Guia Ágil

TechGuide - Alura

C#

Nivel 1

- [ ] **C# - Fundamentos**:

- C# es un lenguaje de programación, multiparadigma, con tipado fuerte, desarrollado por Microsoft como parte de la plataforma .NET. El código fuente se compila a Common Intermediate Language (CIL), que es interpretado por la máquina virtual Common Language Runtime (CLR). Está diseñado para funcionar en la Common Language Infrastructure de la plataforma .NET Framework.
- Conocer los tipos primitivos
- Declarar variables, considerando los diferentes tipos
- Utilizar estructuras condicionales ('if', 'else')
- Conocer los operadores de asignación y comparación
- Utilizar estructuras de repetición y bucles ('while', 'for')
- Utilizar funciones, pasando parámetros y argumentos
- Manejar métodos
- Manejar arrays y listas
- Obtener datos de una API
- Realizar llamadas asíncronas, 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
- [ ] **C# - 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 Dictionary, List, Queue, SortedList y Stack
  - Aprende a trabajar con ArrayList y HashTable
  - Iterators
- [ ] **NuGet**:
  - NuGet es un gestor de paquetes para la plataforma .NET. Define cómo se crean, publican y consumen los paquetes en esta plataforma, y proporciona herramientas para cada una de estas funciones.
  - Gestión de paquetes

- Compartir bibliotecas

- [ ] **C# - System.IO**:

- El espacio de nombres (namespace) System.IO consiste en clases, estructuras, delegates y enumeraciones relacionadas con la entrada y salida de datos (IO). Estas clases pueden ser utilizadas para leer y escribir datos en archivos o streams de datos. También contiene clases de soporte para archivos y directorios.
- Leer datos de archivos
- Escribir datos en archivos
- Gestionar archivos con el uso de Using

- [ ] **C# - Administración de Memoria**:

- El manejo automático de memoria es uno de los servicios que el Common Language Runtime ofrece durante la Ejecución Administrada (Managed Execution). El recolector de basura (Garbage Collector) del Common Language Runtime gestiona la asignación y liberación de memoria para una aplicación.
- Comprender cómo se administra la memoria
- Conocer el Recolector de Basura (Garbage Collector)
- Entender el Stack y el Heap administrado

- [ ] **C# - 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 unitárias
- Usar pruebas de integración
- Usar pruebas de comportamiento (behavior)
- Usar mocks

- [ ] **ADO.NET**:

- ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para desarrolladores del .NET Framework. ADO.NET provee un conjunto rico de componentes para crear aplicaciones distribuidas y compartir datos. Forma parte del .NET Framework, proporcionando acceso a datos de aplicaciones relacionales y XML.
- Manipular bases de datos
- Conocer el DataSet y DataTable
- Realizar conexiones
- Manipular documentos XML

- [ ] **Entity Framework Core**:

- Entity Framework Core es un mapeador objeto-relacional (ORM). El mapeo objeto-relacional es una técnica que permite a los desarrolladores trabajar con datos de manera orientada a objetos, realizando el trabajo necesario para mapear entre objetos definidos en una lenguaje de programación de la aplicación y los datos almacenados en fuentes de datos relacionales.
- Manipular bases de datos usando objetos .NET
- Crear modelos correspondientes a una base de datos
- Realizar consultas

- [ ] **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

- [ ] **CLR**:

- Common Language Runtime (CLR) es el componente de máquina virtual de la plataforma .NET de Microsoft que administra la ejecución de programas .NET.
- Entender cómo funciona CLR
- Entender la gestión de memoria
- Conocer la CIL y el JIT

- [ ] **LINQ**:

- LINQ (Language-Integrated Query) es el nombre de un conjunto de tecnologías basado en la integración de recursos de consulta directamente en el lenguaje C#.
- Crear consultas
- Conocer las cláusulas Select y Where
- Consultar colecciones de objetos en memoria
- Mapear la base de datos con LINQ to Sql

- [ ] **C# - Serialización**:

- Serialización es el proceso de convertir un objeto en un flujo de bytes para almacenar el objeto o transmitirlo a la memoria, a una base de datos o a un archivo. Su objetivo principal es guardar el estado de un objeto para poder recrearlo cuando sea necesario.
- Enviar un objeto a una aplicación remota mediante un servicio web
- Pasar un objeto como una cadena JSON o XML
- Pasar información específica del usuario o de seguridad entre aplicaciones

- [ ] **C# - Red y Sockets**:

- El término programación con sockets se refiere a la escritura de programas que se ejecutan en varios ordenadores en los que los dispositivos están todos conectados entre sí utilizando una red.
- Abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor

- Enviar mensajes a un servidor y recibir respuestas sin consultar al servidor

- [ ] **ASP.NET Core**:
  - ASP.NET Core es un framework de código abierto y multiplataforma para la construcción de aplicaciones basadas en la nube, tales como aplicaciones web, aplicaciones IoT y backends móviles.
  - Crear aplicaciones y servicios web
  - Mantener una aplicación MVC
  - Desarrollar interfaz de usuario web del lado del cliente
  - Crear una API web
- [ ] **Dapper**:
  - Dapper es un producto de mapeo objeto-relacional (ORM) para la plataforma Microsoft .NET. Proporciona un framework para asignar un modelo de dominio orientado a objetos a bases de datos relacionales.
  - Realizar consultas a bases de datos como 'select', 'Insert', 'update', 'delete'
  - Manipular bases de datos
- [ ] **Inyección de Dependencias**:
  - La inyección de dependencias es un patrón de diseño en el que una clase solicita dependencias de fuentes externas en lugar de crearlas.
  - Evitar el alto nivel de acoplamiento de código dentro de una aplicación
  - Implementar la inversión de control
- [ ] **C# - Multithreading**:
  - Multithreading es la capacidad de realizar múltiples operaciones al mismo tiempo. Las operaciones con el potencial de retrasar otras operaciones se pueden realizar en subprocesos separados.
  - Realizar múltiples tareas simultáneamente
  - Entender cómo se ejecutan los hilos
  - Aprender cómo hacer que un hilo espere en un punto específico

Nivel 3

- [ ] **C# - Delegates y Eventos**:
  - Delegates son objetos que se usan como punteros de función para referirse a un método asignado a ellos.
  - Los eventos son acciones que cambian el estado de un objeto. Los eventos se declaran utilizando delegates - ellos proporcionan una encapsulación a los delegates.
  - Entender el concepto de delegate
  - Crear una referencia para una función con una cierta lista de parámetros
  - Entender el concepto de evento
  - Manipular diferentes tipos de eventos
- [ ] **C# - Métodos anónimos y lambda expressions**:
  - Los métodos anónimos son métodos sin nombre que se pueden definir usando la palabra clave delegate.
  - Lambda Expressions se utilizan como funciones anónimas, pero no es necesario especificar el tipo de valor que se escribe, lo que las hace más flexibles de usar.
  - Crear funciones anónimas que puedes utilizar para crear delegates
  - Crear roles locales que se pueden pasar como argumentos
- [ ] **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

☐ **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

☐ **Reflection y atributos**:

- Los objetos de Reflection (reflexión) se utilizan para obtener información del tipo en tiempo de ejecución. Las clases que dan acceso a los metadatos de

un programa en ejecución se encuentran en el espacio de nombres System.Reflection.

- Escribir código que lee la información y metadatos de objetos en tiempo de ejecución
- Obtener nombres de clases en tiempo de ejecución y crear objetos de una clase

- [ ] **MAUI**:

- .NET Multi-platform App UI (.NET MAUI) es un framework multiplataforma para crear aplicaciones nativas móviles y de escritorio con C# y XAML.
- Crear aplicaciones móviles y de escritorio nativas con C# y XAML.
- Desarrollar aplicaciones multiplataforma.
- Compartir el diseño y la interfaz de usuario entre plataformas.

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.

- [ ] **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 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.

- [ ] **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.
