## O que estudar para entrar na IBM?

Para se preparar para a entrevista de desenvolvedor Java pleno na IBM, é importante ter um conhecimento sólido sobre os seguintes tópicos:

1. Fundamentos de Java: certifique-se de ter uma boa compreensão dos fundamentos da linguagem Java, como classes, objetos, tipos de dados, estruturas de controle de fluxo, exceções, entrada/saída, etc.
2. Frameworks e bibliotecas Java: a IBM trabalha com vários frameworks e bibliotecas Java, como Spring, Hibernate, JPA, JMS, JDBC, JSP, Servlets, entre outros. Portanto, é importante que você tenha conhecimento desses frameworks e saiba como utilizá-los.
3. Arquitetura de software: a IBM valoriza a compreensão da arquitetura de software, incluindo design patterns, arquitetura de microsserviços, RESTful APIs, etc.
4. Bancos de dados: é importante ter conhecimento em bancos de dados relacionais e não-relacionais. Alguns bancos de dados comuns utilizados pela IBM incluem Oracle, DB2, PostgreSQL, MongoDB, entre outros.
5. Desenvolvimento ágil: a IBM adota práticas de desenvolvimento ágil, portanto, é importante ter conhecimento em metodologias ágeis, como Scrum, Kanban, XP, entre outras.
6. Ferramentas de desenvolvimento: a IBM utiliza várias ferramentas de desenvolvimento, como Git, JIRA, Jenkins, Eclipse, entre outras. É importante ter conhecimento em pelo menos algumas dessas ferramentas.
7. Boas práticas de codificação: a IBM valoriza a escrita de código limpo e eficiente, portanto, é importante estar familiarizado com as boas práticas de codificação, como SOLID, DRY, KISS, TDD, entre outras.

Além desses tópicos, a IBM também pode avaliar outras habilidades específicas para a vaga em questão, como experiência em desenvolvimento de aplicações móveis, conhecimento em inteligência artificial, machine learning, entre outros. Por isso, é importante estar atento às descrições da vaga e às informações fornecidas pelo recrutador durante o processo seletivo.

## Projetos para praticar:

Os seguintes projetos tem como intuito, evoluir minhas habilidades em java.

projeto banco

Crie um programa que simule um banco. O programa deve permitir que o usuário realize operações bancárias como depósito, saque e transferência, além de fornecer informações como saldo e extrato de conta.

Algumas especificações do projeto:

- Crie uma classe `Conta` que represente uma conta bancária. A classe deve ter as propriedades `numero` (int), `saldo` (double) e `titular` (String), além de métodos como `depositar`, `sacar` e `transferir`.
- Utilize a estrutura de controle de fluxo `if-else` para garantir que o usuário não possa sacar ou transferir um valor maior do que o seu saldo atual.
- Utilize a estrutura de controle de fluxo `switch-case` para permitir que o usuário selecione qual operação bancária ele deseja realizar (depósito, saque, transferência ou consulta de saldo/extrato).
- Utilize a estrutura de exceção `try-catch` para tratar erros que possam ocorrer durante as operações bancárias, como tentativa de sacar mais dinheiro do que o disponível na conta.
- Utilize a classe `Scanner` para receber entrada do usuário via console.
- Utilize a classe `System.out` para exibir informações sobre as operações bancárias realizadas pelo usuário.

Esse projeto permitirá que você pratique muitos dos fundamentos da linguagem Java, como classes, objetos, tipos de dados, estruturas de controle de fluxo, exceções, entrada/saída, etc. Além disso, ele é bastante prático e pode ser útil para você no seu dia a dia como desenvolvedor. Boa sorte!

Projeto gerenciador de tarefas:

Crie um sistema de gerenciamento de tarefas (task management system) que permita que usuários criem e gerenciem tarefas. Algumas especificações do projeto incluem:

- Utilize uma arquitetura de microsserviços para criar um sistema modular e escalável. Cada microsserviço pode ser responsável por uma funcionalidade específica, como autenticação de usuários, gerenciamento de tarefas, notificações, etc.
- Utilize RESTful APIs para criar uma interface de comunicação entre os diferentes microsserviços. Isso permitirá que os microsserviços se comuniquem de forma padronizada e independente da linguagem de programação ou plataforma utilizada.

