Guia Ágil

TechGuide - Alura

Angular

Nivel 1

- [ ] **HTML - Fundamentals**:

- HTML is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.
- Learning which tags are required for basic HTML
- Creating a text paragraph
- Displaying an image
- Knowing the difference between 'h1', 'h2', 'h3', etc.
- Creating a hyperlinked text
- Creating a form with relevant fields
- Creating an ordered or unordered list of items
- Creating a list of items within a dropdown list
- Linking to a CSS file
- Creating a table
- Adding IDs and classes

- [ ] **CSS - Fundamentals**:
    - Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS can be used for very basic document text styling — for example, for changing the color and size of headings and links. It can be used to create a layout — for example, turning a single column of text into a layout with a main content area and a sidebar for related information. It can even be used for effects such as animation.
    - Learning the visual structure of a page, with 'margin' and 'padding'
    - Establishing the size with 'width' and 'height'
    - Learning about the position of an element ('static', 'relative' or 'absolute')
    - Learning about the display of an element ('block', 'inline', 'inline-block')
    - Learning how to position images in relation to text
    - Learning about alignment
    - Learning about font style
    - Learning the differences and advantages of using the different units of measurement in CSS (%, relative, etc)
    - Connecting to the elements (IDs, classes) of an HTML file
    - Changing the characteristics of an element when the mouse hovers over it
    - Learning box-sizing
    - Learning Flexbox
    - Learning Grid
- [ ] **JavaScript - Fundamentals**:
    - JavaScript is the world's most popular programming language and is one of the core technologies of the World Wide Web, alongside HTML and CSS. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles.
    - Knowing the primitive types

- Declaring variables, considering the difference between 'var', 'let' and 'const
- Using conditional structures ('if', 'else')
- Know the assignment and comparison operators ('=', '==', '===')
- Using repetition structures and loops ('while', 'for')
- Using functions, passing parameters and arguments
- Manipulating arrays and lists
- Getting data from an API
- Making asynchronous calls using 'Async/Await', 'Promise', etc

☐ **TypeScript - Fundamentals**:

- TypeScript is a strongly typed programming language that builds on JavaScript.
- Understanding in depth what types are and the importance of typed programming
- Learning what TypeScript is, why it was created, how it works and its relationship with JavaScript
- Knowing the TypeScript tools (integration with the code editor, static checker and compiler)
- Writing code in TypeScript using its tools (interfaces, enum, decorators, etc.)

☐ **Angular - Fundamentos**:

- Angular is an application-design framework and development platform built on TypeScript for creating efficient and sophisticated single-page apps.
- Building interfaces using HTML, CSS and TypeScript
- Creating SPA applications
- Building web, mobile or desktop applications
- Integrating data with REST APIs
- Using composition to create reusable components
- Using Resolver services
- Manipulating requests by creating Interceptor services

- [ ] **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
- [ ] **RxJS - Fundamentals**:
    - RxJS is a library for composing asynchronous and event-based programs by using observable sequences. It provides one core type, the Observable, satellite types (Observer, Schedulers, Subjects) and operators inspired by Array methods (map, filter, reduce, every, etc) to allow handling asynchronous events as collections.
    - Creating asynchronous programs
    - Creating event-based programs
    - Understanding the concept of observables and sequences of observables
    - Understanding how to use Observers, Subscription, Subject and others
- [ ] **Observer pattern**:

- In software design, the Observer pattern is a software design pattern in which an object, named the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.
- Understanding what Design Patterns are
- Updating several elements simultaneously using Observers
- Declaring subjects
- Creating event-based programs

Nivel 2

- [ ] **Angular - Templates**:

- In Angular, a template is a blueprint for a fragment of a user interface (UI). Templates are written in HTML, and special syntax can be used within a template to build on many of Angular's features.

