# Guia Ágil

TechGuide - Alura

## Java

### Nivel 1

- [ ] **Java - Fundamentos**:

- Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Java es un lenguaje multiplataforma, orientado a objetos y centrado en red que se puede utilizar como una plataforma en sí. Es un lenguaje de programación rápido, seguro y confiable para codificar todo, desde aplicaciones móviles y software empresarial hasta aplicaciones de big data y tecnologías de servidor.
- Conocer los tipos primitivos.
- Declarar variables, considerando los diferentes tipos.
- Usar estructuras condicionales ('if', 'Else').
- Conocer a los operadores de comparación.
- Utilizar estructuras de repetición y bucles ('while', 'for').
- Usar funciones, pasar parámetros y argumentos.
- Manipular métodos.
- Manipular arrays y listas.
- Obtener datos de una API.
- Hacer llamadas asíncronas 'Future', etc.
- Crear constructores.

- [ ] **Conceptos de Orientación a Objetos**:
    - La Programación Orientada a Objetos es un paradigma de programación de software basado en la composición e interacción entre diversas unidades llamadas 'objetos' y las clases, que contienen una identidad, propiedades y métodos. Se basa en cuatro componentes de la programación - abstracción digital, encapsulación, herencia y polimorfismo.
    - Cómo funcionan los objetos
    - Crear y utilizar constructores
    - Qué son las clases
    - Crear y utilizar métodos
    - Cómo funciona la encapsulación
    - Qué es la herencia
    - Qué es el polimorfismo
    - Cómo funcionan las interfaces
    - Qué son las abstracciones
- [ ] **Java - Manejo de Errores**:
    - El tratamiento de errores se refiere a los procedimientos de respuesta y recuperación de condiciones de error presentes en una aplicación de software. En otras palabras, es el proceso compuesto de anticipación, detección y resolución de errores de aplicación, de programación o de comunicación.
    - Tratamiento de excepciones predefinidas
    - Uso de 'Try' y 'catch'
    - Crear excepciones específicas
    - Hacer el proceso de depuración
- [ ] **Java - Colecciones**:
    - Una colección representa un grupo de objetos, conocidos como sus elementos. Son como contenedores que agrupan varios elementos en una

sola unidad. Algunas colecciones permiten la duplicación de elementos y otras no. Algunas son ordenadas y otras no ordenadas.

- Aprender los usos y diferencias entre List, Set y Map
- Aprender los usos y diferencias entre Equals y hashcode
- Aprende a trabajar con ArrayList, LinkedList o Vector

- [ ] **Java - Pruebas**:

- La prueba de software es el proceso de evaluación y verificación de que un software realmente hace lo que debe hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento.
- Usar pruebas unitarias
- Usar pruebas de integración
- Usar pruebas de comportamiento (behavior)
- Usar mocks

- [ ] **Java - Paquetes**:

- Un paquete (package) en Java se utiliza para agrupar clases relacionadas, de forma similar a una carpeta en un directorio de archivos. Los paquetes se utilizan para evitar conflictos de nombres y para escribir un código de mejor mantenimiento.
- Utilice Imports y organice su código a través de paquetes (packages)
- Conocer la java.lang
- Comprender la inmutabilidad y la clase String
- Entender la clase java.lang.Object
- Conocer la java.io

- [ ] **Estructura de Datos**:

- En el contexto de los ordenadores, una estructura de datos es una forma específica de almacenar y organizar los datos en la memoria del ordenador

para que esos datos puedan ser fácilmente recuperados y utilizados de forma eficiente cuando sea necesario posteriormente.

- Conocer las principales estructuras de datos
- Implementar las principales estructuras de datos

## Nivel 2

- [ ] **JVM**:

- Máquina virtual Java (en inglés, Java Virtual Machine, JVM) es un programa que carga y ejecuta las aplicaciones Java, convirtiendo los bytecodes en código ejecutable de máquina. JVM es responsable de administrar las aplicaciones a medida que se ejecutan. Gracias a la máquina virtual Java, los programas escritos en Java pueden funcionar en cualquier plataforma de hardware y software que tenga una versión de JVM, haciendo así que estas aplicaciones sean independientes de la plataforma donde funcionan.
- Entender cómo funciona la máquina virtual de Java.

- [ ] **Java - Administración de Memoria**:

- En Java, la administración de memoria es el proceso de asignación y desalojo de objetos, llamado administración de memoria. Java administra la memoria automáticamente. Java utiliza un sistema de gestión automática de memoria llamado Garbage Collector (colector de basura). Por lo tanto, no estamos obligados a implementar la lógica de gestión de memoria en nuestra aplicación.
- Entender cómo funciona la memoria y su gestión en Java.
- Entender cómo funciona la memoria de Garbage Collector.

