Guia Ágil

TechGuide - Alura

Java

Nivel 1

- [ ] **Lógica de Programação**:
  - Aprender lógica de programação, fundamental para o desenvolvimento de software
  - Conhecer as bases para se criar, analisar e resolver problemas computacionais de forma estruturada e eficiente
  - Entender o que são tipos de dados
  - Declarar variáveis, considerando os diferentes tipos
  - Conhecer os operadores de atribuição e comparação
  - Usar estruturas condicionais
  - Usar estruturas de repetição e laços
  - Usar funções, passando parâmetros e argumentos
- [ ] **Java - Fundamentos**:
  - Java é uma linguagem de programação amplamente usada para codificar aplicações Web. Java é uma linguagem multiplataforma, orientada a objetos e centrada em rede que pode ser usada como uma plataforma em si. É uma linguagem de programação rápida, segura e confiável para codificar tudo, desde aplicações móveis e software empresarial até aplicações de big data e tecnologias do servidor.

- Conhecer os tipos primitivos
- Declarar variáveis, considerando os diferentes tipos
- Usar estruturas condicionais ('if', 'else')
- Conhecer os operadores de atribuição e comparação
- Usar estruturas de repetição e laços ('while', 'for')
- Usar funções, passando parâmetros e argumentos
- Manipular métodos
- Manipular arrays e listas
- Obter dados de uma API
- Criar construtores

☐ **Conceitos de Orientação a Objetos**:

- A Programação Orientada a Objetos é um paradigma de programação de software baseado na composição e interação entre diversas unidades chamadas de 'objetos' e as classes, que contêm uma identidade, propriedades e métodos). Ela é baseada em quatro componentes da programação: abstração digital, encapsulamento, herança e polimorfismo.
- Como funcionam objetos
- Criar e utilizar construtores
- O que são classes
- Criar e utilizar métodos
- Como funciona encapsulamento
- O que é herança
- O que é polimorfismo
- Como funcionam interfaces
- O que são abstrações

☐ **Java - Manipulação de Erros**:

- O tratamento de erros refere-se aos procedimentos de resposta e recuperação de condições de erro presentes em um aplicativo de software.

Em outras palavras, é o processo composto de antecipação, detecção e resolução de erros de aplicação, de programação ou de comunicação.

- Tratar exceções pré-definidas
- Uso de 'try' e 'catch'
- Criar exceções específicas
- Fazer o processo de Debug

- [ ] **Java - Coleções**:

- Uma coleção representa um grupo de objetos, conhecidos como seus elementos. Eles são como recipientes que agrupam vários itens em uma única unidade. Algumas coleções permitem a duplicação de elementos e outras não. Algumas são ordenadas e outras não ordenadas.
- Aprender os usos e diferenças entre List, Set e Map
- Aprender os usos e diferenças entre Equals e HashCode
- Saiba trabalhar com ArrayList, LinkedList ou Vector
- Classes Wrappers

- [ ] **Java - Testes**:

- O teste de software é o processo de avaliação e verificação de que um software realmente faz o que deveria fazer. Os benefícios dos testes incluem a prevenção de bugs, a redução dos custos de desenvolvimento e a melhoria do desempenho.
- Usar testes unitários
- Usar testes de integração
- Usar testes de comportamento (behavior)
- Usar mocks

- [ ] **Java - Pacotes**:

- Um pacote (package) em Java é usado para agrupar classes relacionadas, de forma semelhante a uma pasta em um diretório de arquivos. Os pacotes

são usados para evitar conflitos de nomes e para escrever um código de melhor manutenção.

- Use imports e organize o seu código através de packages
- Conhecer a java.lang
- Entender a imutabilidade e a classe String
- Entender a classe java.lang.Object
- Conhecer a java.io

- [ ] **Estruturas de Dados**:

- No contexto dos computadores, uma estrutura de dados é uma forma específica de armazenar e organizar os dados na memória do computador para que esses dados possam ser facilmente recuperados e utilizados de forma eficiente quando necessário posteriormente.
- Conhecer as principais estruturas de dados
- Implementar as principais estruturas de dados

Nivel 2

- [ ] **JVM**:

- Máquina virtual Java (em inglês, Java Virtual Machine, JVM) é um programa que carrega e executa os aplicativos Java, convertendo os bytecodes em código executável de máquina. A JVM é responsável pelo gerenciamento dos aplicativos, à medida que são executados. Graças à máquina virtual Java, os programas escritos em Java podem funcionar em qualquer plataforma de hardware e software que possua uma versão da JVM, tornando assim essas aplicações independentes da plataforma onde funcionam.
- Entender como funciona a máquina virtual do Java

- [ ] **Java - Gerenciamento da Memória**:

- Em Java, o gerenciamento de memória é o processo de alocação e desalocação de objetos, chamado de gerenciamento de memória. Java faz

o gerenciamento de memória automaticamente. Java usa um sistema de gerenciamento automático de memória chamado de Garbage Collector (coletor de lixo). Assim, não somos obrigados a implementar a lógica de gerenciamento de memória em nossa aplicação.

