# Go

TechGuide - Alura, FIAP e PM3

## Go

### 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
- [ ] **Go (GoLang) - Fundamentos**:
  - Go é uma linguagem de programação de alto nível compilada e estaticamente tipada, desenvolvida no Google. Ela é sintaticamente semelhante ao C, mas também tem memory safety, coletor de lixo, tipagem estrutural e concorrência.
  - Entender a sua sintaxe
  - Conhecer os tipos primitivos

- Declarar e usar variáveis e constantes
- Usar estruturas condicionais (if, else)
- Usar estruturas de repetição e laços (while, for)
- Usar funções, passando parâmetros e argumentos
- Implementando métodos e reutilizando eles
- Tratamento de erros
- Convenções de código
- Manipular Coleções, arrays e listas

- [ ] **Go - Orientação a Objetos**:

- Entender como métodos são definidos e associados a tipos em Go.
- Aprender a diferenciar métodos e funções dentro do contexto de Go.
- Explorar a aplicação prática de métodos para manipulação de estruturas e interfaces.

- [ ] **Go - Estruturas**:

- Aprender a definir e utilizar estruturas em Go.
- Entender como estruturas podem ser utilizadas para modelar dados e objetos.
- Explorar a relação entre estruturas e métodos para simular a orientação a objetos.

- [ ] **Go - Tratamento de Erros**:

- Compreender como Go trata erros através de seus mecanismos integrados.
- Aprender a utilizar o tipo `error` para reportar e tratar falhas.
- Explorar padrões de tratamento de erro para manter o código limpo e eficiente.

## Nivel 2

- [ ] **Go - Ponteiros**:

- Entender o conceito de ponteiros e sua sintaxe em Go.
- Aprender a usar ponteiros para manipulação direta de memória e passagem por referência.
- Explorar técnicas avançadas de ponteiros, incluindo ponteiros para estruturas.

- [ ] **Go - Interfaces**:
  - Compreender o papel das interfaces em Go e como elas contribuem para o polimorfismo.
  - Aprender a definir e implementar interfaces, facilitando a interação entre diferentes tipos.
  - Explorar o uso de interfaces para desacoplamento e testabilidade do código.
- [ ] **Go - Concorrência**:
  - Entender o modelo de concorrência em Go e como ele difere de outros modelos de programação paralela.
  - Aprender a criar e gerenciar goroutines para tarefas concorrentes.
  - Utilizar canais para a comunicação segura entre goroutines.
- [ ] **Go - Pacotes**:
  - Compreender a estrutura e o propósito dos pacotes em Go.
  - Aprender a criar, importar e exportar pacotes para modularizar o código.
  - Explorar o gerenciamento de dependências utilizando o Go Modules.
- [ ] **Go - Testes**:
  - Aprender a escrever testes unitários e de integração em Go.
  - Entender a importância dos testes para manter a qualidade do código.
  - Utilizar ferramentas e frameworks de teste disponíveis na comunidade Go.
- [ ] **Go - Persistência de Dados**:
  - Aprender os fundamentos da persistência de dados em Go.
  - Explorar a integração de Go com bancos de dados SQL e NoSQL.

- Entender melhores práticas para modelagem e acesso a dados em aplicações Go.

- [ ] **Go - Goroutines e Canais**:

- Entender o modelo de concorrência baseado em goroutines e canais em Go.
- Aprender a criar e sincronizar goroutines para tarefas concorrentes.
- Explorar padrões de uso de canais para comunicação segura entre goroutines.

- [ ] **Go - Programação Web**:

- Aprender a construir servidores web e APIs RESTful com Go.
- Explorar o ecossistema de frameworks e bibliotecas web disponíveis para Go.
- Entender práticas recomendadas para o desenvolvimento web seguro e eficiente com Go.

## Nivel 3

- [ ] **Go - Construindo APIs REST**:

- Compreender os princípios básicos de APIs RESTful e sua implementação em Go.
- Aprender a estruturar uma aplicação Go para suportar o desenvolvimento de APIs.
- Explorar ferramentas e bibliotecas úteis para facilitar o desenvolvimento de APIs em Go.

