1 و 2 و3 و 4 و5 و 6 و 7و 8 و23 و 24 و 27 و31 و33 و35 و38 و37 و 39 و43 و 45

تعالة النهاردة نعرف اهم الاسألة اللي ممكن تتسالها في الانترفيو لو انت

اوبجكت اورينتد بروجرامينجOop

هي طريقه تنظيميه للكود بيوفر ف الوقت فيها مبادئ بتحسن من الكود بتاعي يتكون (من اوبجكت وكلاس)

( اوبجكت ) باخد نسخه من الكلاس

كلاس هو الديزاين او الملف بتاعي ف بكون محتاجه كلاس واحد وانتج عدد لا نهائي من الاوبجكت

الكلاس دا بيكون فيه صفات اتربيوت والافعال ميثود والكونستراكتور

كل اوبجكت ليه نسخته المنفصله من الاتربيوت والميثود اللي موجوده ف الكلاس

دا هيجبلي كل الاتربيوت $user = new User

دا اتربيوت ال الاسم فقط $cat->name = "nemo";

دا بنادي ع الميثود $user->getUser() .

$this-> عن طريق method او attrb انا جوه ميثود عايزه ااكسس اي حاجه ف كلاس

------------------------------------------------------------------------------------------------------------------

**المقارنة بين الـ Constructor و الـ Destructor:**

بتتنفد اول حاجه لما باخد الاوبجكت Constructor

|  |
| --- |
|  |

ممكن ياخد برامتير

هو دايما public  
مش ليه return type and void

انا مطلبهاش لكن هي بتتنفذ لوحدها اول ما بنادي الاوبجكت اخر حاجه بتشتغل اخر حاجه ف الصفحه **Destructor**

-------------------------------------------------------------------------------------------------------------------

مبادئ ال opp

الوراثه Inheritance1-

كلاس بيورث من كلاس بيورث صفاته والميثود بيورث كل شئ جواه

عن طريق keyword extends

دا single Inheritance

في حاجه اسمها حلتلي مشكله السنجل trait , interface

لو عندي ميثود ف كلاس الاب واستخدمتها ف الابن وغيرت اللوجيك بيعمل اوفررايد ف كلاس الابن

اي هيا access modefiers 2- ..

انا بستعملها علشان احدد ان الداتا اشوفها في انهي مكان واحمي الداتا

access 3 اماكن (inside class يعني جوه ميثود من خلال this , inheretance child , opject)

public (inside class يعني جوه ميثود من خلال this , inheretance child , opject)

protecte (inside class يعني جوه ميثود من خلال this , child )

private (inside class يعني جوه ميثود من خلال this) ->public set method بحط قيمه , public get method بطبع او بحصل او بشوف قيمه

اية هو overloading & overriding ؟

عند استخدام **الـ Overriding**

يشير إلى **إعادة تعريف دالة** موجودة في الفئة الأصلية (أو الفئة الأب) داخل الفئة الفرعية (أو الفئة المشتقة) بحيث يتم **تغيير سلوكها**

**تتم كتابة نفس اسم الدالة** بنفس **الوسائط** ولكن مع **تنفيذ مختلف** في الفئة الفرعية

**keyword Final class**

مينفعش اغير من قيمه بتاعته

final -----> بنديها (calss ----> not extend (admin), method ---> not override)

بنديها ل كلاس او ميثود

مينفعش تورثه لو ادتها ل كلاس بكتب فاينل وبعدها كلاس وبعدها اسم الكلاس not extends

final class Admin

لو ادتها ل ميثود مينفعش اعمل عليها not override

methodChaning

بكتب الميثود بتاعتي ورا بعض عن طريق ->

وبنده الاوبجكت مره واحده بس

-3 absttraction

بستخدمه ف حالتين abstract , interface

abstract بستخدمه علشان اخفي البودي {}

interface & abstract

**keyword abstract** **class**

وظيفته انه ب hide implementation

الميثود ف الكلاس ملهاش بودي(){}

public abstract function setDesc();

abstract class ....

keyword abstract هو عكس final

abstractكونستراكتور و اكسس موديفير كلاس عادي بياخد ميثود وصفات بياخد كل حاجه

abstract بيتورث لكن مش ينفع اعمل منه اوبجكت

inheretance بيتورث لكن سنجل

بتتكتب مره واحده بس

abstract لو الميثود

لازم الكلاس abstract

abstract مش يكون جواه ميثود abstract ممكن الكلاس

-4Interface

غرض منه ان اورثه implements ممكن اعمل اكتر من

interface operation{}

abstract هو بيشبه

interface فيها ايه ؟ فيها مجموعه ميثود

abstract لكن مش بكتب abstract الميثود اللي موجوده فيه نوعها

abstract methode

public function test1();

