INSCRIÇÕES ABERTAS

UECE · UNIVERSIDADE ESTADUAL DO CEARÁ

ESPECIALIZAÇÃO EM

# ENGENHARIA DE SOFTWARE COM DEVOPS

CAMPUS ITAPERI

Prof. Dr. Joaquim Celestino Júnior — Coordenador da Especialização em Engenharia de Software com DevOps
COORDENAÇÃO DA ESPECIALIZAÇÃO

| MODALIDADE | AULAS | DURAÇÃO | INÍCIO |
|---|---|---|---|
| Presencial, Campus Itaperi | Terças e quintas, 19h às 22h | 18 meses | 27/10/2026 |

## APRESENTAÇÃO DO FOLDER

Esta é a versão completa do folder de divulgação da turma T4 da Especialização em Engenharia de Software com DevOps, ofertada pela Universidade Estadual do Ceará (UECE) no Campus Itaperi. O documento reúne, disciplina por disciplina, o corpo docente com as fotografias e os nomes completos, as ementas integrais, os objetivos gerais e específicos, o conteúdo programático completo com módulos e tópicos, as metodologias de ensino, os workshops e projetos práticos previstos, a distribuição da carga horária, os critérios de avaliação, as ferramentas e tecnologias utilizadas, as bibliografias e as competências desenvolvidas.

## FORMAÇÃO PRÁTICA EM

- Engenharia de Software Ágil com DevOps
- Desenvolvimento Web com Frameworks
- Infraestrutura em Nuvem e como Código
- Testes Ágeis e Segurança de Sistemas
- Big Data / Machine Learning
- Governança em TI e Metodologia do Trabalho Científico

## MATRIZ CURRICULAR — 11 DISCIPLINAS

Engenharia de Software Ágil com DevOps · Big Data · Desenvolvimento Web com Frameworks · Governança em TI · Sistemas Distribuídos · Arquitetura Orientada a Serviços e APIs Modernas · Infraestrutura em Nuvem e como Código · Testes Ágeis · Segurança em Desenvolvimento de Sistemas · Integração Contínua e Gerenciamento de Configuração · Metodologia do Trabalho Científico

INSCREVA-SE

**(85) 9 8781-7200**
eesdevops@gmail.com
https://pdv.iepro.org.br/cursos/devops_T4/

**Sua próxima conquista começa aqui!**

## COORDENAÇÃO DA ESPECIALIZAÇÃO

### Prof. Dr. Joaquim Celestino Júnior

**Coordenador da Especialização em Engenharia de Software com DevOps**

É Professor Emérito da Universidade Estadual do Ceará (UECE), possui graduação em Engenharia Eletrônica pela Pontifícia Universidade Católica do Rio de Janeiro (1985), mestrado em Ciência da Computação pela Universidade Federal da Paraíba/Campina Grande (1989), doutorado em informática pela Université de Paris VI (Pierre et Marie Curie) (1994) e pós-doutorado pela Columbia University in New York City (2010). Possui experiência na área de redes de comunicação, segurança e saúde digital. Professor de várias disciplinas: redes de computadores, segurança, sistemas distribuídos, inteligência artificial aplicada a redes de computadores e inteligência artificial aplicada à saúde digital. Coordenou cursos de graduação e pós-graduação.

**Currículo Lattes: http://lattes.cnpq.br/7227955029154651**

## SECRETARIA DO CURSO

### Rochely Campos

**Secretaria do curso — Especialização em Engenharia de Software com DevOps**

Responsável pelo atendimento aos candidatos e alunos: informações sobre o processo seletivo, matrícula, documentação, calendário das aulas e demais demandas administrativas do curso.
Telefone / WhatsApp: (85) 9 8781-7200 · E-mail: eesdevops@gmail.com

## CORPO DOCENTE — NOMES COMPLETOS E FOTOGRAFIAS

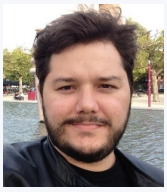
Todos os docentes abaixo integram o corpo docente da turma T4, com as respectivas fotografias, titulações, disciplinas sob sua responsabilidade e trajetória profissional, conforme o projeto do curso.

### Prof. MSc Joaquim Pedro Carvalho

**Engenharia de Software Ágil com DevOps · Integração Contínua e Gerenciamento de Configuração**

Formado em Ciência da Computação pela UFC e mestre em Engenharia de Software pela UFPE. Analista de sistemas no SERPRO desde 2009, liderando o time de arquitetura de nuvem desde 2020. Acumula experiência em instituições como o Instituto Atlântico e o C.E.S.A.R.

**Currículo Lattes: http://lattes.cnpq.br/7381054192994709**



### Prof. Dr. Alex Ferreira Ramires Trajano

Sistemas Distribuídos

Pesquisador e Líder de Plataforma no Instituto Atlântico, onde lidera projetos de P&D em conectividade, sistemas distribuídos, IoT, computação em nuvem e inteligência artificial. Graduado em Sistemas de Informação pela UECE (2016) e doutor em Ciências da Computação pela UFC, na área de Edge Computing.

Currículo Lattes: http://lattes.cnpq.br/9347392389583733

## CORPO DOCENTE · CONTINUAÇÃO

### Prof. Dr. Ricardo Holanda Nobre

**Big Data**

Doutor em Engenharia de Teleinformática e professor nas áreas de Engenharia de Software e Inteligência Artificial. Possui sólida trajetória acadêmica e profissional nas áreas de Engenharia de Software, Arquitetura de Sistemas, Sistemas Distribuídos, Banco de Dados e Tecnologia da Informação. Com ampla experiência na docência de graduação e pós-graduação, bem como em projetos de pesquisa, desenvolvimento tecnológico e inovação, focados em qualidade de software, automação, arquiteturas modernas e engenharia de sistemas complexos. Atua no setor de tecnologia e software corporativo, onde concilia a vivência acadêmica com a prática profissional em ambientes de alta complexidade.

**Currículo Lattes: http://lattes.cnpq.br/1723134991744757**

### Prof. Esp. Wellington Costa Pereira

**Desenvolvimento Web com Frameworks**

Especialista em Engenharia de Software com ênfase em Padrões de Software. Possui mais de dez anos de experiência em desenvolvimento e arquitetura de software, com passagens pelos setores público, privado e acadêmico. Sua atuação abrange as áreas de Engenharia de Software, Arquitetura de Sistemas e desenvolvimento de produtos, experiência em soluções Web e Mobile, liderança técnica, definição arquitetural e adoção de boas práticas de engenharia de software. Experiência como docente de pós-graduação, ministrando disciplinas relacionadas ao desenvolvimento de software e ao uso de frameworks modernos.

**Currículo Lattes: http://lattes.cnpq.br/7899547573886914**

### Prof. Dr. Eliseu Castelo Branco Júnior

**Governança em TI**

Formado em Ciências da Computação, professor de Estratégia e Gestão, executivo com formação acadêmica sólida (Doutorado em Ciências da Computação, Mestrado em Informática Aplicada) e mais de 30 anos de experiência combinando gestão de projetos, governança de TI, avaliação de qualidade de software e pesquisa em privacidade de dados e computação em nuvem. Atuação destacada na implantação e gestão de Escritório de Projetos (PMO) estratégico em ambiente bancário, docência universitária e coordenação de pós-graduação. Certificado PMP (2004) e experiência prática em adoção de boas práticas (PMBOK, ITIL) e métodos ágeis.

**Currículo Lattes: http://lattes.cnpq.br/9494944933209872**

## Prof. MSc. Robério Gomes Patrício

**Arquitetura Orientada a Serviços e APIs Modernas**

Mestre e Doutorando em Ciência da Computação pela UECE. Atua como docente em disciplinas de Programação Orientada a Objetos e Sistemas Distribuídos.