- Entender como funciona a memória e seu gerenciamento em Java
- Entender como funciona a memória o Garbage Collector

- [ ] **Spring Framework**:

- O Spring é um framework open source para a plataforma Java. Trata-se de um framework não intrusivo, baseado nos padrões de projeto (design patterns) de inversão de controle (IoC) e injeção de dependência. No Spring o contêiner se encarrega de "instanciar" classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do framework, o que é chamado de auto-wiring ou ainda anotações nas classes, métodos e propriedades. Dessa forma, o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.
- Entender o conceito de Injeção de Dependências
- Entender o padrão MVC
- Usar o Spring Data para manipular dados

- [ ] **Spring Boot**:

- O Spring Boot é um framework de código aberto baseado em Java usado para criar microserviços com o Spring Framework. Ele é usado para construir aplicações Spring independentes e prontas para produção.
- Criar aplicações Spring standalone
- Usar os servidores HTTP embutidos

- [ ] **Build tools Java**:

- Uma build tool é um sistema que permite automatizar todas as tarefas rotineiras de um projeto de uma forma organizada e que evite que o desenvolvedor tenha que perder tempo. Em outras palavras, adicionar uma nova biblioteca, realização de testes, empacotamento e deploy, ou até

mesmo, a compatibilidade entre as diversas IDEs são tarefas facilmente resolvidas com uma build tool.

- Conheça as principais ferramentas de build do Java, como Maven, Jenkins, Apache Ant, Gradle, etc., e como usá-las

- [ ] **Java - Persistência**:

- O conceito de "persistência de dados" refere-se a garantir que as informações inseridas na aplicação serão armazenadas em um meio em que possam ser recuperadas de forma consistente. Ou seja, são registros permanentes e que não são perdidos quando há o encerramento da sessão.
- Entender sobre JDBC e JPA
- Usar frameworks como Spring Data e Hibernate
- Comunicar-se com um banco de dados relacional
- Entender a diferença entre relacionamentos EAGER e LAZY
- Planejar queries com join fetch
- Encapsular o acesso em um DAO
- Entender como a memória funciona nessa situação

Nivel 3

- [ ] **Arquitetura de Microsserviços**:

- Microsserviços são uma abordagem de arquitetura na qual o software consiste de pequenos serviços independentes que se comunicam entre si e são organizados de acordo com seus domínios de negócio.
- Aprender o conceito de arquitetura planejada para microsserviços
- Realizar a comunicação usando APIs
- Melhorar a escalabilidade de um sistema

- [ ] **Java - Concorrência**:

- Programação concorrente é um paradigma de programação para a construção de programas que fazem uso da execução simultânea de várias

tarefas computacionais interativas, que podem ser implementadas como programas separados ou como um conjunto de threads criadas por um único programa.

- Executar tarefas simultaneamente
- Colocar tarefas para aguardar até que um determinado evento ocorra
- Entender como a memória funciona nessa situação

- [ ] **Contêineres**:

- Os contêineres são pacotes de software que contêm todos os elementos necessários para serem executados em qualquer ambiente. Gerenciamento de contêineres é uma área crucial na computação em nuvem e DevOps, que envolve o uso de tecnologias para automatizar o processo de criação, implantação, escalonamento e monitoramento de contêineres. Contêineres são unidades de software padronizadas que permitem aos desenvolvedores empacotar todas as dependências de um aplicativo (código, bibliotecas, configurações, etc.) em um único pacote. Isso permite que o aplicativo seja executado de forma consistente em qualquer ambiente de infraestrutura.
- A tecnologia de contêineres, como exemplificada pelo Docker, fornece um ambiente consistente e portátil para desenvolvimento, teste e implantação de aplicativos, o que é vital para o trabalho eficiente de engenharia de dados. Além disso, o Kubernetes, um sistema de orquestração de contêineres, permite o gerenciamento, a automação e a escalabilidade de aplicações baseadas em contêineres em ambientes de produção. Dominar esses conceitos e tecnologias possibilita a engenheiros de dados construir e manter pipelines de dados eficientes e confiáveis.
- O Kubernetes (também conhecido como k8s ou kube) é uma plataforma de orquestração de containers open source que automatiza grande parte dos processos manuais necessários para implantar, gerenciar e escalar aplicações em containers.
- Isolar seu software para funcionar independentemente
- Implantar software em clusters
- Modularizar seu sistema em pacotes menores
- Conhecer a plataforma Docker

- Conhecer Kubernetes

- [ ] **Kafka**:

