**Desafio:**

Utilizando qualquer linguagem de programação crie uma funcionalidade que receba como parâmetro de entrada uma **lista de números naturais** e como resultado da sua funcionalidade retorne **somente os números primos** que foram contidos na lista. Os números retornados deverão estar **ordenados de forma crescente.**

Um exemplo:

* Entrada = {6, 4, 2, 3,1,5,10,7,9,8}
* Saída: {1, 2, 3, 5,7}

Link do repositório do desafio:

<https://github.com/vichelly/desafio-junior-riscos-de-credito>

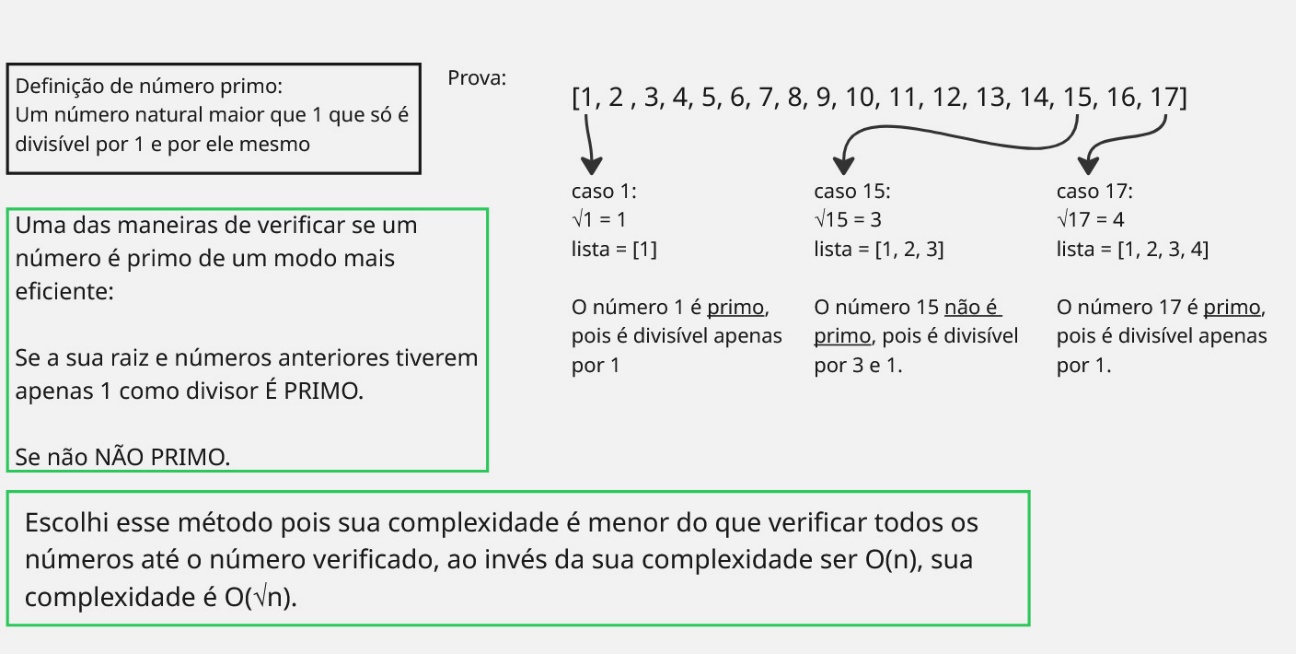
1. Qual o motivo fez você escolher a linguagem de programação utilizada para a construção do desafio acima? Alguma vantagem em utilizá-la em detrimento de outra?

Eu escolhi C# para o desafio, uma das linguagens mais usadas no back-end do banco. Considerei também Java, Go e Python. O Java, por exemplo, também é totalmente orientado a objetos. No entanto, tenho mais familiaridade com Java, e por isso escolhi o C# como forma de me desafiar tecnicamente e aprofundar meu conhecimento na stack .NET especialmente na estruturação de projetos, modularização e testes unitários.

2. Você encontrou algum problema durante a construção do seu código? Se sim, como tentou ou conseguiu solucioná-lo?

Não tive muitos problemas durante a construção da lógica em si, pois já havia resolvido o mesmo problema em ocasiões passadas. Passei mais tempo tentando recordar e encontrando o melhor jeito de resolver o problema de checagem de um número primo.

No começo fiz um algoritmo de checagem de primos que roda em O(n), mas sabia que ele poderia ter sua complexidade reduzida. Depois de pensar mais um pouco percebi que poderia otimiza-lo tornando o O(√n), fui fazendo anotações que comprovavam que o meu algoritmo seria o melhor possível em termos de tempo de execução.