**Currículo Lattes: http://lattes.cnpq.br/4524543853252705**

## Prof. Dr. Carlos Mendes Aderaldo

**Infraestrutura em Nuvem e como Código**

Doutor pela Universidade de Fortaleza e professor de Computação em Nuvem. Executivo de tecnologia com mais de 18 anos de experiência em desenvolvimento de software, computação em nuvem, segurança de aplicações e cibersegurança. Construiu sua carreira próximo ao código e uma forte experiência prática em design de software, testes e práticas modernas de Engenharia. Sua carreira acadêmica inclui um doutorado focado em resiliência de sistemas distribuídos. Possui mais de 10 artigos publicados em conferências e periódicos internacionais, com mais de 345 citações e dois prêmios de melhor artigo em conferências internacionais.

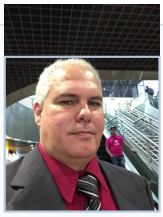
**Currículo Lattes: http://lattes.cnpq.br/1654263373910785**

## Prof. MSc. Francisco Gutenberg da Silva Filho

**Testes Ágeis**

Formado em Engenharia de Software, professor de Qualidade de Software, Mestre em Ciências da Computação (2023) e Bacharel em Engenharia de Software (2020) pela Universidade Federal do Ceará. Vasta experiência em projetos nacionais e internacionais nos domínios de sistemas ERP, sistemas de gestão em saúde, jogos mobile e soluções financeiras. Experiência profissional em gerenciamento de testes, elaboração de artefatos, execução e automação de testes funcionais e não funcionais, testes de API e integração, além da aplicação de metodologias ágeis e práticas de integração contínua com pipelines de CI/CD, bem como o uso de ferramentas e tecnologias como Jira, Testlink, Git/Gitlab, Selenium, Cypress, Rest Assured, Postman, bancos de dados relacionais e administração de servidores Linux/CentOS.

**Currículo Lattes: https://lattes.cnpq.br/3398355347992925**

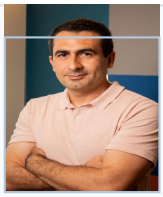


## Prof. Dr. André Luís Moura Santos

**Segurança em Desenvolvimento de Sistemas**

Formado em Engenharia Eletrônica com PhD em computação pela Universidade da Califórnia e bacharelado pelo Instituto Tecnológico de Aeronáutica, ITA (concluído em 1988). Professor da UECE.

**Currículo Lattes: http://lattes.cnpq.br/0486779090155155**



## Prof. Dr. Paulo Henrique Mendes Maia

**Metodologia do Trabalho Científico**

PhD em Computing pelo Imperial College London (Reino Unido) e professor de Engenharia de Software. Com vasta experiência acadêmica, será o responsável pela condução da disciplina de Metodologia do Trabalho Científico.

**Currículo Lattes: http://lattes.cnpq.br/7227955029154651**

DISCIPLINA 01

# Engenharia de Software Ágil com DevOps

**30h**
CARGA HORÁRIA

## Prof. MSc Joaquim Pedro Carvalho

### Ementa

Introdução a Engenharia de Software: histórico, conceitos e principais desafios da Engenharia de Software. Modelos tradicionais de desenvolvimento: tradicional, espiral, iterativo-incremental, processo unificado. Desafios e evolução da Engenharia de Software. Modelos ágeis de desenvolvimento de software: o Manifesto Ágil e o que é ser Ágil hoje; visão geral dos Métodos Ágeis de Desenvolvimento de Software; os métodos Scrum e XP; os métodos Kanban e Burndown; Introdução ao DevOps: principais conceitos e tecnologias. Introdução a Engenharia de Plataformas: conceitos, Internal Developer Platforms (IDP), experiência do desenvolvedor, infraestrutura como produto. Introdução a MLOps e IAOps: diferença entre sistemas de machine learning (ML) e sistemas tradicionais; ciclo de vida da ML; pipelines de ML; uso de IA para operações de TI.

### Conteúdo programático

**1. Engenharia de Software**
- Histórico e principais conceitos

**2. Modelos tradicionais de desenvolvimento de software**
- Cascata
- Espiral
- Iterativo-incremental
- Processo Unificado / RUP

**3. Modelos ágeis de desenvolvimento de software**
- Manifesto Ágil
- Ciclo de vida ágil
- Métodos ágeis de gerenciamento e desenvolvimento: Scrum e XP
- Kanban e métricas ágeis

**4. Introdução ao DevOps**
- Histórico
- Principais conceitos e ferramentas
- DevOps como cultura, como papel e como serviço

**5. Introdução a Engenharia de Plataformas**
- Engenharia de Plataformas vs DevOps
- Conceito de Internal Developer Platform (IDP)
- Experiência do Desenvolvedor (DevEx)
- Infraestrutura como produto

**6. Introdução a MLOps e IAOps**
- Conceitos de MLOps e IAOps
- Diferenças entre software tradicional e sistemas de machine learning (ML)
- Ciclo de vida de ML: coleta de dados, treinamento, validação, deploy, monitoramento
- Versionamento de dados e modelos
- Pipelines de ML

- Uso de IA para operações de TI: detecção de anomalias, correlação de eventos, automação inteligente de incidentes

## Bibliografia


- DevOps na prática: entrega de software confiável e automatizada. Danilo Sato. Casa do Código, 1ª edição, 2014.
- What is DevOps. Mike Loukides. O'Reilly Media, 1ª edição, 2012.
- The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Gene Kim, Jez Humble, Patrick Debois, John Willis. IT Revolution Press, 2ª edição, 2021.
- Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. Nicole Forsgren, Jez Humble, Gene Kim. IT Revolution Press, 2018.
- Introducing MLOps: How to Scale Machine Learning in the Enterprise. Mark Treveil, Nicolas Omont, Clément Stenac. O'Reilly Media, 2021.

DISCIPLINA 02

# Big Data

**45h**
CARGA HORÁRIA

## Prof. Dr. Ricardo Holanda Nobre

| | |
|---|---|
| CARGA HORÁRIA | 60 horas (40h teóricas + 20h práticas em laboratório) |
| MODALIDADE | Presencial / Híbrido |
| PRÉ-REQUISITO | Fundamentos de Programação Python; conceitos básicos de Banco de Dados |

## Ementa

Arquiteturas de dados modernas orientadas a IA (Data Lakehouse, Data Mesh). Processamento distribuído em tempo real com Apache Kafka, Apache Flink e Spark Structured Streaming. Feature Engineering distribuído e Feature Stores (Feast, Hopsworks). Bancos de dados vetoriais e técnicas de Retrieval-Augmented Generation (RAG) para LLMs em escala. MLOps: ciclo de vida de modelos de ML em produção, automação de retreinamento e monitoramento de Data Drift. Uso de Agentes de IA e Assistentes de Código (Claude AI, GitHub Copilot, Cursor) para automação de pipelines de dados, geração de código e orquestração de workflows de Big Data com ferramentas acessíveis e com planos gratuitos ou de avaliação.

## Objetivo geral

Capacitar o aluno a projetar, construir e governar ecossistemas de dados massivos nativamente orientados ao treinamento, implantação e monitoramento de modelos de Inteligência Artificial, integrando infraestruturas de Big Data com o ciclo de vida completo de MLOps e o uso produtivo de Agentes de IA como copiloto de engenharia.

## Objetivos específicos

- Projetar arquiteturas Lakehouse e Data Mesh adequadas a cargas de trabalho de ML;
- Construir e operar pipelines de ingestão e processamento de dados em tempo real;
- Implementar Feature Stores para consistência entre treinamento offline e inferência online;
- Criar pipelines de embeddings vetoriais e sistemas RAG integrados a LLMs;
- Automatizar o ciclo de vida de modelos com práticas de MLOps e monitoramento de drift;
- Utilizar Agentes de IA (Claude, GitHub Copilot, Cursor, n8n, Flowise) para geração de código, automação de tarefas de engenharia de dados e orquestração de workflows;
- Realizar todas as atividades com ferramentas open-source, gratuitas ou com planos de avaliação acessíveis em notebooks e PCs dos alunos.

