Guia Ágil

TechGuide - Alura

C#

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

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

- C# é uma linguagem de programação, multiparadigma, de tipagem forte, desenvolvida pela Microsoft como parte da plataforma .NET. O código fonte é compilado para Common Intermediate Language (CIL) que é interpretado pela máquina virtual Common Language Runtime (CLR). É projetada para funcionar na Common Language Infrastructure da plataforma .NET Framework.

- Conhecer os tipos primitivos
- Declarar variáveis, considerando os diferentes tipos
- 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 métodos
- Manipular arrays e listas
- Obter dados de uma API
- Fazer chamadas assíncronas, etc
- Criar construtores

☐ **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

☐ **C# - Coleções**:

- 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 Dictionary, List, Queue, SortedList e Stack
- Saiba trabalhar com ArrayList e HashTable
- Iterators

- [ ] **NuGet**:

- NuGet é um gerenciador de pacotes para a plataforma .NET. Ele define como os pacotes desta plataforma são criados, publicados e consumidos, e fornece ferramentas para cada uma dessas funções.
- Gerenciar pacotes
- Compartilhar bibliotecas

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

- O namespace System.IO consiste em classes, estruturas, delegates e enumerações relacionadas com entrada e saída de dados (IO). Estas classes podem ser utilizadas para ler e escrever dados em arquivos ou streams de dados. Também contém classes para suporte a arquivos e diretórios.
- Ler dados de arquivos
- Escrever dados em arquivos
- Gerenciar arquivos com Using

- [ ] **C# - Gerenciamento de Memória**:

- O gerenciamento automático de memória é um dos serviços que o Common Language Runtime oferece durante a Execução Gerenciada (Managed Execution). O coletor de lixo do Common Language Runtime gerencia a alocação e liberação de memória para uma aplicação.
- Entender como a memória é administrada
- Conhecer o Garbage Collector

- Entender sobre o Stack e o Heap gerenciado

- [ ] **C# - Testes**:

- O teste de software é o processo de avaliação e verificação de que um software realmente faz o que deveria fazer. Os benefícios dos testes incluem a prevenção de bugs, a redução dos custos de desenvolvimento e a melhoria do desempenho.
- Usar testes unitários
- Usar testes de integração
- Usar testes de comportamento (behavior)
- Usar mocks

- [ ] **ADO.NET**:

- O ADO.NET é um conjunto de classes que expõem serviços de acesso a dados para desenvolvedores do .NET Framework. O ADO.NET fornece um conjunto rico de componentes para criar aplicativos distribuídos e de compartilhamento de dados. Faz parte do .NET Framework, fornecendo acesso a dados de aplicativo relacionais e XML.
- Manipular bancos de dados
- Conhecer o DataSet e DataTable
- Realizar conexões
- Manipular documentos XML

- [ ] **Entity Framework Core**:

- O Entity Framework Core é um mapeador objeto-relacional (ORM). O mapeamento objeto-relacional é uma técnica que permite aos desenvolvedores trabalhar com dados de forma orientada ao objeto, realizando o trabalho necessário para mapear entre objetos definidos em uma linguagem de programação da aplicação e dados armazenados em fontes de dados relacionais.
- Manipular bancos de dados usando objetos .NET
- Criar modelos correspondentes a um banco de dados

- Realizar consultas

- [ ] **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

Nivel 2

- [ ] **CLR**:

- O Common Language Runtime (CLR) é o componente de máquina virtual da plataforma .NET da Microsoft que gerencia a execução de programas .NET.
- Entender como o CLR funciona
- Entender o gerenciamento de memória
- Conhecer a CIL e o JIT

- [ ] **LINQ**:

- O LINQ (Language-Integrated Query) é o nome de um conjunto de tecnologias com base na integração de recursos de consulta diretamente na linguagem C#.
- Criar consultas
- Conhecer as cláusulas Select e Where
- Consultar coleções de objetos em memória
- Mapear o banco de dados com Linq to Sql

- [ ] **C# - Serialização**:

- Serialização é o processo de converter um objeto em um stream de bytes para armazenar o objeto ou transmiti-lo à memória, a um banco de dados ou

a um arquivo. Seu principal objetivo é salvar o estado de um objeto a fim de poder recriá-lo quando necessário.

