



# Módulo 8

**Princípios de design de  
microsserviços - Os 12  
fatores**

## Módulo 8

### **Aula 1 – Princípios de design de software como serviço**

- Aplicações modernas são costumeiramente entregues como serviço
- Para que isso pudesse ser feito de forma padronizada, surgiu uma metodologia chamada de Twelve Factors, ou 12 fatores
- Esta metodologia pode ser utilizada em aplicações escritas em qualquer linguagem de programação e que utilizem quaisquer componentes de suporte (bases de dados, caches, etc.)

## Módulo 8

### Aula 1 – Princípios de design de software como serviço

- Ao usarmos esta metodologia, escreveremos serviços que:
  1. Usam formatos declarativos para automatizar sua configuração inicial e minimizar o tempo de onboarding de novos desenvolvedores no projeto
  2. Tem um contrato claro com o sistema operacional onde rodam, garantindo o máximo de portabilidade entre os ambientes onde são executados
  3. São adequados para implantação em plataformas de nuvem, evitando a necessidade de administração do sistema e de servidores físicos
  4. Minimizam a divergência entre ambientes de desenvolvimento e produção, permitindo deployment contínuo e máxima agilidade
  5. Sejam facilmente escaláveis sem mudanças significativas em ferramentas, arquiteturas ou práticas de desenvolvimento

## Módulo 8

### Aula 2 – Base de código, dependências e configurações

#### Base de código

- A aplicação deve ter seu versionamento controlado por um sistema de controle de versão (Git, Mercurial, etc.)
- Existe sempre uma correlação 1:1 entre a aplicação e a sua base de código
- Múltiplas aplicações não devem compartilhar a mesma base de código
- Existe apenas uma base de código por aplicação, mas podem existir diversos deploys por aplicação. Um deploy nada mais é do que uma instância da sua aplicação rodando num determinado ambiente (produção, staging, etc.)

## Módulo 8

### **Aula 2 – Base de código, dependências e configurações**

- A base de código é a mesma entre todos os deploys
- Um ambiente de testes pode ter código ainda não entregue em produção
- Todos os ambientes compartilham a mesma base de código

## Módulo 8

### **Aula 2 – Base de código, dependências e configurações**

#### Dependências

- Define que todas as dependências devem ser declaradas e isoladas
- Nunca confia na existência de implícita de pacotes em todo o sistema, ou seja, declara todas elas, completa e exatamente, através de um manifesto de declaração de dependência
- Tal especificação de dependências é aplicada para todos os ambientes da aplicação

## Módulo 8

### **Aula 2 – Base de código, dependências e configurações**

- Um dos principais benefícios é a simplificação da configuração da aplicação para novos desenvolvedores, dado que os scripts de build e os sistemas de gerenciamento de dependências devem garantir uma instalação unificada
- Se a aplicação precisa executar alguma ferramenta de sistema (Imagemagik, etc.), esta ferramenta precisa ser empacotada junto da aplicação

## Módulo 8

### Aula 2 – Base de código, dependências e configurações

#### Configurações

- Armazenar as configurações no ambiente
- A configuração de uma aplicação é tudo o que pode variar entre deploys (endereços de bases de dados, usuários, etc.)
- Configurações deste tipo não podem ser armazenadas como constantes (chumbadas no código), pois configurações variam entre deploys, mas não o código



## Módulo 8

### **Aula 2 – Base de código, dependências e configurações**

- Para saber se sua aplicação está bem configurada, pense em como seria se seu código fosse liberado para o público: as pessoas seriam capazes de enxergar suas credenciais?
- Idealmente, armazene suas configurações em variáveis de ambiente, pois são agnósticas de linguagem e de sistema operacional

### **Aula 3 – Serviços de apoio, build, processos e vínculos de porta**

#### Serviços de apoio

- Trate serviços de apoio como serviços anexados
- Um serviço de apoio é qualquer serviço que sua aplicação consuma via rede durante seu funcionamento: bases de dados, filas, etc.
- Uma aplicação 12 fatores não faz distinção entre serviços locais e de terceiros: ambos são considerados anexados.