## Conteúdo programático

### Módulo 1 | Arquiteturas de Dados Modernas para IA — Lakehouse & Mesh (20% da CH)

#### 1.1 De Data Lake a Data Lakehouse AI-Ready

- Evolução das arquiteturas: HDFS/Data Lake Data Lakehouse AI-Ready Platform;
- Implementação prática com Delta Lake (open-source) sobre MinIO (S3 local) ou Google Cloud Storage Free Tier;
- Transações ACID, versionamento de dados (Time Travel) e governança: base para reprodutibilidade de modelos de ML;
- Apache Iceberg e Apache Hudi como alternativas open-source — comparativo e casos de uso.

#### 1.2 Data Mesh e IA Descentralizada

- Conceito de Data Mesh: domínios de dados como produtos para consumo por equipes de Data Science;
- Como evitar monólitos de dados ingeríveis em ambientes distribuídos de IA;
- Prática: modelagem de um Data Domain com Delta Lake e catálogo de metadados simples (Apache Atlas ou DataHub — Free Tier).

## Módulo 2 | Processamento Preditivo e em Tempo Real — Streaming para ML (25% da CH)

### 2.1 Streaming de Alta Performance

- Apache Kafka (via Confluent Cloud Free Tier ou Docker local): padrões publish-subscribe, tópicos, partições e consumidores;
- Apache Flink (processamento stateful) e Spark Structured Streaming: comparativo e cenários de aplicação;
- Construção de pipelines de ingestão de eventos com latência de milissegundos.

### 2.2 Inferência em Streaming

- Técnicas para embarcar modelos de ML pré-treinados (scikit-learn, PyTorch) diretamente na pipeline de streaming;
- Predições on-the-fly sem dependência de banco de dados externo;
- Laboratório: pipeline de detecção de anomalias financeiras com Kafka + Spark Structured Streaming rodando em Docker no notebook do aluno.

### 2.3 Geração de Código com IA para Pipelines de Streaming

- Uso do Claude AI (claude.ai — plano gratuito) para geração e revisão de código PySpark e Kafka Producer/Consumer;
- GitHub Copilot (plano gratuito para estudantes) como copiloto de codificação em tempo real no VS Code;
- Cursor IDE (plano gratuito) para refatoração inteligente de pipelines.

## Módulo 3 | Feature Engineering, Vector DBs e Agentes de IA (30% da CH)

### 3.1 Feature Engineering Distribuído

- Extração e transformação de features em escala com PySpark e Ray;
- Geração automática de features com assistência do Claude AI via prompts especializados.

### 3.2 Feature Stores em Escala

- Implementação de Feature Store open-source com Feast (gratuito): arquitetura online/offline, definição de entidades e features;
- Eliminação do training-serving skew: garantia de consistência entre features de treinamento e inferência em produção;
- Hopsworks Community Edition como alternativa com suporte nativo a Vector DB.

### 3.3 Vector Databases e RAG em Escala

- Fundamentos de embeddings vetoriais e busca por similaridade (ANN — Approximate Nearest Neighbor);
- Ferramentas open-source e gratuitas: Chroma DB (local), Milvus Lite, Weaviate (Sandbox), Qdrant (Free Tier);
- Construção de pipeline RAG (Retrieval-Augmented Generation): ingestão de documentos geração de embeddings (Hugging Face) armazenamento vetorial consulta + geração via LLM;
- Integração com LLMs gratuitos: Claude AI (API Free Tier), Ollama (execução local de LLMs — Llama 3, Mistral).

### 3.4 Agentes de IA para Engenharia de Dados

- Conceito de AI Agents e Agentic Workflows aplicados a Big Data;
- n8n (open-source, self-hosted ou Cloud Free Tier): criação de workflows de automação de pipelines de dados com agentes de IA integrados;
- Flowise (open-source): construção de agentes conversacionais com acesso a bases de dados vetoriais e Feature Stores;
- LangChain / LlamaIndex (gratuito): orquestração de agentes de IA para automação de tarefas de ETL, geração de relatórios e monitoramento de qualidade de dados;
- Laboratório prático: agente de IA com n8n que monitora um tópico Kafka, aciona retreinamento automático ao detectar data drift e notifica via Slack/e-mail.

## Módulo 4 | MLOps e Governança de Big Data para IA (25% da CH)

### 4.1 Ciclo de Vida do Modelo em Escala — MLOps

- MLflow (open-source, gratuito): rastreamento de experimentos, registro de modelos e deploy como API REST;
- Kubeflow Pipelines no Google Kubernetes Engine (Free Tier) ou Minikube local;
- Automação de retreinamento baseado em gatilhos de dados (Data Drift / Model Drift) detectados na pipeline de Big Data;
- Deploy de modelos como microsserviços com FastAPI + Docker (gratuito).

### 4.2 Governança, Qualidade e Ética de Dados para IA

- Great Expectations (open-source): checks automatizados de qualidade de dados distribuídos no Spark;
- Prevenção de bias e dados corrompidos que degradam modelos de IA;
- Apache Atlas / DataHub (Community Free): catálogo de dados, lineage e governança;
- Princípios de ética em IA e compliance com LGPD/GDPR aplicados a pipelines de Big Data.

### 4.3 AIOps e Observabilidade de Pipelines

- OpenTelemetry (CNCF, gratuito): rastreamento distribuído de pipelines de dados desde a ingestão até a inferência;
- Grafana + Prometheus (gratuito, auto-hospedado): dashboards de monitoramento de latência, throughput e model drift;
- Uso de Claude AI para análise inteligente de logs e alertas: prompts para diagnóstico automático de falhas em pipelines.

## Ferramentas, tecnologias e disponibilidade

| Ferramenta / Plataforma | Finalidade na disciplina | Acesso |
|---|---|---|
| **Delta Lake + MinIO (Docker)** | Data Lakehouse local | Open-source / Gratuito |
| **Apache Kafka (Docker / Confluent Cloud)** | Streaming de eventos | Open-source / Free Tier |
| **Apache Spark (PySpark)** | Processamento distribuído | Open-source / Gratuito |
| **Apache Flink (Docker)** | Streaming stateful / Inferência | Open-source / Gratuito |
| **Feast Feature Store** | Feature Store online/offline | Open-source / Gratuito |
| **Chroma DB / Qdrant** | Vector Database local/cloud | Open-source / Free Tier |
| **Ollama + Llama 3 / Mistral** | LLM local para RAG | Open-source / Gratuito |
| **Claude AI (claude.ai)** | Geração de código e agentes | Free Tier disponível |
| **GitHub Copilot** | Copiloto de código no VS Code | Gratuito p/ estudantes |
| **Cursor IDE** | Refatoração inteligente de código | Free Tier disponível |
| **n8n (self-hosted ou cloud)** | Workflows de agentes de IA | Open-source / Free Tier |
| **Flowise** | Agentes conversacionais RAG | Open-source / Gratuito |
| **LangChain / LlamaIndex** | Orquestração de agentes LLM | Open-source / Gratuito |
| **MLflow** | Rastreamento e registro de modelos | Open-source / Gratuito |
| **Great Expectations** | Qualidade de dados distribuída | Open-source / Gratuito |
| **Docker + Docker Compose** | Ambiente de laboratório local | Open-source / Gratuito |
| **Google Colab / Kaggle Notebooks** | Computação na nuvem gratuita | Free Tier (GPU disponível) |
| **Grafana + Prometheus** | Observabilidade de pipelines | Open-source / Gratuito |

## Metodologia

A disciplina adota abordagem ativa centrada em laboratórios práticos, onde cada tópico teórico é imediatamente seguido de implementação hands-on. O ambiente de laboratório é 100% executável no notebook ou PC do aluno via Docker Compose, sem necessidade de infraestrutura corporativa.

