INJEÇÃO DE DEPENDÊNCIAS E SEUS CONCEITOS

O que preciso saber?

PRIMEIRA SEMANA

1. Component-Based Architectural Style
2. Inversion of Control
3. Dependency Inversion
4. Class Coupling

SEGUNDA SEMANA

5. *Service Locator*
6. *Dependency Injection*
7. *Post-Construction Resolve*

RESTO DA SUA VIDA

Padrões ou Estilos de Arquitetura

Architecture Patterns or Styles

Representam um conjunto de princípios que dão forma a uma aplicação, especificam responsabilidades, regras e instruções para o relacionamento entre os componentes de um sistema.

Arquitetura Baseada em Componentes

Component-Based Architectural Style

Se baseia na decomposição do projeto em componentes funcionais ou lógicos que expõem interfaces de comunicação bem definidas, fornecendo assim um nível mais alto de abstração.

Inversão de Controle

Inversion of Control - IoC

Inverte fluxo de controle de um sistema, onde as classes deixam de se preocupar com QUAIS classes precisam para funcionar, e se focam em COMO utilizá-las.

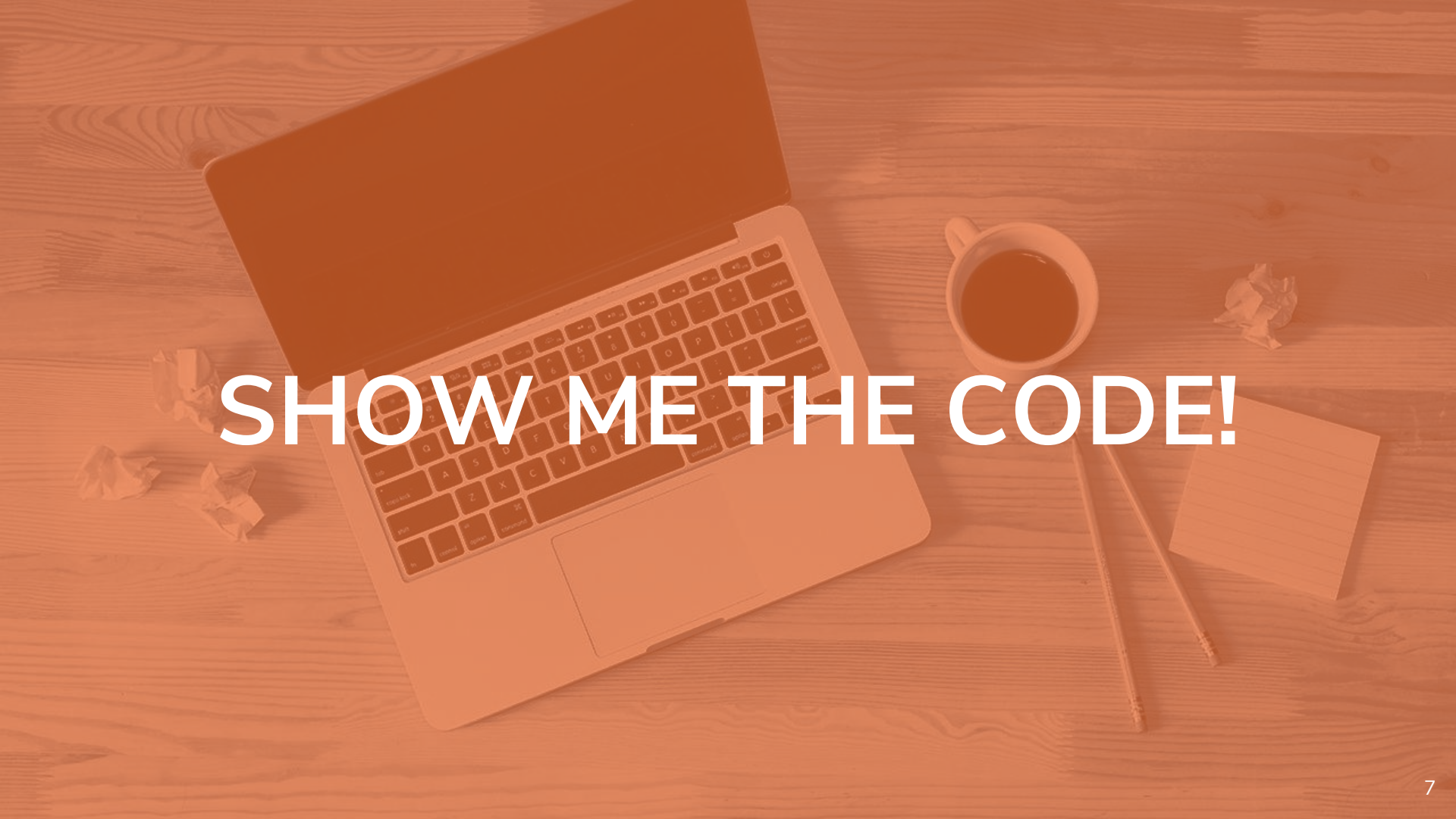
Sem IoC

Com IoC

Exemplo das Frutas

Sem IoC: você pede por uma “maçã” e sempre recebe mais “maçãs” quando pede mais.

Com IoC: você pede por “frutas” e pode receber frutas diferentes sempre que pedir mais, ex.: “maçã”, “banana” etc.


