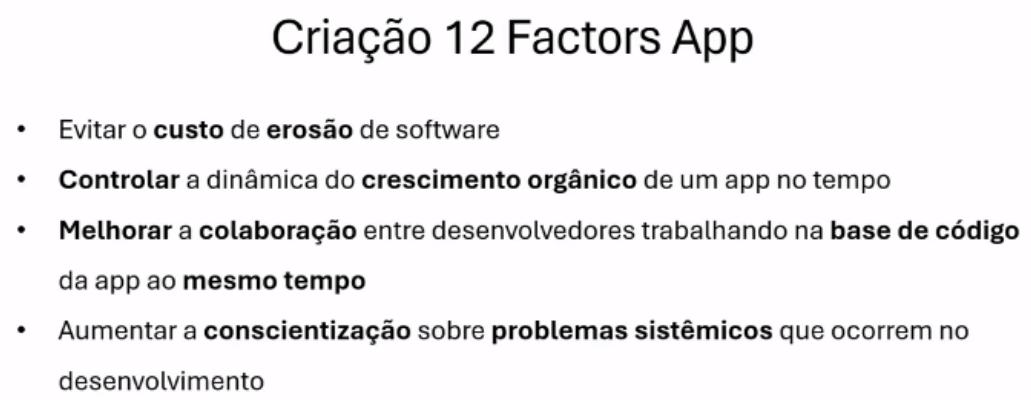
DevOps Modulo 3 - IT Talent

Escrito por: Gabriel Oliveira dos Santos

GIthub: https://github.com/Hypothasis

O terceiro módulo de DevOps do IT Talent vem como tema principal 12 Factos APP (Aplicativo de 12 Fatores), são conjunto de regras e softwares que auxiliam na produção de soluções escalavel, robustas e mais fácil de ser executado.

Objetivos Principais do 12 Factors App



Como a imagem está descrita, a nossa aplicação deve ser robusta, evitando custos adicionais, melhorar a integração entre todos os integrantes do desenvolvimento dessa aplicação (aumentando a colaboração de todos).

Essa metodologia ajuda a visualizar os desafios e problemas de uma forma moderna e otimizada, assim temos todo o planejamento inteiro do software antes da produção, e na produção construindo uma aplicação sólida e robusta para ser um desenvolvimento duradouro.

Os 12 Fatores

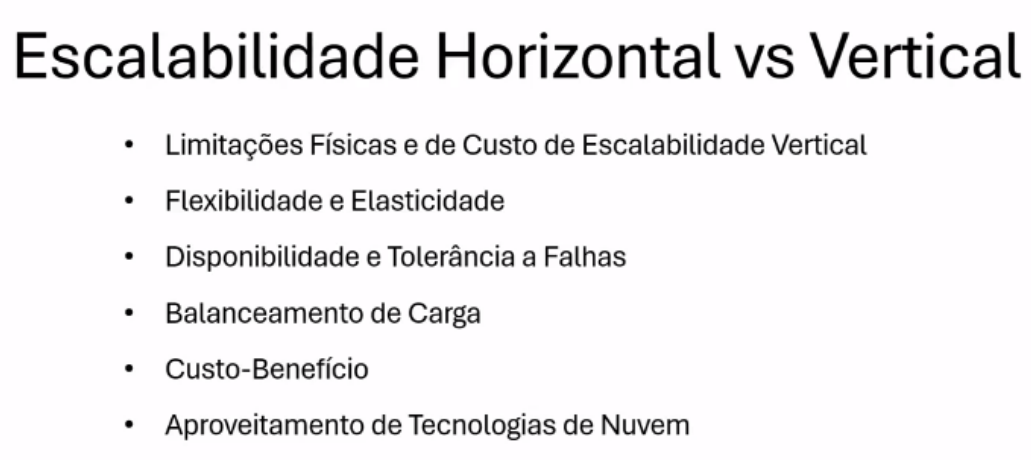
1. **Base de Código**: Todo código-fonte do aplicativo deve ser armazenado em um sistema de controle de versão, como Git.
2. **Dependências**: Todas as dependências externas, como bibliotecas de código, devem ser explicitamente declaradas e isoladas do código do aplicativo.
3. **Configurações**: As configurações do aplicativo (por exemplo, credenciais de banco de dados, chaves de API) devem ser armazenadas em variáveis de ambiente e não no código-fonte.
4. **Serviços de Apoio:** O aplicativo deve tratar serviços de suporte (como bancos de dados, filas de mensagens) como recursos externos que podem ser conectados ou desconectados facilmente.
5. **Build, release, run**: O processo de construção (compilação e empacotamento do código), lançamento (combinação de código com configuração) e execução (execução do aplicativo) deve ser separado e automatizado.
6. **Processos**: O aplicativo deve ser executado como um ou mais processos que são completamente independentes, sem estado compartilhado. Isso permite fácil escalabilidade horizontal.
7. **Vínculo de Porta:** O aplicativo expõe serviços via portas, e é responsabilidade do ambiente de execução fornecer a URL de acesso.
8. **Concorrência**: Os processos do aplicativo devem ser capazes de escalar horizontalmente e aproveitar ao máximo o poder de processamento disponível.
9. **Descartabilidade**: O aplicativo deve ser projetado para ser robusto, capaz de iniciar e encerrar rapidamente, facilitando a implantação e a