الميثود اللي فيها بستخدمها ك توجيهي ف اسماء الميثود

\_\_construct ولا attrb مش فيه ميثود عاديه ولا

access public بياخد

مش ينفع اخد منه اوبجكت

اي الفرق بين class & trait؟

1. الـ Class

الـ class هي قالب يُستخدم لتعريف (objects) واحتواء (المتغيرات) و (الدوال). يمكن اعتبار الـ class تعريفًا لنوع جديد من الكائنات. يمكن إنشاء objects من الـ class باستخدام الـ constructor (الدالة الخاصة بالبناء). الـ class يمكن أن تحتوي على:

خصائص (Properties) أو متغيرات.

دوال (Methods) تنفذ سلوكيات معينة.

إمكانية الوراثة (Inheritance) حيث يمكن لـ class أن ترث من class آخر.

الـ class يمكن أن تحتوي على:

مُحددات وصول (مثل: private, public, protected).

يمكن أن تحتوي على (Constructors).

Trait

بستخدمه في اكتر من كلاس

بستخدمه ف اعاده استخدام الكود بين الكلاسات

مش ينفع اعمل \_\_construct

abstract مش ينفع احط فيه ميثود

مش بتتورث ومش باخد منها اوبجكت

inheretance هيحل مشكله كلاس سنجل

مش بياخد اتربيوت

بياخد ميثود عاديه

trait

يمكن أن يتم دمجه في الـ class

Const ..

ثابته محدش يعمل عليها override

مفيش this

:: هاكسس عليها من اسم الكلاس

// او

لو انا ف الكلاس هاكسس عليها من SELF::

لو انا ف الابن هاكسس عليها من parent::

لو هاكسس من الاوبجت اسم الكلاس::

Static ..

بتفيدني ان اطلبها باسم الكلاس::

بتاخدها (atrrbu , methode) ولا يمكن استخدمهم

access method static جوه (static -> static only)

access method non static جوه (non static -> all)

access (inside class , child -- parent:: , opject ---> calss name :: )

يتم الوصول إلى static عبر class::property

Self::$country =

يمكن استدعاء الدالة باستخدام class::method بدلاً من إنشاء اوبجكت من الـ class

اي الفرق بين self & this ؟ ..

self تُستخدم للوصول إلى الدوال والخصائص الثابتة (static) داخل الـ class. ولا تشير إلى الكائنات.

this تُستخدم للوصول إلى الدوال والخصائص غير الثابتة (non-static) داخل الـ object (الكائن).

Namespace

هو ميزة تُستخدم لتنظيم الأكواد وتقسيمها إلى مجموعات منطقية لتجنب التعارضات بين الأسماء

هو عبارة عن طريقة لتنظيم وتجميع الكلاسات

يعمل كعنوان فريد لكل عنصر من هذه العناصر، مما يمنع التعارضات بين الأسماء ويجعل من السهل العثور على العناصر

يتم عادة تعريفها في أعلى ملفات الكود باستخدام الكود التالي

namespace App\Http\Controllers;

use الوصول إلى الكائنات يمكنك فعل ذلك عن طريق استخدام

هناك قسم يسمى autoload

في هذا الملف، حيث يتم تحديد كيفية تحميل الكلاسات بناءً على هيكل المجلدات

ال namespace هي متل ال path يلي بدلني عالفايل يلي عم استخدمو

وهاد الشي بخليني اقدر سمي اكتر من فايل بنفس الاسم واستخدمن بدون ما يصير في تعارض بيناتن

