PÓS TECH

FIAP + alura

//SEGURANÇA

PROFISSIONAL DE

ARQUITETURA DE SOFTWARE

//TESTABILIDADE

//SOFT SKILLS

TÉCNICAS, ESTRATÉGIAS E HABILIDADES PESSOAIS.

//DOMAIN-DRIVE

03 INTRO →

04 CONHECIMENTOS GERAIS →

04 DOMAIN-DRIVEN DESIGN →

05 GUIA DE ESTUDOS →

07 PADRÕES E PRÁTICAS →

08 SEGURANÇA →

09 RESILIÊNCIA →

10 MONITORAMENTO E ALERTAS →

11 EVOLUÇÃO DO SISTEMA →

12 TESTABILIDADE E QUALIDADE →

13 SOFT SKILLS E LIDERANÇA →

14 CONCLUSÃO →

QUE BOM TER VOCÊ POR AQUI

Ser uma pessoa que atua com Arquitetura de Software exige muito mais do que habilidades técnicas avançadas. É preciso ter uma visão estratégica para garantir que as soluções de software sejam escaláveis, sustentáveis e alinhadas aos objetivos dos negócios. E isso é, sim, uma boa notícia.

Existem vários guias essenciais que ensinam como estruturar sistemas complexos e promover uma colaboração eficaz entre equipes técnicas e times de negócios. Mas isso é apenas o ponto inicial da sua trajetória.

Para realmente se destacar nesta profissão é preciso compreender profundamente os fundamentos, os princípios que geraram os padrões de arquitetura e as práticas ágeis, além de dominar técnicas avançadas de escalabilidade, segurança e resiliência no dia a dia dos projetos.

Mas, o sucesso não depende apenas de conhecimento técnico.
Habilidades interpessoais, como comunicação eficaz, gestão de conflitos e liderança nos concorridos ambientes corporativos, são fundamentais na construção de equipes coesas e no desenvolvimento de soluções que sejam consideradas bem-sucedidas.
Neste mercado, o equilíbrio é tudo.

Comece bem, por aqui. Boa leitura.

FUNDAMENTOS QUE TODO ARQUITETO DE SOFTWARE DE SUCESSO DEVE TER

Ser um(a) Arquiteto(a) de Software exige não apenas uma profunda compreensão técnica, mas também uma visão estratégica de como as soluções de software podem ser escaláveis, sustentáveis e alinhadas aos objetivos de negócios.

Para entender melhor, um bom caminho é ler o livro Domain-Driven Design de Eric Evans, uma referência crucial para qualquer arquiteto(a) que deseja entender como estruturar sistemas complexos de maneira que eles evoluam de forma eficiente e atendam a todas as reais necessidades de um negócio.

DOMAIN-DRIVEN DESIGN

Para quem ainda não conhece, o DDD (Domain-Driven Design) ajuda a criar uma linguagem comum entre as equipes técnica e de negócios, promovendo uma colaboração mais efetiva, permitindo que a arquitetura seja desenvolvida com uma base sólida que favoreça a clareza e a flexibilidade, mesmo no caso de cenários de alta complexidade.

Através dos conceitos de modelagem de domínio, entidades, agregados e eventos, o DDD fornece uma abordagem prática para projetar sistemas que não apenas resolvem problemas técnicos, mas que também suportam o crescimento e as mudanças do negócio a longo prazo, o que é comum.

Mas saber o DDD é só o começo. Não basta entender um único conceito ou abordagem. O verdadeiro desafio está em aplicar esses conceitos de forma mais eficaz, levando em consideração a variedade de contextos e as necessidades que surgem ao longo do ciclo de vida de um sistema.

FIAP + alura

NADA COMO COMEÇAR COM UM BOM GUIA DE ESTUDOS

A seguir, apresentaremos um Guia de Estudos com tópicos e áreas que vão ajudar você a se tornar um(a) excelente profissional de Arquitetura de Software, com fundamentos técnicos e o desenvolvimento de algo importante: habilidades de liderança e comunicação, essenciais para o sucesso na profissão.

