Curso: Análise e Desenvolvimento de Sistemas
4º Período “A”
Aluno: José Henrique da Cunha Pedrosa Neto
Matrícula: 01593364

**Roteiro Back-end**

**1 - Funcionamento da Internet e protocolos**

É importante compreender a internet, uma rede global que conecta dispositivos e permite a troca de informações por protocolos como HTTP e HTTPS, que organizam a comunicação entre clientes (navegadores) e servidores. Navegadores interpretam dados do servidor em interfaces visíveis para o usuário. No backend, o DSN (Data Source Name) mapeia conexões com bancos de dados, facilitando o acesso a informações. Já a hospedagem (hosting) mantém sites online através de servidores dedicados, assegurando que estejam acessíveis a qualquer momento. Esses conceitos formam a base da comunicação entre sistemas e usuários.

**2 - Conhecimento sobre Lógica de Programação**

O conhecimento em lógica de programação é fundamental para resolver problemas de maneira estruturada e eficiente. Ele envolve entender conceitos básicos como **variáveis**, **operadores**, **estruturas condicionais** (como "if" e "else") e de **repetição** (como "for" e "while"), além de **funções** e **algoritmos**. A lógica de programação permite decompor problemas complexos em etapas simples, criando sequências lógicas que o computador pode seguir. Dominar essa habilidade é essencial para desenvolver soluções claras, eficientes e escaláveis em qualquer linguagem de programação.

**3 - JavaScript**

JavaScript é uma linguagem essencial para o desenvolvimento web, tanto no Front-End, quanto no Back-End, sendo usada para tornar páginas interativas. Sua sintaxe é simples e utiliza variáveis para armazenar dados. Com comandos de entrada e saída (como a biblioteca *prompt-sync*), permite capturar e exibir informações. A linguagem inclui operadores matemáticos, de comparação e lógicos para realizar cálculos e verificações. Usando estruturas de condição (if/else) e funções, é possível criar blocos de código reutilizáveis.

JavaScript também trabalha com arrays para organizar dados em listas, oferecendo métodos para manipulá-los. Promises são usadas para lidar com operações assíncronas, informando se uma tarefa foi concluída com sucesso. Outras funcionalidades incluem objetos para estruturar dados complexos, métodos específicos para cada tipo de variável, manipulação de datas e várias bibliotecas para aumentar seu potencial de uso.

Para a construção e estruturação do Back-End utilizando o JavaScript, é possivel utulizarmos um framework, chamado Node.js, que traz o JavaScript ou o TypeScript para o desenvolvimento do Back-End

**4 - Gerenciador de pacotes**

Um Package Manager (Gerenciador de Pacotes) é uma ferramenta que facilita o gerenciamento de bibliotecas, frameworks e dependências em projetos de software, automatizando a instalação, atualização e remoção de pacotes. Essas ferramentas ajudam a garantir a integridade e compatibilidade das dependências, tornando o trabalho dos

desenvolvedores mais eficiente. Além de facilitar a instalação e atualização, eles reforçam a segurança e permitem a reutilização de código.
O npm (Node Package Manager) é o gerenciador de pacotes oficial do Node.js e possibilita instalar e gerenciar bibliotecas e ferramentas JavaScript, simplificando a criação de aplicações complexas. Yarn é outra opção popular, oferecendo benefícios como maior velocidade e verificação de integridade dos pacotes instalados.

**5 - Git e GitHub**
Git é um sistema de controle de versão distribuído, usado para gerenciar e acompanhar alterações no código. Ele permite que desenvolvedores colaborem de forma organizada, criando "snapshots" (commits) do projeto em diferentes etapas, possibilitando voltar a versões anteriores, experimentar novas funcionalidades sem afetar o código principal (branches) e revisar o histórico completo das modificações.
GitHub é uma plataforma que hospeda repositórios Git na nuvem, facilitando o compartilhamento e colaboração online em projetos. Com o GitHub, desenvolvedores podem fazer pull requests, revisar código, gerenciar issues (problemas ou tarefas) e usar ferramentas de integração contínua. Ele também permite que equipes acompanhem o progresso e trabalhem juntas em um mesmo projeto, independentemente da localização.
Alguns comandos básicos do Git:
1 - Inicia um repositório
`git init`
2 - Adicionar arquivos ao staging
`git add .`
3 - Confirmação de mudanças
`git commit -m "Descrição das mudanças"`
4 - Verificação do status
`git status`
5 - Enviar Commits para o repositório local
`git push origin <nome-da-branch>`