- [ ] **Angular - Rendering**:

- A normal Angular application executes in the browser, rendering pages in the DOM in response to user actions. Angular Universal executes on the server, generating static application pages that later get bootstrapped on the client.
- Showing an Angular page on the browser
- Performing Server-side rendering

- [ ] **Angular - Services**:

- Service is a broad category encompassing any value, function, or feature that an application needs. A service is typically a class with a narrow, well-defined purpose. Angular distinguishes components from services to increase modularity and reusability.
- Criar dados que estarão sempre disponíveis
- Dividir a aplicação web em diversas partes

- [ ] **Angular - Routing**:

- In a Single Page Application (SPA), all of your application's functions exist in a single HTML page. As users access your application's features, the browser needs to render only the parts that matter to the user, instead of loading a new page. This pattern can significantly improve your application's user experience. To define how users navigate through your application, you use routes. Add routes to define how users navigate from one part of your application to another. You can also configure routes to guard against unexpected or unauthorized behavior.
- Navigating to a component
- Including a route parameter
- Controllung your user's navigation flow with route guarding

- [ ] **Angular - CLI (Command-Line Interface)**:

- The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.
- Learning the syntax 'ng [optional-arg] [options]'
- Getting to know important commands, like 'ng add', 'ng build', 'ng update', 'ng deploy', 'ng new', 'ng test', among others

Nivel 3

- [ ] **Angular - State Management**:

- A 'state' is whatever data necessary in order to rebuild the UI at any moment in time. When this data change, it will trigger a redraw of the user interface. State management is the concept of adding, updating, removing and reading these data and their state in an application.
- Updating components in real time
- Waiting for changes in a component and performing changes
- Using Redux, NGXS and others

- [ ] **Angular - Forms**:

- A web form is an online page that allows for user input. It is an interactive page that mimics a paper document or form, where users fill out particular fields.
- Creating forms with Template Forms
- Creating reactive forms with Reactive Forms

- [ ] **Angular - Modules**:

- A Module, unlike a Component, does not control any view. A Module consists of one or more Components. An Angular application must have at least one Module that contains at least one Component.
- Using the ngModule class
- Declaring which components can be used by components from other modules
- Declaring which services will be injected
- Learning how to modularize an application
- Lazy-loading modules

- [ ] **Angular - Dependency Injection**:

- Dependency Injection (DI) allows you to declare the dependencies of your TypeScript classes without taking care of their instantiation. Instead, Angular handles the instantiation for you. This design pattern allows you to write more testable and flexible code.
- Declaring your classes' dependencies
- Injecting dependencies into a Component
- Getting to know Injection Tokens
- Configuring providers
- Understanding Hierarchical Dependency Injection

- [ ] **Angular - Testes**:

- 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
- Getting acquainted with Jasmine and/or Karma

- [ ] **GraphQL**:

- GraphQL is a new API standard that provides a more efficient, powerful and flexible alternative to REST. It was developed and open-sourced by Facebook and is now maintained by a large community of companies and individuals from all over the world.
- Understanding how GraphQL is used in API development
- Creating APIs using GraphQL libraries and frameworks

- [ ] **Apollo Client**:

- Apollo Client is a comprehensive state management library for JavaScript that enables you to manage both local and remote data with GraphQL.

Habilidade Auxiliar: Infrastructure and Back-end

- [ ] **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

- [ ] **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

Habilidade Auxiliar: UX and Design

- [ ] **Design Systems**:
  - A design system is a collection of reusable components, guided by clear standards, that can be assembled together to build applications.
  - Creating and maintaining libraries that will be consumed and used as a standard for building a project
- [ ] **Figma - Fundamentals**:
  - Figma is a collaborative web application for interface design. The feature set of Figma focuses on user interface and user experience design, with an emphasis on real-time collaboration, utilising a variety of vector graphics editor and prototyping tools.
  - Creating page layouts and components
- [ ] **Design components**:
  - Knowing the components that describe a layout or interface
- [ ] **Color systems**:
  - Defining a color palette that makes sense for a given interface
- [ ] **How to use Fonts**:
  - Choosing the most appropriate font for a given project