- Aulas expositivas dialogadas: apresentação de conceitos arquiteturais com exemplos reais da indústria.
- Laboratórios (mão na massa): implementação guiada em duplas, com código disponibilizado no GitHub da disciplina.
- AI-Assisted Coding: uso obrigatório de Claude AI e GitHub Copilot para geração, revisão e refatoração de código — desenvolvendo o skill de prompt engineering para engenharia de dados.

- Pair Programming com Agentes: criação de agentes autônomos (n8n/Flowise) que executam tarefas de engenharia de dados, reduzindo trabalho manual repetitivo.
- Projeto Integrador: desenvolvimento incremental de uma arquitetura completa ao longo da disciplina.

## Workshops práticos

### Workshop 1 — Data Lakehouse Local com Delta Lake e IA-Assisted Setup

- Subir ambiente completo com Docker Compose (MinIO + Apache Spark + Delta Lake) no notebook
- Usar Claude AI para gerar os scripts de inicialização do ambiente e schemas Delta Lake
- Demonstrar transações ACID, Time Travel e reprodutibilidade de datasets de treinamento
- Ferramentas: Docker, PySpark, Delta Lake, MinIO, Claude AI (geração de código)

### Workshop 2 — Pipeline de Streaming com Inferência On-the-Fly

- Criar producer Kafka que simula transações financeiras em tempo real
- Construir pipeline Spark Structured Streaming que consome, processa features e aplica modelo de ML para detecção de fraude
- Usar GitHub Copilot / Claude AI para gerar e iterar sobre o código PySpark
- Ferramentas: Kafka (Docker), Spark, scikit-learn, Claude AI, GitHub Copilot

### Workshop 3 — Feature Store + Vector DB + RAG Pipeline

- Implementar Feature Store com Feast para servir features online (Redis) e offline (Parquet)
- Construir pipeline RAG: ingestão de documentos PDF embeddings (Hugging Face) Chroma DB consulta com Ollama (Llama 3 local)
- Usar Flowise para criar interface conversacional no-code sobre o RAG pipeline
- Ferramentas: Feast, Chroma DB, Ollama, Hugging Face, Flowise, LangChain

### Workshop 4 — Agente de IA para Automação de Pipeline de Dados com n8n

- Criar agente n8n que monitora tópico Kafka, detecta anomalias nos dados (Great Expectations) e aciona retreinamento MLflow
- Integrar Claude AI API ao workflow n8n para análise inteligente de alertas e geração de relatórios automáticos
- Deploy do modelo retreinado como endpoint FastAPI via Docker
- Ferramentas: n8n, MLflow, Great Expectations, FastAPI, Docker, Claude AI API

### Projeto Integrador Final — Arquitetura Completa de Big Data para IA

- Projetar e implementar end-to-end: ingestão Kafka Lakehouse Delta Lake Feature Store Feast treinamento MLflow deploy FastAPI monitoramento Grafana
- Incorporar agente de IA (n8n ou Flowise) para automação de retreinamento e alertas
- Usar Claude AI como arquiteto consultor: gerar diagramas de arquitetura, revisar código e sugerir otimizações via prompts
- Documentar decisões arquiteturais com auxílio de IA generativa
- Entrega: repositório GitHub documentado + apresentação de 15 minutos demonstrando o sistema em execução

## Avaliação

| Instrumento de avaliação | Peso | Módulo(s) |
|---|---|---|
| **Workshops 1 e 2 — Relatório técnico + código GitHub** | 25% | 1 e 2 |
| **Workshop 3 — Pipeline RAG funcional documentado** | 20% | 3 |
| **Workshop 4 — Agente de IA operacional (n8n/Flowise)** | 20% | 3 e 4 |
| **Projeto Integrador Final — Arquitetura completa + apresentação** | 35% | 1 a 4 |

Critério de aprovação: média ponderada maior ou igual a 7,0 e frequência maior ou igual a 75%. Todos os workshops devem ser entregues com repositório GitHub público contendo README, código comentado e instruções de execução via Docker Compose.

## Referências bibliográficas

### 9.1 Arquiteturas AI-Ready — Lakehouse, Mesh e Governança

- Dehghani, Z. (2022). Data Mesh: Delivering Data-Driven Value at Scale. O'Reilly Media.

- Armbrust, M. et al. (2021). Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. VLDB.
- Great Expectations. (2026). Official Documentation: Data Quality in Spark Pipelines. greatexpectations.io
- CNCF. (2026). OpenTelemetry: Official Documentation. opentelemetry.io

### 9.2 Processamento e Streaming

- Apache Software Foundation. (2026). Apache Kafka: Official Documentation. kafka.apache.org
- Apache Software Foundation. (2026). Apache Flink: Official Documentation. flink.apache.org
- Anyscale. (2026). Ray: Official Documentation. docs.ray.io

### 9.3 Feature Stores e Engenharia Distribuída

- Chambers, B., & Zaharia, M. (2018). Spark: The Definitive Guide. O'Reilly Media. (Atualizações em spark.apache.org)
- Feast. (2026). Official Documentation: Online and Offline Feature Serving. docs.feast.dev
- Hopsworks. (2026). Feature Store Documentation & Vector DB Integration. docs.hopsworks.ai

### 9.4 Vector Databases, RAG e LLMs

- Chroma. (2026). Official Documentation. docs.trychroma.com
- Weaviate. (2026). Official Documentation. weaviate.io/developers/weaviate
- Ollama. (2026). Official Documentation. ollama.com/docs
- LangChain. (2026). Official Documentation. python.langchain.com

### 9.5 MLOps e Ciclo de Vida de Modelos

- Databricks/Linux Foundation. (2026). MLflow: Official Documentation. mlflow.org
- Google Cloud. (2025). MLOps: Continuous Delivery and Automation in ML. Google Cloud Whitepapers.

### 9.6 Agentes de IA e Ferramentas de Automação

- Anthropic. (2026). Claude AI: Official Documentation & API Reference. docs.anthropic.com
- n8n. (2026). Official Documentation: Workflow Automation. docs.n8n.io
- Flowise. (2026). Official Documentation: Low-Code LLM App Builder. docs.flowiseai.com

## Orientações para configuração do ambiente

Todo o ambiente da disciplina é executável localmente via Docker e Docker Compose, sem necessidade de conta em nuvem paga. Os requisitos mínimos recomendados são: 8 GB de RAM (16 GB recomendado), 50 GB de espaço em disco livre e conexão à internet. O repositório oficial da disciplina no GitHub conterá arquivos docker-compose.yml pré-configurados para cada workshop, permitindo que o ambiente completo seja iniciado com um único comando.

### Contas gratuitas a criar antes do início da disciplina

- Claude AI: claude.ai (Free Tier com acesso a claude.ai e API com créditos iniciais);
- GitHub: github.com (gratuito; solicitar GitHub Student Developer Pack para Copilot gratuito);
- Cursor IDE: cursor.com (Free Tier com 2.000 completions/mês);
- Confluent Cloud: confluent.io (Kafka Free Tier com 5 GB/mês);
- Hugging Face: huggingface.co (acesso gratuito a modelos e Inference API);
- Google Colab: colab.research.google.com (gratuito, com acesso a GPUs T4).

DISCIPLINA 03

# Desenvolvimento Web com Frameworks

**45h**
CARGA HORÁRIA

## Prof. Esp. Wellington Costa Pereira

### Ementa

Conceitos de REST, padrões de software e boas práticas de código, com breve introdução ao uso de ferramentas de IA como apoio ao desenvolvimento. Desenvolvimento de APIs RESTful com Java e Spring Boot, incluindo consumo de APIs externas e de Inteligência Artificial Generativa via HTTP. Persistência de dados com PostgreSQL. Desenvolvimento de interfaces web modernas com React, com ênfase em consumo de APIs e exibição de respostas em streaming. Projeto prático integrado full-stack.

### Justificativa e princípios norteadores

A proposta tem como objetivo incorporar conceitos de Inteligência Artificial à ementa já consolidada da disciplina, sem desvirtuar o seu foco central nem tornar o conteúdo superficial dentro da carga horária de 45 horas.