recuperação de falhas.

10. **Dev/prod semelhantes:** Os aplicativos devem ser construídos como conjuntos de serviços pequenos e independentes, cada um com sua própria responsabilidade clara e interface bem definida.
11. **Logs**: O aplicativo deve produzir logs como fluxo de eventos e deve tratar os logs como fluxos de eventos, sem armazenamento local. Os logs devem ser tratados como dados de fluxo, com uma função de processamento centralizada.
12. **Processos Administrativos**: O aplicativo deve ser projetado para permitir escalabilidade rápida e fácil, adicionando ou removendo instâncias de serviços conforme necessário.

Escalabilidade Horizontal e Vertical

Escalabilidade vertical e escalabilidade horizontal são dois conceitos importantes em arquitetura de sistemas, especialmente quando se trata de grande carga e demanda de tráfego de dados nessa aplicação.



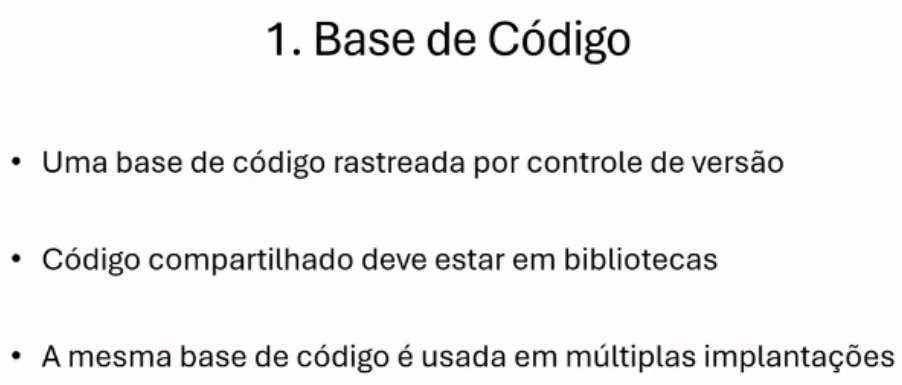
A Escalabilidade Vertical (Scale-Up) pode ser uma solução muito viável para vários Devs, porém com o tempo (se pensarmos como um servidor rodando uma aplicação inteira na nuvem) podemos sobrecarregar esse servidor e assim tendo que invetir mais dinheiro, que pode crescer exponencialmente, custando muito para empresas.Estão abaixo os principais conceitos da escalabilidade vertical:

- Aumentar a capacidade do poder de processamento, a capacidade de armazenamento ou outros recursos de hardware de um único servidor ou máquina. Todo poder de procaessamento é feito em um servidor.
- Um único servidor ou máquina pode manipular toda a carga de trabalho e trafego de dados. Isso faz com que se um a necessidade de uma melhor rentabilidade do processamento de dados, é necessário um servidor mais pontente, mais RAM e CPU.
- Escalabilidade vertical é mais fácil de implementar, porém requer mais verba para mais processamento do servidor.