- O Apache Kafka é uma plataforma distribuída de transmissão de dados que é capaz de publicar, subscrever, armazenar e processar fluxos de registro em tempo real. Essa plataforma foi desenvolvida para processar fluxos de dados provenientes de diversas fontes e entregá-los a vários clientes.
- Utilizar o Kafka para comunicação assíncrona
- Criar microsserviços com Kafka
- Criar produtores e consumidores
- Entender como usar o Kafka para paralelismo e execução serializada
- Obter garantias relativas ao envio ou entrega das mensagens

Habilidade Auxiliar: Infraestrutura

- [ ] **Git e GitHub - Fundamentos**:

- Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com tudo, desde projetos pequenos a muito grandes com velocidade e eficiência.
- GitHub é um serviço de hospedagem para desenvolvimento de software e controle de versão usando Git.
- Criar um repositório
- Clonar um repositório
- Fazer commit, push e pull de e para o repositório
- Reverter um commit
- Criar branches e pul requests
- Lidar com merge e conflitos

- [ ] **HTTP - Fundamentos**:

- HTTP significa Hyper Text Transfer Protocol. A comunicação entre computadores cliente e servidores web é feita enviando solicitações HTTP e

recebendo respostas HTTP.

- Entender a diferença dos verbos HTTP
- Testar os requests e ver os status codes no navegador
- Saber fazer uma requisição HTTP na linha de comando com WGET
- Baixar uma imagem com WGET
- Fazer um post

- [ ] **JSON**:

- JSON significa JavaScript Object Notation (notação de objeto JavaScript). É um formato de texto para armazenar e transmitir dados.
- Criar um objeto
- Transformar um objeto em uma string
- Transformar uma string em objeto
- Manipular um objeto

- [ ] **Linha de comando - Fundamentos**:

- CLI é um programa de linha de comando que aceita entradas de texto para executar funções do sistema operacional.
- Conhecer os principais comandos

- [ ] **Cloud - Fundamentos**:

- Cloud, ou computação em nuvem é a distribuição de serviços de computação pela Internet usando um modelo de preço pago conforme o uso. Uma nuvem é composta de vários recursos de computação, que abrangem desde os próprios computadores (ou instâncias, na terminologia de nuvem) até redes, armazenamento, bancos de dados e o que estiver em torno deles. Ou seja, tudo o que normalmente é necessário para montar o equivalente a uma sala de servidores, ou mesmo um data center completo, estará pronto para ser utilizado, configurado e executado.
- Conhecer a diferença entre IaaS, PaaS e SaaS
- Conhecer os maiores provedores de cloud

- Especializar-se em algum provedor

- [ ] **SQL - Fundamentos**:

- SQL (Structured Query Language, traduzindo, Linguagem de Consulta Estruturada) é uma linguagem de programação padronizada que é usada para gerenciar bancos de dados relacionais e realizar várias operações sobre os dados neles contidos.
- Conhecer os comandos mais comuns do SQL
- Usar SELECT para consultar uma tabela
- Usar INSERT para inserir dados em uma tabela
- Usar UPDATE para atualizar uma tabela
- Usar DELETE para remover dados de uma tabela
- Usar JOIN para conectar os dados de múltiplas tabelas
- Conhecer as cláusulas (FROM, ORDER BY, etc)

Habilidade Auxiliar: Boas práticas

- [ ] **SOLID**:

- O Solid possui cinco princípios considerados como boas práticas no desenvolvimento de software que ajudam os programadores a escrever os códigos mais limpos, separando as responsabilidades, diminuindo acoplamentos, facilitando na refatoração e estimulando o reaproveitamento do código.

- [ ] **Clean Architecture**:

- A Clean Architecture (Arquitetura Limpa) é uma forma de desenvolver software, de tal forma que apenas olhando para o código fonte de um programa, você deve ser capaz de dizer o que o programa faz.

- [ ] **Design Patterns**:

- Na engenharia de software, um "padrão de projeto" (Design Pattern em inglês) é uma solução geral e reutilizável para um problema que ocorre normalmente dentro de um determinado contexto de projeto de software.

- Conhecer e aplicar os principais Design Patterns

- [ ] **Clean Code**:

- Aplicar técnicas simples que visam facilitar a escrita e leitura de um código
- Refatorar seu código para que fique mais claro

- [ ] **Conceitos de Design Orientado a Domínio (Domain-Driven Design - DDD)**:

- O Design Orientado a Domínio (DDD) é uma abordagem ao projeto e desenvolvimento de software que é primeiramente informado pelos requisitos de negócios. Os componentes do programa (objetos, classes, matrizes, etc.) indicam a indústria, setor ou domínio empresarial em que o negócio opera.
- Modelar domínios de forma efetiva
- Basear projetos complexos em modelos do domínio
- Conhecer os blocos de construção de DDD
