Exercícios - Funcionamento da TI nas Organizações e Introdução à Programação

EXERCÍCIOS TEÓRICOS

Exercício 1

Questão: Explique a diferença entre as responsabilidades de um Administrador de Sistemas e um Engenheiro de Infraestrutura, citando pelo menos 3 atividades específicas de cada função.

Gabarito:

- **Administrador de Sistemas:**
  - Gerencia servidores (Linux, Windows, etc.)
  - Cuida de atualizações, patches e backups
  - Administra serviços como Active Directory, DNS, DHCP
- **Engenheiro de Infraestrutura:**
  - Atua em projetos de expansão ou modernização da infraestrutura
  - Planeja arquiteturas escaláveis e resilientes
  - Trabalha com cloud computing (AWS, Azure, GCP), virtualização e containers

Exercício 2

Questão: Descreva as 5 etapas do ciclo de desenvolvimento de software, explicando brevemente o objetivo de cada uma e dando um exemplo prático para cada etapa.

Gabarito:

1. **Análise de Requisitos:** Entender o problema e necessidades do cliente
   - *Exemplo:* Definir que um banco precisa de funcionalidades de transferência, consulta de saldo e histórico
2. **Projeto/Design:** Definir arquitetura do sistema e escolher tecnologias
   - *Exemplo:* Decidir usar banco de dados MySQL, interface web responsiva e integração com APIs de pagamento
3. **Programação (Codificação):** Traduzir especificações em código executável
   - *Exemplo:* Programar as telas de login, função de transferência e validações de segurança
4. **Testes:** Verificar se o software funciona corretamente
   - *Exemplo:* Testar cenários como senha incorreta, saldo insuficiente, valores inválidos
5. **Implantação e Manutenção:** Colocar em produção e manter o sistema
   - *Exemplo:* Publicar o app, corrigir bugs reportados e adicionar novas funcionalidades

Exercício 3

Questão: Analise a seguinte situação: "Uma empresa está implementando metodologias DevOps para melhorar a integração entre as equipes." Explique como essa implementação beneficia a colaboração entre os times de Infraestrutura e Desenvolvimento, citando pelo menos 3 ferramentas mencionadas no texto.

Gabarito: A implementação de DevOps melhora a colaboração porque:

- Integra os times de desenvolvimento e infraestrutura
- Automatiza o ciclo de vida do software através de CI/CD

- Promove automação, confiabilidade e agilidade

Ferramentas mencionadas:

- Docker (containerização)
- Kubernetes (orquestração de containers)
- Terraform (infraestrutura como código)
- Ansible (automação)
- Jenkins/GitLab CI (integração contínua)

Exercício 4

Questão: Compare as responsabilidades de um Desenvolvedor Front-End, Back-End e Full Stack. Em que situações seria mais vantajoso contratar um desenvolvedor Full Stack ao invés de especialistas separados?

Gabarito: Responsabilidades:

- **Front-End:** Interface do usuário, HTML/CSS/JavaScript, frameworks como React/Angular/Vue
- **Back-End:** Lógica de negócios, comunicação com banco de dados, linguagens como Java/Python/Node.js
- **Full Stack:** Atua em ambas as frentes

Vantagens do Full Stack:

- Times menores onde há necessidade de versatilidade
- Acelerar entregas com menos dependência entre papéis
- Reduzir custos em projetos de menor complexidade
- Maior agilidade na comunicação e tomada de decisões

Exercício 5

Questão: Explique por que "programar não é tudo" no desenvolvimento de software. Use a analogia da construção de uma casa apresentada no texto e elabore sobre como isso se aplica ao desenvolvimento de um sistema de e-commerce.

Gabarito: Programar é apenas uma etapa do processo completo, assim como "colocar tijolos" é apenas parte da construção de uma casa. O desenvolvimento completo envolve todo o processo: planejar, projetar, calcular, testar e construir.

Aplicação ao e-commerce:

- **Planejamento:** Definir que produtos vender, público-alvo, funcionalidades necessárias
- **Projeto:** Arquitetura do sistema, escolha de tecnologias, design da interface
- **Programação:** Codificar carrinho de compras, sistema de pagamentos, cadastros
- **Testes:** Verificar funcionalidades, performance, segurança
- **Manutenção:** Corrigir bugs, adicionar funcionalidades, atualizações de segurança

EXERCÍCIOS PRÁTICOS

Exercício 6

Situação Prática: Você é o responsável pela TI de uma empresa que está crescendo rapidamente. Atualmente vocês têm apenas um técnico de suporte, mas estão recebendo muitas chamadas complexas que ele não consegue resolver.

Tarefa: Elabore um plano de estruturação do time de suporte em 3 níveis, definindo responsabilidades específicas para cada nível e o perfil profissional necessário.

Gabarito: Estrutura proposta:

Nível 1 - Técnico de Suporte Júnior:

- Atendimento inicial e triagem
- Resetar senhas, suporte a software básico
- Orientações gerais sobre uso de sistemas
- Perfil: Técnico em informática, conhecimentos básicos

Nível 2 - Analista de Suporte Pleno:

- Problemas complexos (impressoras, rede, configurações avançadas)
- Instalação e configuração de software especializado
- Suporte a hardware mais complexo
- Perfil: Superior em TI ou experiência comprovada, conhecimento em redes

Nível 3 - Especialista/Engenheiro:

- Suporte especializado e correção de falhas sistêmicas
- Interação com fornecedores
- Projetos de infraestrutura
- Perfil: Especialização ou vasta experiência, conhecimento avançado em sistemas

Exercício 7

Situação Prática: Uma startup de delivery de comida contratou você para criar seu primeiro aplicativo. Eles querem um app simples onde restaurantes podem cadastrar pratos e clientes podem fazer pedidos.

