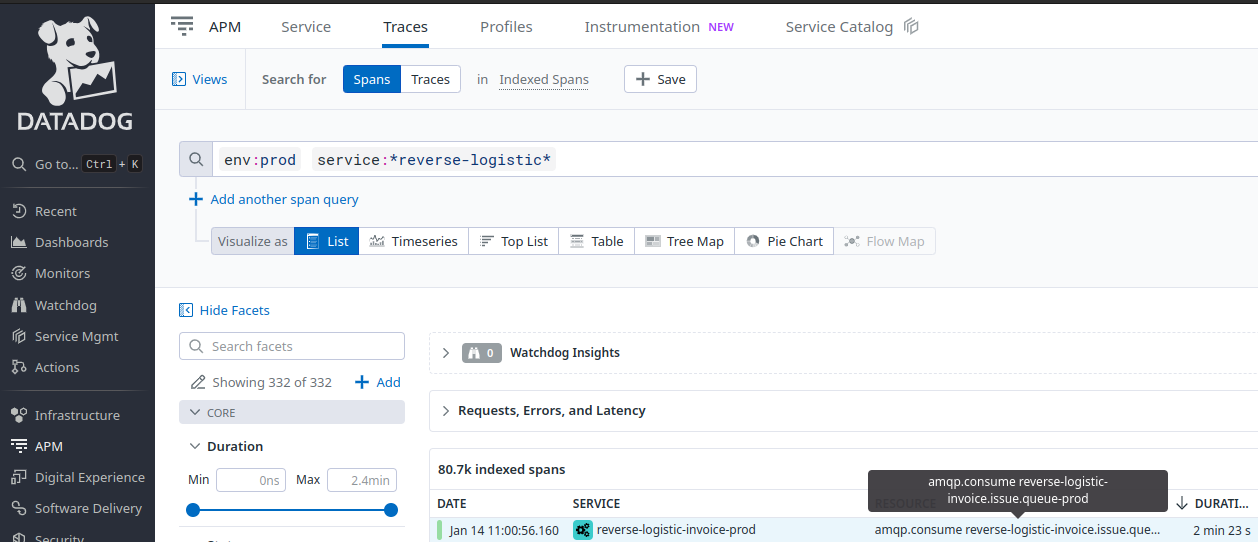
  
**<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%205-2.png>**

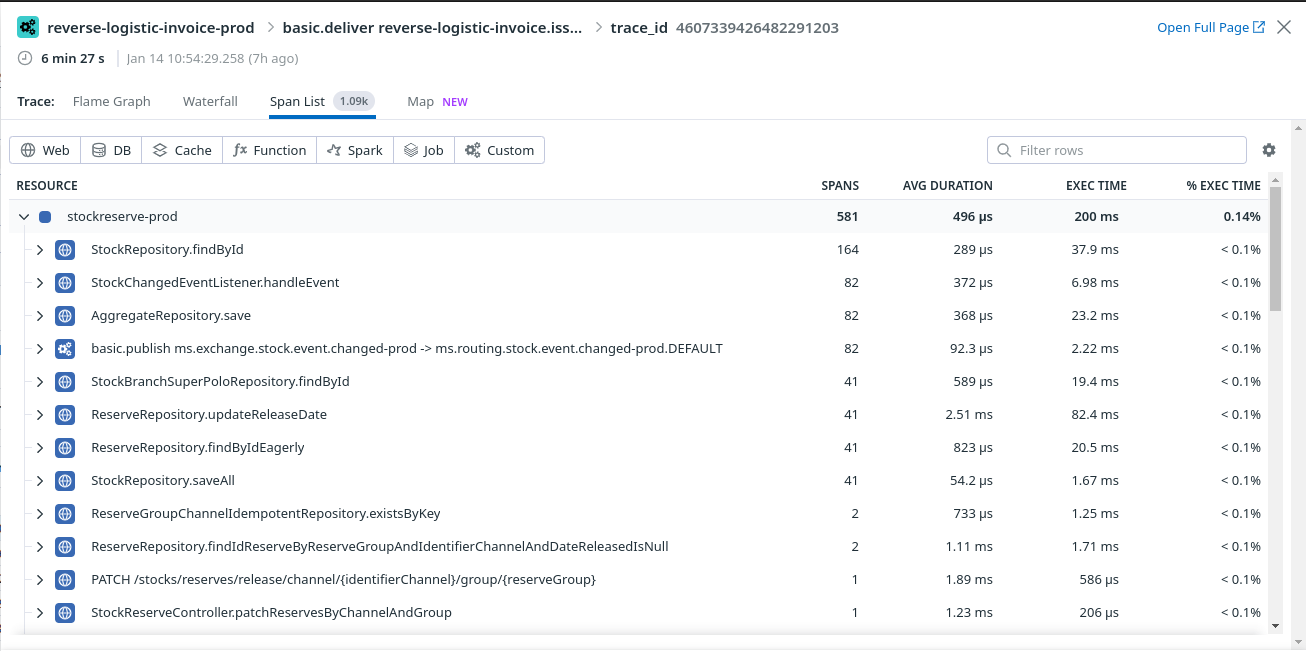
1. **Describe the steps you would take to diagnose and improve the performance of a batch process that interacts with a database and an FTP server. Explain how you would identify bottlenecks, optimize database queries, improve logic execution, and enhance file transfer efficiency. Provide examples of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