No fim ele ficou com complexidade O(m√n), onde “m” é o tamanho da lista recebida e “n” é o tamanho da lista de números checados em cada número da lista.

3. Como e onde você documentaria o seu código construído?

Para documentar o código utilizei a ferramenta miro e o próprio readme.md, é possível visualizar tudo no github do projeto. Expliquei o problema que estava resolvendo no desafio, como executar o programa, como testar e a complexidade do meu algoritmo.

4. Qual a importância de versionar o código e qual ferramenta você utilizaria para versiona-lo?

É muito importante versionar o código e adotar boas práticas como o gitflow, esse flow de trabalho ajuda muito no CI/CD da aplicação e é essencial para trabalho colaborativo num software em construção ou em manutenção.

Porém no caso deste desafio posso salientar a importancia de fazer bons commits em cada implementação nova que funciona, penso que o versionamento serve como um checkpoint, caso eu quebre a aplicação com um bug indecifravel posso apenas voltar para a versão anterior e tentar de novo, além de facilitar muito o desenvolvimento do software em diferentes CPUs.

No meu caso utilizei o git e tentei fazer commits que descrevem exatamente o que foi implementado em cada versão.

5. Você utilizou algum pilar de OOP durante a construção do seu código, como herança, encapsulamento ou polimorfismo? Se sim, por que e onde? Se não, por quê?

Utilizei de forma leve alguns conceitos da Programação Orientada a Objetos. A classe PrimeNumberIdentifier representa uma entidade com responsabilidade bem definida identificar números primos o que já demonstra um uso básico de abstração, mesmo sem recorrer a interfaces ou classes abstratas. Também apliquei o encapsulamento de forma mínima, tornando o método IsPrime privado, para ocultar a lógica interna de verificação e manter a classe mais modular. Embora não tenha utilizado herança ou polimorfismo, por não serem necessários para a simplicidade do desafio, busquei estruturar o código de forma clara e separada, alinhando com boas práticas da OOP, mesmo que não em seu uso mais completo.

6. Em cima do desafio criado, você consegue identificar algum requisito funcional e algum requisito não funcional? Se sim, quais?

Entre os requisitos funcionais do desafio temos: Receber uma lista de números, os números recebidos devem ser naturais, devemos retornar os números primos da lista em uma lista ordenada.

Entre os requisitos não funcionais: O algoritmo deve verificar os números primos de maneira eficiente, o código deve ser claro, compreensível e bem estruturado para possíveis mudanças futuras, a aplicação deve permitir testes unitarios.

7. Como você explicaria sua funcionalidade criada para uma pessoa não familiarizada com programação, por exemplo, alguém do time de negócios?

Esta funcionalidade possui o objetivo de analisar uma lista de números naturais e retornar os valores dela que são primos de maneira ordenada crescente. Os números naturais são aqueles inteiros e positivos, ou seja [1, 2, 3 ...], primos são por definição aqueles que são divisíveis por 1 e por ele mesmo por exemplo [1, 2, 3, 5, 7].

8. Você utilizou alguma IDE para construir seu código, se sim, qual?

Sim, utilizei o VSCode.

9. Quais os testes você fez para validar se seu código estava se comportando conforme o esperado e como fez?

Utilizei o xUnit para realizar testes unitários, criei uma classe para testes e por meio de reflexão verificamos diretamente o método privado IsPrime.

10. É comum realizar pesquisas, ler ou relembrar termos esquecidos durante nossos desafios, a fim de conseguirmos resolvê-lo da melhor forma possível e superamos nossos problemas. Fazemos isso o tempo inteiro!

Pode compartilhar conosco qual foi sua experiencia durante esse desafio? Encontrou alguma dificuldade, ou precisou lembrar de algo? Como tentou superar a dificuldade, onde pesquisou, que ferramentas utilizou, para quem perguntou?

Gostei bastante do desafio, levei muito a sério cada etapa e me aprofundei na solução que eu proponho. Para responder as perguntas relembrei de conceitos teóricos pesquisando sobre temas de engenharia de software, analise de algoritmos, OOP e também mais técnicos pesquisando no stackoverflow, assisti tutoriais no youtube ou até utilizando o chatgpt.

Não tive muitas dificuldades em escrever o código, pois a sua sintaxe é muito amigável, uma das coisas que aprendi foi Linq que facilitou muito no retorno do meu filtro. Uma de minhas dificuldades foi aprender o xunit e faze-lo funcionar, estou constantemente aprendendo mais sobre testes unitários e já possuo alguma experiencia em testes unitários dentro do banco, o que me ajudou bastante no processo. Também pedi opiniões de colegas da área a respeito do algoritmo que resolvi usar e sua complexidade. No fim descobri que o Visual Studio pode ser melhor para o desenvolvimento em C# do que o Vscode.