Escalabilidade Horizontal (Scale-Out):

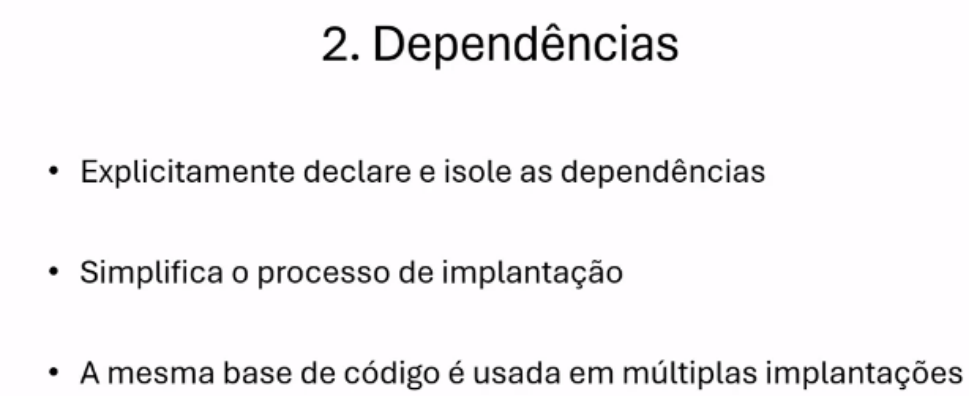
- A escalabilidade horizontal tirar a autonomia de apenas um servidor, mas agora são tarefas divididas para cada servidor, assim tirando o monopólio de um servidor e criando um oligopólio para vários servidores terem seus objetivos.
- Assim facilitando a demanda, aumentando com o tempo os recursos de forma mais flexível e economica.
- Sistema distribuidos são excessiais para escalabilidade horizontal, em que cada servidor tem um papel para a execução da aplicação.
- Apesar de ser mais díficil de implementar, essas diversificação dos processamento dos servidores perante o a nossa aplicação, viabiiza a economia e flexibilidade da nossa aplicação.

1 - Base de código



Todo código fonte (no GIt), deve ter a sua versão correspondente, assim tendo controle do projeto e das atualizações do mesmo. Assim cada modificação fica documentada, assim o usuário acompanhando cada evolução do projeto.

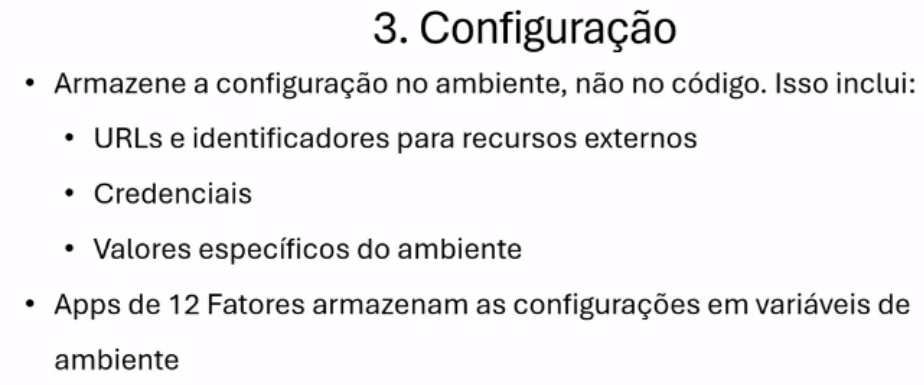
2 - Dependências



Toda aplicação tem suas dependências, podendo ser uma ou mais bibliotecas. É sem dúvida, que enfatizar cada demonstração dessas dependencias explicitamente é crucial para uma aplicação robusta e flexivel de problemas. Assim facilitando o gerenciamento dessas dependencias, a atualização dessas dependencias e elas em si não deverm estar presentes no codigo fonte.