Já trabalhei com new relic e grafana, mas atualmente estamos usando o datadog, quando temos algum problema de performance eu abro o service no datadog e investigo o que está acontecendo, se há algum problema com esse serviço

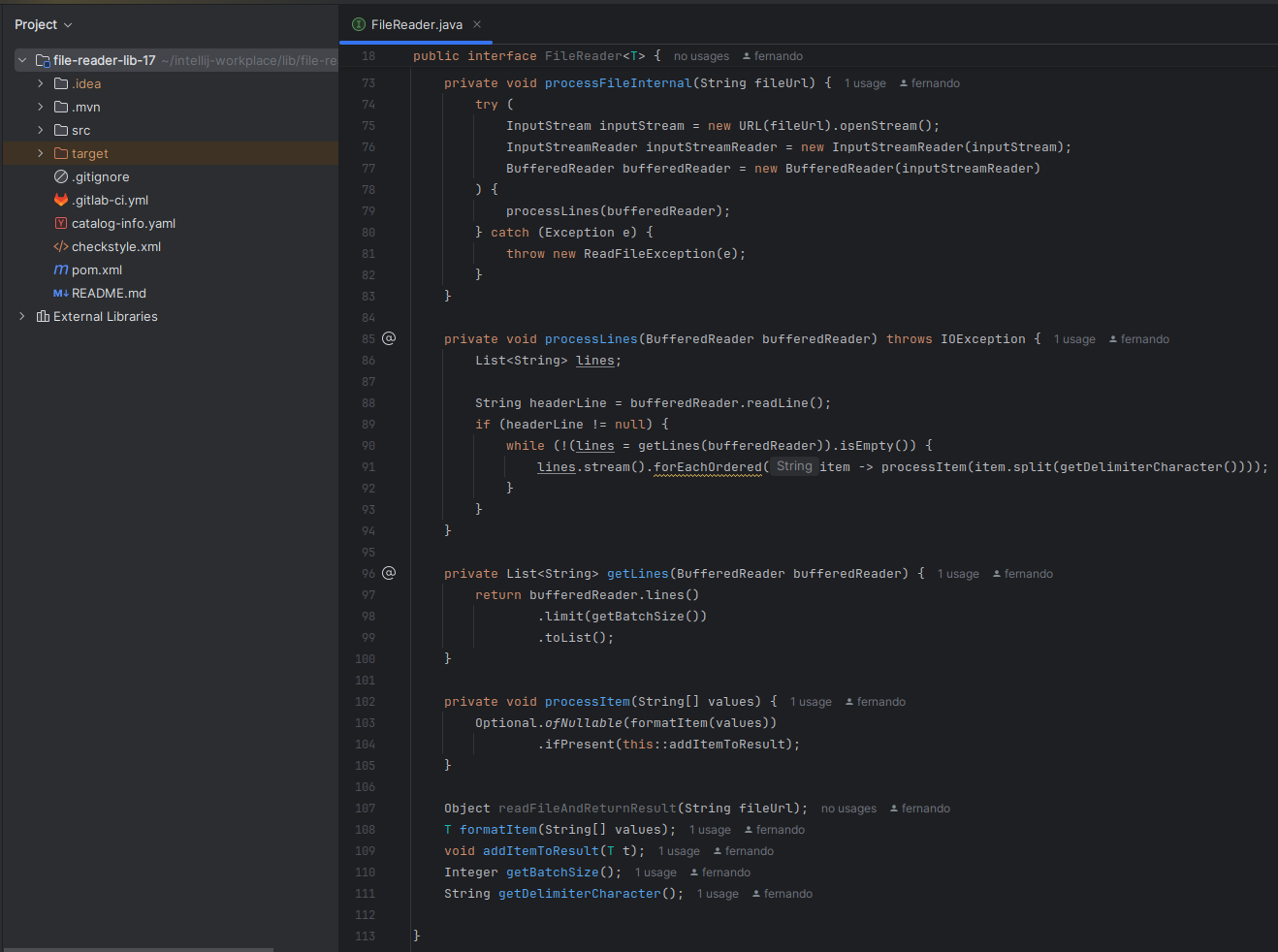
  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%206-1.png>