SHOW ME THE CODE!

Inversão de Dependências / Acoplamento de Classes

Dependency Inversion - DIP / Class Coupling

Tem como objetivo desacoplar os módulos de um sistema através de abstrações.

A. Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações;
B. Abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

Generalização de Padrões de Inversão de Dependências

Pattern Generalization

Em muitos projetos o conceito de inversão de dependências é considerado um conceito que deve ser generalizado por dois motivos:

- É mais fácil aplicar um princípio como um padrão de código.
- Como muitas ferramentas de teste de unidade dependem de herança para criar mocks, a utilização de interfaces genéricas entre classes tornou-se a regra.

Pontos Negativos

Major Drawbacks

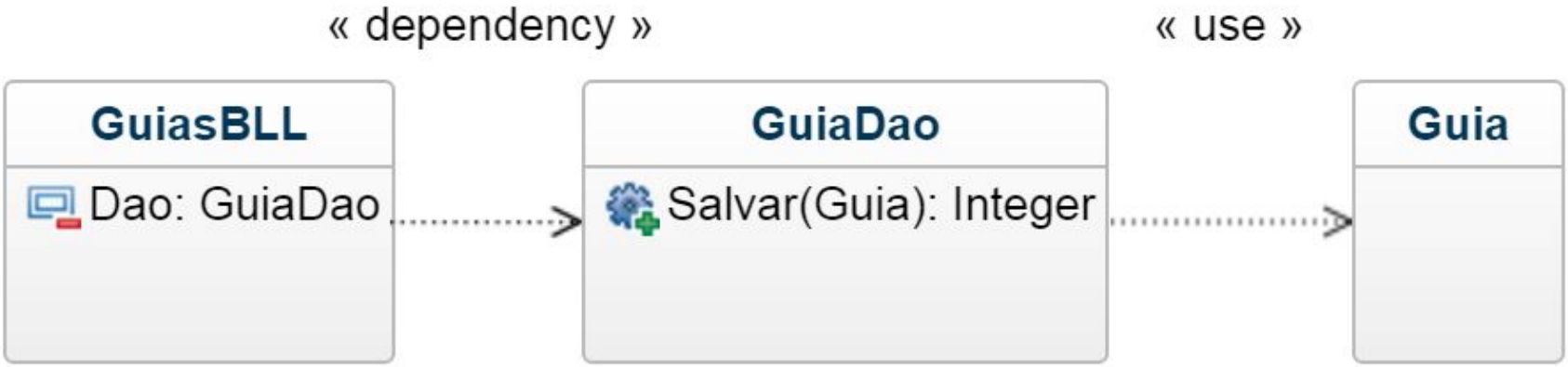
- Apenas utilizar uma interface no lugar de uma classe não reduz acoplamento, apenas pensar sobre a abstração de interações pode levar a um design menos acoplado.
- Utilizar interfaces por todo seu sistema o torna mais complexo e difícil de entender
- A generalização de interfaces requer mais código, em especial factories que geralmente estão ligadas a injeção de dependências.

