Guia Ágil

TechGuide - Alura

C#

Nivel 1

- [ ] **C# - Fundamentals**:

- C# is a multi-paradigm, strongly typed programming language developed by Microsoft as part of the .NET platform. The source code is compiled into Common Intermediate Language (CIL) which is interpreted by the Common Language Runtime (CLR) virtual machine. It is designed to run on the Common Language Infrastructure of the .NET Framework platform.
- Knowing the primitive types
- Declaring variables, considering the different types
- Using conditional structures ('if', 'else')
- Knowing the assignment and comparison operators
- Using repetition structures and loops ('while', 'for')
- Using functions, passing parameters and arguments
- Manipulating methods
- Manipulating arrays and lists
- Getting data from an API
- Making asynchronous calls
- Creating constructors

- [ ] **Object-oriented Programming Concepts**:
  - Object-oriented programming (OOP) is a programming paradigm based on the concept of 'objects', which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A common feature of objects is that procedures (or methods) are attached to them and can access and modify the object's data fields. Some of the main concepts are classes and instances, inheritance, and encapsulation.
  - How objects work
  - Creating and using constructors
  - What classes are
  - Creating and using Methods
  - How encapsulation works
  - What inheritance is
  - What polymorphism is
  - How interfaces work
  - What abstractions are
- [ ] **C# - Collections**:
  - A collection represents a group of objects, known as its elements. They are like containers that group multiple items in a single unit. Some collections allow duplicate elements and others do not. Some are ordered and others unordered.
  - Learn the difference between Dictionary, List, Queue, SortedList and Stack
  - Get to know how to work with ArrayList e HashTable
  - Iterators
- [ ] **NuGet**:
  - NuGet is a package manager for the .NET platform. It defines how packages for this platform are created, published and consumed, and provides tools for each of these functions.

- Gerenciar pacotes
- Compartilhar bibliotecas

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

- The System.IO namespace consists of IO related classes, structures, delegates and enumerations. These classes can be used to reads and write data to files or data streams. It also contains classes for file and directory support.
- Ler dados de arquivos
- Escrever dados em arquivos
- Gerenciar arquivos com Using

- [ ] **C# - Memory Management**:

- Automatic memory management is one of the services that the Common Language Runtime provides during Managed Execution. The Common Language Runtime's garbage collector manages the allocation and release of memory for an application.
- Understand how the memory is managed
- Getting to know the Garbage Collector
- Understanding about the managed Stack and Heap

- [ ] **C# - Testing**:

- Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.
- Using unit tests
- Using integration testing
- Using behavioral testing
- Using mocks

- [ ] **ADO.NET**:

- ADO.NET is a set of classes that expose data access services for .NET Framework programmers. ADO.NET provides a rich set of components for creating distributed, data-sharing applications. It is an integral part of the .NET Framework, providing access to relational, XML, and application data.
- Handling databases
- Getting to know DataSet and DataTable
- Performing connections
- Handling XML documents

- [ ] **Entity Framework Core**:

- Entity Framework Core is an object-relational mapper (ORM). Object-relational mapping is a technique that enables developers to work with data in object-oriented way by performing the work required to map between objects defined in an application's programming language and data stored in relational datasources.
- Manipulating databases using .NET objects
- Creating models corresponding to a database
- Performing queries

- [ ] **Data Structures**:

- In the context of computers, the data structure is a specific way of storing and organizing data in the computer's memory so that these data can be easily retrieved and efficiently used when needed later.
- Knowing the main data structures (linked list, stack, queue, tree, etc)
- Implementing the main data structures

Nivel 2

- [ ] **CLR**:

- The Common Language Runtime (CLR) is the virtual machine component of Microsoft .NET Framework that manages the execution of .NET programs.
- Understanding how the CLR works

- Understanding memory management
- Getting to know CIL and JIT

- [ ] **LINQ**:

- Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language.
- Criar consultas
- Conhecer as cláusulas Select e Where
- Consultar coleções de objetos em memória
- Mapear o banco de dados com Linq to Sql

- [ ] **C# - Serialization**:

- Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed.
- Sending an object to a remote application using a web service
- Passing an object as a JSON or XML string
- Passing user-specific or security information between applications

- [ ] **C# - Networking and Sockets**:

- Networking is a concept of connecting two or more computing devices together so that we can share resources. Socket programming provides facility to share data between different computing devices through a network.
- Opening an interactive communication session between the user's browser and a server
- Sending messages to a server and receive replies without querying the server

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

- ASP.NET Core is an open-source and cross-platform framework for building modern cloud-based applications, such as web apps, IoT apps and mobile backends.
- Creating applications and web services
- Maintaining an MVC application
- Developing a client-side web user interface
- Creating a Web API

- [ ] **Dapper**:

- Dapper is an object–relational mapping (ORM) product for the Microsoft .NET platform. It provides a framework for mapping an object-oriented domain model to relational databases.
- Performing database queries such as 'select', 'insert', 'update', 'delete'
- Manipulating databases

- [ ] **Dependency Injection**:

- Dependency Injection is a design pattern in which a class requests dependencies from external sources instead of creating them.
- Avoiding high level of code coupling within an application
- Implementing inversion of control