Em seguida, analiso todos os traces da aplicação, procurando os processos mais “demorados”, abro eles e analiso o que está demorando mais, para identificar possíveis lentidões graves no processo.

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%206-2.png>

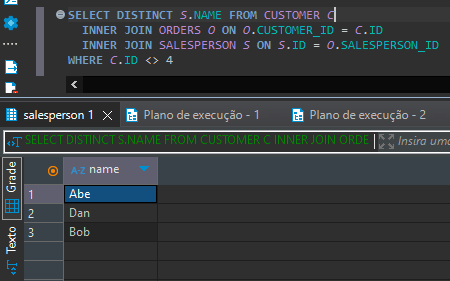
  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%206-3.png>

Sobre integração com ftp, já participei de diversas integrações nesse sentido, mas geralmente como consumidor, neste caso sempre utilizo a leitura via Stream para evitar alto consumo de memória (grande problema de integração com arquivos), para envio eu sei que é sempre bom compactar o arquivo, para “reduzir” o tamanho, diminuindo o custo de infra para a transferência

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%206-4.png>   
(I created this lib)

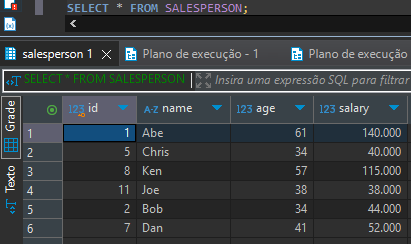
|  |  |
| --- | --- |
| **Salesperson** | **Customer** |
| |  |  |  |  | | --- | --- | --- | --- | | **ID** | **Name** | **Age** | **Salary** | | 1 | Abe | 61 | 140000 | | 2 | Bob | 34 | 44000 | | 5 | Chris | 34 | 40000 | | 7 | Dan | 41 | 52000 | | 8 | Ken | 57 | 115000 | | 11 | Joe | 38 | 38000 | |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

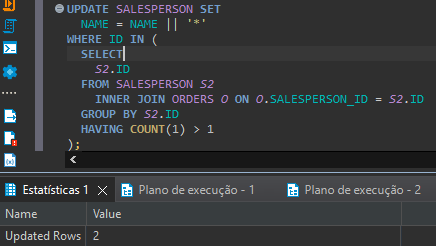
|  |
| --- |
| **Orders** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ID** | **order\_date** | **customer\_id** | **salesperson\_id** | **Amount** | | 10 | 8/2/96 | 4 | 2 | 540 | | 20 | 1/30/99 | 4 | 8 | 1800 | | 30 | 7/14/95 | 9 | 1 | 460 | | 40 | 1/29/98 | 7 | 2 | 2400 | | 50 | 2/3/98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

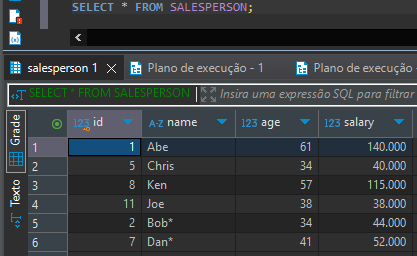
1. **Given the tables above, write the SQL query that:   
     
   a. Returns the names of all Salesperson that don’t have any order with Samsonic.**

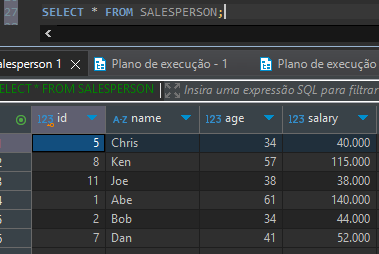
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%207-1.sql>

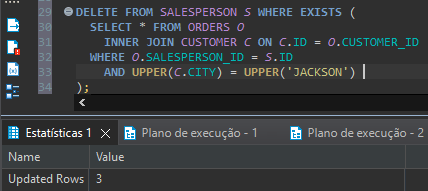
**b. Updates the names of Salesperson that have 2 or more orders. It’s necessary to add an ‘\*’ in the end of the name.**