Tarefa: Aplicando o ciclo de desenvolvimento de software, detalhe cada etapa para este projeto específico, incluindo perguntas que você faria e decisões que tomaria em cada fase.

Gabarito: 1. Análise de Requisitos:

- Perguntas: Quantos restaurantes? Tipos de pagamento? Delivery próprio ou terceirizado?
- Definições: Cadastro de restaurantes, cardápios, pedidos, pagamentos, rastreamento

2. Projeto/Design:

- Arquitetura: App mobile (React Native/Flutter) + API REST + Banco PostgreSQL
- Integrações: Gateway de pagamento, mapas para localização
- Interface: Wireframes das telas principais

3. Programação:

- Backend: API para cadastros, pedidos, autenticação
- Frontend: Telas de login, listagem de restaurantes, carrinho, pagamento
- Integrações: Implementar pagamentos e notificações

4. Testes:

- Funcionais: Fluxo completo de pedido
- Performance: Tempo de resposta da API
- Usabilidade: Facilidade de navegação

5. Implantação e Manutenção:

- Deploy: Play Store/App Store, servidores de produção
- Monitoramento: Logs, métricas de uso
- Evolução: Novas funcionalidades baseadas no feedback

Exercício 8

Situação Prática: Você precisa montar uma equipe de desenvolvimento para criar um sistema interno de RH que será usado por 500 funcionários. O orçamento permite contratar 6 profissionais.

Tarefa: Defina a composição ideal da equipe, justificando cada escolha e explicando como esses profissionais irão colaborar durante o projeto.

Gabarito: Composição sugerida:

1. **Product Owner (1):** Para definir requisitos e prioridades com o RH
2. **Scrum Master (1):** Para facilitar processos ágeis e remover impedimentos
3. **Arquiteto/Desenvolvedor Sênior (1):** Para definir arquitetura e liderar tecnicamente
4. **Desenvolvedor Full Stack (2):** Para desenvolvimento do frontend e backend
5. **QA/Tester (1):** Para garantir qualidade antes da entrega

Colaboração:

- PO trabalha com stakeholders do RH para definir funcionalidades
- Scrum Master organiza sprints e dailies
- Arquiteto define padrões e orienta developers
- Developers implementam funcionalidades em pares/colaboração
- QA testa cada funcionalidade antes de ir para produção
- Todos participam de retrospectivas para melhoria contínua

Exercício 9

Situação Prática: Uma empresa tradicional que sempre manteve seus servidores localmente está considerando migrar para a nuvem. Como especialista em cloud, você precisa apresentar um plano de migração.

Tarefa: Elabore uma estratégia de migração considerando os papéis de Engenheiro de Infraestrutura, Especialista em Cloud e Engenheiro de Segurança. Inclua riscos e benefícios.

Gabarito: Estratégia de Migração:

Fase 1 - Análise (Engenheiro de Infraestrutura):

- Inventário completo da infraestrutura atual
- Mapeamento de dependências entre sistemas
- Análise de performance e capacidade atual

Fase 2 - Planejamento (Especialista em Cloud):

- Escolha da plataforma (AWS/Azure/GCP)
- Definição da arquitetura cloud (híbrida/pública)
- Provisioning de infraestrutura como código (Terraform)
- Estimativa de custos e timeline

Fase 3 - Segurança (Engenheiro de Segurança):

- Análise de conformidade (LGPD, ISO 27001)
- Configuração de firewalls e VPNs
- Políticas de acesso e criptografia
- Plano de resposta a incidentes

Benefícios: Escalabilidade, redução de custos operacionais, alta disponibilidade **Riscos:** Dependência de internet, curva de aprendizado, possível aumento inicial de custos

Exercício 10

Situação Prática: Você está liderando um projeto de modernização de um sistema legado de uma empresa financeira. O sistema atual é monolítico e precisa ser transformado em microserviços.

Tarefa: Descreva como os diferentes profissionais (Arquiteto de Software, DevOps Engineer, Desenvolvedor Back-End e Analista de Segurança) trabalhariam juntos neste projeto. Inclua desafios específicos e soluções propostas.

Gabarito: Colaboração entre profissionais:

Arquiteto de Software:

- Define a decomposição do monólito em microserviços
- Estabelece padrões de comunicação entre serviços (APIs REST, mensageria)
- Define arquitetura de dados e estratégias de migração

DevOps Engineer:

- Implementa pipeline CI/CD para múltiplos serviços
- Configura orquestração com Kubernetes
- Automatiza deployment e monitoring de cada microserviço

Desenvolvedor Back-End:

- Implementa novos microserviços seguindo padrões definidos
- Refatora código legado gradualmente
- Implementa APIs e integração entre serviços

Analista de Segurança:

- Define autenticação/autorização entre microserviços (JWT, OAuth)
- Implementa gateway de API com controles de segurança
- Monitora vulnerabilidades em múltiplos pontos de entrada

Desafios e Soluções:

- **Desafio:** Manter sistema funcionando durante migração
- **Solução:** Migração gradual (Strangler Fig Pattern)
- **Desafio:** Complexidade de monitoramento
- **Solução:** Observabilidade distribuída com logs centralizados
- **Desafio:** Transações distribuídas
- **Solução:** Implementação de padrões como Saga Pattern