**Boas práticas de Mercado**

Neste módulo, abordamos as melhores práticas no desenvolvimento de aplicações, destacando a importância do versionamento de código, as melhores práticas no desenvolvimento Java, e os conceitos de entrega e melhoria contínua. Também explora tendências atuais na criação de aplicações e oferece orientações para a criação de um portfólio profissional.

**Boas Práticas de Mercado**

**1. Importância do Versionamento de Código**

O versionamento de código é crucial para o gerenciamento eficiente das mudanças no código-fonte ao longo do tempo. Ele permite rastrear modificações, colaborar com a equipe e garantir a integridade do código.

**2. Boas Práticas no Desenvolvimento Java**

Para garantir código Java eficiente e sustentável, é importante seguir boas práticas, incluindo:

a) Convenções de nomenclatura: melhora a legibilidade e manutenção do código.

b) Documentação do código: documentar o código facilita o entendimento e a manutenção por outros desenvolvedores.

c) Código limpo e legível: código claro e organizado reduz erros e facilita o processo de revisão.

d) Boas práticas de Orientação a Objetos: aplicar princípios de orientação a objetos promove um design modular e reutilizável.

e) Gerenciamento de recursos de forma adequada: uso eficiente de recursos evita desperdícios e melhora o desempenho.

f)Tratamento de exceções: o tratamento apropriado de exceções melhora a robustez do sistema.

g) Implementação de testes: testes garantem que o código funcione  conforme o esperado e ajudam a identificar problemas precocemente.

h) Segurança: práticas de segurança ajudam na proteção contra vulnerabilidades.

i) Versionamento: manter o versionamento atualizado facilita o gerenciamento de mudanças e colaboração.

**3. Conceitos de Entrega e Melhoria Contínua**

A entrega contínua serve como um facilitador para a melhoria contínua, e a melhoria contínua pode ser aplicada aos processos de entrega para otimizar continuamente os pipelines de CI/CD e outros aspectos do desenvolvimento.

**Entrega Contínua**

Envolve uma série de práticas para automatizar e otimizar o processo de integração e entrega de software. As principais práticas incluem:

• Integração Contínua (CI): Integra e testa alterações de código frequentemente.

• Automação de Builds e Testes: Automatiza a construção e verificação do software.

• Implantação Automatizada: Facilita a entrega contínua de software em produção.

• Monitoramento e Feedback: Permite a análise contínua do desempenho do software.

**Benefícios da Entrega Contínua**

•Redução do Tempo de Entrega: Acelera o ciclo de desenvolvimento.

• Melhoria da Qualidade: Identifica e corrige problemas mais rapidamente.

• Maior Flexibilidade: Adapta-se rapidamente às mudanças.

**Melhoria Contínua**

Este conceito busca aprimorar continuamente processos e práticas por meio de:

• Ciclo PDCA (Plan-Do-Check-Act): Estrutura para planejamento e implementação de melhorias.

• Feedback Contínuo: Coleta e análise de feedback para ajustes e melhorias.

• Análise de Métricas: Utiliza dados para identificar áreas de melhoria.

• Adaptação e Aprendizado: Ajusta processos com base em novas informações e resultados.

**Benefícios da Melhoria Contínua**

• Aumento da Eficiência: Melhora a produtividade e a eficácia.

• Maior Qualidade: Eleva o padrão dos produtos e serviços.

• Inovação e Adaptação: Estimula a inovação e a capacidade de adaptação.

**Tendências na Criação de Aplicações**

Desenvolvimento de Aplicações Mobile e Multiplataforma através de *frameworks* como *Flutter, React Native*e*Xamarin* permitindo o desenvolvimento de aplicações para várias plataformas a partir de um único código-fonte é uma das tendências de mercado que estão em alta.

*Progressive Web Apps (PWAs)* oferecem uma experiência de usuário similar à dos aplicativos nativos com as vantagens das aplicações web e são outro foco do mercado de trabalho hoje.

A experiência do usuário e design também são tendências que seguem em alta no mercado de tecnologia como um todo. Aqui, focamos em design responsivo para assegurar uma experiência consistente em diversos dispositivos e tamanhos de tela. Também focamos em design intuitivo onde as interfaces devem ser simples e intuitivas, promovendo um design centrado no usuário.

Outro ponto muito forte no mercado, e que seguirá assim por tempo indeterminado é a segurança e privacidade. A Autenticação Multifator (MFA) é aquela que adiciona camadas adicionais de segurança além da senha e precisa fazer parte da sua solução de sistemas e aplicações. A privacidade dos dados precisa estar em conformidade com regulamentações e práticas para proteger os dados dos usuários.

**Criação de Portfólio**

Um portfólio bem estruturado deve demonstrar suas habilidades técnicas e a qualidade do seu trabalho em Java. Ele deve incluir exemplos de projetos, descrições detalhadas e evidências das melhores práticas adotadas. Uma das formas mais simples de criar um portfólio, é utilizar o seu próprio perfil do GitHub.

**Próximos Passos na Criação de Aplicações Java**

• Planejamento e Design: Definição de requisitos, arquitetura de software e modelagem.

• Configuração do Ambiente: Preparação do ambiente de desenvolvimento e gestão de dependências.

• Desenvolvimento: Implementação, aplicação de boas práticas e versionamento.

• Testes: Execução de testes unitários, de integração e de aceitação.

• Documentação: Criação de documentação de código, de usuário e técnica.

• Monitoramento e Manutenção: Monitoramento contínuo, gestão de logs e manutenção do software.

Este módulo fornece uma visão abrangente das melhores práticas no desenvolvimento, enfatizando a importância do versionamento de código e práticas recomendadas em Java. Os conceitos de entrega e melhoria contínua são fundamentais para a eficiência e qualidade do desenvolvimento. Além disso, as tendências atuais e a criação de um portfólio são aspectos essenciais para o sucesso no mercado de trabalho.

**Conteúdo Bônus**

**Título:**[CODE] BOAS PRÁTICAS PARA DESENVOLVER UMA APLICAÇÃO

**Canal:** CORTES JAVA

**Plataforma:** YouTube

**Descrição:**Neste corte, Paula Santana compartilha algumas boas práticas essenciais durante o desenvolvimento de um software, focando em como garantir eficiência, organização e qualidade no código.

​

**Referências Bibliográficas**

ASCENCIA, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da programação**: algoritmos, Pascal, C/C++ e Java. 2. ed. Pearson, 2007.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. **Sistemas operacionais**. 3. ed. Pearson, 2005.

FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de programação**: a construção de algoritmos e estruturas de dados. 3. ed. Pearson, 2005.

GUEDES, S. (Org.). **Lógica de programação algorítmica**. Pearson, 2014.

LEE, V.; SCHNEIDER, H.; SCHELL, R. **Aplicações móveis**: arquitetura, projetos e desenvolvimento. Pearson, 2005.

MELO, A. C. V. de; SILVA, F. S. C. da.**Princípios de linguagens de programação**. Blucher, 2014.

MENEZES, A. M. de. **Os paradigmas de aprendizagem de algoritmo computacional**. Blucher, 2015.

PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados, com aplicações em Java**. Pearson, 2016.