- Preservar o núcleo da disciplina: REST, padrões de software, boas práticas de código, Java/Spring, React e PostgreSQL continuam sendo a espinha dorsal do curso. Esses fundamentos são condição necessária para que qualquer integração com IA seja feita com qualidade e critério.
- Adicionar IA onde ela agrega valor direto ao conteúdo já existente: integração com APIs de IA Generativa é uma extensão natural do consumo de APIs REST já ensinado no backend; Generative UI é uma evolução orgânica do desenvolvimento de interfaces com React; desenvolvimento assistido por IA é uma prática que impacta imediatamente a produtividade do estudante durante a própria disciplina.

A disciplina não aborda microserviços (contemplados na disciplina específica), infraestrutura em nuvem, MLOps, bancos de dados vetoriais em profundidade, nem frameworks Python — todos fora do escopo e/ou tratados em outras disciplinas da grade.

### Objetivo geral

Capacitar o estudante a projetar e implementar aplicações web full-stack modernas utilizando Java/Spring Boot no backend e React/Next.js no frontend, incorporando de forma prática e responsável recursos de Inteligência Artificial Generativa.

### Objetivos específicos

- Aplicar os princípios REST, padrões de projeto e boas práticas de código em aplicações web com IA.
- Desenvolver APIs RESTful robustas com Spring Boot capazes de consumir APIs de LLMs (ex.: OpenAI, Anthropic) com suporte a streaming.
- Realizar persistência e modelagem de dados com PostgreSQL integrados a um backend Spring.
- Construir interfaces com React e introduzir o modelo de Server Components do Next.js para suporte a Generative UI.
- Implementar padrões de Generative UI: geração dinâmica de componentes e respostas em streaming na interface.
- Utilizar ferramentas de IA (GitHub Copilot, Cursor, Claude) como aceleradores do desenvolvimento, compreendendo seus limites e riscos.

### Conteúdo programático

A disciplina está organizada em quatro módulos progressivos, cada um construindo sobre o anterior e culminando em um projeto integrado.

**Módulo 1: Fundamentos — ~8h (18%)**

- 1.1 REST e design de APIs: revisão de REST — verbos HTTP, status codes, recursos, versionamento; boas práticas de design de API: nomenclatura, contratos, documentação com OpenAPI/Swagger.

- 1.2 Padrões de software e boas práticas: padrões de projeto relevantes para aplicações web — Repository, Service Layer, Strategy, Factory; princípios SOLID aplicados ao contexto web; Clean Code: nomenclatura, funções, organização de código; uso de ferramentas de IA generativa (Copilot, Cursor) como apoio ao desenvolvimento: potencial e limitações; revisão crítica de código gerado por IA: o que verificar antes de aceitar uma sugestão.

### Módulo 2: Backend com Java e Spring Boot — ~18h (40%)

- 2.1 Desenvolvimento de APIs com Spring Boot: arquitetura de um projeto Spring Boot — camadas Controller, Service, Repository; Spring Web MVC: criação de endpoints REST, tratamento de exceções, validação; Spring Data JPA com PostgreSQL: mapeamento O/R, JPQL, consultas customizadas; segurança básica com Spring Security: autenticação stateless com JWT.
- 2.2 Persistência de dados com PostgreSQL: modelagem relacional para aplicações web — entidades, relacionamentos, índices; queries complexas e otimização básica.
- 2.3 Consumo de APIs externas — incluindo APIs de IA generativa: consumo de APIs REST externas com WebClient/RestClient no Spring Boot; APIs de IA Generativa (OpenAI, Anthropic) como qualquer outro endpoint REST — autenticação, envio de payload JSON, leitura da resposta; streaming de respostas via Server-Sent Events (SSE): como expor para o frontend sem bloquear o servidor; boas práticas transversais: tratamento de erros, timeout, variáveis de ambiente para credenciais.

### Módulo 3: Frontend com React — ~14h (31%)

- 3.1 React moderno: componentes funcionais, hooks (useState, useEffect, useContext, useMemo); gerenciamento de estado — Context API e introdução ao Zustand; consumo de APIs REST no frontend: fetch, axios, tratamento de loading/error; componentização, reuso e design systems.
- 3.2 Consumo de APIs com resposta em streaming: leitura de respostas em streaming com fetch + ReadableStream; exibição incremental de texto na interface — padrão de atualização de estado a cada chunk recebido; prática: interface React que consome o endpoint do Módulo 2.

### Módulo 4: Projeto prático integrado — ~5h (11%)

O projeto integrador percorre toda a stack da disciplina e materializa as competências adquiridas nos módulos anteriores. Os estudantes desenvolvem, em grupos de 2 a 3 pessoas, uma aplicação web full-stack com as seguintes características obrigatórias:

- Backend em Spring Boot
- Persistência de dados em PostgreSQL
- Consumo de uma API de IA Generativa
- Frontend em React consumindo o backend
- Revisão de código em pares ao longo do desenvolvimento

## Distribuição da carga horária

| Módulo | Horas | CH |
|---|---|---|
| **Módulo 1 — Fundamentos** | ~8h | 18% |
| **Módulo 2 — Backend com Java e Spring Boot** | ~18h | 40% |
| **Módulo 3 — Frontend com React** | ~14h | 31% |
| **Módulo 4 — Projeto Prático Integrado** | ~5h | 11% |
| **Total** | 45h | 100% |

## Metodologia

A disciplina combina aulas expositivo-dialogadas com laboratórios práticos. A proporção sugerida é de 40% teoria e 60% prática, dado o perfil aplicado da especialização.

- Aulas teóricas com exemplos de código ao vivo e discussão de trade-offs de design.
- Laboratórios guiados nos quais o estudante implementa incrementos da aplicação completa ao longo dos módulos.
- Code reviews em pares ao longo da disciplina.
- Projeto integrador ao final, com documento de escopo entregue antes do início da implementação.

## Referências bibliográficas


### Base da disciplina — referências mantidas

- FLANAGAN, David. JavaScript: The Definitive Guide — Master the World's Most-Used Programming Language. 7. ed. O'Reilly Media, 2020.
- BLOCH, Joshua. Effective Java. 3. ed. Addison-Wesley, 2017.
- FERREIRA, Alexandre. APIs REST: Seus serviços prontos para o mundo real. Casa do Código, 2017.
- STEFF, Michael. Spring Boot: Acelere o desenvolvimento de microsserviços. Casa do Código, 2016.
- REACT DOCUMENTATION. React: react.dev. Disponível em https://react.dev. Acesso contínuo.

### Adições — consumo de APIs e integração com IA

- OPENAI. OpenAI Platform Documentation — API Reference. https://platform.openai.com/docs. Referência para os parâmetros da API consumida no laboratório do Módulo 2 (modelo, mensagens, streaming).
- ANTHROPIC. Anthropic API Documentation. https://docs.anthropic.com. Alternativa à OpenAI; documentação sobre estrutura de mensagens e streaming via SSE.
- SPRING PROJECTS. WebClient Documentation (Spring WebFlux). Referência para consumo de APIs externas assíncronas no Spring Boot, incluindo leitura de respostas em streaming.

### Qualidade de código e boas práticas com IA

- FOWLER, Martin. Refactoring: Improving the Design of Existing Code. 2. ed. Addison-Wesley, 2018.
- MARTIN, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

### Leituras complementares recomendadas

- HUNAIN, Muhammad. Spring Boot 3 + Spring AI Tutorials. Baeldung.com, 2024–2026. https://www.baeldung.com/spring-ai
- POSTGRESQL GLOBAL DEVELOPMENT GROUP. pgvector: Open-source vector similarity search for Postgres. https://github.com/pgvector/pgvector
- WIERUCH, Robin. The Road to React. Leanpub, 2024.

