Aplicando design patterns na prática com C#

Desenvolver aplicações em C# confiáveis e estruturadas com as melhores práticas do mercado.

avanade

Victor Fructuoso

Manager

Professional background

Arquiteto de Soluções com ênfase em ChatBot e RPA. Responsável técnico por projetos de grande impacto com mais de 15 anos de experiência. Especialista no ecossistema Microsoft deste o saudoso VB6 + ASP até o .NET Core e entusiasta do Azure.

Principais Projetos

Companhia Aérea

Arquiteto responsável pelo desenvolvimento do chatbot para atendimento de clientes externos.

Arquiteto responsável pelo desenvolvimento de RPA para automatizar atividades do BackOffice - BluePrism

Cosméticos

Líder Técnico responsável pelo desenvolvimento do chatbot para atendimento das consultoras

Telefonia

Arquiteto responsável pelo desenvolvimento de RPA para automatizar atividades do BackOffice

Coordenador Técnico responsável por acompanhar tecnicamente mais de 40 desenvolvedores distribuídos em 8 squads distintas em projetos focados na melhoria de produtividade do backoffice

Laboratório de Análise Clinica

Coordenador Técnico responsável por acompanhar o time de desenvolvimento.

Arquiteto responsável por serviços de autorização / elegibilidade de exames junto as operadores de planos de saúde.

Certificações

- Microsoft 70-480: Programming int HTML5 with JavaScript and CSS3
- Microsoft 70-461: Querying Microsoft SQL Server 2012
- Microsoft 70-486: Developing ASP.NET MVC 4 Web Applications
- Microsoft 70-483: Programming in C#
- Microsoft 70-515: Web Applications Development with Microsoft .NET Framework 4
- Microsoft 70-487: Developing Windows Azure and Web Services
- Microsoft 74-343: Managing Projects with Microsoft Project 2013
- Microsoft 70-481 : Essentials of Developing Windows Store Apps Using HTML5 and JavaScript;
- Scrum Org PSM I: Professional Scrum Master I

Experiências

- .NET Core, .NET Framework, MVC, Web Forms, WCF, Windows Forms, VB6 e ASP Clássico;
- SQL Server, Mongo, CosmosDB e Storage Account;
- TDD e LoadTests (Jmeter);
- RPA (Automation Anywhere e Blue Prism)
- Chatbot e Serviços Cognitivos (Microsoft BotFramework);

Definição

Design Patterns são soluções reutilizáveis para problemas comumente ocorridos (no contexto do design de software). Estes padrões foram iniciados como melhores práticas que foram aplicadas repetidamente a problemas semelhantes encontrados em diferentes contextos. Eles se tornaram populares depois que foram apresentados, de forma estruturada, no livro "Design Patterns - Elements of Reusable Object-Oriented Software" (Gang Of Four) em 1994.

O "Gang of Four" representa apenas uma de muitas coleções.