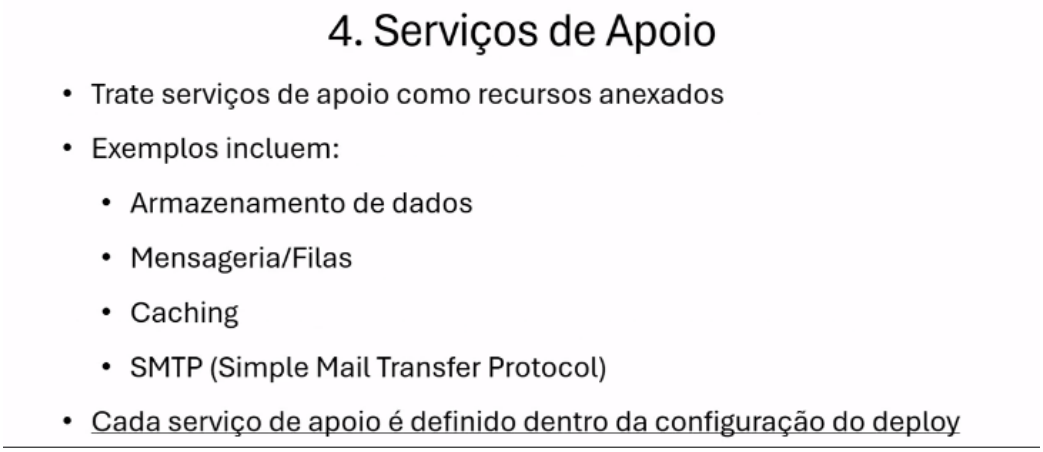
Práticas comuns para alcançar esses gerenciamentos são o Maven (para Java), npm (para JavaScript), pip (para Python), que gerenciam automaticamente as dependências do seu aplicativo.

3 - Configuração



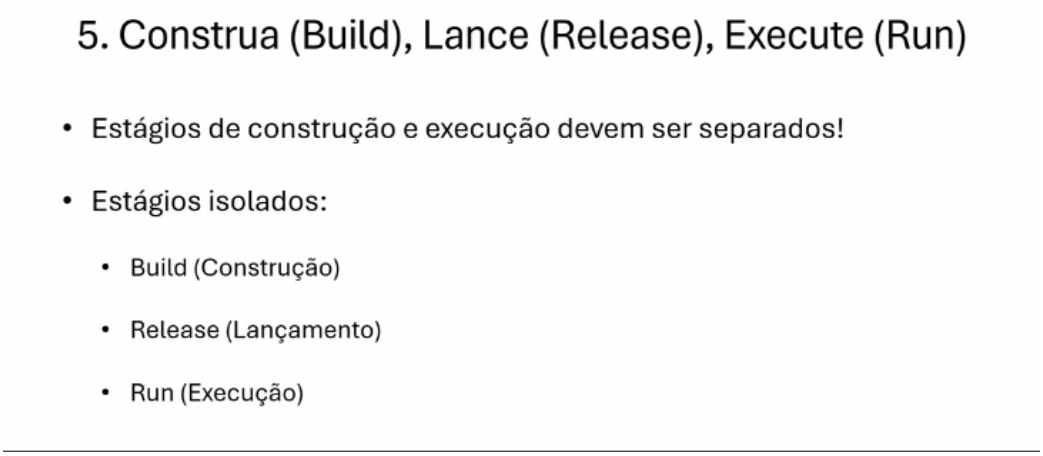
Credenciais, dados de configurações e de acesso ao aplicativo não tevem estar dentro da aplicação, mas sim nas vaiaveis de ambiente, deixando nossa aplicação mais robusta.

4 - Serviços de Apoio



Apoio e contato com o suporte devem ser implementados no sistema da nossa aplicação, em caso de erro inesperado do usuario perante a nossa aplicação, o mesmo usuario tem a possibilidade de entrar em contato com o suporte.