مبادئ ال بيخيلي السوفت وير قوي اطور فيه اعدل عليه اضيف فيوتشر solid

**singel Responsability (SRP)**

يعني كل كلاس بيكون مسؤول عن حاجه واحده

بيحل مشكله ان في كلاس واحد في كميه ميثود كتير مش مرتبطه ببعض

بنحلها ب ان في كلاس كتير بس فيها ميثود قليله ومتشابهه مع بعض

يعني نعمل كل كلاس ميثود اللي هو متخصص فيها

مميزاته سهل ان افهمه / كلين كود / كود منظم

مثال كلاس ال auth

**open for extension, close for edit , modification**

مقفول للتعديل ومفتوح للوراثه

هعمل كلاس intereface , abstract /وجواه ميثود abstract

ومش هعدل ع main function

intereface يكون افضل

**liskove substation**

تكلم عن فكره الوراثه

متورثش كلاس مش هتستعمل كل الميثود اللي فيه حتى لو كلاس عادي

لو كلاس عادي هنرجع ل اول مبدا

لكن لو كلاس ابستركت او انترفيس كلاس يورث كلاس

**Interface Segregation (ISP)**

متعملش امبيليمت كلاس مش هتستعمل كل الميثود اللي فيه

إذا كنت بحاجة إلى تخزين بيانات صغيرة وأقل حساسية على جانب العميل مثل تفضيلات المستخدم، استخدم Cookies.

اما cookies بتخزن البيانات مباشرا علي جهاز المستخدم و بيرسل مع كل طلب الي السيرفر ال session بتكون امان لكن cookies اخف في الحمل .

أما إذا كنت بحاجة إلى إدارة بيانات حساسة أو كبيرة أو تستمر طوال الجلسة مثل تفاصيل تسجيل الدخول، استخدم Sessions.

session بيخزن البيانات علي السيرفر و بيستخدم معرف فريد اسمو session\_id بيكون محفوظ علي cookies علي بيانات المستخدم

LARAVEL

اي هو template engine 1 في لارافيل

الـ Template Engine بتساعدك تدمج الأكواد البرمجية زي PHP مع HTML بطريقة سهلة ومرتبة. بدل ما تستخدم أكواد PHP جوه الـ HTML بشكل تقليدي، الـ Template Engine زي Blade في Laravel بيخليك تكتب الأكواد بطريقة أنظف وأوضح.

ليه Blade يعتبر مميز؟

1- تبسيط كتابة الأكواد: Blade بيبسط كتابة الكود اللي فيه دمج بين HTML و PHP. وده بيخلي شغلك أسهل في التعامل مع تصميمات الصفحات.

2- التوريث (Inheritance): مع Blade، تقدر تعمل Layout رئيسي تستخدمه في كل الصفحات، بدل ما تكرر نفس الهيدر والفوتر في كل صفحة. الفكرة إنك بتكتب تصميم واحد وتورثه لكل الصفحات المختلفة، وده بيختصر وقت كبير.

3- إعادة الاستخدام: تقدر تعيد استخدام أجزاء معينة من التصميم في أكتر من مكان من غير ما تكرر الكود. يعني لو عندك جزء ثابت، ممكن تعمله مرة وتستخدمه في كل مكان تحتاجه.

4- سهولة التعامل مع الأكواد الشرطية: Blade بيسهل عليك كتابة الشروط والحلقات في الكود، بطريقة منظمة وواضحة، من غير تعقيد.

ليه Blade مهم في Laravel؟

لأن الـ Blade بيساعدك تبني تطبيقات ويب بشكل أسرع وأنظف. بيبسط عليك كتابة الأكواد المعقدة ويخلي تصميماتك أكثر تنظيمًا. وده بيسهل صيانة الكود وتطويره على المدى الطويل.

2 هنكمل كلامنا عن ال Global Scopes, Local Scopes و إيه الفرق بينهم.

ال Global Scopes ف لاراڤيل هي طريقة بنستخدمها عشان نضيف conditions معينه بشكل تلقائي على كل ال queries ال بتتعمل على Model محدد.

يعني هنحط سيناريو إنك عندك موديل معين و كل مره بتجيب الداتا منه بتحتاج تضيف شرط معين، بدل ما تضيف الشرط دا كل مره فالكود، ممكن تستخدم ال Global Scope عشان تطبق الشرط دا automatically على كل ال queries الخاصة بالموديل دا.

خلينا نجرب نعمل Global Scope،

لو دلوقتي عندنا table لل users و ال users دول ممكن يكونوا active أو لا و إحنا فكل مره عايزين لما نعمل أي query ال يجيلنا اليوزرس الأكتيڤ بس.

هنكريت Scope مثلًا اسمه ActiveScope، و هنحط فيه ال condition ال إحنا عايزينه، عن طريق ميثود apply ال ف ال scope زي ما هيتوضح فالمثال تحت.

بعد كده إحنا محتاجين نربط ال Scope دا بالموديل بتاعنا، هنيجي ف ال model فحالتنا ال User ف ال booted function و هنقوله ي add ال global scope ال إحنا عايزينه للموديل دا.

