Node.js

TechGuide - Alura, FIAP e PM3

Node.js

Nivel 1

- [ ] **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

- [ ] **Node.js - Fundamentals**:

  - Node.js is a JavaScript execution environment that allows you to run applications developed with the language autonomously, without depending

on a browser.

- Getting to know blocking and non-blocking operations
- Learning the concept of event loops
- Learning how to use Node.js libraries, such as 'net', 'fs', 'http', 'path', among others
- Understanding how Timers work

- [ ] **JavaScript - Callbacks and Promises**:

- A Promise is a proxy for a value not necessarily known when the promise is created. This lets asynchronous methods return values like synchronous methods - instead of immediately returning the final value, the asynchronous method returns a promise to supply the value at some point in the future.
- A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.
- An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.
- Understanding the concept of asynchronous programming
- Writing asynchronous code understanding the concept of promises in JavaScript
- Using JavaScript methods, keywords and objects for handling promises like 'Async/Await', '.then()', 'Promise', etc
- Learning in which situations you need to use asynchronous programming
- Calling APIs with `fetch()`

- [ ] **JavaScript - Error handling**:

- Error handling refers to the response and recovery procedures from error conditions present in a software application. In other words, it is the process comprised of anticipation, detection and resolution of application, programming or communication errors.

- Knowing and handling the most common exceptions
- Knowing the types of errors and in which situations they can occur
- Using 'try' and 'catch' for error handling
- Learning on what occasions and how to use `throw`
- Creating specific exceptions according to your application's needs

- [ ] **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

- [ ] **Node.js - 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
- Learn the concepts of BDD (Behavior-Driven Development) and TDD (Test-Driven Development)

Nivel 2

- [ ] **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.

- [ ] **Node.js - Express**:

- Express is a flexible Node.js web application framework that provides a robust feature set for web and mobile applications.
- Using the Express framework to build REST APIs with Node.js
- Managing requests for different HTTP verbs on different URLs
- Defining ports that will be used for connection and the location of the templates that are used to render the response
- Creating route handlers using the 'router' method
- Getting to know the 'Router' libraries and their HTTP verbs, such as 'get', 'post', 'put', etc
- Defining endpoints with route paths

- [ ] **Node.js - ORM**:

- Object-Relational Mapping (ORM) is a technique used to map object-oriented systems to relational databases, where the database tables are represented in classes and the table records would be instances of these classes.
- Understanding what ORMs are and what they are used for
- Knowing SQL and its database managers

- Working with Sequelize, an ORM used with Node.js
- Knowing other Node.js ORMs, like Prisma

- [ ] **Node.js - Authentication and Tokens**:

- JSON Web Tokens (JWT) are an open, industry standard RFC 7519 method for representing claims securely between two parties.
- Building an authentication system using tokens
- Understanding how JSON Web Tokens (JWT) work
- Building an allowlist to store opaque tokens
- Implementing methods for updating tokens

- [ ] **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.)

Nivel 3

- [ ] **Nest.js - Fundamentals**:

- NestJS is a Node framework with full TypeScript support that runs on top of HTTP frameworks like ExpressJS or Fastify. It uses many elements of object-oriented programming and a lot of TypeScript functionality.
- Learning what NestJS is and why it is used
- Using specific NestJS features, such as providers, modules and controllers
- Developing APIs using NestJS

- Using the Nest.js CLI

- [ ] **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

- [ ] **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

- [ ] **WebSockets**:

- The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.
- Getting to know the WebSocket protocol and its use in client-server communication
- Learning about the various uses of WebSockets on the web
- Creating applications that use WebSockets with Node.js APIs and libraries

- [ ] **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 good practices

- [ ] **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

- [ ] **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

- [ ] **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

Habilidade Auxiliar: Front-end

- [ ] **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

☐ **DOM - Fundamentals**:

- The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.
- Understanding how the DOM tree works
- Accessing and manipulating HTML and CSS elements
- Accessing the parents and children of an element
- Inserting a new element to the tree
- Removing an element from the tree
- Waiting for an event on a certain page element using addEventListener()

☐ **Javascript - Accessibility**:

- Web accessibility is the inclusive practice of ensuring there are no barriers that prevent interaction with, or access to, websites by people with physical disabilities, situational disabilities, and socio-economic restrictions on bandwidth and speed.

- Writing code with accessibility in mind

- [ ] **SEO Strategies**:

- Search engine optimization (SEO) is the process of improving the quality and quantity of website traffic to a website or a web page from search engines.
- Choosing keywords
- Understanding how Google ranks pages
- Knowing the ranking factors
- Performing Link Building
