Lumen: API Rest com o Micro-framework do Laravel

- Esse curso capacita a:
  - Entender o estilo arquitetural REST e seus padrões;
  - Usar Eloquent para persistir os dados;
  - Definir recursos e rotas com Lumen;
  - Saber como trabalha com erros e códigos HTTP;
  - Aprender como funciona a autenticação com tokens JWT;

Aulas:

1. **Primeiro Endpoint:**
   - Para criar um projeto local: **composer create-project –prefer-dist laravel/lumen <nomeDaAPI>**;
     - Uma API devolve dados através do protocolo HTTP, muito usado para atender aplicações mobile ou desktop;
     - Podemos criar uma API com Laravel, e as rotas da API ficam no arquivo **api.php**;
     - O Laravel usa o prefixo **/api** nas rotas da API;
     - Uma coleção do Eloquent retornada no método se torna um JSON;
     - O Lumen é uma versão leve do Laravel específica para a criação de uma API;
     - Pode-se agrupar rotas para definir características comuns (por exemplo, prefixo);
     - O Eloquent fica desabilitado por padrão no Lumen, e pode-se habilitá-lo no arquivo **bootstrap/app.php**;

2. **Acesso ao recurso pela API:**
   - Verbos HTTP:
     GET [Pega todas ou pega uma específica do banco],
     POST [Insere um dado],
     PUT [Atualiza um dado por completo],
     DELETE [Remove um dado];
   - Padrão de Mensagens HTTP:
     2xx, indica que a requisição foi processada com sucesso
     3xx, indica que ao cliente uma ação a ser tomada para que a requisição possa ser concluída
     4xx, indica erro(s) na requisição causado pelo cliente
     5xx, indica que a requisição não foi concluída devido a erro(s) ocorrido(s) no servidor;
     - REST é um padrão arquitetural, que define regras como acessar/publicar um recurso;
     - O padrão REST aproveita o máximo do protocolo HTTP;
     - Usa-se as URLs, métodos, cabeçalhos e códigos para definir o acesso;

- Através do objeto **Illuminate\Http\Request**, pode-se ler os dados em JSON;
- Através do método **response()**, pode-se definir a resposta JSON e o código de resposta;
- O código 201 siginifica **Created**, 204 significa **No Content** e 404 significa **Not Found**;

3. **Manipulando dados pela interface HTTP:**
    - As rotas podem ter grupos e sub-grupos para não repetir nenhuma configuração;
    - A herança pode ser utilizada para reutilizar código entre controllers;
    - O ORM Eloquent possui Accessors e Mutators, para personalizar a forma de como os atributos são preenchidos e devolvidos:
        - Para personalizar o acessor, pode-se implementar o método **getNomeDoSeuAtributoAttribute($valor)**;
        - Para personalizar o mutator, pode-se implementar o método **setNomeDoSeuAtributoAttribute($valor);**

4. **Navegando e paginando recursos:**
    - Enviando dados que não são somente texto, como imagens, vídeos ou links, são comumente tratados como hypermidia ou hipermídia. A utilização de hipermídia como o motor de uma **API RESTful** é muito comum e amplamente utilizada. Para este tipo de técnica, se deu o nome de Hypermidia As The Engine Of the Application State, ou, **HATEOAS**;
    - Este componente de uma API é o que diferencia o padrão REST de qualquer outro padrão de arquitetura. Fornecer informações para que o cliente consiga navegar entre os recursos é extremamente útil e pode facilitar a utilização da solução;
        - É comum usar query params para definir o número da página e a quantidade de elementos por página;
        - O Eloquent já vem preparado para a paginação e possui um método **paginate()**;
            - A quantidade de itens por página pode ser definida na URL ou no próprio modelo (**$perPage)**;
            - Ao usar a paginação do Eloquent, o Laravel já devolve mais informações sobre a paginação (quantidade, primeira e última página, etc.);
            - Recursos podem ter sub-recursos, para reforçar o relacionamento;
            - A apresentação do recurso pode fornecer informações, como navegar para o sub-recurso, no qual, normalmente é usado o atributo **link** para navegar ao sub-recurso;

5. **Autenticação e Proteção das rotas:**
    - Precisa-se informar para o Lumen como o seu Middleware de autenticação pode realizar o login de um usuário, a partir de uma requisição. Para isso, utiliza-se o Service Provider, chamado **AuthServiceProvider**;
    - Service providers são formas de registrar e informar configurações para pacotes que serão utilizados na aplicação;
    - Caso queira-se adicionar uma ação sempre que um Model for salvo, por exemplo, pode-se configurar o que o Eloquent deve fazer neste caso, criando um novo Service Provider;
        - Normalmente uma API não guarda dados da sessão do usuário, devido a isso, é muito comum que ela autentique o usuário através de um token;
        - Um token é uma identificação (parecido com session id);
        - JWT (JSON Web Token) é um padrão para definir o token;
        - No Lumen, já existe um provedor de autenticação (**AuthServiceProvider),** ele também já fornece classes para representar o usuário (**User** e **GenericUser**);
        - O Lumen também já possui provedores de autenticação e middleware para proteger as rotas;