1) **Front-end:** O front-end é a parte da aplicação que os usuários interagem diretamente. Ele lida com a apresentação visual e interação do usuário.

**HTML (Hypertext Markup Language):** HTML é a linguagem de marcação usada para estruturar o conteúdo da página, como textos, imagens e links. Ele define a estrutura da página.

**CSS (Cascading Style Sheets):** CSS é uma linguagem de estilo que controla a aparência visual dos elementos HTML. Ele define cores, fontes, layouts e outros aspectos visuais.

**Back-end:** O back-end é a parte da aplicação que lida com a lógica de negócios, processamento de dados e interações com o banco de dados.

**Node.js:** Node.js é um ambiente de execução JavaScript que permite criar aplicativos de back-end usando JavaScript. Ele é conhecido por sua eficiência e escalabilidade.

**Python com Django:** Python é uma linguagem de programação versátil para o desenvolvimento de back-end. O Django é um framework em Python que facilita a criação de aplicativos web robustos e complexos.

2) a) **Controller e Models:**

* **Controller:** Um controller em um padrão arquitetural MVC (Model-View-Controller) é responsável por gerenciar a interação entre o modelo (dados) e a interface do usuário. Ele recebe solicitações (geralmente por meio de rotas) do cliente, processa essas solicitações usando o modelo apropriado e retorna a resposta apropriada de volta ao cliente. Os controllers manipulam a lógica de negócios e coordenam as operações que devem ocorrer em resposta a uma ação do usuário.
* **Model:** Os modelos representam os dados e as regras de negócios subjacentes de um aplicativo. Eles encapsulam a lógica para acessar e manipular os dados, garantindo a consistência e integridade dos mesmos. Os modelos podem incluir operações de consulta, validação e atualização dos dados.

**Herança de ControllerBase:** Ao criar um controller que herda de ControllerBase, geralmente estamos aproveitando as funcionalidades e comportamentos básicos fornecidos pelo ControllerBase. Ele fornece recursos comuns, como gerenciamento de rotas, manipulação de solicitações HTTP, autenticação básica, entre outros. Ao herdar de ControllerBase, podemos estender esses recursos básicos para atender às necessidades específicas do nosso aplicativo.

b) No contexto de desenvolvimento web, as rotas GET e POST são dois métodos HTTP diferentes usados para enviar solicitações para um servidor.

* **GET:** O método GET é usado para solicitar dados do servidor. Ele é seguro e não deve ter efeitos colaterais no servidor ou nos dados. Ele é facilmente executado em um navegador, pois o navegador pode fazer uma solicitação simples de obter informações de um URL e exibir essas informações no navegador.
* **POST:** O método POST é usado para enviar dados para o servidor, muitas vezes para criar, atualizar ou excluir recursos. Ele pode ter efeitos colaterais no servidor ou nos dados. Devido à natureza dos efeitos colaterais, os navegadores geralmente não podem executar diretamente solicitações POST ao acessar uma URL, porque isso poderia causar mudanças indesejadas nos dados do servidor sem o conhecimento do usuário.

c) O Swagger é uma ferramenta amplamente utilizada para documentar APIs de serviços web. Ele permite criar, visualizar e interagir com a documentação da API de forma amigável. O papel do Swagger inclui:

Documentação: O Swagger gera automaticamente uma documentação interativa para a API, incluindo informações sobre endpoints, parâmetros, tipos de dados, autenticação, etc. Isso ajuda os desenvolvedores a entenderem rapidamente como usar a API.

Interface Interativa: O Swagger UI oferece uma interface interativa baseada em navegador que permite testar os endpoints da API diretamente na documentação. Isso é extremamente útil para depuração e validação.

Geração de Clientes e Servidores: O Swagger pode gerar automaticamente código cliente e, em alguns casos, código servidor, em várias linguagens de programação, a partir da descrição da API. Isso acelera o desenvolvimento, pois os desenvolvedores não precisam escrever manualmente chamadas de API ou estruturas de dados.

Em resumo, o Swagger é usado para melhorar a documentação, testar, validar e gerar código relacionado a APIs, tornando o processo de desenvolvimento mais eficiente e colaborativo.

3) a) Criar aula Verbo HTTP: POST

b) Remover um material Verbo HTTP: DELETE

c) Listar as disciplinas disponíveis para o professor no semestre Verbo HTTP: GET

d) Listar os alunos da disciplina Verbo HTTP: GET

e) Atualizar um material Verbo HTTP: PUT

f) Lançar uma nota Verbo HTTP: POST

4) Certamente, esses métodos **Ok()**, **BadRequest()**, **NotFound()**, são retornos de status HTTP frequentemente usados em frameworks de desenvolvimento web para indicar o resultado de uma ação executada em um controller. Cada um deles tem um propósito específico:

**Ok():** O método **Ok()** é usado para indicar que a solicitação foi bem-sucedida e retornou um resultado válido. Ele geralmente é usado para retornar dados em resposta a uma solicitação GET, POST ou outra operação. O status HTTP retornado é o 200 OK.

**BadRequest():** O método **BadRequest()** é usado quando a solicitação feita pelo cliente contém informações inválidas, incorretas ou malformatadas. Ele indica que a requisição não pode ser processada devido a erros nos dados fornecidos. O status HTTP retornado é o 400 Bad Request.

**NotFound():** O método **NotFound()** é usado quando o servidor não encontra o recurso solicitado na URL especificada. É comumente utilizado quando uma página, endpoint ou recurso não está disponível. O status HTTP retornado é o 404 Not Found.

**5) https://**: Isso indica o protocolo utilizado para acessar a API. "HTTPS" significa Hypertext Transfer Protocol Secure. É uma versão segura do protocolo HTTP, que fornece uma camada adicional de segurança através do uso de criptografia. Isso ajuda a proteger os dados transmitidos entre o cliente (quem faz a requisição) e o servidor (onde a API está hospedada) contra possíveis interceptações ou manipulações maliciosas.

**minhaapi.com.br**: Isso é o domínio da API. É o endereço na web onde a API está hospedada. No exemplo, "minhaapi.com.br" é um nome de domínio fictício, e normalmente isso seria substituído pelo domínio real da API. O domínio é usado para identificar de forma exclusiva a localização do servidor que hospeda a API.

**/api/vendas**: Esta é a parte do caminho do endpoint que especifica o recurso ou a funcionalidade da API que você deseja acessar. No exemplo:

**/api**: Isso indica que a solicitação está indo para um caminho dentro da API, e "api" é uma convenção comum para endpoints de API.

**/vendas**: Isso especifica que o recurso sendo acessado é relacionado a vendas. Dependendo da estrutura da API, "/vendas" pode indicar diferentes ações relacionadas a vendas, como listar vendas, criar uma nova venda, atualizar vendas existentes etc.

Portanto, o exemplo completo do endpoint "https://minhaapi.com.br/api/vendas" indica que você está fazendo uma solicitação segura (HTTPS) para um recurso relacionado a vendas dentro da API hospedada no domínio "minhaapi.com.br". O servidor da API entenderá a solicitação e responderá com os dados ou ações apropriados relacionados a vendas.