5- Build, release, run



- A construção (building) só pode ser feita no estágio de Build usando a Base de Código
- O lançamento (releasing) combina o build do estágio de construção com a configuração para o deploy
- A execução roda um conjunto de processos do app em uma release específica

---

Contrução é o devenvolvimento nato da nossa aplicação, em que no final desse processo obtemos um arquivo executavel binário, chamado de build. O Release tem o build e tranforma e combina a configuração para que seja possível obter a implantação, e o Run executa o código em um ambiente alfa.

6- Processos

6. Processos

- Execute o app como ou mais processos sem estado (stateless)
- Dados persistentes devem ser armazenados em um serviço de apoio com estado (stateful)

Um processo é um script padronizado, ou conteinerizado. Se o processo necessita de um ou mais micro serviços, é melhor ser conteinerizado o nosso processo. Assim o nosso aplicativo roda com a escalibilidade horizontal, em que cada microserviço ou processamento necessário é rodado em um sistema distribuido, com servidores específicos para essa aplicação, como foi descrito escalibilidade horizontal acima.

7- Vínculo de Porta

7. Vínculo de Porta

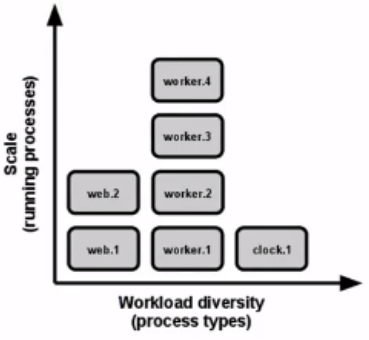
- Exportação de serviços através de vinculação de porta
- Apps de 12 Fatores são completamente autocontidos e provêm serviços vinculando-os a portas. Por exemplo, um app web exportaria HTTP como um serviço vinculando-o a uma porta e escurando requisições àquela porta

URL e portas de acessos são essenciais para a comunição, e nossa aplicação deve ser autocontida (tudo que a nossa aplicação nessecita deve estar com ela). Nossa solução nao deve ter a necessidade de uma porta com servidores Web externos para a comunicação e ser acessivel na Web. A nossa aplicaçãpo deve iniciar uma aplicação de Web interno e se vincula a um a porta disponivel através dessa porta de comunicação. Na nuvem, nossa aplicação se vincula com uma porta (para “escutar”), permitindo que a nossa aplicação tem acesso pela internet, assim nossa aplicação tem uma aplicação mais fácil e escalação também, promovendo uma independencia da nossa aplicação com a nuvem, e promovendo diferentes comunicações com a nuvem que no espedam.

8- Concorrência

8. Concorrência

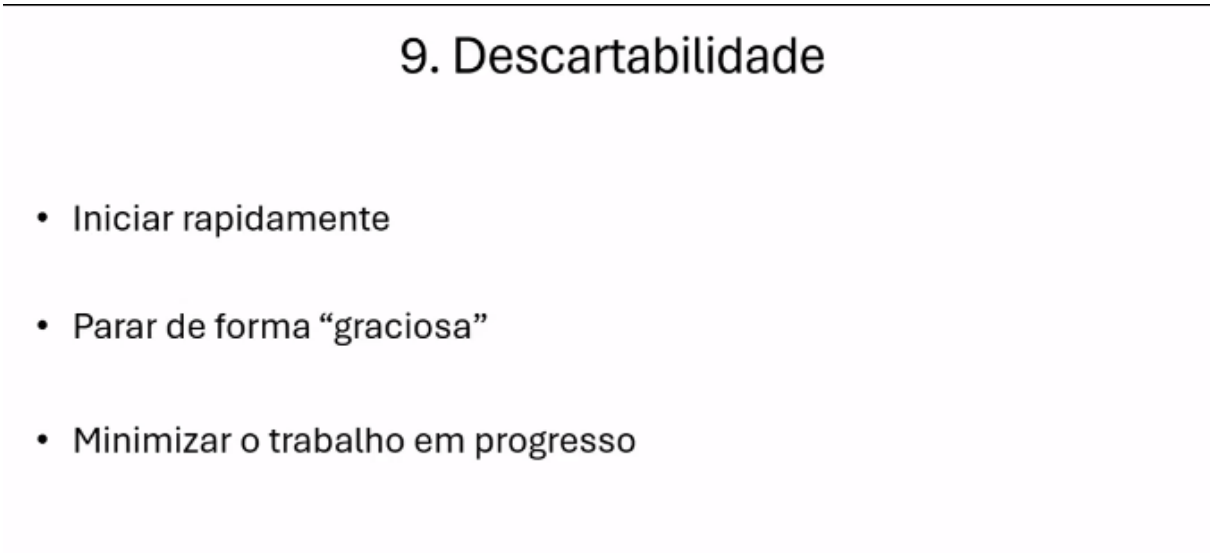
- Escala através do modelo de processo.



