

Kotlin

TechGuide - Alura, FIAP e PM3

Kotlin

Nível 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

☐ Kotlin - Fundamentos:

- Kotlin é uma linguagem de programação multiplataforma, orientada a objetos, funcional e estaticamente tipada. Ela compila para a máquina virtual Java e também pode ser traduzida para a linguagem JavaScript e compilada para código nativo (via LLVM). É a linguagem oficial do sistema Android da Google.
- 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
- Null Safety (Elimine o perigo de referências nulas)
- Exceptions e Throwables
- Convenções de código
- Manipular Coleções, arrays e listas
- Orientação a Objetos com Kotlin (Properties, Data class, Companion Objects, Delegation)
- Receber dados de uma API

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

☐ 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

☐ Kotlin - 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

☐ Kotlin - Collections:

- 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 read only e mutable

☐ Kotlin - Extensões:

- Como criar e usar extensões simples para melhorar a compreensão dos fundamentos da linguagem.

Nível 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

☐ Kotlin - Gerenciamento de memória:

- O gerenciamento de memória é o processo de alocação e desalocação de objetos, chamado de gerenciamento de memória. Kotlin faz o gerenciamento de memória automaticamente. Kotlin 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 Kotlin
- 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

☐ **Kotlin - Coroutines:**

- A programação assíncrona ou sem bloqueio é uma parte importante do cenário de desenvolvimento. Ao criar aplicativos do lado do servidor, desktop ou dispositivos móveis, é importante fornecer uma experiência que não seja apenas fluida do ponto de vista do usuário, mas também escalável quando necessário.

☐ **Kotlin - Multiplatform:**

- Como o Kotlin pode ser usado em várias plataformas, como JVM, JavaScript e código nativo.

Nível 3

☐ **Microserviços:**

- Aprender o conceito de arquitetura planejada para microserviços
- Realizar a comunicação usando APIs
- Melhorar a escalabilidade de um sistema

☐ **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

☐ **Kotlin - Type Safe Builders:**

- Criar construtores type-safe usando recursos avançados de Kotlin, para construir DSLs (Domain-Specific Languages) personalizadas.

☐ **Kotlin - Scope Functions:**

- Funções de escopo (let, run, with, apply, also) e como elas podem ser usadas para simplificar o código e melhorar a legibilidade.

☐ **Kotlin - Date Serialization Library:**

- Como trabalhar com bibliotecas de data e serialização em Kotlin para manipular datas e objetos serializados.

☐ **KDoc:**

- Mostrar a documentação KDoc e como escrever documentação de código em Kotlin.

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 pull 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

☐ **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

☐ **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

Habilidade Auxiliar: Front-end

☐ HTML - Fundamentos:

- HTML é uma linguagem de marcação que define a estrutura do seu conteúdo. HTML consiste em uma série de elementos que você usa para mostrar algo de uma determinada maneira ou agir de uma certo modo. As tags podem criar um hiperlink de uma palavra ou imagem para outro lugar, podem colocar palavras em itálico, podem aumentar ou diminuir a fonte e assim por diante.
- Aprender quais tags são necessárias para um HTML básico
- Criar um parágrafo de texto
- Exibir uma imagem
- Conhecer a diferença entre 'h1', 'h2', 'h3', etc
- Criar um texto com hyperlink
- Criar um formulário com campos relevantes
- Criar uma lista de itens ordenada ou não ordenada
- Criar uma lista de itens dentro de uma lista suspensa (dropdown list)
- Conectar com um arquivo de CSS
- Criar uma tabela
- Adicionar IDs e classes

☐ JavaScript - Fundamentos:

- JavaScript é a linguagem de programação mais popular do mundo e é uma das principais tecnologias da World Wide Web, juntamente com HTML e CSS. Ela possui tipagem dinâmica, orientação a objetos baseada em protótipos e funções de primeira classe. Ela é multi-paradigma e suporta estilos de programação orientados a eventos, funcionais e imperativos.
- Conhecer os tipos primitivos
- Declarar variáveis, considerando a diferença entre 'var', 'let' e 'const'
- 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 arrays e listas
- Aprender o conceito de Orientação a Objetos
- Fazer um CRUD
- Obter dados de uma API
- Fazer chamadas assíncronas usando 'Async/Await', 'Promise', etc

☐ CSS - Fundamentos:

- Cascading Style Sheets (CSS) é uma linguagem usada para descrever a apresentação de um documento escrito em uma linguagem de marcação como HTML ou XML. CSS pode ser usado para estilos de texto de documentos muito básicos — por exemplo, para alterar a cor e o tamanho de títulos e links. Ele pode ser usado para criar um layout — por exemplo, transformar uma única coluna de texto em um layout com uma área de conteúdo principal e uma barra lateral para informações relacionadas. Pode até ser usado para efeitos como animações.
- Aprender a estrutura visual de uma página, com 'margin' e 'padding'
- Estabelecer o tamanho com 'width' e 'height'
- Aprender sobre a posição de um elemento ('static', 'relative' ou 'absolute')
- Aprender sobre o 'display' de exibição de um elemento ('block', 'inline', 'inline-block')
- Aprender a posicionar imagens em relação ao texto
- Aprender sobre alinhamento
- Aprender sobre estilo de fontes
- Aprender as diferenças e vantagens de usar as diferentes unidades de medida em CSS (% , relativas, etc)
- Conectar com os elementos (IDs, classes) de um arquivo HTML

- Alterar características de um elemento quando o mouse passar por cima dele ('hover')
- Aprender box-sizing
- Aprender Flexbox
- Aprender Grid

TechGuide - Alura
Alura, PM3 e FIAP
O Techguide.sh é um projeto open source