UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE – CCENS/ UFES
Departamento de Computação

Trabalho Prático – Tópicos Especiais em Redes – 2025/2

Implementação de uma API em JavaScript na AWS utilizando Arquitetura de Microsserviços e Pipeline CI/CD com GitHub

1. Introdução

Com o crescimento das aplicações distribuídas e da computação em nuvem, a adoção de arquiteturas baseadas em microsserviços tornou-se uma prática comum no desenvolvimento de software moderno. Paralelamente, a automação de processos de integração e entrega contínua (CI/CD) permite maior confiabilidade, escalabilidade e agilidade no ciclo de desenvolvimento.

Este trabalho propõe o desenvolvimento de uma API utilizando JavaScript, hospedada na Amazon Web Services (AWS), estruturada em microsserviços independentes e integrada a um pipeline CI/CD utilizando o GitHub.

2. Objetivos

2.1 Objetivo Geral

Desenvolver e implantar uma API em JavaScript na AWS utilizando arquitetura de microsserviços e um pipeline de CI/CD automatizado via GitHub.

2.2 Objetivos Específicos

- Implementar microsserviços independentes utilizando Node.js
- Utilizar de containers ECS da SWS
- Utilizar o padrão blue/green para a implantação de novas versões
- Utilizar um serviço RDS (Mysql) para persistência
- Configurar versionamento de código com GitHub
- Implementar pipeline CI/CD automatizado
- Garantir boas práticas de segurança, testes e monitoramento

3. Justificativa

A escolha da arquitetura de microsserviços permite maior escalabilidade, facilidade de manutenção e independência entre componentes. A AWS oferece uma infraestrutura robusta e amplamente

utilizada no mercado, enquanto o GitHub fornece ferramentas eficientes para controle de versão e automação de pipelines CI/CD, tornando este projeto alinhado às práticas profissionais atuais.

4. Tecnologias Utilizadas

4.1 Linguagem e Framework

- JavaScript
- Node.js

4.2 Infraestrutura em Nuvem (AWS)

- AWS ECS
- AWS Auto Scaling
- Amazon RDS
- Amazon S3 (artefatos e logs)
- AWS IAM (controle de acesso)

4.3 CI/CD e Versionamento

- GitHub
- GitHub Actions
- AWS CLI / AWS SDK (opcional)

4.4 Ferramentas Complementares

- Jest ou Mocha (testes automatizados)
- ESLint (padronização de código)

5. Arquitetura do Sistema

A aplicação será composta por múltiplos microsserviços independentes, cada um responsável por uma funcionalidade específica da API. Os serviços se comunicarão por meio de endpoints REST.

5.1 Microsserviços

- Serviço de autenticação
- Serviço de gerenciamento de usuários
- Serviço de recursos principais da API

Cada microsserviço terá:

- Repositório ou diretório próprio
- Deploy independente
- Testes unitários específicos

6. Processo de CI/CD

6.1 Integração Contínua (CI)

- Execução automática de testes a cada *push* ou *pull request*
- Análise estática de código
- Validação de build

6.2 Entrega Contínua (CD)

- Deploy automático para ambiente de testes
- Deploy manual ou automático para produção
- Versionamento e rollback de versões

O pipeline será implementado utilizando **GitHub Actions**, integrando-se diretamente com a AWS.

7. Segurança

- Autenticação e autorização via tokens (JWT)
- Controle de permissões com AWS IAM
- Uso de variáveis de ambiente seguras
- Restrição de acesso aos endpoints

8. Metodologia

O desenvolvimento seguirá uma abordagem incremental, baseada em:

- Planejamento dos microsserviços
- Implementação individual dos serviços
- Criação do pipeline CI/CD
- Testes e validação contínuos
- Documentação técnica da API (Swagger/OpenAPI)

9. Resultados Esperados

- API funcional e escalável
- Pipeline CI/CD totalmente automatizado
- Código versionado e documentado

10. Considerações Finais

Cada grupo deverá escolher um caso prático a ser abordado (cadastro de produtos, cadastro de clientes, etc, **sem coincidir com os demais grupos**). A api deverá ter endpoints para as funcionalidades básicas de um CRUD.