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

- Multithreading is the ability to perform multiple operations at the same time. Operations with the potential of holding up other operations can execute on separate threads.
- Running multiple tasks simultaneously
- Understanding how threads are executed
- Learning how to make a thread wait at a specific point

Nivel 3

- [ ] **C# - Delegates and Events**:

- Delegates are objetcs that are used as function pointers to refer to a method assigned to them.
- Events are the action performed which changes the state of an object. Events are declared using delegates - they provide encapsulation to the delegates.
- Understanding the concept of delegate
- Creating a reference to a function with a certain list of parameters
- Understanding the concept of event
- Handling different types of events

- [ ] **C# - Anonymous methods and lambda expressions**:

- Anonymous methods are unnamed methods that can be defined using the delegate keyword.
- Lambda expressions are used like anonymous functions, but you don't need to specify the type of the value that you input, making them more flexible to use.
- Creating anonymous functions that you can use to create delegates
- Creating local functions that can be passed as arguments

- [ ] **Containers**:

- Containers are software packages that contain all the elements needed to run in any environment.
- Kubernetes (also known as k8s or “kube”) is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications.
- Isolating your software to run independently
- Deploying software in clusters
- Modularizing your system into smaller packages
- Getting to know the Docker platform
- Getting to know Kubernetes

- [ ] **Microservices architecture**:
  - Microservices are an architectural approach in which software consists of small independent services that communicate with each other and are organized according to their business domains.
  - Learning the concept of planned architecture for microservices
  - Performing communication using APIs
  - Improving the scalability of a system
- [ ] **Reflection and attributes**:
  - Reflection objects are used for obtaining type information at runtime. The classes that give access to the metadata of a running program are in the System.Reflection namespace.
  - Writing code that reads object information and metadata at runtime
  - Getting class names at runtime and creating objects of a class
- [ ] **MAUI**:
  - .NET Multi-platform App UI (.NET MAUI) is a cross-platform framework for creating native mobile and desktop apps with C# and XAML.
  - Creating native desktop and mobile applications with C# and XAML
  - Creating cross-platform applications
  - Sharing user interface layout and design across platforms

Habilidade Auxiliar: Infrastructure

- [ ] **Git & GitHub - Fundamentals**:
  - Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
  - GitHub is a hosting service for software development and version control using Git.
  - Creating a repository

- Cloning a repository
- Committing, pushing and pulling to and from the repository
- Reversing a commit
- Creating branches and pull requests
- Handling merge and conflicts

- [ ] **HTTP - Fundamentals**:

- HTTP stands for Hyper Text Transfer Protocol. Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses.
- Understanding the difference between HTTP verbs
- Testing requests and checking the status codes in the browser
- Learning how to make a HTTP request on the command line with WGET
- Downloading an image with WGET
- Performing a POST

- [ ] **JSON**:

- JSON stands for JavaScript Object Notation. It is a text format for storing and transporting data.
- Creating an object
- Transforming an object into a string
- Transforming a string into an object
- Manipulating an object

- [ ] **Command Line - Fundamentals**:

- CLI is a command line program that accepts text input to execute operating system functions.
- Knowing the most important commands

- [ ] **Cloud - Fundamentals**:

- Cloud, or cloud computing, is the distribution of computing services over the Internet using a pay-as-you-go pricing model. A cloud is composed of various computing resources, ranging from the computers themselves (or instances, in cloud terminology) to networks, storage, databases, and everything around them. In other words, everything that is normally needed to set up the equivalent of a server room, or even a complete data center, will be ready to use, configured, and run.
- Knowing the difference between IaaS, PaaS and SaaS
- Knowing the largest cloud providers
- Specializing in a specific provider of your choice

- [ ] **SQL - Fundamentals**:

- Structured Query Language (SQL) is a standardized programming language that is used to manage relational databases and perform various operations on the data in them.
- Knowing the most common SQL commands
- Using SELECT to query a table
- Using INSERT to insert data into a table
- Using UPDATE to update a table
- Using DELETE to remove data from a table
- Using JOIN to connect data from multiple tables
- Knowing the clauses (FROM, ORDER BY, etc.)

Habilidade Auxiliar: Good practices

- [ ] **SOLID**:

- SOLID has five principles that are considered best practices in software development that help programmers write cleaner code by separating responsibilities, reducing docking, easing refactoring, and encouraging code reuse.

- [ ] **Design Patterns**:

- In software engineering, a Design Pattern is a general, reusable solution to a commonly occurring problem within a given context in software design. It is a description or template for how to solve a problem that can be used in many different situations. Design Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.
- Getting familiarized with and applying the main Design Patterns

- [ ] **Clean Architecture**:

- Clean architecture is a way of developing software, such that just by looking at the source code of a program, you should be able to tell what the program does.

- [ ] **Clean Code**:

- Applying simple techniques that aim to make a code easier to write and read
- Refactoring your code to make it clearer

- [ ] **Domain-Driven Design (DDD) Concepts**:

- Domain-Driven Design (DDD) is an approach to software design and development that is first informed by business requirements. The program components (objects, classes, arrays, etc.) indicate the industry, sector, or business domain in which the business operates.
- Modeling domains effectively
- Basing complex projects on domain models
- Getting to know the building blocks of DDD