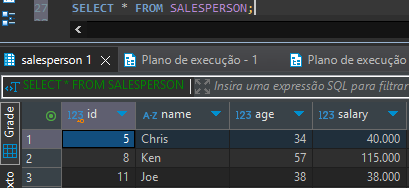
  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%207-2.sql>

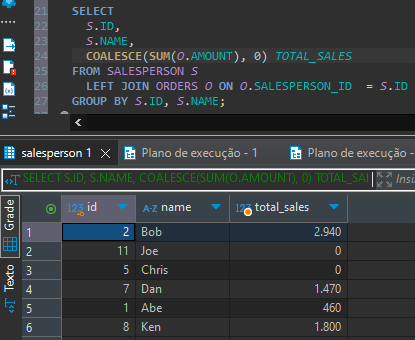


**c. Deletes all Ssalesperson that placed orders to the city of Jackson.**



<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%207-3.sql>

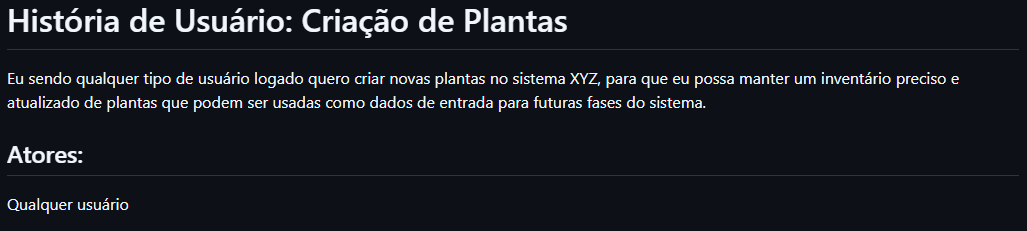


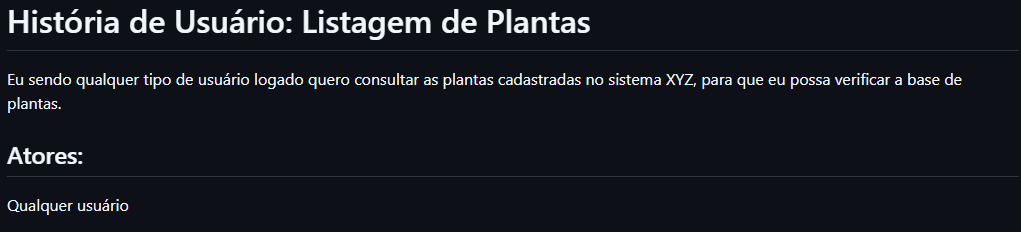
**d. The total sales amount for each Salesperson. If the salesperson hasn’t sold anything, show zero.**  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%207-4.sql>

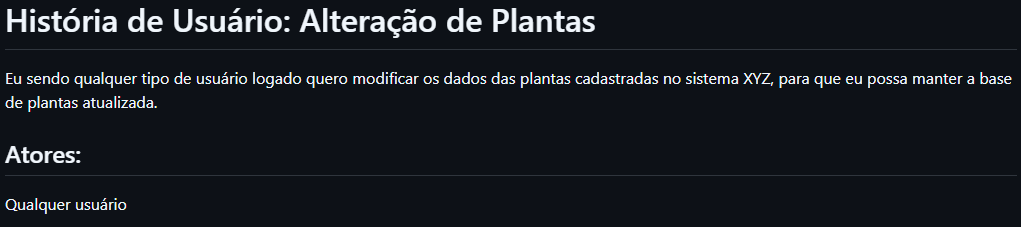
1. **The customer has a system called XYZ and intends to start updates split into 3 phases. The requirements for the first phase are as follows:**
2. **Enable new data entries in the system, which will serve as input for the second phase.**
3. **Implement functionality to create, update, delete, and 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.**

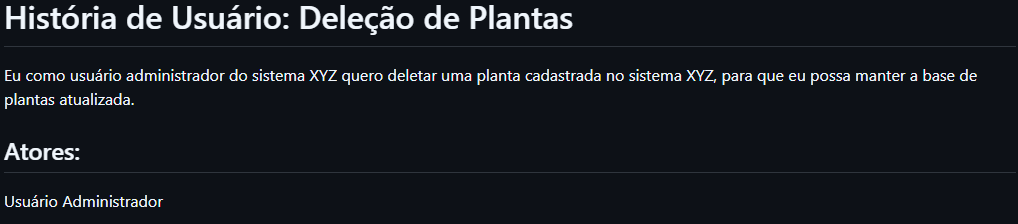
**Task:  
Based on the above information:**