- Enviar um objeto para um aplicativo remoto usando um serviço Web
- Passar um objeto como uma cadeia de caracteres JSON ou XML
- Passar informações específicas do usuário ou de segurança entre aplicativos

- [ ] **ASP.NET Core**:

- ASP.NET Core é um framework de código aberto e multi-plataforma para a construção de aplicações baseadas na nuvem, tais como aplicações web, aplicações IoT e backends mobile.
- Criar aplicações e serviços Web
- Manter uma aplicações MVC
- Desenvolver de interface do usuário da Web do lado do cliente
- Criar uma API Web

- [ ] **Dapper**:

- O Dapper é um produto de mapeamento objeto-relacional (ORM) para a plataforma Microsoft .NET. Ele fornece um framework para mapear um modelo de domínio orientado a objetos para bancos de dados relacionais.
- Realizar consultas a bancos de dados como 'select', 'insert', 'update', 'delete'
- Manipular bancos de dados

- [ ] **Injeção de Dependências**:

- Injeção de Dependências é um padrão de projeto no qual uma classe solicita dependências de fontes externas ao invés de criá-las.
- Evitar o alto nível de acoplamento de código dentro de uma aplicação
- Implementar a inversão de controle

- [ ] **C# - Multithreading**:

- Multithreading é a capacidade de realizar múltiplas operações ao mesmo tempo. Operações com o potencial de atrasar outras operações podem ser executadas em threads separadas.
- Executar múltiplas tarefas simultaneamente
- Entender como threads são executadas
- Aprender como fazer uma thread esperar em um ponto específico

Nivel 3

- [ ] **C# - Delegates e Eventos**:

- Delegates são objetos que são usados como ponteiros de função para se referirem a um método atribuído a eles.
- Eventos são a ação realizada que altera o estado de um objeto. Eventos são declarados usando delegates - eles fornecem um encapsulamento aos delegates.
- Entender o conceito de delegate
- Criar uma referência para uma função com uma certa lista de parâmetros
- Entender o conceito de evento
- Manipular diferentes tipos de eventos

- [ ] **C# - Métodos anônimos e lambda expressions**:

- Métodos anônimos são métodos sem um nome que podem ser definidos usando a palavra-chave delegate.
- Lambda expressions são usadas como funções anônimas, mas você não precisa especificar o tipo do valor que você digita, tornando-as mais flexíveis de usar.
- Criar funções anônimas que você pode usar para criar delegates
- Criar funções locais que podem ser passadas como argumentos

- [ ] **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

☐ **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

- [ ] **Reflection e atributos**:

- Os objetos de Reflection (reflexão) são usados para obter informações do tipo em tempo de execução. As classes que dão acesso aos metadados de um programa em execução estão no namespace System.Reflection.
- Escrever código que lê as informações e metadados de objetos em tempo de execução
- Obter nomes de classes em tempo de execução e criar objetos de uma classe

- [ ] **MAUI**:

- .NET Multi-platform App UI (.NET MAUI) é um framework multi-plataforma para criar aplicativos nativos móveis e desktop com C# e XAML.
- Criar aplicativos móveis e desktop nativos com C# e XAML
- Criar aplicativos multiplataforma
- Compartilhar o layout e o design da interface do usuário entre plataformas

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 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

- [ ] **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

- [ ] **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)

Habilidade Auxiliar: Boas práticas

- [ ] **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.

- [ ] **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

- [ ] **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.

- [ ] **Clean Code**:

- Aplicar técnicas simples que visam facilitar a escrita e leitura de um código
- Refatorar seu código para que fique mais claro

- [ ] **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