FUNDAMENTOS TÉCNICOS

O primeiro passo é ter uma base sólida em Engenharia de Software. Isso inclui:

SOLID

Ajuda na criação de código modular.

DRY

Evita repetições desnecessárias.

YAGNI

(You Aren't Gonna Need It)

Assegura que não há funcionalidades desnecessárias.

KISS

(Keep It Simple, Stupid)

Orienta na construção de soluções simples e eficazes.

DESIGN PATTERNS

Fornecem soluções para problemas recorrentes.

Esses princípios são fundamentais para garantir que o código seja bem estruturado, de fácil manutenção e escalabilidade.

A ESCOLHA DO PADRÃO ARQUITETURAL CERTO É CRUCIAL PARA O SUCESSO DE UM SISTEMA

Um(a) arquiteto(a) deve entender como e quando usar cada padrão, conforme as necessidades de escalabilidade, flexibilidade e resiliência do sistema.

Estude os principais padrões de arquitetura, como:

MONOLÍTICO

Abordagem em que todos os componentes e funcionalidades de um sistema são integrados em uma única aplicação ou módulo.

MICROSSERVIÇOS

Destaca-se devido à capacidade de escalar de forma independente.

EDA

A Event-Driven Architecture oferece flexibilidade e desacoplamento para sistemas que precisam reagir a eventos em tempo real.

SCRUM

É uma metodologia ágil que organiza o trabalho em ciclos curtos, chamados sprints, para a entrega incremental de software.

KANBAN

É um método visual de gestão de fluxo de trabalho que foca na melhoria contínua e na entrega de tarefas.

DEVOPS

Promove a automação e integração contínua, permitindo que o software seja entregue de modo rápido e sem problemas.

FAMILIARIZE-SE COM PRÁTICAS ÁGEIS COMO: SCRUM, KANBAN E DEVOPS

Esses métodos garantem que o desenvolvimento de software seja contínuo e eficiente.

FIAP + alura

PADRÕES E PRÁTICAS SÃO FUNDAMENTAIS

ESTEJA SEMPRE ATENTO(A) AOS DOIS

Para ser um arquiteto de software, é necessário ir além dos conceitos básicos e explorar padrões e práticas avançadas que ajudam a garantir que o sistema seja robusto, escalável e mantenha a mais alta qualidade ao longo do tempo.

AQUI ESTÃO ALGUMAS DAS PRINCIPAIS ÁREAS QUE UM ARQUITETO DEVE DOMINAR:

- **Escalabilidade e performance**
  A escalabilidade é um dos principais desafios ao projetar sistemas modernos. Um arquiteto deve estudar técnicas para garantir que o sistema consiga lidar com o aumento de carga, tanto verticalmente (aumento da capacidade do servidor) quanto horizontalmente (adicionando mais servidores).

- **Balanceamento de carga**
  Distribuir o tráfego entre diferentes servidores ou instâncias para evitar sobrecarga em um único ponto e garantir alta disponibilidade.

- **Sharding**
  Dividir grandes volumes de dados em fragmentos menores (shards), o que facilita o gerenciamento e melhora a performance de bancos de dados.

- **Cache**
  Utilizar caching (memória, como Redis ou Memcached) para melhorar a performance de leituras frequentes, reduzindo o tempo de resposta.

- **Otimização de consultas**
  Otimizar consultas de banco de dados usando índices, evitando consultas n+1 e buscando soluções como leitura e gravação assíncronas para reduzir a latência.

FIAP + alura

SEGURANÇA

UM FATOR CADA VEZ MAIS ESSENCIAL

Em um cenário em que ataques cibernéticos são cada vez mais comuns, a segurança se torna uma prioridade. Como arquiteto(a), você deve garantir que os sistemas sejam projetados com segurança desde o início, incluindo:

CRIPTOGRAFIA

Usá-la para proteger dados sensíveis tanto em repouso quanto em trânsito (SSL/TLS).