دلوقتي لو إحنا عملنا أي query ال results بتاعتها إنه يرجع يوزرس هيرجعلنا ال Active Users بس من غير ما نكتب أي كود زياده.

• طب لو أنا عايزه كل اليوزرس ف case معينه ؟ هستخدم withoutGlobalScope و هكتب اسم ال scope ال مش عايزاه يتنفذ، لو مش عايزه أي scope من ال global يتنفذوا هنكتب withoutGlobalScopes

• طيب بالنسبة لل Local Scopes ؟ دا بستخدمه

دا بنستخدمه فحالة إن إحنا مش هنبقى محتاجين فكل مره أنادي الموديل إنه يحقق ال Scope automatically .

و فالحاله دي مش هنادي الموديل بس، هنادي الموديل بال scope بتاعه ال أنا عايزاه يتحقق ف ال case المعينه دي.

و ساعتها هنكتب ال scope ف ال model بالطريقه دي (scopeName) ال هي camelCase و ال condition ال عايزينه بردو.

 public function scopeActive($query){

        $query->where("status",1);

    }

    public function scopeActiveUser($query){

        $query->whereHas('user',function($user){

            $user->whereStatus(1);

        });

    }

        $query = Post::query()->with(['user' , 'category' , 'images'])->activeUser()->activeCategory()->active();

• وفيه حاله كمان من ال global scopes ال هي ال anonymous ، و دي لو إحنا مش عايزين نكريت فايل مخصوص لل global scope، و الطريقة بتاعتها ال ف آخر صورتين، هو نفس الموضوع بردو هنحط الميثود booted فالموديل و هنكتب اللوچيك ال إحنا عايزينه .

3 النهارده هنتكلم عن مفهوم Routing in Laravel، واللي هو واحد من أساسيات الفريم وورك القوي ده. زي ما شفنا في MVC، التنظيم بيخلي الكود أسهل في التعامل والصيانة، والراوتنج بيكون الخطوة الأولى لتنظيم الروابط (URLs) وتوجيهها للوظائف المناسبة في التطبيق. تعالوا نفهم الموضوع ببساطة 👇🏻👇🏻

إيه هو الراوتنج (Routing)؟

هو المسؤول عن تحديد إيه اللي هيحصل لما المستخدم يفتح رابط معين.

يعني ببساطة لما حد يكتب عنوان معين في المتصفح الراوت بيقرر هيوديه لأنهي جزء من التطبيق

🚀 إزاي بيشتغل Routing في Laravel؟

Laravel بيستخدم ملف اسمه web.php (موجود في فولدر routes)

عشان نحدد الروابط المطلوبة كل راوت بنكتبه بيكون مربوط بصفحة، دالة، أو كلاس

🎯 ليه الراوتنج مهم🔔؟

1.تنظيم التطبيق: الروابط كلها بتبقى واضحة ومترتبة.

2.مرونة عالية: تقدر تربط الروابط بوظائف أو كلاسز بسهولة.

3.الأمان: باستخدام Middleware، تقدر تتحكم في مين يقدر يدخل روابط معينة

4 اية هو composer ؟

تول بنضيف منها المكتبات والباكدجات اللى محتاجين نتعمل معاها فى المشروع

**3. PSR-4: Autoloading Standard**

* **الوصف**: يحدد كيفية تنظيم الفئات (Classes) في المشروع وربطها بمسارات الملفات لدعم التحميل التلقائي (Autoloading) عبر أدوات مثل Composer.

 **تقليل الأخطاء**: من خلال اتباع ممارسات موحدة.

* جعل الكود أكثر قابلية للقراءة والفهم.
* PSR-7 يوفر واجهات لمعالجة طلبات واستجابات HTTP، وهو مفيد في تطبيقات API

5 إيه هو الـ Seeder في Laravel؟ 🌱🤔

ببساطة أداة بتساعدك تملأ قاعدة البيانات بتاعتك ببيانات مبدئية أو بيانات تجريبية بسرعة. يعني بدل ما تكتب بيانات يدوية كل مرة، تقدر تستخدم الـ Seeder وتخلي Laravel يضيف البيانات دي بشكل آلي، وده بيبقى مفيد جداً وقت تطوير التطبيق أو تجربته

إزاي نعمل Seeder؟

أول حاجة، بننشئ Seeder جديد باستخدام الأمر ده في التيرمينال:

php artisan make:seeder UserSeeder

بعد كده، هتلاقي ملف جديد اسمه UserSeeder.php في المسار database/seeders، وده الملف اللي هنكتب فيه البيانات اللي عايزين نضيفها.