- [ ] **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

- [ ] **Go - Debugging**:
  - Aprender a utilizar as ferramentas de debugging disponíveis para Go.
  - Compreender técnicas eficazes para identificar e resolver bugs em Go.
  - Explorar melhores práticas para prevenir erros comuns em desenvolvimento Go.
- [ ] **Streaming**:
  - Streaming é o processamento e transmissão contínua de dados em tempo real, à medida que são gerados. Diferentemente do armazenamento tradicional, onde os dados são coletados e processados posteriormente, o streaming permite a análise e o uso imediato dos dados à medida que são recebidos. Essa abordagem é ideal para lidar com grandes volumes de dados em alta velocidade, permitindo a detecção de padrões em tempo real, tomada de decisões instantâneas e resposta rápida a eventos em andamento.
  - O streaming desempenha um papel essencial na engenharia de dados, capacitando as organizações a processar, analisar e tomar decisões com base em dados em tempo real. Isso resulta em maior agilidade, escalabilidade e reatividade, possibilitando uma tomada de decisões mais informada e orientada por insights atualizados.
- [ ] **Go - gRPC**:
  - Compreender os fundamentos e a arquitetura do gRPC e como ele se integra ao ecossistema Go.
  - Aprender a definir serviços gRPC usando protocol buffers, facilitando a interoperabilidade e a eficiência na comunicação entre serviços.
  - Desenvolver servidores e clientes gRPC em Go, explorando recursos avançados como streaming, autenticação e interceptores.
  - Adotar melhores práticas para o desenvolvimento de APIs gRPC em Go, garantindo segurança, performance e escalabilidade.
- [ ] **Go - Melhores Práticas de Segurança**:

- Compreender as vulnerabilidades comuns em aplicações Go e como mitigá-las.
- Aprender a implementar técnicas de segurança, como autenticação e criptografia.
- Explorar ferramentas e bibliotecas de segurança disponíveis para Go.

- [ ] **Go - Otimização de Performance**:

- Aprender a medir e analisar a performance de aplicações Go.
- Identificar gargalos de performance e aplicar técnicas de otimização.
- Explorar ferramentas e práticas para escrever código Go de alta performance.

## Habilidade Auxiliar: DevOps

- [ ] **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

☐ **Entrega e integração contínuas (CI/CD)**:

- CI/CD é a abreviação de Continuous Integration/Continuous Delivery, traduzindo para o português "entrega e integração contínuas". Trata-se de uma prática de desenvolvimento de software que visa tornar a integração de código mais eficiente por meio de builds e testes automatizados.
- Automatizar a integração de código entre varias partes da equipe se tornou cada vez mais importante, ja que assim é possivel acelerar o desenvolvimento e diminiur o tempo de entrega de software.
- Executar testes automatizados da aplicação para verificar seu funcionamento.
- Realizar a entrega de atualizações de forma automatica e com segurança.
- Realizar testes de conexão e testes de carga para evitar que a aplicação apresente problemas ao ser atualizada.

## Habilidade Auxiliar: Best Practices

☐ **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.

☐ **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

- [ ] **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

- [ ] **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

☐ **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)

☐ **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

- [ ] **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

- [ ] **Infraestrutura como código (IaC)**:

- Infraestrutura como código (IaC) é uma abordagem em que a infraestrutura de TI, incluindo servidores, redes e recursos relacionados, é gerenciada e provisionada por meio de código. Em vez de configurar manualmente a infraestrutura, o IaC utiliza arquivos de configuração ou scripts que descrevem a infraestrutura desejada de forma automatizada e reproduzível.
- A infraestrutura como código é de extrema importância na engenharia de dados, pois permite uma gestão eficiente e escalável dos recursos de infraestrutura necessários para processamento e armazenamento de dados. Com o IaC, é possível definir, provisionar e configurar ambientes de maneira consistente, rápida e controlada. Isso facilita a implantação e manutenção de pipelines de dados, ambientes de desenvolvimento e testes, clusters de processamento distribuído e outros componentes necessários para a engenharia de dados. Além disso, o IaC permite a automação e o

versionamento da infraestrutura, proporcionando maior agilidade, rastreabilidade e garantia de qualidade no gerenciamento dos recursos de TI.

---