Compreender algoritmos como AES para criptografia simétrica e RSA para criptografia assimétrica.

AUTENTICAÇÃO E AUTORIZAÇÃO

Garantir que os sistemas possuam mecanismos de autenticação robustos, como o uso de OAuth, OpenID Connect e autenticação multifatorial (MFA).

A autorização deve ser baseada em papéis (RBAC) ou atributos (ABAC) para garantir que os usuários acessem apenas os recursos para os quais têm permissão.

PREVENÇÃO DE ATAQUES

Proteção contra os ataques comuns como SQL Injection, Cross-Site Scripting (XSS) e Cross-Site Request Forgery (CSRF).

Implementação de práticas como validação e sanitização de entradas de usuários, uso de parâmetros preparados em consultas SQL e políticas de segurança de conteúdo (CSP).

FIAP + alura

RESILIÊNCIA

FUNDAMENTAL PARA O SISTEMA

A resiliência é essencial para garantir que o sistema continue funcionando, mesmo diante de falhas. Para isso, existem técnicas como:

CIRCUIT BREAKER

Implemente o padrão Circuit Breaker para evitar chamadas repetidas a serviços que estão com falha, dando tempo para recuperação e evitando sobrecarregar o sistema.

RETRY LOGIC

Automatizar tentativas de reprocessamento de operações falhas de forma controlada (com backoff exponencial, por exemplo), para reduzir o impacto de falhas temporárias.

TIMEOUTS E FALLBACKS

Defina timeouts adequados para chamadas de rede e interações com serviços externos, além de implementar alternativas de fallback (como respostas de cache ou valores padrão) para não interromper a experiência do usuário.

FIAP + alura

MONITORAMENTO E ALERTAS

Configure monitoramento para detectar anomalias no sistema, como: aumentos inesperados na latência, falhas de rede ou alta utilização de recursos, e configure alertas para garantir respostas mais rápidas e precisas à ocorrência de incidentes.

PARA ISSO, O(A) ARQUITETO(A) DEVE SABER RESPONDER:

- **Quais métricas monitorar?**
  Identificar as métricas essenciais, como latência, taxa de erro, tempo de resposta, utilização de CPU/memória, taxa de requisições, entre outras. Essas métricas ajudam a compreender o comportamento do sistema em tempo real e a antecipar problemas.

- **Qual é o limiar para gerar um alerta?**
  Um arquiteto precisa definir limiares apropriados para alertas. Não é eficiente disparar alertas para qualquer pequena variação, mas também não se pode esperar que os problemas maiores ocorram antes de serem notados. Os limiares devem ser ajustados com base no contexto e na criticidade do serviço.

- **Como responder a um alerta?**
  O arquiteto deve trabalhar com a equipe para garantir que haja processos bem-definidos para reagir a alertas. Isso inclui ter playbooks para cada tipo de incidente, que detalham como lidar com diferentes falhas ou anomalias. A capacidade de agir rapidamente é vital para você restaurar o serviço sem causar impacto significativo.

- **Quais ferramentas utilizar para monitoramento e alertas?**
  Saber escolher ferramentas de monitoramento que permitem uma visualização clara e eficiente do estado do sistema, como:

  - Prometheus
  - Grafana
  - Datadog
  - ELK Stack

  Além disso, é importante integrar essas ferramentas com sistemas de notificação que garantem que os alertas sejam recebidos pelos responsáveis no momento certo, como:

  - Slack
  - PagerDuty

FIAP + alura

COMO ACOMPANHAR A EVOLUÇÃO DO SISTEMA?

É fundamental que a arquitetura seja projetada de maneira a permitir a coleta contínua de dados relevantes, como métricas de desempenho, uso e erros.

Um bom profissional deve antecipar a necessidade de monitoramento e estabelecer indicadores-chave que ajudem a identificar problemas e oportunidades de melhoria ao longo do tempo.