زاي نضيف البيانات؟

داخل ملف الـ Seeder اللي أنشأناه، هنستخدم كود Laravel لإضافة البيانات اللي عايزينها. على سبيل المثال، لو عايز تضيف مستخدمين جدد، هتكتب حاجة شبه كده:

()public function run

}

;()App\Models\User::factory()->count(10)->create\

{

الكود ده بيقول للـ Seeder إنه يضيف 10 مستخدمين للجدول بتاع المستخدمين (users table)، وده عن طريق Factory اللي بتساعدك تولد بيانات وهمية بطريقة سهلة وسريعة.

تشغيل الـ Seeder

بعد ما تخلص كتابة الكود، تيجي للخطوة الأخيرة اللي هي تنفيذ الـ Seeder فعلياً وتشغيله على قاعدة البيانات. بنعمل كده بالأمر ده:

php artisan db:seed

ولو عايز تشغّل Seeder معين بس مش كل الـ Seeders اللي عندك، ممكن تستخدم الأمر ده وتحدد اسم الـ Seeder:

php artisan db:seed --class=UserSeeder

الفايدة من الـ Seeder

الـ Seeder مفيد في مراحل التطوير والتجريب، بحيث تقدر تعبّي قاعدة البيانات ببيانات واقعية بدون تعب أو وقت طويل، وتبدأ تجرّب على التطبيق كأنك عندك بيانات حقيقية.

5 . ما هي الـ Factories ؟

تُستخدم Factories لإنشاء بيانات وهمية (Fake Data) بسهولة وسرعة أثناء تطوير التطبيقات. هذا يسهل عليك اختبار الميزات الجديدة دون الحاجة إلى إدخال البيانات يدويًا.

توفير الوقت: يمكنك إنشاء كميات ضخمة من البيانات الوهمية في ثوانٍ،

1. **اشرح ما هو Event و Listener؟**

* Event: حدث يحصل في النظام.
* Listener: يقوم بالاستجابة للحدث.
* $validator = Validator::make($request->all(), [
* $validated = $request->validate([
* public function authorize()
* {
* return true; // السماح للمستخدم بتنفيذ هذا الطلب
* }
* public function rules()
* {
* return [
* 'name' => 'required|string|max:255',
* 'email' => 'required|email|unique:users,email',
* 'password' => 'required|min:8|confirmed',
* ];

6عارف Accessors & Mutators!؟

**Mutators**: بتستخدم لما حابب تعدل البيانات قبل ما تتخزن في الداتابيز/ تعديل البيانات قبل التخزين

يتم استخدامها لتعديل البيانات قبل حفظها في قاعدة البيانات

* **Accessors**: بتستخدم لما حابب تعدل البيانات قبل ما تُعرض أو بعد ما تسحبها من الداتابيز.  
  طب ازاي يعني لو عايز تعدل البيانات قبل ما تتخزن، زي إنك تخزن الأرقام بتنسيق معين، تقدر تعمل كده في الموديل

    public function setPhoneNumberAttribute($value){

          return $this->attributes[PhoneNumber] = :+2: . $ value;

    }

### هي طريقة لتحويل البيانات عند استرجاعها من قاعدة البيانات قبل عرضها Accessor:

 // accessories بعدل عليها وهي طالعه من الداتا بيز

    // بجبها من الداتا بيز get

    // set بنحطها ف الداتا بيز

    // public function getStatusAttripute(){

    //     return $this->attributes['status'] == 1 ? 'Active' : 'Not Active';

    // }

public function getStatusAttribute($value){

        return $value == 1 ? 'Active' : 'Inactive' ;

    }

7 شرح الـ CSRF

ده نوع من الهجمات الإلكترونية اللي بتحصل يحاول يخلّي مستخدم ينفذ طلبات بدون ما هو يعرف

كيف نحمي نفسنا؟

في Laravel، الـ CSRF بتعمل حماية ضد النوع ده من الهجمات. Laravel بيضيف توكن خاص في كل فورم وبراوزر العميل عشان يتأكد من صحة الطلب. لما المستخدم يرسل طلب، Laravel بيتحقق من التوكن ده وإذا ما كان موجود أو صحيح، بيرفض الطلب.