Uma aplicação que lida com várias tarefas de forma eficiente, a concorrência em si, multi processos que executam a nossa aplicação simultaneamente. É importante a concorrência, pois é uma forma de atender vários usuários de

forma eficiente, vários processos em que são executados na nossa aplicação, usando os recursos do sistema de forma completa.

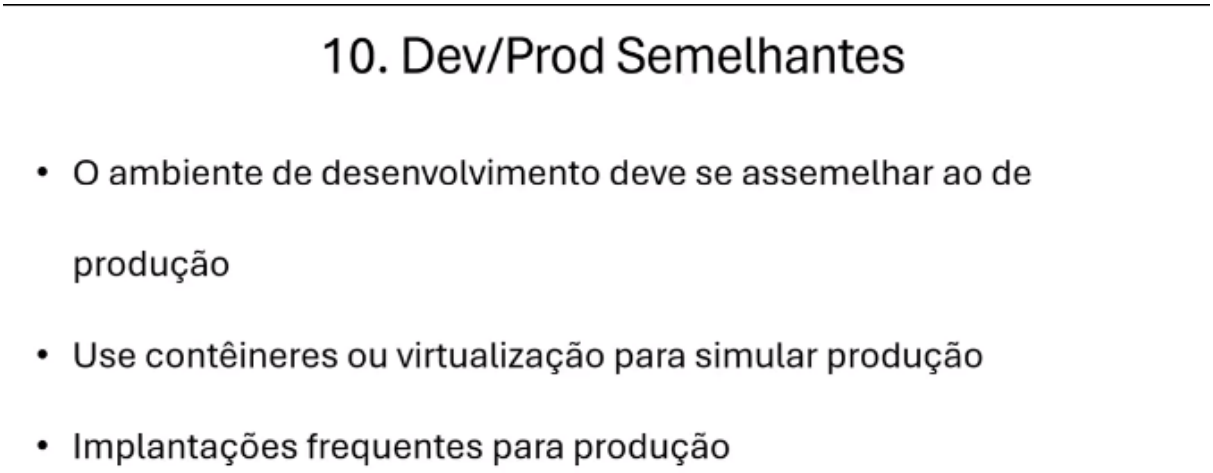
9- Descartabilide



Nossas aplicações devem iniciar e encerrar de forma continua, sem interrupções e de forma ágil. Deve ser uma aplicação que deve ser robusto para uma atualização de dependencia, ajustar as instancias de forma que satisfaça a demanda do usuário.

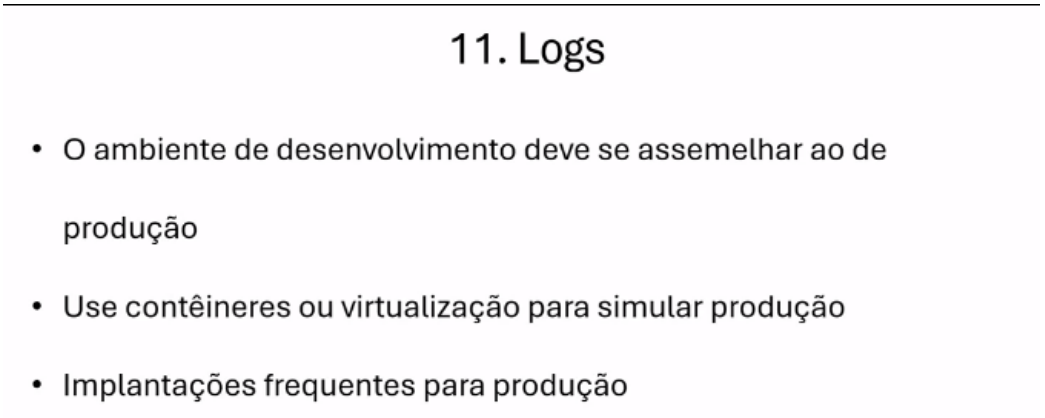
“A parada graciosa” é salvar dados e não obter erros quando o usuário interrompe de forma inesperada a nossa aplicação, assim obtendo confiabilidade da nossa aplicação para com o usuário. Capacidade da nossa aplicação ser menos trabalhosa na no progresso do desenvolvimento.

