**1.**

**a) Papel de *Controller* e *Model***

* **Model**  
  Representa a **camada de dados** da aplicação. É onde ficam as classes que definem as **entidades** (por exemplo, Produto, Cliente) e as regras de negócio relacionadas. O *Model* lida com:
  + Estrutura dos dados
  + Validações
  + Interações com banco de dados
* **Controller**  
  É a **camada de controle** que recebe as requisições do usuário (via rota/endpoint), processa essas requisições (chamando os *models* ou serviços) e retorna a resposta apropriada.  
  Ele funciona como um “intermediário” entre a **view** (ou cliente, como o navegador) e o **model**.
* **Por que herdamos de ControllerBase?**  
  Porque o ControllerBase já traz funcionalidades prontas do ASP.NET Core para:
  + Lidar com requisições HTTP
  + Retornar respostas no formato certo (JSON, XML, etc.)
  + Trabalhar com *Model Binding* e *Model Validation*  
    Herdar de ControllerBase evita que a gente precise reescrever todo esse código manualmente.

**b) Por que conseguimos executar apenas a rota GET pelo navegador?**

O navegador, quando você digita uma URL e aperta **Enter**, **sempre** faz uma requisição HTTP **GET** por padrão.  
Isso acontece porque:

* O campo de endereço do navegador serve para “buscar” recursos.
* Para fazer **POST**, **PUT** ou **DELETE**, precisamos enviar dados no corpo da requisição, o que não é suportado só digitando o endereço.  
  Para isso, usamos:
  + Formulários HTML com método POST
  + Ferramentas como Postman, Insomnia, cURL
  + Chamadas via JavaScript (fetch, axios, etc.)

**2.**

**a) Criar a aula** → **POST**

Criar um novo recurso no servidor (neste caso, uma aula).

**b) Remover um material** → **DELETE**

Apagar um recurso específico.

**c) Listar as disciplinas disponíveis para o professor no semestre** → **GET**

Buscar/consultar informações sem modificar nada no servidor.

**d) Listar os alunos da disciplina** → **GET**

Apenas leitura de dados, sem alterações.

**e) Atualizar um material** → **PUT** ou **PATCH**

* PUT → atualização completa do recurso.
* PATCH → atualização parcial (apenas alguns campos).

**f) Lançar uma nota** → **POST** ou **PUT** (depende)

* POST → se a nota está sendo lançada pela primeira vez.
* PUT → se está substituindo uma nota já existente.

**Resumo rápido:**

* **Ok** → Deu certo.
* **BadRequest** → O problema veio do cliente.
* **NotFound** → Não existe o que pediu.
* **Created** → Novo recurso criado.
* **Unauthorized / Forbid** → Problema de autenticação/autorização.

 **https://**

* **O que é:** Protocolo de comunicação usado na web.
* **Função:** Diz ao navegador ou cliente qual protocolo usar para se comunicar com o servidor.
* **Detalhe:** O https é a versão segura do http, usando criptografia (SSL/TLS) para proteger os dados durante a transmissão.

 **minhaapi.com.br**

* **O que é:** Nome de domínio (endereço do servidor).
* **Função:** Identifica o local onde a API está hospedada.
* **Detalhe:** O navegador ou cliente resolve esse domínio para um **endereço IP** através do DNS para encontrar o servidor correto.

 **/api/vendas**

* **O que é:** Caminho (path) do recurso.
* **Função:** Indica ao servidor **qual recurso** ou funcionalidade você quer acessar dentro do site ou API.
* **Detalhe:**
  + /api → normalmente indica que é um endpoint de API (Application Programming Interface).
  + /vendas → recurso específico, neste caso, provavelmente lista, cria ou manipula informações de vendas.