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.**

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%208-1.md>

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%208-2.md>

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%208-3.md>

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%208-4.md>

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.**

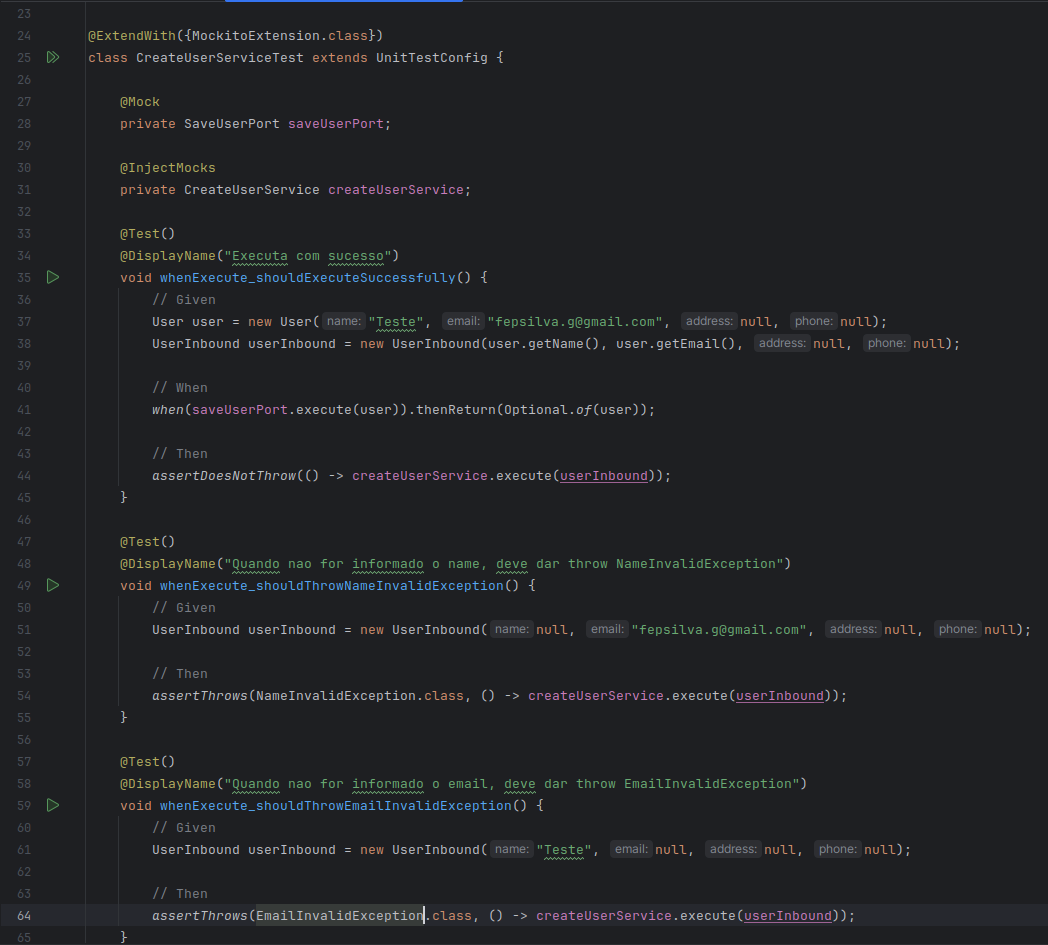
**Task:**

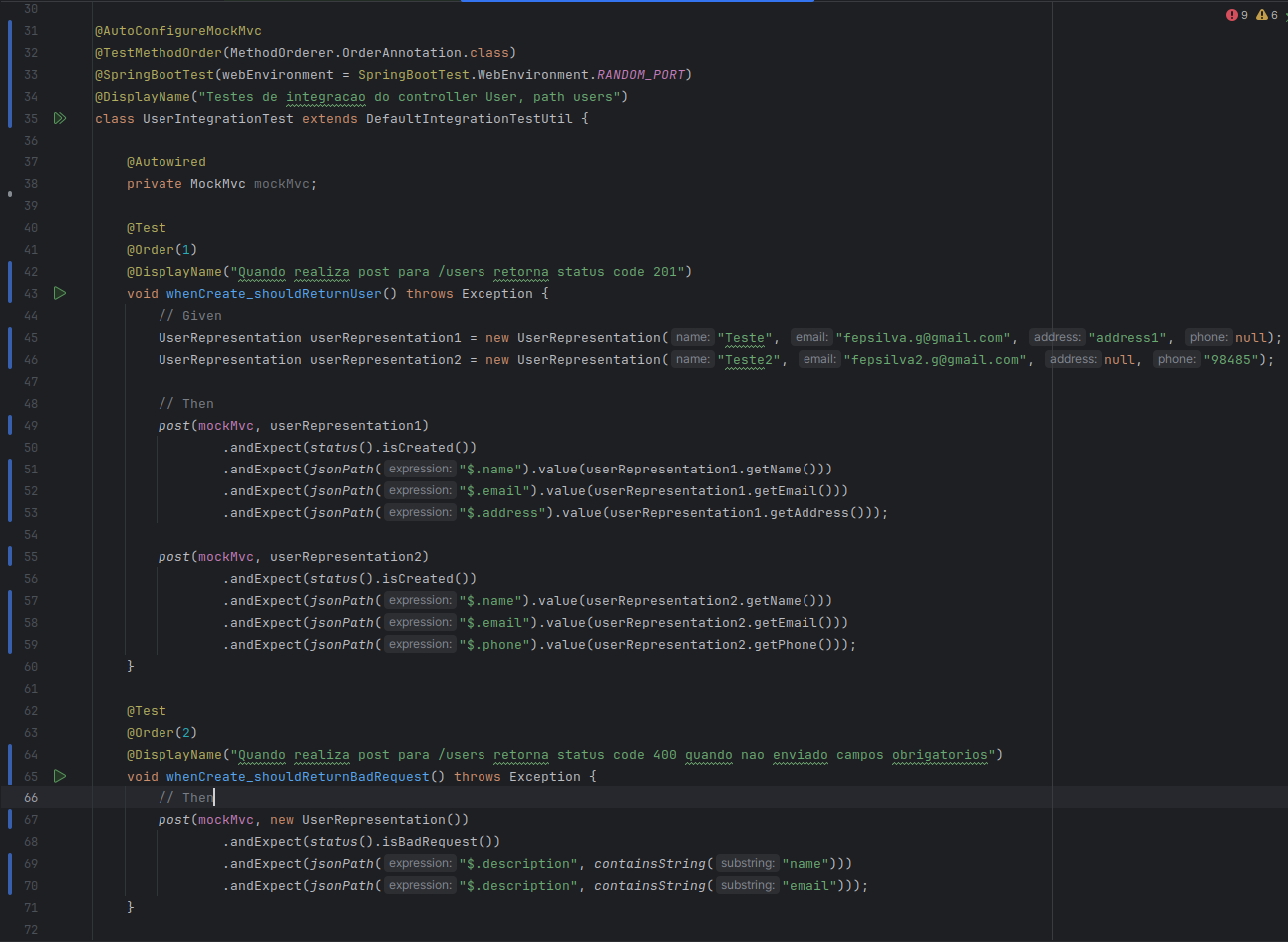
1. **Describe the types of tests you would implement (e.g., unit, integration, or end-to-end tests) and explain the scenarios you would test to ensure the functionality works as expected.**
2. **Provide examples of edge cases and how you would handle them.**
3. **Include an example of a test case in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

Testes Unitários: Temos atualmente uma política de cobertura de testes em 90%, eu prezo pelo 100%, eu utilizaria eles principalmente para validar o retorno dos usecases (já que com os testes unitários não daria para ter um teste a nível de banco de dados), enviando todas as possibilidades de campos vazios, mockando as chamadas externas.

Testes Integrados: Acredito ser necessário principalmente para testar o retorno HTTP para cada input, então caso o campo email não seja preenchido no POST, deveria retornar 400 por exemplo, assim testaria todos os tipos de envios e retornos das APIs, porém mockando chamadas externas, geralmente o teste usa o H2 como banco

Testes End-to-End: Este teste tem como função testar sem mockar chamadas externas, então de fato o registro irá para o banco de dados e irá bater em outras APIs, utilizaria para saber se nada do que foi desenvolvido vai quebrar outros sistemas e vice versa, dependendo da linguagem utilizada é possível até envolver o front em testes End-to-End, na RD é utilizado o ruby para isso (mexi poucas vezes no ruby)

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%209-1.png>

  
<https://github.com/fepsilvag/dse-technical-test/blob/main/Question%209-2.png>