**o que são e para que servem as blades.**

Blades são componentes de visualização utilizados no Laravel para criar a camada de apresentação da aplicação. Eles são arquivos de template escritos em Blade, que é o mecanismo de template do Laravel. As blades servem para separar o HTML do código PHP, permitindo um design mais limpo e facilitando a manutenção e reutilização de trechos de código.

**Como funciona os comandos para criar a migrates e controller**

No Laravel, podemos criar migrations e controllers através de comandos Artisan, que é a interface de linha de comando do framework.

Criar uma Migration:

“php artisan make:migration create\_nome\_da\_tabela\_table”

Este comando gera um arquivo de migration na pasta database/migrations.

Criar um Controller:

“php artisan make:controller NomeDoController”

Este comando cria um novo controller na pasta app/Http/Controllers.

**Ccomo funciona o processo de criação de cada etapa**

Processo de Criação de Cada Etapa

Migration:

* + Crie uma migration para definir a estrutura das tabelas do banco de dados.
  + Escreva os campos e suas especificações dentro da migration.
  + Execute a migration com o comando php artisan migrate.

Model:

* + Crie um model para a tabela com o comando php artisan make:model NomeDoModel.
  + Defina os relacionamentos e atributos necessários dentro do model.

Controller:

* + Crie um controller para gerenciar as requisições HTTP e manipular os dados da aplicação.
  + Defina métodos dentro do controller para cada operação CRUD (Create, Read, Update, Delete).

Routes:

* + Defina as rotas no arquivo routes/web.php para mapear as URLs aos métodos do controller.

Views (Blades):

* + Crie arquivos de blade para a interface do usuário dentro da pasta resources/views.
  + Utilize os métodos do controller para enviar dados para as views e renderizá-las.

**explicar o crud**

CRUD é um acrônimo para Create, Read, Update, Delete. Representa as operações básicas de persistência de dados em um banco de dados:

* **Create:** Inserir novos dados.
* **Read:** Recuperar dados existentes.
* **Update:** Atualizar dados existentes.
* **Delete:** Excluir dados.

Cada uma dessas operações é tipicamente implementada como um método separado dentro de um controller no Laravel.

A pasta vendor é um diretório crucial em projetos que utilizam o Composer, um gerenciador de dependências para PHP. Ela armazena todos os pacotes de código de terceiros (bibliotecas externas) que o projeto utiliza. A pasta garante que todas as dependências do projeto sejam facilmente gerenciáveis e acessíveis, melhorando a modularidade e facilitando a manutenção do código.

Atualizações feitas para passar o CRUD do laravel 7 para o laravel 11:

Atualizar o composer pelo comando “ composer self-update” no terminal

Atualizar a rotas(routes) para:

<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\ProdutosController;

Route::get('/', function () {

    return view('welcome');

});

Route::resource('/produtos', ProdutosController::class);

Route::prefix('produto')->group(function () {

    Route::get('/novo', [ProdutosController::class, 'create']);

    Route::post('/novo', [ProdutosController::class, 'store'])->name('registrar\_produto');

    Route::get('/ver/{id}', [ProdutosController::class, 'show']);

    Route::get('/editar/{id}', [ProdutosController::class, 'edit']);

    Route::post('/editar/{id}', [ProdutosController::class, 'update'])->name('alterar\_produto');

    Route::get('/excluir/{id}', [ProdutosController::class, 'delete']);

    Route::post('/excluir/{id}', [ProdutosController::class, 'destroy'])->name('excluir\_produto');

});