10- Dev/Prod Semelhantes



Ambiente de Desenvolvimento, no Ambiente Software, deve se assemelhar o mais proximo ao de produção, reduzindo a chance de ter problemas no desenvolvimento do código. Ferramentas como Docker, possibilitam que o desenvolvedor possa usar essa ferramenta como um ambiente de teste simulando um usuário comum que não tem nada instalado (na sua máquina), que se assemelhe com ambiente de desenvolvimento. Quanto mais frequentemente você implantar na produção, mais você irar se familiarizar com o processo, e mais rápido e automatizado será a execução da produção.

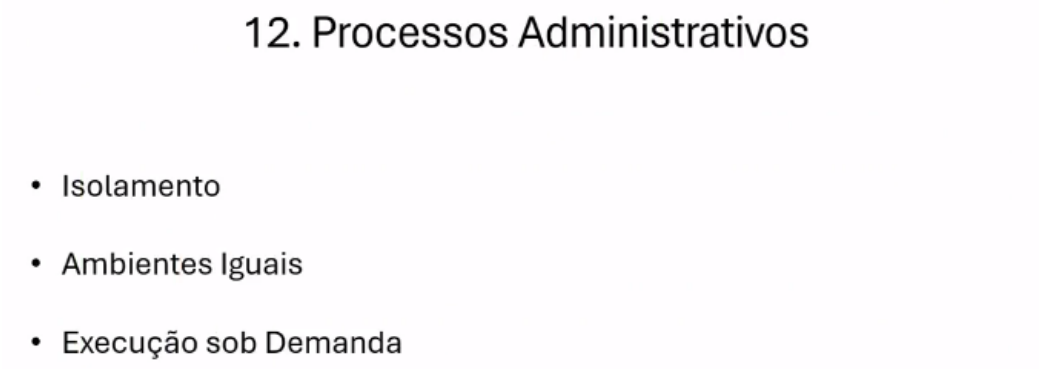
11- Logs



Registro de eventos, os Logs, devem ser tratados com o fluxo de eventos para o usuário. Uma aplicação robusta deve informar o que está acontecendo com o nosso código, enviando status para o Log, em que é possível detectar erros de Debug, que são dificéis de ver na compilação, sejam vistas de forma mais clara

e rápida, sem ter que lidar com o código direto, mas com eventos da aplicação.O monitoramento desses Logs podem ser designados pela nuvem, ou ter um sistema de monitoramento que possibilitam a visualização desses eventos de forma eficaz.

12- Processos Administrativos



Processos Administrativos ou de manutenção, migrações de banco de dados, limpeza de dados são processos que caracterizam a manutenção da nossa aplicação. Verificações de seguranças, limpeza de dados e atualizaçoes de bibliotecas são exempĺos de manutenção no ramo da nossa aplicação. Processos administrativos devem ser executados diferentes do ambiente da nossa aplicação, garantindo que não tenham interferência na nossa aplicação principal. Essas verificações de processos admins, devem ser feitos quando forem necessarios. Com tudo, esses processos de manutenção, devem ser isolados e executados em curta duração em um ambiente fiel ao de produção, garantindo a segurança, estabilidade e presivibilidade da nossa aplicação .