Além de reagir a falhas, o sistema deve ser capaz de fornecer métricas que possibilitem a análise de tendências, como o crescimento da carga de trabalho, mudanças no comportamento dos usuários e a performance em diferentes cenários.

O monitoramento não deve ser estático, à medida que o sistema evolui, é necessário adaptar e expandir os pontos de coleta de dados, incluindo novas métricas conforme a arquitetura se transforma.

Por exemplo, novos componentes ou funcionalidades podem exigir a implementação de novas ferramentas de observação ou ajustes nas já existentes.

Isso permite um ciclo contínuo de análise, diagnóstico e otimização, garantindo que o sistema permaneça eficiente, escalável e alinhado com as necessidades do negócio e dos usuários.

A capacidade de fazer ajustes e melhorias de forma proativa é essencial para manter a sustentabilidade e o sucesso do sistema a longo prazo.

FIAP + alura

TESTABILIDADE E QUALIDADE DE CÓDIGO

PRIMORDIAIS PARA A MANUTENÇÃO E A EVOLUÇÃO CORRETA DO SISTEMA

Como arquiteto(a), você deve garantir que a arquitetura suporte boas práticas de teste e que a equipe siga padrões para garantir a qualidade:

AUTOMAÇÃO DE TESTES

Defina uma estratégia de testes automatizados, incluindo testes unitários, de integração e end-to-end. Ferramentas como: JUnit, NUnit, Selenium e Cypress podem ser úteis.

TEST-DRIVEN DEVELOPMENT (TDD)

Incentive a equipe a escrever testes antes de escrever o código, isso garante que o software seja testável desde o início e que cada nova funcionalidade tenha uma cobertura de testes adequada.

ANÁLISE DE QUALIDADE DE CÓDIGO

Utilize ferramentas de análise estática de código, como: SonarQube e ESLint para garantir que o código esteja em conformidade com os padrões estabelecidos, sem bugs ou problemas de segurança.

CODE REVIEWS

Fundamental para detectar erros antes que o código seja integrado ao sistema.

SOFT SKILLS E LIDERANÇA

MUITO ALÉM DO DOMÍNIO TÉCNICO

Um(a) arquiteto(a) de software deve ter uma forte habilidade em liderança e comunicação para coordenar equipes, tomar decisões estratégicas e garantir o sucesso de seus projetos.

AQUI ESTÃO ALGUMAS DAS SOFT SKILLS ESSENCIAIS PARA ESSA FUNÇÃO:

- **Comunicação eficaz**
  Seja capaz de comunicar ideias técnicas complexas de forma clara e acessível é fundamental, principalmente ao interagir com seus stakeholders que não são técnicos.

  Isso inclui traduzir as decisões de design em uma linguagem compreensível e conectar os objetivos de negócio com as soluções técnicas propostas. Você deve ser capaz de comunicar riscos, benefícios e trade-offs de decisões arquiteturais com transparência.

- **Gestão de conflitos**
  Em qualquer projeto, especialmente em sistemas complexos, é natural que surjam desacordos técnicos. Como arquiteto(a), você precisa ser capaz de gerenciar e resolver esses conflitos de forma construtiva.

  Isso significa mediar discussões entre os membros da equipe, equilibrando diferentes pontos de vista, levando em consideração as necessidades tanto técnicas quanto de negócios, encontrando soluções que atendam a todos, de perfil técnico e não técnico.

- **Mentoria e desenvolvimento de equipe**
  Um(a) arquiteto(a) de software não apenas toma decisões técnicas, mas também investe no desenvolvimento da equipe. Oriente os membros sobre as boas práticas de codificação, as práticas ágeis e de design de arquitetura e incentive a aprendizagem contínua.

