
Semana 21 - Aula 1

Tópico Principal da Aula: Frameworks back-end e integração com serviços externos

Subtítulo/Tema Específico: Gestão de configurações e segredos

Código da aula: [SIS]ANO2C2B3S21A1

Objetivos da Aula:

- Implementar práticas de gestão de configurações e segredos, utilizando ferramentas apropriadas.
- Desenvolver o autocontrole ao focar a segurança e eficiência da gestão de configurações.

Recursos Adicionais:

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Tema implícito nos Slides 9 e 10 - Separação de Configurações por Ambiente

- **Definição:** **A gestão de configurações é o processo de gerenciar e controlar as configurações de um sistema de software. Uma prática fundamental é separar as configurações por ambiente (desenvolvimento, homologação/testes e produção). Isso garante que cada ambiente utilize os parâmetros corretos, como conexões de banco de dados, chaves de API e níveis de log, evitando que dados sensíveis de produção sejam expostos em ambientes menos seguros.**
- **Aprofundamento/Complemento:** A separação de configurações previne erros críticos, como um desenvolvedor se conectar acidentalmente ao banco de dados de produção e causar perda de dados. Além disso, permite que cada ambiente seja otimizado para seu propósito. O ambiente de desenvolvimento pode ter logs mais detalhados e ferramentas de depuração ativas, enquanto o de produção deve focar em performance e segurança máxima.
- **Exemplo Prático:** **Uma aplicação web se conecta a um banco de dados. No ambiente de desenvolvimento, o arquivo de configuração aponta para um banco de dados local (localhost:5432). No ambiente de homologação, aponta para um banco de dados de testes compartilhado pela equipe (db.test.empresa.com). Já no ambiente de produção, a configuração aponta para o banco de dados real, acessado pelos clientes (db.prod.empresa.com). Essa separação evita o uso indevido de segredos de produção em outros ambientes.**

Referência do Slide: Tema implícito nos Slides 7, 8 e 11 - Gestão de Segredos

- **Definição:** **A gestão de segredos é a prática de proteger informações sensíveis (segredos), como senhas, chaves de API, tokens e certificados. Em vez de armazenar esses dados diretamente no código-fonte ou em arquivos compartilhados de forma insegura, como por e-mail , utilizam-se ferramentas especializadas que centralizam, protegem e controlam o acesso a essas informações.**
- **Aprofundamento/Complemento:** Ferramentas de gestão de segredos, como HashiCorp Vault , AWS Secrets Manager e Azure Key Vault , oferecem recursos robustos de segurança, como criptografia, controle de acesso granular, políticas de rotação de senhas e trilhas de auditoria detalhadas para rastrear quem acessou qual segredo e quando. Elas complementam, mas não substituem completamente, a necessidade de arquivos de configuração.
- **Exemplo Prático:** **Uma aplicação precisa de uma chave de API para se comunicar com um serviço de pagamentos. Em vez de escrever a chave no código (`apiKey = "chave_secreta_123"`), a aplicação é configurada para solicitar essa chave ao HashiCorp Vault no momento da inicialização. O Vault, por sua vez, verifica se a aplicação tem permissão para acessar aquele segredo específico. Isso centraliza e protege a chave, permitindo que ela seja alterada ou revogada sem a necessidade de modificar e reimplantar o código da aplicação.**

Semana 21 - Aula 2

Tópico Principal da Aula: Frameworks back-end e integração com serviços externos

Subtítulo/Tema Específico: Monitoramento e logs

Código da aula: [SIS]ANO2C2B3S21A2

Objetivos da Aula:

- Implementar monitoramento e logging para rastrear e resolver problemas em aplicações back-end.
- Compreender como ferramentas de monitoramento e logging podem identificar e prevenir problemas.
- Aprender a configurar logging distribuído e centralizado para uma análise eficiente.

Recursos Adicionais:

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou interne

Exposição do Conteúdo:

Referência do Slide: Tema implícito no Slide 10 - Ferramentas e Métricas de Monitoramento

- **Definição:** **O monitoramento de aplicações é o processo de coletar e analisar dados (métricas) para entender o desempenho e a saúde de um sistema em tempo real. As métricas-chave a serem observadas inicialmente incluem latência (tempo de resposta), uso de CPU, consumo de memória e a taxa de erros das requisições.**
- **Aprofundamento/Complemento:** Ferramentas como **Prometheus** são usadas para coletar e armazenar essas métricas. O **Grafana** é então utilizado para criar dashboards (painéis visuais) que exibem as métricas de forma intuitiva, com gráficos e alertas. Essa visualização permite que as equipes identifiquem rapidamente anomalias, como um pico no uso de CPU ou um aumento súbito na latência, que podem indicar um problema iminente.
- **Exemplo Prático:** **Uma equipe de desenvolvimento configura o Prometheus para coletar métricas de uma API. Eles criam um dashboard no Grafana que mostra gráficos do tempo de resposta de cada endpoint da API. Ao observar o dashboard, eles percebem que a latência do endpoint /api/pedidos aumenta drasticamente todos os dias às 18h. Com essa informação, eles podem investigar e otimizar a consulta ao banco de dados que é executada naquele horário, resolvendo o problema de lentidão antes que gere reclamações dos usuários.**

