# Teste - Desenvolvimento de Software

1. Crie um fork desse repositório, faça os testes, responda as perguntas e depois submeta um pull request.

2. Os testes envolvendo código devem ser feitos preferencialmente em **C# ou Java/Kotlin (Android)**. Cada codificação deve estar em uma pasta com o nome que está entre perênteses nas questões. As questões teóricas devem ser respondidas em um **pdf** que também será adicionado ao GIT. O primeiro teste será pegar os códigos e dar um run. Tem que rodar de primeira.

3. Como você se atualiza tecnicamente?

   Já fiz alguns cursos na plataforma Udemy, atualmente tenho feito cursos na plataforma Alura. Tenho acompanhado também os canais .Net pelo youtube e facebook. Sigo e acompanho blogs e sites com matérias técnicas como Lambda3, de Geovanni Bassi e instrutores do Alura.

4. Crie uma função para calcular o *n-ésimo* elemento da Sequência de Fibonacci (fibonacci).
   i. Qual solução é mais performática, iterativa ou recursiva? Por que?

      A função mais performática é a iterativa. Porque a solução iterativa utiliza laços para controlar o fluxo, o esforço é realizado em cada ciclo de repetição.

   ii. *Opcional:* Qual é o 5287º elemento da sequência?

      36997212111737864844378387651231484194667078982513515224481374
      80285857913565834984520929216272716418016617994714385706765899
      53011935414072876218520281577733814700890397774965730960629258
      03725031253390159006480374461310174544558628276672868924544402
      41565074532997862818830158805546197789485773192422757030757222
      26158144434097222670108572976030847233141767331277010055581751
      03530852448415667029324338388798940215910163493340699051580812
      54197855346709280263141652935496803290379831445043302749345673
      84664198449417836276751750879491684363632464138900052994674579
      91243252569796171131894085801209823963740996936485539601850952
      18493316678336284241270766611700540081238466107264659575911480
      92483415116275698917995188427352618386247100375163314592145521
      21113994575685035145644178266195909188893352340882666972801054
      28040803227654120652991398143150951746110043739443754100271772
      15184965522946052345643801779700328787713211314112230048483118
      47155667626216567697299077588376173307779062849404617349744089

116477389323975899297749521848146440064937630684967271526885985221533435372979915706327747996886366132632501219 33

5. O que significa SOLID?

   SOLID são princípios que quando aplicados, trazem benefícios da orientação a objetos e quando não aplicados podem resultar em problemas.

   SOLID é um acrônimo dos cinco primeiros princípios da programação orientada a objetos.

   Os princípios são:

   - S – Principio da Responsabilidade Única;
   - O – Principio Aberto e Fechado;
   - L – Princípio da Substituição de Liskov;
   - I – Princípio da Segregação da Interface;
   - D - Princípio da Inversão da Dependência.

   As definições:

   - SRP - Uma classe deve ter um, e somente um, motivo para mudar;
   - OCP – Você deve ser capaz de estender um comportamento de uma classe, sem modificá-lo;
   - LSP – As classes base devem ser substituíveis por suas classes derivadas;
   - ISP – Muitas interfaces específicas são melhores do que uma interface única;
   - DIP - Dependa de uma abstração e não de uma implementação.

   Os benefícios:

   - Código fácil de se manter, adaptar e se ajustar às alterações de escopo;
   - Código testável e de fácil entendimento;
   - Código extensível para alterações com o mínimo de esforço necessário;
   - Que forneça o máximo de reaproveitamento;
   - Que permaneça o máximo de tempo possível em utilização.

   Os problemas mais comuns são:

   - Dificuldade para criação de testes de unidade;
   - Código sem estrutura ou padrão;
   - Dificuldades de isolar funcionalidades;
   - Duplicação de código, uma alteração precisa ser feita em N pontos;
   - Fragilidade, o código quebra facilmente em vários pontos após alguma mudança.

6. O que são design patterns?

   Design Patterns são um conjunto de ideias que auxiliam na modelagem e na solução de problemas conhecidos no desenvolvimento de software.

   i. Quais são os tipos de design patterns?