Enquete

Acesse https://www.menti.com/iq57cex4f5

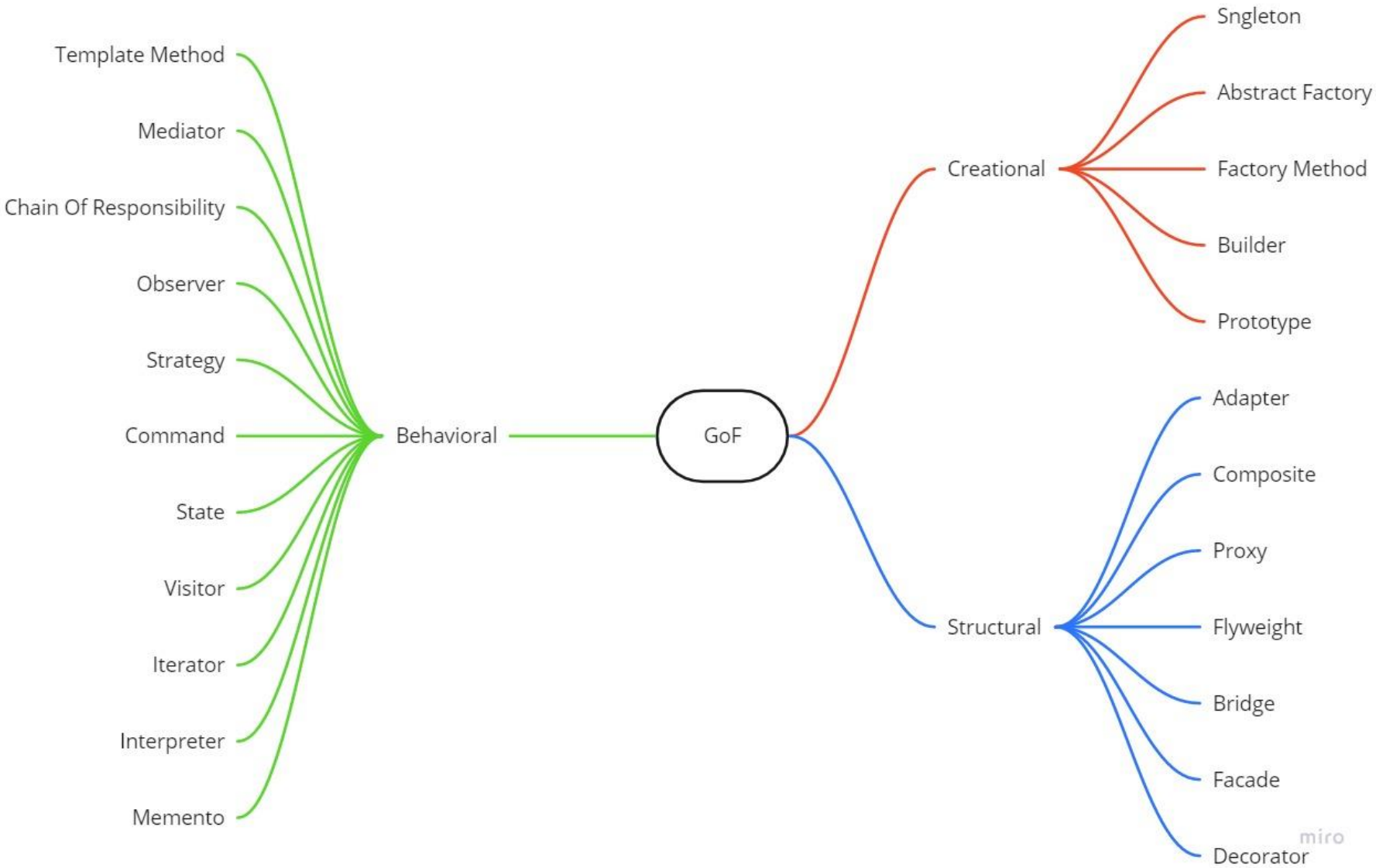
Acesse via QRCode

Resultado: https://www.mentimeter.com/s/21741c4e4451b342e8014277e3e0e79e/427a1b293813

avanade

Gang Of Four

avanade



Porque eu devo usar?

- **Produtividade:** Estes padrões são modelos de resolução de problemas que já foram utilizados e testados inúmeras vezes;
- **Manutenção:** Os padrões são baseados em soluções de baixo acoplamento e padronização de soluções;
- **Temos Universais:** Os projetos são amplamente conhecidos desta forma as discussões técnicas são facilitadas, é mais simples falar o nome de um "design pattern" do que toda vez ter que explicar o seu comportamento.

ATENÇÃO!!!

Antes de começar a aplicar padrões de projetos precisamos entender algumas coisas...

avanade

Desuso

Alguns padrões surgiram para solucionar limitações de linguagens de programação com menos recursos no que diz respeito à abstração, nestes casos os padrões eram como “gambiarras” que proporcionavam à linguagem a possibilidade de fazer implementações que não eram possíveis nativamente.

Linguagens mais recentes trazem alguns destes recursos nativamente, em alguns outros casos os padrões foram substituídos por padrões mais recentes.

O padrão Strategy, por exemplo, pode ser substituído pelo uso de uma função anônima.

Soluções “Prontas”

Os padrões não são soluções prontas, códigos que podemos pegar prontos e “jogar” dentro do projeto, em alguns casos é necessário ajustar o padrão ao contexto em que o projeto necessita, e isso costuma demandar um conhecimento mais profundo por parte da equipe de desenvolvimento.

A “bala de prata”

É comum ver desenvolvedores que ao conhecer um novo padrão / técnica, tentam encaixar ele em todos os cenários, inclusive em situações onde uma abordagem mais simples seria suficiente para resolver o problema.

Um martelo é ótimo para colocar um prego na parede, mas não funciona tão bem se você tiver um parafuso.

Lembrem-se: Não é uma competição para ver quem usa mais padrões.

S.O.L.I.D.

avanade

Os Princípios do S.O.L.I.D.

