**Lista 1 – Desenvolvimento Web**

1-   
  
Front-end:

O front-end é a parte da aplicação que os usuários interagem diretamente. É responsável por exibir informações e permitir a interação do usuário. Isso inclui a interface do usuário, o design, a usabilidade e a experiência do usuário.

Duas tecnologias de front-end populares são:

HTML (HyperText Markup Language): É a linguagem de marcação usada para estruturar e organizar o conteúdo da página, como textos, imagens e links.

CSS (Cascading Style Sheets): É usado para estilizar a aparência do conteúdo HTML. Ele define cores, fontes, posicionamento e layout, tornando a página visualmente atraente e responsiva.

Back-end:

O back-end é a parte da aplicação que lida com a lógica de negócios, o processamento de dados, o armazenamento e a comunicação com bancos de dados e outros serviços. Ele cuida das operações por trás dos bastidores que tornam a aplicação funcional.

Duas tecnologias de back-end populares são:

Node.js: É um ambiente de tempo de execução baseado em JavaScript que permite a criação de aplicativos do lado do servidor. Ele é conhecido por sua eficiência e capacidade de construir aplicativos de rede escaláveis.

Python com Django: Django é um framework de desenvolvimento web em Python. Ele oferece ferramentas poderosas para criar aplicativos web robustos e seguros, automatizando muitos aspectos do desenvolvimento back-end.  
  
2 a)Controller:

O controller em uma arquitetura MVC (Model-View-Controller) é responsável por receber e processar as requisições do cliente (geralmente um navegador web) e coordenar as interações entre a interface do usuário (View) e a lógica de negócios (Model). Ele atua como um intermediário que encaminha as solicitações recebidas para os modelos apropriados, realiza operações de negócios e, em seguida, seleciona a visualização apropriada para exibir os resultados ao usuário. O controller lida com a lógica de controle, que determina como as requisições devem ser tratadas e como as respostas devem ser geradas.

Model:

O model representa a camada de negócios e lida com a lógica do domínio, o acesso aos dados e o gerenciamento do estado da aplicação. Ele encapsula as regras de negócios, as operações de banco de dados e a manipulação dos dados subjacentes. Os modelos são independentes da interface do usuário e das interações do cliente, o que os torna reutilizáveis e separados das preocupações de apresentação.

Herança de ControllerBase:

Ao criar um controller que herda de ControllerBase, você está seguindo um padrão comum em muitos frameworks web, como ASP.NET MVC e ASP.NET Core MVC. A classe ControllerBase é uma classe base fornecida pelo framework que contém funcionalidades comuns para os controladores. A herança de ControllerBase traz vantagens como:   
  
b)No contexto da navegação web, quando você digita uma URL no navegador e pressiona "Enter", o navegador faz uma solicitação HTTP para o servidor correspondente àquela URL. Existem vários tipos de solicitações HTTP, mas as duas mais comuns são as solicitações GET e POST. No entanto, apenas a solicitação GET pode ser facilmente executada diretamente no navegador, enquanto a solicitação POST não pode ser executada da mesma forma. Isso se deve principalmente à forma como o navegador interage com o servidor.

Aqui está o motivo pelo qual apenas a solicitação GET é facilmente executada no navegador:

Solicitação GET:

A solicitação GET é usada para buscar recursos do servidor, como páginas HTML, imagens, arquivos CSS, etc.

Quando você digita uma URL no navegador e pressiona "Enter", o navegador envia automaticamente uma solicitação GET para o servidor correspondente.

A solicitação GET é considerada segura e idempotente, o que significa que fazer várias solicitações GET para a mesma URL não deve causar efeitos colaterais.

Solicitação POST:

A solicitação POST é usada para enviar dados ao servidor para serem processados.

Ao contrário da solicitação GET, a solicitação POST não pode ser facilmente executada digitando uma URL no navegador. Isso ocorre porque a solicitação POST geralmente envolve o envio de dados do formulário no corpo da solicitação, enquanto a URL é apenas a parte de endereço.

Para executar uma solicitação POST a partir do navegador, normalmente é necessário um formulário HTML que defina o método POST e especifique a ação (URL) para a qual os dados do formulário serão enviados.

A solicitação POST não é considerada idempotente, pois pode ter efeitos colaterais no servidor, como inserção ou atualização de dados.

Portanto, a razão pela qual só conseguimos executar a solicitação GET diretamente no navegador é porque a interação com o servidor através do navegador é mais adequada para buscar recursos e exibir informações, enquanto o envio de dados e execução de ações geralmente requer o uso de formulários ou outras formas de interação mais complexas do usuário.  
  
c) O Swagger é uma ferramenta de código aberto amplamente utilizada para documentar, testar e gerenciar APIs de serviços da web. Ele permite que os desenvolvedores definam a estrutura da API, suas rotas, parâmetros, tipos de dados, métodos de autenticação e muito mais, tudo em um formato legível por máquina e humano. O Swagger é amplamente adotado na indústria de desenvolvimento de software devido aos benefícios que oferece. Seu papel principal é facilitar a criação, manutenção e uso de APIs de forma eficiente. Aqui estão algumas razões pelas quais o Swagger é utilizado:

Documentação Automática: O Swagger gera automaticamente documentação interativa da API a partir das definições fornecidas. Isso inclui detalhes sobre todas as rotas, parâmetros, métodos, tipos de dados, códigos de status, exemplos de solicitações e respostas, o que facilita a compreensão e o uso da API.