DISCIPLINA 04

# Governança em TI

**15h**
CARGA HORÁRIA

## Prof. Dr. Eliseu Castelo Branco Júnior

### Ementa

Conceitos fundamentais de Governança Corporativa e Governança de Tecnologia da Informação (TI). Alinhamento estratégico entre TI e negócios em contextos de transformação digital. Modelos de governança, gestão e entrega de valor em TI. Estruturas de decisão, papéis e responsabilidades. Gestão de riscos, compliance e aspectos regulatórios (com ênfase na LGPD). Segurança da informação e cibersegurança como pilares da governança. Métricas, indicadores de desempenho (KPIs) e geração de valor com TI. Frameworks e boas práticas: COBIT 2019, ITIL 4, ISO/IEC 27001, ISO/IEC 38500 e Balanced Scorecard (BSC). Arquitetura corporativa e governança de dados. Cultura ágil, DevOps e o fim da TI bimodal. Tendências em Governança de TI: ESG, inteligência artificial, cloud computing e governança de dados.

### Conteúdo programático

**1. Fundamentos de Governança**
- Governança Corporativa vs Governança de TI
- Princípios, objetivos e estruturas de governança
- Criação de valor com TI

**2. Alinhamento Estratégico**
- Planejamento estratégico de TI
- Business-IT Alignment
- Cadeia de valor e transformação digital

**3. Estruturas de Governança**
- Papéis e responsabilidades (CIO, CDO, CISO)
- Comitês de TI
- Modelos de tomada de decisão

**4. Gestão de Riscos e Compliance**
- Gestão de riscos em TI
- LGPD e proteção de dados
- Auditoria e conformidade
- Controles internos

**5. Segurança da Informação e Cibersegurança**
- Princípios de segurança
- Gestão de incidentes
- Continuidade de negócios (BCP/DRP)

**6. Frameworks e Boas Práticas**
- COBIT 2019 — governança e gestão corporativa de TI; domínios e objetivos de controle
- ITIL 4 — Sistema de Valor de Serviço (SVS); cadeia de valor de serviços; práticas de gerenciamento
- ISO/IEC — ISO 38500 (Governança de TI); ISO 27001 (Segurança da Informação)
- BSC (Balanced Scorecard) — indicadores estratégicos; performance e valor

**7. Métricas e Indicadores**
- KPIs e OKRs em TI

- Avaliação de desempenho
- ROI em TI

**8. Arquitetura e Governança de Dados**

- Governança de dados
- Qualidade e segurança da informação
- Data-driven organizations

**9. Modelos Modernos de TI**

- DevOps e Agile
- Cultura digital
- Fim da TI Bimodal

**10. Tendências em Governança de TI**

- ESG e tecnologia
- Inteligência Artificial e ética
- Cloud governance
- Transformação digital

## Bibliografia básica

- ISACA. COBIT 2019 Framework: Governance and Management Objectives. ISACA, 2019.
- AXELOS. ITIL Foundation: ITIL 4 Edition. TSO, 2019.
- WEILL, P.; ROSS, J. IT Governance: How Top Performers Manage IT Decision Rights for Superior Results. Harvard Business School Press.

## Bibliografia complementar

- FERNANDES, A.; ABREU, V. Implantando a Governança de TI. Brasport.
- MANSUR, R. Governança de TI: Metodologias, Frameworks e Melhores Práticas. Brasport.
- MAGALHÃES, I.; PINHEIRO, W. Gerenciamento de Serviços de TI na Prática. Novatec.

## Bibliografia atual e avançada

- ISACA. COBIT 2019 Design Guide.
- ISACA. COBIT 2019 Implementation Guide.
- ISO/IEC 38500:2015 – Governance of IT.
- ISO/IEC 27001:2022 – Information Security Management.
- KIM, G.; HUMBLE, J.; DEBOIS, P.; WILLIS, J. The DevOps Handbook.
- WESTERMAN, G.; BONNET, D.; MCAFEE, A. Leading Digital.
- DAVENPORT, T. Competing on Analytics.

DISCIPLINA 05

# Sistemas Distribuídos

**30h**
CARGA HORÁRIA

## Prof. Dr. Joaquim Celestino Júnior

| | |
|---|---|
| CARGA HORÁRIA | 30 horas |
| MODALIDADE | Teórico-prática (Hands-on) |
| PREVISÃO | Especialização em Engenharia de Software com DevOps. |

## Objetivo geral

Capacitar o aluno a projetar, governar e operar ecossistemas de sistemas distribuídos modernos e "AI-Native", centrados em arquiteturas escaláveis, resilientes, observáveis e governáveis via GitOps.

## Objetivos específicos

- Fundamentos: compreender os desafios inerentes de rede (falácias) e tempo em SD.
- Dados & Consenso: analisar os trade-offs de consistência (CAP/PACELC) e algoritmos de consenso na prática.
- Arquiteturas: projetar ecossistemas de microsserviços leves com APIs modernas (gRPC) e arquiteturas baseadas em eventos (EDA).
- Operação (DevOps): integrar práticas de Engenharia de Plataforma e GitOps para orquestração declarativa e governança automática.

## Ementa consolidada

Introdução aos sistemas distribuídos modernos: as 8 Falácias da Computação Distribuída. Modelos de comunicação síncronos (gRPC) e assíncronos (Event Streaming vs. Filas). Idempotência operacional. Consistência de dados massivos e preditivos e Teorema CAP/PACELC. Algoritmos de consenso na prática (etcd/Raft). Padrões avançados de microsserviços: Saga Pattern e introdução a CQRS. Orquestração Nativa com Kubernetes como Plano de Controle Distribuído. Cultura de Plataforma e Service Mesh (mTLS, resiliência no "edge"). Observabilidade Nativa e AIOps: OpenTelemetry. Práticas DevOps Avançadas: Governança via GitOps e Chaos Engineering. Pipelines de IA Distribuída.

## Conteúdo programático detalhado (30h)

### Módulo 1 – Fundamentos e o Novo Desafio (5h)

- 1.1. De monólitos a microsserviços e cloud native: a evolução cultural e tecnológica.
- 1.2. As 8 falácias da computação distribuída: o "banho de realidade" sobre rede.
- 1.3. O desafio DevOps em SD: projetar para a falha; reprodutibilidade de ambientes e dados.

### Módulo 2 – Comunicação e o Coração dos Dados (10h)

- 2.1. APIs modernas: REST (API-First), gRPC (HTTP/3 e Protobuf) e GraphQL.
- 2.2. O novo middleware de dados: arquiteturas orientadas a eventos (EDA). Diferença entre filas (RabbitMQ) vs. streaming de eventos (Kafka/Pulsar).
- 2.3. Idempotência operacional: por que retries automáticos exigem operações idempotentes.
- 2.4. Teorema CAP & PACELC: consistência estrita vs. eventual.
- 2.5. Consenso na prática: padrões Raft/Paxos simplificados. Uso de etcd/ZooKeeper via GitOps.

### Módulo 3 – Arquiteturas e Resiliência DevOps (10h)

- 3.1. Orquestração nativa: Kubernetes (K8s) como plano de controle distribuído (Service Discovery, Self-Healing).
- 3.2. Padrões de resiliência no código: Circuit Breaker, retries (exponential backoff/jitter), idempotency.
- 3.3. Transações distribuídas: Saga Pattern (orquestração vs. coreografia).

- 3.4. Service Mesh (Istio/Linkerd): abstração de rede, mTLS automático, gerenciamento de tráfego (canary) e resiliência.

**Módulo 4 – Observabilidade, Segurança e AIOps (5h)**

- 4.1. Observabilidade nativa: mover do monitoramento reativo para proativo baseado em IA. Os três pilares: logs estruturados, métricas distribuídas e tracing distribuído (OpenTelemetry/Jaeger).
- 4.2. DevSecOps em SD (zero-trust): mTLS automático via mesh, OAuth2/OIDC para identidade de serviços e gestão de segredos integrada via GitOps.
- 4.3. Pipelines de IA distribuída: como treinar e servir modelos (LLMs) massivos distribuindo inferência e dados.