Letra	Sigla	Nome	Definição
S	SRP	Principio da Responsabilidade Única	Uma classe deve ter um, e somente um, motivo para mudar.
O	OCP	Princípio Aberto-Fechado	Você deve ser capaz de estender um comportamento de uma classe, sem modificá-lo.
L	LSP	Princípio da Substituição de Liskov	As classes base devem ser substituíveis por suas classes derivadas.
I	ISP	Princípio da Segregação da Interface	Muitas interfaces específicas são melhores do que uma interface única.
D	DIP	Princípio da inversão da dependência	Dependa de uma abstração e não de uma implementação.

Problemas comuns em aplicações que NÃO usam o S.O.L.I.D.

- Duplicidade de Código;
- Código sem estrutura coesa;
- Dificuldade de manter / evoluir;
- Pequenos ajustes podem quebrar o código, inclusive em outras partes do sistema;
- Dificuldade para executar e criar testes unitários;
- Dificuldade de reaproveitar código para outras aplicações.

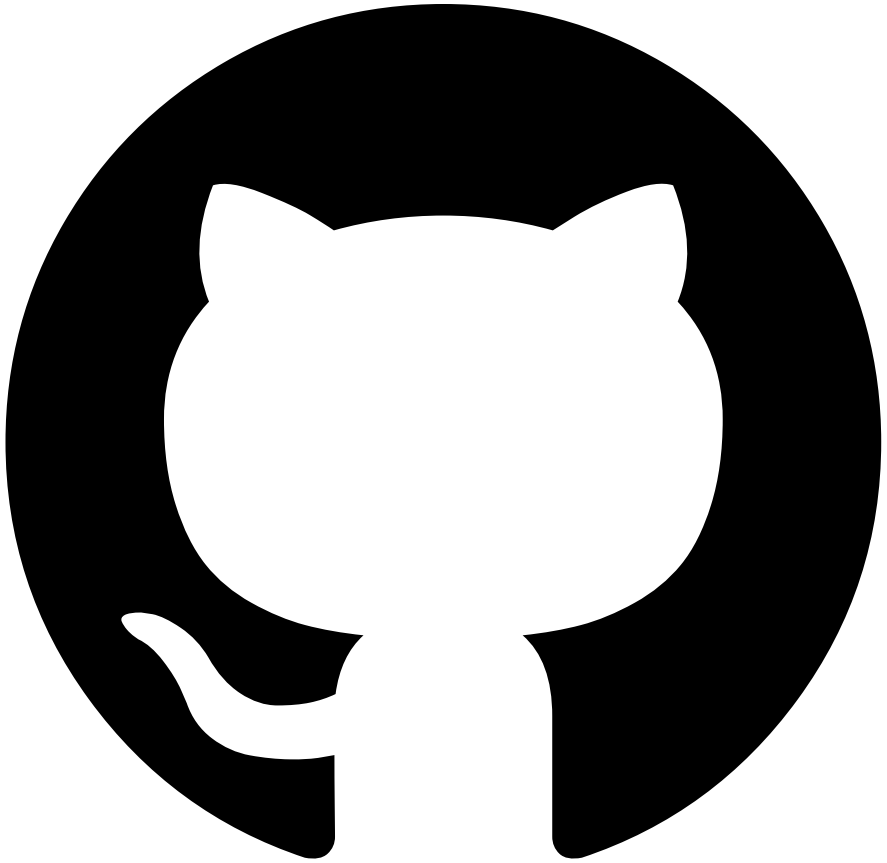
Principais benefícios

- Fácil manutenção;
- Fácil entendimento;
- Organização;
- Aberta a receber novas funcionalidades sem danos colaterais;
- Reaproveitamento de código;
- Fácil adaptação a mudanças no escopo do projeto.

Exemplos Práticos (WebAPI REST .NET Core)

Como os padrões de projeto podem nos ajudar a ‘buscar’ os princípios do S.O.L.I.D.?

avanade



https://github.com/fructuoso/DesignPatternSamples

avanade

Dúvidas?

avanade

Estudos Complementares

Cloud Design Patterns: Apresenta os principais desafios do desenvolvimento na nuvem e padrões difundidos no mercado para supera-los.

Link: https://docs.microsoft.com/en-us/azure/architecture/patterns/

Referências:

- https://www.oodesign.com
- https://www.dofactory.com/net/design-patterns
- https://refactoring.guru/pt-br/design-patterns/
- “Agile Principles, Patterns, and Practices in C#”
- “Design Patterns - Elements of Reusable Object-Oriented Software”

avanade