- Utilize design patterns para criar uma arquitetura flexível e fácil de manter. Alguns padrões que podem ser úteis incluem o padrão Repository, para separar a lógica de acesso a dados do restante do código, e o padrão Observer, para implementar notificações de tarefas para os usuários.
- Utilize um banco de dados para armazenar informações sobre as tarefas e os usuários. É possível utilizar tanto bancos de dados relacionais quanto NoSQL, dependendo das necessidades do projeto.
- Implemente um sistema de autenticação de usuários para garantir a segurança das informações. Isso pode ser feito utilizando tokens JWT (JSON Web Tokens), por exemplo.
- Utilize tecnologias de front-end, como HTML, CSS e JavaScript, para criar uma interface de usuário (UI) intuitiva e fácil de usar. É possível utilizar frameworks como Angular, React ou Vue.js para acelerar o desenvolvimento da UI.

Esse projeto permitirá que você pratique vários aspectos da arquitetura de software, incluindo design patterns, arquitetura de microsserviços e RESTful APIs. Além disso, ele é bastante prático e pode ser útil para você no seu dia a dia como desenvolvedor. Boa sorte!

Projeto cadastro de clientes:

Crie uma aplicação de cadastro de clientes (customer registration system) que permita que usuários adicionem, removam e atualizem informações sobre clientes. Algumas especificações do projeto incluem:

- Utilize o princípio SOLID para criar um código modular, flexível e fácil de manter. Divida o código em classes e módulos responsáveis por funcionalidades específicas e utilize as melhores práticas de orientação a objetos para garantir uma boa organização e legibilidade do código.
- Utilize o princípio DRY (Don't Repeat Yourself) para evitar repetição de código. Crie funções e métodos reutilizáveis que possam ser chamados de diferentes partes do código, em vez de replicar o mesmo código várias vezes.
- Utilize o princípio KISS (Keep It Simple, Stupid) para manter o código simples e fácil de entender. Evite complexidade desnecessária e opte por soluções simples e eficazes sempre que possível.
- Utilize TDD (Test-Driven Development) para criar testes automatizados para cada parte do código. Isso permitirá que você valide e verifique as funcionalidades da aplicação em diferentes cenários e evite erros e bugs no código.

- Utilize boas práticas de versionamento de código, como Git, para gerenciar e controlar o histórico do código. Crie ramificações (branches) específicas para cada nova funcionalidade ou correção de bug e utilize pull requests para revisão do código por outros desenvolvedores.
- Utilize boas práticas de documentação de código, incluindo comentários claros e concisos em cada parte do código, para facilitar a compreensão e manutenção do código no futuro.

Esse projeto permitirá que você pratique boas práticas de codificação, como SOLID, DRY, KISS e TDD. Além disso, ele é bastante prático e pode ser útil para você no seu dia a dia como desenvolvedor. Boa sorte!

## Lista de Projetos a serem concluídos:

Os seguintes projetos tem como intuito, desenvolver um portfólio com os requisitos mínimos para atuar como desenvolvedor na IBM.

1. Crie uma aplicação web utilizando o framework Spring Boot:
    - Implemente um sistema de cadastro de produtos com as funcionalidades de listar, adicionar, atualizar e excluir produtos.
    - Adicione autenticação de usuários utilizando Spring Security.
    - Utilize o banco de dados relacional MySQL para armazenar os dados.
2. Implemente um algoritmo de ordenação de dados:
    - Crie um programa que recebe um conjunto de números aleatórios e os ordena utilizando um dos algoritmos de ordenação disponíveis (ex: Bubble Sort, Insertion Sort, Quick Sort, etc).
    - Implementar testes unitários para garantir que o algoritmo está ordenando corretamente.
3. Desenvolva um sistema de gerenciamento de projetos:
    - Crie um sistema que permita a criação de projetos com suas respectivas tarefas e prazos de entrega.
    - Implemente um sistema de notificações que avisa os usuários quando uma tarefa está atrasada ou próxima do prazo de entrega.
    - Utilize um banco de dados relacional para armazenar os dados.
4. Desenvolva um aplicativo de gerenciamento de finanças pessoais:
    - Crie um aplicativo que permita aos usuários registrar suas despesas e receitas.
    - Implemente gráficos e relatórios que mostrem o balanço financeiro dos usuários.

- Utilize o banco de dados não relacional MongoDB para armazenar os dados.

5. Crie uma aplicação web utilizando o framework Angular:
    - Implemente um sistema de cadastro de usuários com as funcionalidades de listar, adicionar, atualizar e excluir usuários.
    - Utilize o serviço HttpClient para fazer requisições HTTP para o backend.
    - Utilize o Firebase Authentication para autenticação de usuários.

Essas são apenas algumas ideias de projetos que podem ajudá-lo a praticar e se preparar para uma entrevista de desenvolvedor Java Pleno na IBM. Lembre-se de sempre estar atualizado sobre as melhores práticas de desenvolvimento, padrões de projeto e tecnologias utilizadas pela empresa. Boa sorte!