Teste da API: A interface gerada pelo Swagger permite que os desenvolvedores testem a API diretamente na documentação. Isso é útil para verificar o funcionamento da API antes de integrá-la em um aplicativo e para experimentar diferentes parâmetros e entradas.

Padronização e Consistência: O Swagger define um formato padrão para documentar APIs, o que promove a consistência na equipe de desenvolvimento e facilita a compreensão da API por parte de outros desenvolvedores.

Em resumo, o Swagger desempenha um papel crucial na criação, documentação e gerenciamento de APIs. Ele ajuda a economizar tempo, evitar erros de integração, melhorar a colaboração entre equipes e fornecer uma experiência mais fácil e eficiente tanto para os desenvolvedores que criam a API quanto para aqueles que a utilizam.

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 - Eles são usados em frameworks de desenvolvimento web para enviar respostas apropriadas para as solicitações feitas pelos clientes (geralmente navegadores ou outros sistemas) às APIs ou aplicativos web. Cada um desses métodos indica o estado da solicitação e a operação que foi realizada. Vou explicar a diferença entre alguns desses métodos comuns:

Ok():

Código HTTP: 200 OK

Significado: Indica que a solicitação foi bem-sucedida. Geralmente usado para retornar dados em uma resposta positiva. Por exemplo, após criar, atualizar ou recuperar dados com sucesso.

BadRequest():

Código HTTP: 400 Bad Request

Significado: Indica que a solicitação do cliente é inválida ou malformada. É usado quando os dados enviados pelo cliente não correspondem às expectativas do servidor, como parâmetros ausentes, formato incorreto, etc.

NotFound():

Código HTTP: 404 Not Found

Significado: Indica que o recurso solicitado não foi encontrado no servidor. É comumente usado quando o servidor não consegue encontrar a URL ou o caminho especificado na solicitação.

InternalServerError():

Código HTTP: 500 Internal Server Error

Significado: Indica um erro interno no servidor. É usado quando ocorre algum problema inesperado no processamento da solicitação que não pode ser tratado adequadamente. Geralmente, isso ocorre devido a um bug ou falha no servidor.

Unauthorized():

Código HTTP: 401 Unauthorized

Significado: Indica que o cliente não possui as credenciais válidas para acessar o recurso solicitado. É usado quando a autenticação é necessária e as credenciais não foram fornecidas ou são inválidas.

Forbid():

Código HTTP: 403 Forbidden

Significado: Indica que o cliente foi autenticado, mas não tem permissão para acessar o recurso solicitado. Isso ocorre quando o cliente não possui as permissões necessárias.

NoContent():

Código HTTP: 204 No Content

Significado: Indica que a solicitação foi bem-sucedida, mas não há conteúdo para ser retornado na resposta. Isso é comum em solicitações que resultam em alterações no servidor, mas não exigem dados de resposta.

Esses métodos de retorno de status HTTP ajudam a comunicar claramente o resultado da solicitação entre o cliente e o servidor, permitindo um fluxo de comunicação eficiente e indicando problemas específicos que possam ter ocorrido durante o processamento da solicitação.  
  
5 - https://:

O "https://" é o protocolo usado para acessar o recurso na web.

Ele indica que a conexão entre o cliente (navegador, aplicativo, etc.) e o servidor é segura e criptografada. O "s" em "https" representa "secure" (seguro).

O protocolo HTTPS é amplamente utilizado para proteger a privacidade e a integridade dos dados transmitidos entre o cliente e o servidor.

minhaapi.com.br:

Essa parte do URL é o domínio do servidor ao qual você está fazendo a solicitação.

O domínio é o nome do servidor onde a API está hospedada. No exemplo, "minhaapi.com.br" é o endereço da API que está sendo acessada.

Ele é usado para direcionar a solicitação ao servidor correto, identificando sua localização na rede.

/api/vendas:

Essa parte do URL especifica o caminho do recurso que está sendo acessado na API.

O caminho indica o local exato da API onde a solicitação deve ser roteada. No exemplo, "/api/vendas" é o caminho que provavelmente leva a um conjunto de funcionalidades relacionadas às vendas.

O caminho é usado para identificar a rota específica da API a ser acionada.

Portanto, ao juntar todas essas partes, o URL completo "https://minhaapi.com.br/api/vendas" representa uma solicitação segura a um recurso relacionado a vendas hospedado no domínio "minhaapi.com.br". Esse recurso específico será definido pela API e dependerá do que a API foi projetada para fazer em relação às vendas.