- [ ] **Spring Framework**:

- Spring es un framework de código abierto para la plataforma Java. Se trata de un marco no intrusivo, basado en los estándares de diseño (design patterns) de inversión de control (ioc) e inyección de dependencia. En Spring el contenedor se encarga de "instanciar" clases de una aplicación Java y definir las dependencias entre ellas a través de un archivo de

configuración en formato XML, inferencias del framework, lo que es llamado de auto-wiring o incluso anotaciones en las clases, métodos y propiedades. De esta forma, Spring permite el bajo acoplamiento entre clases de una aplicación orientada a objetos.

- Entender el concepto de inyección de dependencias
- Entender el patrón MVC
- Usar Spring Data para manipular datos

- [ ] **Spring Boot**:

- Spring Boot es un framework de código abierto basado en Java que se utiliza para crear Microservicios con Spring Framework. Se utiliza para construir aplicaciones de Spring independientes y listas para la producción.
- Crear aplicaciones independientes de Spring
- Usar los servidores HTTP integrados

- [ ] **Builds Tools Java**:

- Un build tool es un sistema que permite automatizar todas las tareas rutinarias de un proyecto de una manera organizada y que evite que el desarrollador tenga que perder tiempo. En otras palabras, agregar una nueva biblioteca, realizar pruebas, empaquetar y desplegar, o incluso, la compatibilidad entre varios IDEs son tareas fácilmente resueltas con una herramienta de construcción.
- Descubre las principales herramientas de construcción de Java, como Maven, Jenkins, Apache Ant, Gradle, etc., y cómo usarlas.

- [ ] **Java - Persistencia**:

- El concepto de "persistencia de datos" se refiere a garantizar que la información introducida en la aplicación se almacena en un medio en el que se puede recuperar de forma consistente. Es decir, son registros permanentes y que no se pierden cuando se cierra la sesión.
- Entender sobre JDBC y JPA
- Usar frameworks como Spring Data e Hibernate

- Comunicarse con una base de datos relacional
- Entender la diferencia entre las relaciones EAGER y LAZY
- Planificar consultas con Join fetch
- Encapsular el acceso en un DAO (Objeto de Acceso a Datos, en inglés Data Access Objeto)
- Entender cómo funciona la memoria en esta situación

## Nivel 3

- [ ] **Arquitectura de Microservicios**:

- Los microservicios son un enfoque de arquitectura en el que el software consiste en pequeños servicios independientes que se comunican entre sí y se organizan de acuerdo con sus dominios de negocio.
- Aprender el concepto de arquitectura diseñada para microservicios
- Realizar la comunicación mediante API
- Mejorar la escalabilidad de un sistema

- [ ] **Java - Concurrencia**:

- Programación concurrente es un paradigma de programación para la construcción de programas que hacen uso de la ejecución simultánea de varias tareas computacionales interactivas, que pueden ser implementadas como programas separados o como un conjunto de hilos creados por un único programa.
- Realizar tareas simultáneamente.
- Poner tareas a esperar hasta que ocurra un evento determinado.
- Entender cómo funciona la memoria en esta situación.

- [ ] **Contenedores**:

- Los contenedores son paquetes de software que contienen todos los elementos necesarios para ejecutarse en cualquier entorno. La gestión de contenedores es un área crucial en la computación en nube y DevOps, que implica el uso de tecnologías para automatizar el proceso de creación,

implementación, escalado y monitoreo de contenedores. Los contenedores son unidades de software estandarizadas que permiten a los desarrolladores empaquetar todas las dependencias de una aplicación (código, bibliotecas, configuraciones, etc.) en un solo paquete. Esto permite que la aplicación se ejecute de forma consistente en cualquier entorno de infraestructura.

- La tecnología de contenedores, como ejemplifica Docker, proporciona un entorno coherente y portátil para el desarrollo, las pruebas y la implementación de aplicaciones, lo que es vital para el trabajo eficiente de la ingeniería de datos. Además, Kubernetes, un sistema de organización de contenedores, permite la gestión, automatización y escalabilidad de aplicaciones basadas en contenedores en entornos de producción. Dominar estos conceptos y tecnologías permite a los ingenieros de datos construir y mantener canalizaciones de datos eficientes y confiables.
- Kubernetes (también conocido como k8s o Kube) es una plataforma de orquestación de contenedores de código abierto que automatiza gran parte de los procesos manuales necesarios para implementar, gestionar y escalar aplicaciones en contenedores.
- Aislar el software para que funcione independientemente
- Implementación de software en clústeres
- Modularizar su sistema en paquetes más pequeños
- Conocer la plataforma Docker
- Conocer Kubernetes

- [ ] **Java - Kafka**:

- Apache Kafka es una plataforma de transmisión de datos distribuida que es capaz de publicar, suscribir, almacenar y procesar flujos de registro en tiempo real. Esta plataforma está diseñada para procesar flujos de datos provenientes de diversas fuentes y entregarlos a varios clientes.
- Utilizar Kafka para la comunicación asíncrona
- Crear microservicios con Kafka
- Crear productores y consumidores

- Entender cómo usar Kafka para paralelismo y ejecución serializada
- Obtener garantías relativas al envío o entrega de los mensajes

## Habilidade Auxiliar: Infraestructura

- [ ] **Git y GitHub - Fundamentos**:
  - Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia.
  - GitHub es un servicio de hosting para el desarrollo de software y el control de versiones mediante Git.
  - Crear un repositorio
  - Clonar un repositorio
  - Comprometerse, empujar y tirar hacia y desde el repositorio
  - Revertir un commit
  - Crear de ramas y Pull requests
  - Manejar fusiones y conflictos
- [ ] **HTTP - Fundamentos**:
  - HTTP significa Protocolo de transferencia de hipertexto. La comunicación entre las computadoras cliente y los servidores web se realiza mediante el envío de solicitudes HTTP y la recepción de respuestas HTTP.
  - Comprender la diferencia entre los verbos HTTP
  - Probar solicitudes y verificar los códigos de estado en el navegador
  - Aprendiendo a hacer una solicitud HTTP en la línea de comando con WGET
  - Descargar una imagen con WGET
  - Realización de una POST
- [ ] **JSON**:
  - JSON significa Notación de objetos de JavaScript. Es un formato de texto para almacenar y transportar datos.

- Crear un objeto
- Transformar un objeto en una cadena
- Transformar una cadena en un objeto
- Manipular un objeto

- [ ] **Línea de Comando - Fundamentos**:

- CLI es un programa de línea de comandos que acepta la entrada de texto para ejecutar funciones del sistema operativo.
- Conocer los comandos más importantes

- [ ] **Cloud - Fundamentos**:

- La computación en nube, o cloud computing, es la distribución de servicios informáticos a través de Internet mediante un modelo de tarificación de pago por uso. Una nube se compone de varios recursos informatizados, desde los propios ordenadores (o instancias, en terminología de nube) hasta las redes, el almacenamiento, las bases de datos y todo lo que les rodea. En otras palabras, todo lo que normalmente se necesita para montar el equivalente a una sala de servidores, o incluso un centro de datos completo, estará listo para usar, configurar y ejecutar.
- Conocer la diferencia entre IaaS, PaaS y SaaS
- Conocer los mayores proveedores de nube
- Especializarse en un proveedor específico de su preferencia

- [ ] **SQL - Fundamentos**:

- Conocer los comandos más comunes de SQL
- Usar SELECT para consultar una tabla
- Usar INSERT para insertar datos en una tabla
- Usar UPDATE para actualizar una tabla
- Usar DELETE para eliminar datos de una tabla
- Usar JOIN para conectar los datos de múltiples tablas
- Conocer las cláusulas (FROM, ORDER BY, etc.)

## Habilidade Auxiliar: Buenas prácticas

- [ ] **SOLID**:
  - Solid tiene cinco principios considerados como buenas prácticas en el desarrollo de software que ayudan a los programadores a escribir los códigos más limpios, dividiendo las responsabilidades, disminuyendo los acoplamientos, facilitando la refactorización y estimulando el reaprovechamiento del código. Propuesto por Robert C. Martin, SOLID propicia el desarrollo de un código limpio, legible y comprobable.
- [ ] **Clean Architecture**:
  - Clean Architecture (Arquitectura Limpia) es una forma de desarrollar software, de tal forma que solo mirando el código fuente de un programa, debes ser capaz de decir lo que el programa hace.
- [ ] **Design Patterns**:
  - En ingeniería de software, un "patrón de diseño" (Design Pattern en inglés) es una solución general y reutilizable para un problema que ocurre normalmente dentro de un determinado contexto de diseño de software.
  - Conocer y aplicar los principales patrones de diseño.
- [ ] **Clean Code**:
  - Aplicar técnicas sencillas para facilitar la escritura y la lectura de un código.
  - Refactorizar el código para que quede más limpio.
- [ ] **Diseño Orientado a Dominio - DDD**:
  - El Diseño Orientado a Dominio (DDD) es un enfoque de diseño y desarrollo de software que se informa principalmente por los requisitos de negocio. Los componentes del programa (objetos, clases, matrices, etc.) indican la industria, sector o dominio empresarial en que opera el negocio.
  - Modelar dominios de manera efectiva.
  - Basar proyectos complejos en modelos de dominio.
  - Conocer los bloques de construcción de DDD.