Implicações no Design

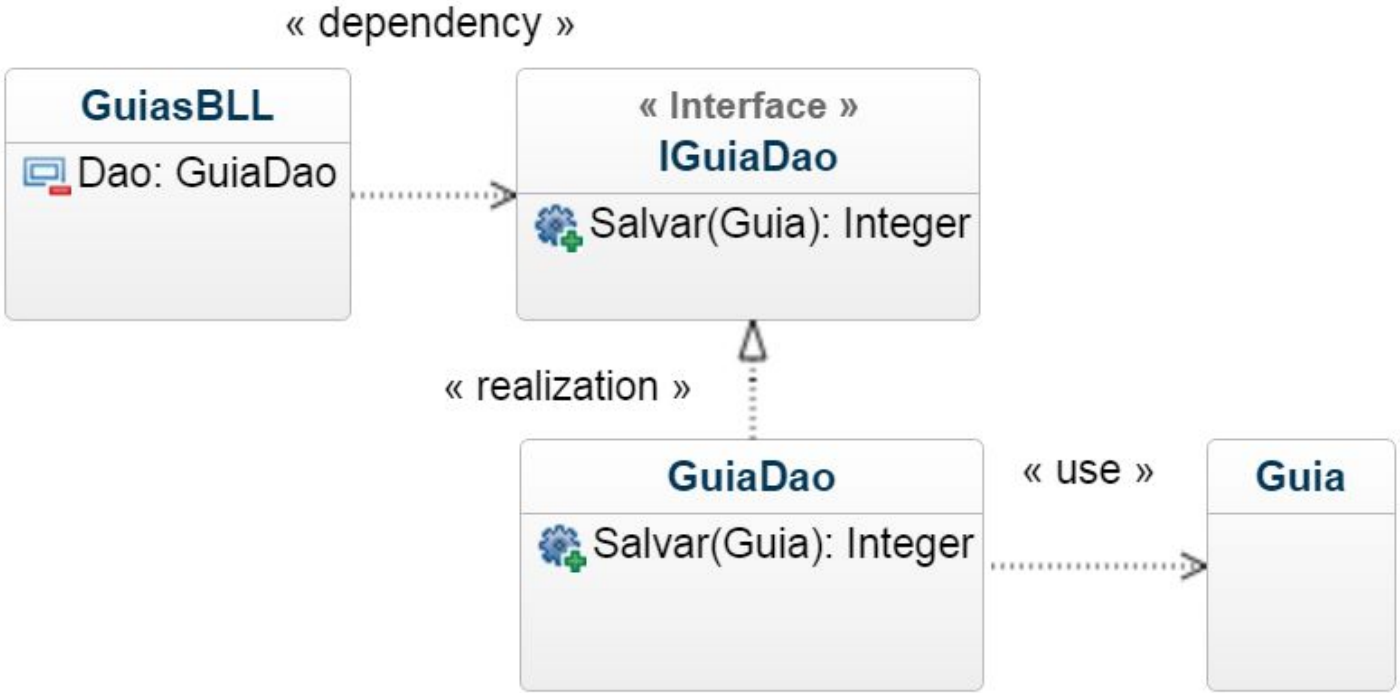
Design Implications

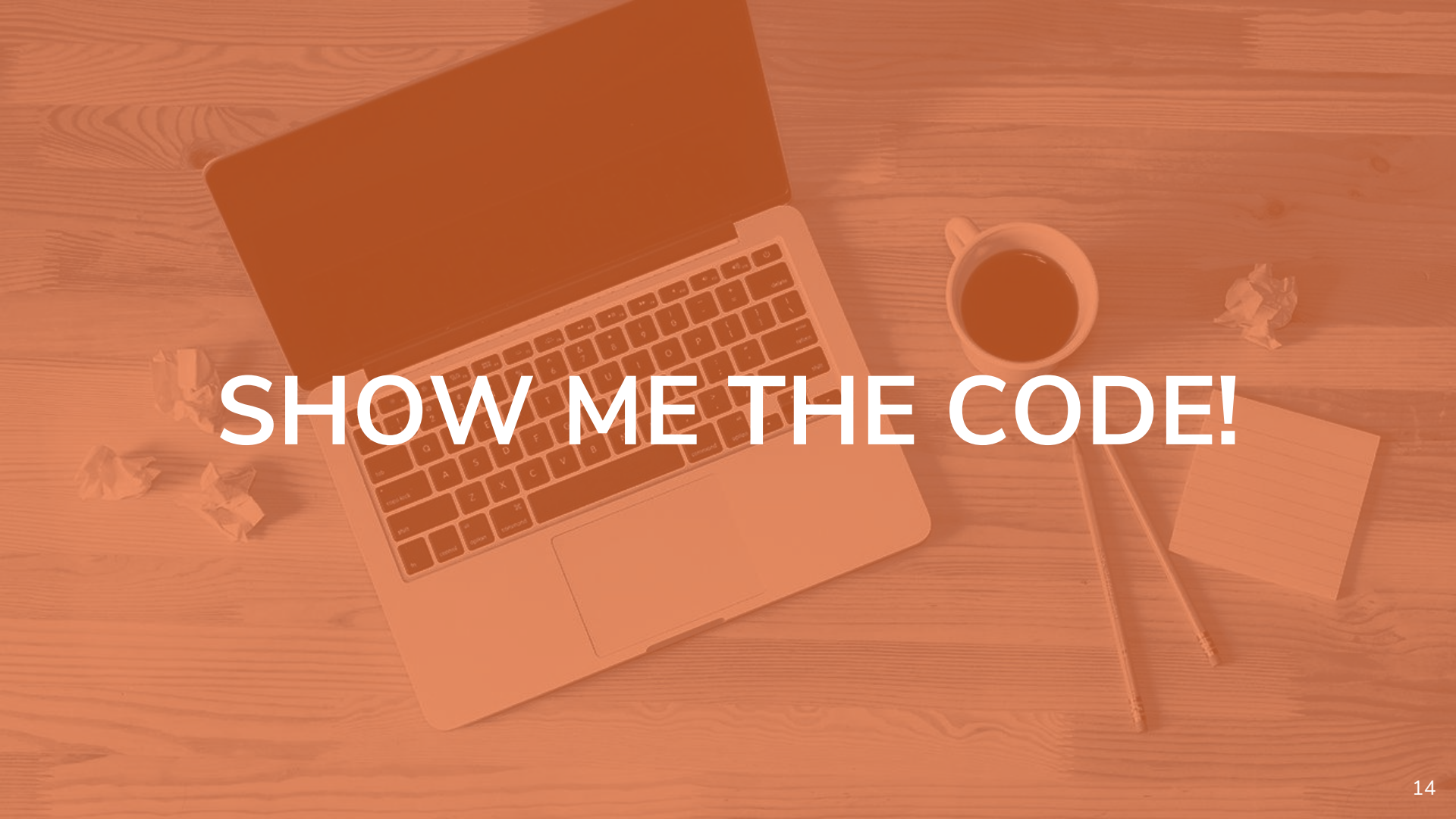
- Todas as variáveis de uma classe devem ser interfaces ou abstrações.
- Classes concretas devem se comunicar apenas através de interfaces ou classes abstratas.
- Nenhuma classe deve derivar de uma classe concreta.
- Nenhum método deve sobrescrever(override) um método já implementado.
- Toda instanciação de variáveis deve depender de um padrão criação(creational) como os padrões de factory, ou utilizar um framework de injeção de dependências.

Sem Inversão de Dependências



Com Inversão de Dependências




SHOW ME THE CODE!

Utilize Abstrações!

Pare de instanciar objetos dentro das classes

Declare dependências como interfaces

Classes podem enviar instâncias

Testes unitários podem enviar mocks