**6 - SQL (Structured Query Language)**
É uma linguagem de consulta estruturada usada para gerenciar e manipular bancos de dados relacionais. Com SQL, é possível criar, ler, atualizar e deletar dados (operações CRUD), definir estruturas de dados, consultar e filtrar informações e realizar junções de tabelas. É essencial para desenvolvedores que trabalham com bancos de dados como MySQL, PostgreSQL e SQL Server.
**->Docker**
É uma plataforma de virtualização que permite criar, implantar e gerenciar aplicações em contêineres. Com ele, desenvolvedores podem empacotar aplicações e todas as suas dependências em um ambiente isolado e portátil, facilitando o desenvolvimento, teste e implantação de software em diversas máquinas de forma consistente.
**->PgAdmin**
É uma ferramenta de administração para bancos de dados PostgreSQL. Ela fornece uma interface gráfica que permite aos usuários gerenciar e consultar bancos de dados PostgreSQL de forma amigável, incluindo a criação de tabelas, execução de consultas SQL, visualização de dados e gerenciamento de permissões.

**7 - Criação de API's com Node.js**

A criação de APIs com Node.js envolve configurar um servidor utilizando o **Express.js** para criar rotas, processar dados de entrada (como body, header e query) e gerenciar requisições HTTP. Ferramentas como **CORS** permitem que diferentes origens se comuniquem com a API. A utilização de middleware ajuda na implementação de funcionalidades adicionais, como logging ou validação. Para facilitar o desenvolvimento, o **Nodemon** recarrega automaticamente o servidor quando há alterações no código, e o **Multer** permite o upload de arquivos.

Ferramentas de interface gráfica como **Postman** e **Insomnia** são usadas para testar as APIs, enquanto o **Swagger** oferece uma maneira de gerar documentação interativa para facilitar o entendimento da API. Por fim, os **ORMs** (como **Sequelize**) simplificam a interação com o banco de dados, permitindo mapear tabelas e realizar operações de forma mais intuitiva.

**8 - Autenticação**

Autenticação Back-end é o processo de verificar a identidade de um usuário para garantir que ele tenha permissão para acessar certos recursos de uma aplicação. Em sistemas de autenticação, o back-end interage com provedores de identidade para validar a identidade do usuário.

**JWT(Json Web Token)**

Um formato de token utilizado para autenticação e troca de informações seguras entre partes, geralmente usado para sessões em APIs.

**Passport**

Um middleware para Node.js que facilita a implementação de autenticação em aplicações, com suporte para diferentes estratégias como login com senha, OAuth, entre outras

**Social Providers**

Serviços de autenticação que permitem aos usuários fazer login com suas contas de redes sociais como Google, Apple, Facebook, etc.

**API's Externas**

Plataformas como Amazon Cognito e Auth0 fornecem serviços de autenticação e gestão de usuários de forma simplificada, permitindo integração com várias fontes de identidade.

9 - **Serviços e Ferramentas de Infraestrutura**

**Heroku**

Plataforma como serviço (PaaS) que facilita a implementação e escalabilidade de aplicativos na nuvem, com suporte a integração de autenticação.

**AWS EC2**

Serviço de computação em nuvem que permite a execução de máquinas virtuais para hospedar aplicações e serviços, incluindo sistemas de autenticação.

**PM2**

Gerenciador de processos para Node.js que ajuda a manter a aplicação rodando de forma estável e reiniciar automaticamente em caso de falhas.

**NGinx**

Servidor web que também pode atuar como proxy reverso para distribuir tráfego entre diferentes servidores, sendo útil para autenticação em sistemas distribuídos.

**AWS S3**

Serviço de armazenamento de objetos da Amazon Web Services, utilizado para armazenar e servir arquivos, como imagens ou documentos, em sistemas que exigem autenticação de usuários.

**10 - TypeScript e Nest.js**

**TypeScript** é uma linguagem de programação baseada em JavaScript que adiciona tipagem estática opcional e outros recursos avançados. Ao permitir que os desenvolvedores definam tipos para variáveis, funções e objetos, o TypeScript ajuda a detectar erros de compilação e facilita a manutenção do código em projetos de maior escala. Ele é amplamente utilizado para melhorar a robustez de projetos em JavaScript, proporcionando uma experiência de desenvolvimento mais segura e eficiente. Para usar o TypeScript, é necessário compilar o código para JavaScript antes de executá-lo.

**NestJS** é um framework para o desenvolvimento de aplicações back-end com Node.js, que utiliza TypeScript como linguagem principal. Baseado no padrão MVC (Model-View-Controller), o NestJS é altamente modular, permitindo a criação de APIs escaláveis e de fácil manutenção. Ele também integra outras bibliotecas populares, como TypeORM para interação com bancos de dados e Passport para autenticação. O NestJS é ideal para aplicações grandes e complexas devido à sua estrutura organizada e suporte a conceitos como injeção de dependências e controle de ciclo de vida dos serviços. Ele oferece uma abordagem similar ao que é encontrado em frameworks como Angular, mas focado no back-end.