Referência do Slide: Tema implícito no Slide 10 - Logging Distribuído e Centralizado

- **Definição:** **Logging é o ato de registrar eventos, erros e informações operacionais que ocorrem durante a execução de uma aplicação. Em arquiteturas de microsserviços, onde múltiplas aplicações rodam de forma independente, o logging distribuído se torna um desafio. Um sistema de logging centralizado resolve isso coletando os logs de todos os serviços em um único local para facilitar a análise.**
- **Aprofundamento/Complemento:** A pilha **ELK (Elasticsearch, Logstash, Kibana)** é uma solução popular para logging centralizado.
  - **Logstash:** Coleta e processa os logs de diferentes fontes.
  - **Elasticsearch:** Armazena e indexa os logs de forma otimizada para busca.
  - **Kibana:** Oferece uma interface web para pesquisar, visualizar e analisar os logs armazenados no Elasticsearch.
- **Exemplo Prático: Um e-commerce possui microsserviços para usuarios, produtos e pagamentos. Quando um cliente realiza uma compra, uma requisição passa por todos esses serviços. Se ocorrer um erro no serviço de pagamentos, o desenvolvedor não precisa acessar o servidor de cada serviço individualmente para encontrar a causa. Com o ELK Stack, ele pode acessar o Kibana, filtrar as requisições pelo ID da transação e ver a sequência completa de logs de todos os três serviços em um único lugar, identificando rapidamente onde e por que a falha ocorreu.**

Semana 21 - Aula 3

Tópico Principal da Aula: Frameworks back-end e integração com serviços externos

Subtítulo/Tema Específico: Orquestração de microsserviços

Código da aula: [SIS]ANO2C2B3S21A3

Objetivos da Aula:

- Implementar orquestração de microsserviços para coordenação de tarefas.
- Compreender como ferramentas como Kubernetes otimizam a implantação e gerenciamento de microsserviços.
- Descobrir como o Kubernetes pode ser usado para lidar com problemas de desempenho e escalabilidade.

Recursos Adicionais:

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Slide 07, 08 e 09 - Ferramentas de Orquestração

- **Definição:** **A orquestração de microsserviços é o processo de gerenciar e coordenar múltiplos microsserviços em ambientes distribuídos, garantindo que eles funcionem juntos como uma unidade coesa**. Isso é crucial para automatizar tarefas e garantir a estabilidade do sistema.
- **Aprofundamento/Complemento: As ferramentas de orquestração resolvem três grandes desafios:**
  - **Escalabilidade: Aumentam ou diminuem automaticamente o número de instâncias de um serviço com base na demanda.**
  - **Gerenciamento de Falhas: Detectam quando um serviço falha e o reiniciam automaticamente, garantindo alta disponibilidade.**
  - **Monitoramento e Logging: Integram-se a ferramentas para coletar métricas e logs, facilitando o rastreamento de problemas.**
  - **Ferramentas populares incluem Kubernetes, Docker Swarm, Apache Mesos e AWS ECS.**
- **Exemplo Prático:** **Imagine um aplicativo de e-commerce durante a Black Friday. O serviço de carrinho está recebendo milhares de acessos por segundo. Uma ferramenta de orquestração detecta esse aumento de tráfego e automaticamente cria novas cópias (instâncias) do serviço para distribuir a carga. Se uma dessas instâncias falhar, a ferramenta a remove e cria uma nova para substituí-la, tudo sem intervenção manual e sem que o usuário perceba.**

Referência do Slide: Slide 10, 11 e 12 - Aprofundamento em Kubernetes (K8s)

- **Definição:** Kubernetes (também conhecido como K8s) é uma plataforma de código aberto, originalmente desenvolvida pelo Google, que se tornou o padrão de

mercado para orquestração de contêineres. Ele automatiza a implantação, o escalonamento e a gestão de aplicações em contêineres.

- **Aprofundamento/Complemento:** Os principais componentes do Kubernetes são:
    - **Pod:** A menor unidade de implantação do K8s, que encapsula um ou mais contêineres.
    - **Node:** Uma máquina (física ou virtual) que faz parte do cluster e executa os pods.
    - **Cluster:** Um conjunto de nodes gerenciados pelo Kubernetes.
    - **Service:** Uma abstração que expõe um conjunto de pods como um serviço de rede único, permitindo a comunicação.
    - O Kubernetes oferece benefícios como gerenciamento automatizado, alta disponibilidade (redistribui pods em caso de falha) e facilidade de integração com outras ferramentas.
- **Exemplo Prático:** Uma empresa de entregas usa microsserviços para gerenciar pedidos, rastreamento e pagamentos. Para garantir que o sistema não fique lento durante picos promocionais, eles usam Kubernetes. Cada microsserviço é implantado em um conjunto de pods. Eles configuram o
- **Horizontal Pod Autoscaler (HPA)**, um recurso do K8s, que monitora o uso de CPU dos pods. Quando a CPU ultrapassa 70%, o HPA automaticamente cria mais pods para aquele serviço, e quando o tráfego diminui, ele remove os pods extras para economizar recursos.