## Metodologia Hands-on (Stack 2026)

- Linguagem: Go (ou Python com AsyncIO).
- Middleware: gRPC e Apache Kafka (Streaming).
- Dados: PostgreSQL (CP) e Redis/Cassandra (AP).
- Orquestração: Kubernetes (via kind local).
- Plataforma: Istio (Mesh), Argo CD (GitOps), OpenTelemetry (Tracing).

## Bibliografia essencial (DevOps Focus)

- KLEPPMANN, M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media. O "padrão de ouro" moderno que substitui Couloris/Tanenbaum para arquitetura de dados e trade-offs de SD.
- NEWMAN, S. (3ª edição, 2025). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. Guia prático essencial para padrões operacionais e de design específicos de microsserviços modernos.
- CNCF (Cloud Native Computing Foundation). (2026). CNCF Project Documentation: Argo CD (GitOps), Istio (Mesh), Prometheus/OpenTelemetry (AIOps). As documentações oficiais dos projetos CNCF são as fontes mais atuais sobre padrões de orquestração autônoma, governança via GitOps e observabilidade distribuída.

DISCIPLINA 06

# Arquitetura Orientada a Serviços e APIs Modernas

**30h**
CARGA HORÁRIA

## Prof. MSc. Robério Gomes Patrício

### Ementa

A disciplina aborda conceitos essenciais de arquitetura orientada a serviços (SOA) com foco em abordagens modernas, como microserviços e APIs cloud-native. Explora padrões, protocolos e ferramentas atuais para o desenvolvimento de serviços escaláveis e seguros, alinhados às demandas do mercado, incluindo API design, gerenciamento de segurança e implantação com containers. Ênfase em RESTful APIs, GraphQL, segurança com OAuth 2.0 e JWT, e princípios de microserviços, priorizando aplicações práticas e desafios reais para otimizar o tempo de aula.

### Conteúdo programático

- Introdução à SOA moderna e evolução para microserviços: motivadores, definições, benefícios e desafios; comparação entre arquiteturas monolíticas, SOA tradicional e microserviços; integração básica com IA para otimização de serviços.
- Características principais, papéis e design de serviços: provider, consumer e registry; API-first design; identificação e modelagem de serviços usando Domain-Driven Design (DDD) simplificado; padrões como loose coupling e high cohesion.
- Estudos de caso e tecnologias essenciais: exemplos reais de empresas como Netflix e bancos brasileiros; tecnologias-chave: JSON, Protobuf; APIs RESTful, GraphQL e gRPC; interoperabilidade com OpenAPI.
- Infraestrutura e orquestração: cloud-native basics; containers (Docker) e orquestração (Kubernetes); API gateways; introdução a serverless (ex.: AWS Lambda).
- Segurança e governança: modelos com OAuth 2.0, OpenID Connect e JWT; princípios de DevSecOps; gerenciamento de APIs e riscos.
- Desenvolvimento com microserviços e tendências: observabilidade básica (monitoring e logging); resiliência (circuit breakers); tendências como sustentabilidade (GreenOps) e migração de monolitos; exercícios práticos.

### Bibliografia

- Building Microservices: Designing Fine-Grained Systems (2ª edição). Sam Newman. O'Reilly Media, 2021.
- Microservices Patterns: With Examples in Java. Chris Richardson. Manning Publications, 2018.
- Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. Sam Newman. O'Reilly Media, 2019.
- Microservice APIs: Using Python, Flask, FastAPI, OpenAPI and More. Jose Haro Peralta. Manning Publications, 2023.
- Spring Microservices in Action (2ª edição). John Carnell e Illary Huaylupo Sánchez. Manning Publications, 2021.

DISCIPLINA 07

# Infraestrutura em Nuvem e como Código

**45h**
CARGA HORÁRIA

## Prof. Dr. Carlos Mendes Aderaldo

### Ementa

- Implantação de aplicações em Computação em Nuvem;
- O conceito de infraestrutura como código;
- Automação com Terraform.

### Conteúdo programático

**1. Introdução à Computação em Nuvens**
- Modelos de serviço: IaaS, PaaS, SaaS
- Modelos de implantação: pública, privada e híbrida

**2. Virtualização de processamento**
- Tecnologias e ferramentas de virtualização
- Desafios da virtualização

**3. Arquitetura na Nuvem**
- Histórico e conceitos
- Escalabilidade
- Tecnologias e ferramentas

**4. Implantação de aplicações na nuvem**

**5. O conceito de infraestrutura como código**

**6. Automação com Ansible**
- Introdução, formato YAML
- Preparação do ambiente
- Inventário, playbooks
- Criação de roles

**7. Infraestrutura como código**
- Introdução
- Variáveis, uso de módulos e gerenciamento de estado
- Criação de módulos e gestão de ambientes

DISCIPLINA 08

# Testes Ágeis

**30h**
CARGA HORÁRIA

## Prof. MSc. Francisco Gutenberg da Silva Filho

### Ementa

Teoria de verificação, validação e confiabilidade de software; Planejamento de testes; Critérios e técnicas de testes: testes funcionais, estruturais, fluxo de dados, regressão, níveis de testes, testes de unidade, testes de integração, testes de aceitação, testes de desempenho; Inspeção, Revisão Técnica; Depuração; Documentação e especificação de testes; Automação e Ferramentas de testes; Conceitos de Test-driven development e Behaviour-driven development.

### Conteúdo programático

**1. Testes Ágeis e os fundamentos da Agilidade**

- Valores e princípios ágeis
- Métodos ágeis: Scrum e Extreme Programming
- O que são testes ágeis
- Transição entre testes tradicionais e testes ágeis

**2. O Time Ágil e os Testes**

- Papéis e responsabilidades do "testador ágil" no time multidisciplinar
- Papéis e responsabilidades do programador nos testes
- O dia-a-dia do testador ágil
- Quadrantes de testes ágeis e o plano de testes
- Planejamento de release/iteração e testes

**3. Práticas Ágeis de Testes**

- Testes de regressão
- Testes unitários e Test-Driven Development (TDD)
- Testes de aceitação e Acceptance Test Driven Development (ATDD)
- Behavior Driven Development (BDD)
- Testes exploratórios
- Testes de usabilidade
- Testes de aceitação do usuário
- Testes não-funcionais

**4. Pipeline e Integração Contínua aplicados a Testes**

- Conceitos: CI (Integração Contínua), CD (Entrega/Deploy Contínuo) e "shift-left testing"
- Estrutura de um pipeline de qualidade (estágios, jobs, gatilhos e artefatos)
- Paralelismo

**5. Ferramentas para Testes**

- Automação de testes funcionais
- Automação de testes de API

**6. Inteligência Artificial (IA) para Testes**

- IA na prática para QAs
- Prompt engineering — como instruir a IA

- Agentes de IA para testes

## Bibliografia


- Software Testing Techniques. B. Beizer. Ed. Van Nostrand Reinhold Company, 1983.
- Java Testing Patterns. Jon Thomas, Matthew Young, Kyle Brown e Andrew Glover. Ed. Wiley, 2004.

DISCIPLINA 09

# Segurança em Desenvolvimento de Sistemas

**30h**
CARGA HORÁRIA

## Prof. Dr. André Luís Moura Santos

### Objetivo geral

Capacitar o aluno a projetar, desenvolver e operar sistemas de software seguros, incorporando princípios de segurança da informação ao longo de todo o ciclo de vida de desenvolvimento, com foco em práticas de DevSecOps.

### Objetivos específicos

- Compreender os fundamentos de segurança da informação
- Identificar vulnerabilidades e ameaças em sistemas modernos
- Aplicar práticas de codificação segura
- Integrar segurança em pipelines de CI/CD
- Avaliar riscos e implementar controles de segurança

### Ementa

Fundamentos de segurança da informação: confidencialidade, integridade e disponibilidade. Modelos de ameaças e análise de riscos. Criptografia aplicada (simétrica, assimétrica, funções hash). Segurança em redes e aplicações web. Principais vulnerabilidades segundo o OWASP (incluindo o Top 10). Autenticação, autorização e controle de acesso. Segurança em APIs e microserviços. Segurança em containers e orquestração. Práticas de DevSecOps: integração de testes de segurança em pipelines CI/CD, análise estática e dinâmica (SAST/DAST), gestão de vulnerabilidades. Segurança em ambientes de nuvem. Monitoramento, resposta a incidentes e conformidade.