## Módulo 8

### **Aula 3 – Serviços de apoio, build, processos e vínculos de porta**

- Deveria ser possível trocar a URL do banco de dados, por exemplo, por uma de um banco gerenciado na nuvem sem qualquer mudança no código
- Cada serviço de apoio distinto é considerado um recurso anexado. Recursos podem ser anexados e desanexados entre deploys e sem mudar nada no código

## Módulo 8

### Aula 3 – Serviços de apoio, build, processos e vínculos de porta

Build (construir, lançar e executar)

- Separe estritamente os estágios de construção e execução
- Uma base de código é transformada em deploy em três estágios distintos:
  1. Construção (build), que transforma um repositório em um pacote executável
  2. Lançamento (release), que combina a construção do estágio anterior com a configuração atual do deploy
  3. Execução, que nada mais é do que rodar a aplicação no ambiente de execução selecionado

## Módulo 8

### **Aula 3 – Serviços de apoio, build, processos e vínculos de porta**

- Os estágios de construção, lançamento e execução devem ser estritamente separados (deveria ser impossível alterar o código em tempo de execução)
- Cada release deve ter um código de identificação único
- Builds podem ser iniciados por desenvolvedores ou de forma automatizada (continuous deployment)

## Processos

- Execute a aplicação com um ou mais processos que não armazenem estado
- Processos não devem compartilhar nada entre si
- Se algo precisar ser compartilhado entre eles, deverá ser persistido em serviços de apoio

## Módulo 8

### Aula 3 – Serviços de apoio, build, processos e vínculos de porta

#### Vínculos de porta (port binding)

- Exporte serviços via vínculo de portas
- A aplicação deve ser completamente auto-contida e não pode depender de injeções em tempo de execução de servidores web para criar para criar um serviço que defronte a web
- Isso significa que sua aplicação deve exportar o protocolo HTTP através da vinculação a uma porta

## Módulo 8

### **Aula 3 – Serviços de apoio, build, processos e vínculos de porta**

- Outros serviços podem ser exportados através de vínculo de porta, como Redis, XMPP, etc.
- Vincular um serviço a uma porta significa que este serviço se transformou num serviço de apoio para outros serviços



## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

#### Concorrência

- Escale através do modelo de processos
- Processo são considerados cidadão de primeira classe, utilizando fortes sugestões do modelo Unix para execução de serviços daemon. Usando esse modelo, o desenvolvedor pode criar um processo diferente para cada carga de trabalho

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

- A natureza stateless e de não compartilhar nada das aplicações 12 fatores permite que escalar aplicações horizontalmente se torne um processo simples e confiável

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

#### Descartabilidade

- Maximize robustez com início rápido e parada graciosa (graceful)
- Aplicações devem ser descartáveis, isto é, podem ser paradas e iniciadas a qualquer momento
- Ao receber um sinal de término, os processos devem morrer de forma graciosa, isto é, permitindo que requisições em andamento possam terminar antes de pararem

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

- Deveriam ser robustos contra morte súbita, isto é, falhas de hardware ou desconexões de clientes

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

#### Ambientes iguais (paridade)

- Mantenha produção, desenvolvimento e staging o mais similares possível
- A aplicação deve ser desenhada para continuous deployment a fim de minimizar o gap entre desenvolvimento e produção

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

- Diferentes serviços de apoio não deveriam ser usados entre ambientes. Um teste que passa em desenvolvimento e staging, por exemplo, pode falhar em produção por este motivo

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

#### Logs

- Trate logs como fluxos (streams) de eventos
- Sua aplicação nunca deverá se preocupar com a rota ou com o armazenamento de seus fluxos de saída. Ao invés disso, deverá sempre escrever na saída padrão

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

- Outros processos devem capturar estes fluxos de saída e redirecioná-los para os locais corretos e especificados pelos administradores ou desenvolvedores da aplicação



## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

#### Processos de administração

- Execute processos de administração/gerenciamento como processos pontuais
- Processos de administração deveriam ser executados contra o mesmo ambiente onde os processos da aplicação estão rodando

## Módulo 8

### **Aula 4 – Concorrência, descartabilidade, ambientes iguais, logs e processos de administração**

- Processos administrativos podem incluir migrações de bases de dados, execução de comandos em um console, backfills, etc.