  Um(a) arquiteto(a) eficaz deve ser um(a) mentor(a), capaz de elevar o nível técnico da equipe, promovendo um ambiente de colaboração e crescimento

- **Tomada de decisão estratégica**
  Como arquiteto, você frequentemente precisará tomar decisões difíceis entre várias alternativas, considerando os trade-offs. Isso envolve analisar os impactos a longo prazo de diferentes soluções, levando em conta custos, desempenho, escalabilidade e manutenção

  Além disso, o(a) arquiteto(a) deve ser capaz de articular essas decisões aos stakeholders de forma que todos entendam o raciocínio por trás das escolhas e como elas alinham-se aos objetivos do negócio

CONCLUSÃO

Imagine um mix de competências complementares.
E você terá uma pessoa de Arquitetura de Software.

Um(a) arquiteto(a) de software de sucesso deve possuir uma combinação de habilidades técnicas e interpessoais para enfrentar os desafios de projetar sistemas complexos e escaláveis.

Além de um sólido domínio de linguagens de programação, padrões de design e arquitetura de software, é essencial que o(a) profissional entenda profundamente a gestão de requisitos e a integração de diferentes sistemas.

O conhecimento em metodologias ágeis, como Scrum ou Kanban, também é fundamental, pois garante uma abordagem iterativa e colaborativa no desenvolvimento de software.

A capacidade de tomar decisões informadas sobre a escalabilidade, segurança e desempenho do sistema é outro ponto crucial, assim como a habilidade de equilibrar as necessidades do cliente com as limitações técnicas.

Além disso, um(a) arquiteto(a) de software deve estar sempre atualizado(a) com as novas tecnologias, tendências e melhores práticas do mercado.

O bom relacionamento com equipes multidisciplinares e a habilidade de comunicar de forma clara e eficaz são qualidades indispensáveis para garantir a execução bem-sucedida dos projetos.

Tudo isso parece difícil? Nem tanto. A experiência do dia a dia dará a vivência necessária para que você se destaque na sua organização.

Apenas comece a colocar em prática e tenha sucesso!

FIAP + alura

PÓS TECH

Pós-Graduações Tech Intensive em Dev. Tenha jornadas de aprendizado contínuo, conectadas às mais recentes demandas de um mercado que não para de evoluir.

CURSOS OFERECIDOS

PÓS TECH - ONLINE

ARQUITETURA DE SISTEMAS .NET

10 MESES - 360 HORAS

SAIBA MAIS

PÓS TECH - ONLINE

ARQUITETURA E DESENVOLVIMENTO JAVA

10 MESES - 360 HORAS

SAIBA MAIS

PÓS TECH - ONLINE

FRONT-END ENGINEERING

10 MESES - 360 HORAS

SAIBA MAIS

PÓS TECH - ONLINE

FULL STACK DEVELOPMENT

10 MESES - 360 HORAS

SAIBA MAIS

PÓS TECH - ONLINE

IA PARA DEVS

10 MESES - 360 HORAS

SAIBA MAIS

PÓS TECH - ONLINE

SOFTWARE ARCHITECTURE

10 MESES - 360 HORAS

SAIBA MAIS

VEJA OUTROS CURSOS: **POSTECH.FIAP.COM.BR**

FIAP + alura

A FIAP E A ALURA FORMAM O MAIOR
E MAIS EFICIENTE ECOSSISTEMA DE EDUCAÇÃO
EM TECNOLOGIA, NEGÓCIOS, EMPREENDEDORISMO
E PRODUTOS DIGITAIS DA AMÉRICA LATINA, COM
SOLUÇÕES DE EDUCAÇÃO E PRODUTOS DIFERENTES
PARA AS DIVERSAS ETAPAS DE DESENVOLVIMENTO
DAS CARREIRAS DE ALUNOS E ALUNAS. A MISSÃO
DAS DUAS FRENTES É A MESMA: TRANSFORMAR
A REALIDADE DE PESSOAS E EMPRESAS PARA QUE
ELAS SE REALIZEM, POR MEIO DE RESULTADOS,
EM UMA JORNADA DE SUCESSO.

FIAP + alura

PARA MAIS INFORMAÇÕES SOBRE OS CURSOS E FORMAÇÕES
EM DEV, VISITE NOSSO SITE:

fiap.com.br