### Conteúdo programático

**Módulo 1 – Fundamentos de Segurança (6h)**

- Princípios: confidencialidade, integridade e disponibilidade (CIA)
- Ameaças, vulnerabilidades e riscos
- Modelagem de ameaças (ex.: STRIDE)
- Políticas e governança de segurança

**Módulo 2 – Criptografia e Proteção de Dados (6h)**

- Conceitos básicos de criptografia
- Criptografia simétrica e assimétrica
- Hashes e assinaturas digitais
- Gestão de chaves e certificados

**Módulo 3 – Segurança de Aplicações (6h)**

- Vulnerabilidades comuns (OWASP Top 10)
- Injeção, XSS, CSRF, autenticação insegura
- Boas práticas de codificação segura
- Testes de segurança em aplicações

**Módulo 4 – DevSecOps e Pipeline Seguro (6h)**

- Integração de segurança no ciclo DevOps
- Ferramentas SAST, DAST e SCA

- Segurança em CI/CD
- Segurança em containers (ex.: Docker)
- Orquestração segura com Kubernetes

**Módulo 5 – Segurança em Nuvem e Operações (6h)**

- Conceitos de segurança em cloud
- Gestão de identidade e acesso (IAM)
- Monitoramento e logging
- Resposta a incidentes
- Compliance e LGPD

## Metodologia

- Aulas expositivas com estudos de caso reais
- Laboratórios práticos (hands-on) com exploração de vulnerabilidades
- Uso de ferramentas de mercado (scanners, análise de código, etc.)
- Desenvolvimento de um mini-projeto com pipeline DevSecOps

## Avaliação

- Projeto prático (pipeline com segurança integrada) – 40%
- Exercícios/labs – 30%
- Participação e estudos dirigidos – 30%

## Bibliografia básica

- STALLINGS, W. Cryptography and Network Security.
- ANDERSON, R. Security Engineering.
- KIM, G. et al. The DevOps Handbook.

## Bibliografia complementar

- The Phoenix Project – Gene Kim
- Designing Data-Intensive Applications – Martin Kleppmann
- Documentação oficial do OWASP
- Guias de segurança em cloud (AWS, Azure, GCP)

## Competências desenvolvidas

Ao final da disciplina, o aluno será capaz de:

- Incorporar segurança desde a concepção do software (security by design)
- Automatizar testes de segurança em pipelines DevOps
- Identificar e mitigar vulnerabilidades em aplicações modernas
- Projetar soluções seguras em ambientes distribuídos e em nuvem

DISCIPLINA 10

# Integração Contínua e Gerenciamento de Configuração

**30h**
CARGA HORÁRIA

## Prof. MSc. Joaquim Pedro Carvalho

### Ementa

Controle de versão: conceitos, estratégias de branches e merges, gitops. Integração contínua: conceitos, automação de builds, pipeline de entrega, uso de IA na integração contínua. Controle de mudanças e rastreabilidade. Introdução à observabilidade.

### Conteúdo programático

1. **Controle de versão**
   - Conceitos iniciais: repositório de versões, branches e merges, "tudo como código"
   - Estratégias de branches: trunk-based / feature branches / git flow
   - GitOps
2. **Integração contínua**
   - Automação de builds
   - Verificações da qualidade da build: testes, análise de código, segurança
   - Esteira (pipeline) de entrega — visão geral das ferramentas
   - Uso de IA na integração contínua: geração de testes, análise de código e de segurança, análise de logs
3. **Infraestrutura como código (IaC)**
   - Conceitos
   - Introdução à configuração da infraestrutura com Ansible
   - Introdução à criação da infraestrutura com Terraform
4. **Controle de mudanças e rastreabilidade**
   - Controle de mudanças em ambientes ágeis
   - Gestão de releases
   - Rastreabilidade ponta a ponta
5. **Introdução à observabilidade**
   - Conceitos: logs, métricas e alarmes
   - Ferramentas de observabilidade
   - Uso de IA na observabilidade e incidentes

### Bibliografia


- Pro Git. Scott Chacon, Ben Straub. Apress, 2014.
- Entrega Contínua: Como Entregar Software de Forma Rápida e Confiável. Jez Humble. Bookman, 1ª edição, 2013.
- Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way. Bas Meijer, Lorin Hochstein, René Moser. O'Reilly Media, 3ª edição, 2022.
- Terraform: Up & Running: Writing Infrastructure as Code. Yevgeniy Brikman. O'Reilly Media, 3ª edição, 2022.
- Site Reliability Engineering: How Google Runs Production Systems. Jennifer Petoff, Betsy Beyer, Chris Jones, Niall Richard Murphy. O'Reilly Media, 2016.

DISCIPLINA 11

# Metodologia do Trabalho Científico

**30h**
CARGA HORÁRIA

## Prof. Dr. Paulo Henrique Mendes Maia

### Ementa

Conhecimento e Ciência. Pesquisa Científica: etapas, modalidade, coleta e análise de dados. Trabalhos Acadêmicos. Monografia: apresentação gráfica, orientações metodológicas, estrutura. Delimitação da Pesquisa: tema, problema, objetivos e etapas do desenvolvimento.

### Conteúdo programático

**1. Método científico**

- Introdução
- Desenvolvimento histórico do método
- Conceito atual do método
- Formas de raciocínio: indução e dedução
- O método hipotético-dedutivo

**2. Noções gerais sobre pesquisa**

- Conceito de pesquisa
- Tipos de pesquisa: bibliográfica, descritiva e experimental
- Projeto de pesquisa

**3. Procedimentos para investigação científica**

- Introdução
- Escolha do assunto (delimitação, formulação do problema, revisão bibliográfica, objetivos da pesquisa)
- Hipóteses
- Definição da amostra

**4. Análise de dados**

- Formas de coleta de dados
- Níveis de mensuração

**5. Divulgação da pesquisa**

- Estruturação de uma monografia
- Artigos e revistas científicas

### Bibliografia

- ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 14724: informação e documentação: trabalhos acadêmicos: apresentação. Rio de Janeiro, 2011.
- Metodologia do trabalho científico. SEVERINO, Antônio Joaquim. 21. ed. São Paulo: Cortez, 2000.
- Introdução à metodologia do trabalho científico. ANDRADE, Maria Margarida de. 5. ed. São Paulo: Atlas, 2001.

DISCIPLINA 12

# Monografia

**90h**
CARGA HORÁRIA

## INFORMAÇÕES E INSCRIÇÕES

| | |
|---|---|
| TELEFONE / WHATSAPP | (85) 9 8781-7200 |
| E-MAIL | eesdevops@gmail.com |
| SITE | https://pdv.iepro.org.br/cursos/devops_T4/ |
| MODALIDADE | Presencial — Campus Itaperi |
| AULAS | Terças e quintas, das 19h às 22h |
| DURAÇÃO | 18 meses |
| INÍCIO | 27/10/2026 |

### Como se inscrever:

- Acesse o site do processo seletivo e faça sua inscrição online.
- Reúna seus documentos pessoais e o comprovante de graduação.
- Acompanhe as etapas do processo seletivo pelos canais oficiais.
- Garanta sua vaga na turma com início em 27/10/2026.

## REALIZAÇÃO

Universidade Estadual do Ceará (UECE) — Campus Itaperi.
Especialização em Engenharia de Software com DevOps — Turma T4.

## INVESTIMENTO

**Inscrição:** R$ 180,00

**Matrícula:** R$ 550,00

**Mensalidades:** 18x de R$ 550,00