      - Padrões de criação;
      - Padrões estruturais;
      - Padrões comportamentais.

   ii. Com quais você está familiarizado? Qual é a função deles?

      Tenho familiaridade com os padrões comportamentais. Eles caracterizam a forma como classes e objetos interagem e distribuem responsabilidade;

   iii. *Opcional:* Qual é sua opinião quanto ao uso de design patterns?

      Procuro aplicar os modelos Design Patterns, seguindo esse conjunto de boas práticas tenho mais facilidade na evolução e no desenvolvimento de software.

7. Qual foi o último livro técnico que você leu? Quando foi isso?

   CARVALHO, Thiago Leite. Orientação a Objetos: Aprenda seus conceitos e suas aplicabilidades de forma efetiva. São Paulo: Casa do Código.

   Lido em março de 2020.

   i. *Observação: se já tivermos lido e você for chamado para uma entrevista, perguntas poderão ser feitas a respeito do mesmo.*

8. Cite 3 maneiras diferentes de implementar Dependency Inversion.

   - Implementando a injeção de dependência no construtor da classe;
   - Implementando o Service Location;
   - Implementado Interfaces;

9. O que são ORMs?

   Em tradução livre Mapeamento de Objetos relacionais. ORM é uma camada que mapeia o modelo de objetos (aplicação) e o modelo relacional (base de dados)

   i. Quais você conhece bem?

      Conheço EF, Conheço Micro ORM Dapper, conheço My Generation, estou estudando EF Core.

   ii. *Opcional:* Cite pelo menos 2 vantagens e 2 desvantagens de seu uso.

      As vantagens são:

      - Produtividade;
      - Facilidade de manutenção;
      - Padronização

      As desvantagens:

      - Performance reduzida;
      - Complexidade;
      - Prévio conhecimento da ferramenta que será utilizada;

10. O que são microsserviços?

    Microsserviços são pequenas funções sendo executadas em um domínio específico, tendo apenas uma responsabilidade.

    Quais são suas vantagens e desvantagens?

    As vantagens:

    - Aumento da resiliência;
    - Escalabilidade aprimorada;
    - Flexibilidade no uso de tecnologias;
    - Redução de Acoplamento;
    - Facilidade de Manutenção;
    - Entregas contínuas;

    As desvantagens:

    - Deve ser bem documentado;
    - Testes podem ser complexos;
    - Deve ser usado a cultura devops;
    - Projetar tendo um plano B contra falhas;
    - Ter uma equipe especialista no assunto;

11. Com a seguinte representação de produto (crud):

(Devido ao tempo não foi possível concluir essa etapa do teste)

```
{
    "sku": 43264,
    "name": "Batata frita Ruffles Cebola & Salsa",
    "inventory": {
        "quantity": 15,
        "warehouses": [
            {
                "locality": "SP",
                "quantity": 12,
                "type": "ECOMMERCE"
            },
            {
                "locality": "MOEMA",
                "quantity": 3,
                "type": "PHYSICAL_STORE"
            }
        ]
    },
    "isMarketable": true
}
```

Crie endpoints para as seguintes ações:

- Criação de produto onde o payload será o json informado acima (exceto as propriedades isMarketable e inventory.quantity)
- Edição de produto por sku
- Recuperação de produto por sku
- Deleção de produto por sku

Requisitos:

- Toda vez que um produto for recuperado por sku deverá ser calculado a propriedade: inventory.quantity
- A propriedade inventory.quantity é a soma da quantity dos warehouses
- Toda vez que um produto for recuperado por sku deverá ser calculado a propriedade: isMarketable
- Um produto é marketable sempre que seu inventory.quantity for maior que 0
- Caso um produto já existente em memória tente ser criado com o mesmo sku uma exceção deverá ser lançada
- Dois produtos são considerados iguais se os seus skus forem iguais
- Ao atualizar um produto, o antigo deve ser sobrescrito com o que esta sendo enviado na requisição
- A requisição deve receber o sku e atualizar com o produto que tbm esta vindo na requisição

## Não é necessário o uso de bancos de dados.

**Testes são bem-vindos.**
**Você não deve levar mais do que 4 horas para o teste todo.**