كيف بنضيف حماية الـ CSRF في Laravel:

1. التوكن تلقائي: Laravel تلقائيًا بيضيف التوكن في أي فورم بتستخدمه.

2. التحقق من التوكن: لمن المستخدم يرسل الطلب، Laravel بيتحقق من التوكن ويتأكد من صحة الطلب.

الـ CSRF بيخليك مطمئن إن طلباتك كلها بتجي من المستخدم الفعلي

8  تحسين أداء Laravel مع Eager Loading

تتيح لك تحميل العلاقات المرتبطة بالنموذج في استعلام واحد

يحسن الأداء بشكل كبير عند التعامل مع بيانات مترابطة

لنفترض أن لديك نموذجين: Post وComment، حيث يحتوي كل Post على عدة Comment. عند استخدام Eager Loading

$posts = Post::with('comments')->get();

تجنب استخدام Lazy Loading في الحلقات (loops) لتفادي الاستعلامات المتكررة التي قد تؤدي إلى بطء التطبيق.

9 اية هو transaction

هي عملية في قاعدة البيانات اللي بتضمن إن مجموعة أوامر (Queries) تتحقق يا كلها مع بعض، يا كلها ما تتحققش خالص. يعني لو حصلت مشكلة في نص الطريق، كل حاجة بترجع زي ما كانت كأنها ما حصلتش

زي إنك تسجل طلب شراء وتخصم الكمية من المخزون، لو واحدة منهم فشلت، لازم التانية كمان ما تحصلش

ببدأ Transaction باستخدام

DB::beginTransaction( )

• لو كل العمليات نجحت، بأكدها بـ ( )DB::commit

• لو حصل أي خطأ، أرجع كل حاجة زي ما كانت ب ( )DB::rollBack دي بحطها ف catch

Laravel request life cycle10

أول ما المستخدم يبعت طلب (request)، سواء فتح صفحة أو بعت فورم، الطلب بيروح على ملف public/index.php.

public/index.php. :- البراوزر او اليوزر قادر انه يوصله بيكون موجود ف الملف دا مثل css , js , img ,

الملف index بيروح ، يستدعي الملف vendor/autoload.php عشان يجيب كل المكتبات المطلوبة،

وكمان ملف bootstrap/app.php اللي بيعمل إنستنس للتطبيق بيسموه بعد كده services container

بعد كده،الطلب بيروح على app/Http/Kernel.php اللي بيحمل الـ middlewares وبيجهز التطبيق.

اول حاجه ف لارافيل ف Kernel بيتم تحميل ملفات ال conf بيحتوي ع ملفات مختلفه كل ملف بحتوي ع موضوع معين بيعمل تحميل بس مش بينفذها الا لو نا اللي طلبتها

تاني حاجه ServiceProvider هو مجرد كلاس بيحتوي ع 2 ميثود بيحط بداخلها كود اللي محتاجها تتنفذ ف كل ريكويست مثل اللغه او صلاحيات

الطلب بعد كده بيعدي على الـ middleware اللي بيتأكد من حاجات زي الأمان والدخول.

عمليه ال dispatch requests : هي بناء ع مفهوم ال mvc

بعد كده، الطلب بيوصل لـ Router اللي بيحدد الـ Controller المناسب على حسب الرابط.

الـ Controller بينفذ منطق التطبيق، سواء جلب بيانات من قاعدة البيانات أو تنفيذ عمليات معينة.

في النهاية، الـ Controller بيرجع الـ Response أو الـ View اللي هيتعرض للمستخدم.

Middleware11

هو حلقة وصل بين (Controller) , (Router)

هو حلقة وصل بين الطلب (Request) والاستجابة (Response)

التحقق من الصلاحيات: التأكد من أن المستخدم لديه الصلاحية للوصول إلى مورد معين.

التحقق من التوثيق: التأكد من أن المستخدم مسجل الدخول.

التحقق من الحاله

بعد إنشاء الميدلوير، يجب تسجيله في ملف app/Http/Kernel.php او bootstrap/app

بعد التسجيل، يمكنك استخدام الميدلوير في المسارات (Routes) أو الـ **Controllers**.

مثال ع **Controllers**

public function \_\_construct(){

        $this->middleware("can:admins");

    }

Migration

// migration اسم الجدول جمع - php artisan make:migration creat\_categories\_table

// php artisan make:migration update\_in\_categories\_table

php artisan make:migration modify\_column\_in\_table\_name --table=table\_name

// php artisan make:migration add\_to\_categories\_table

او

php artisan make:migration add\_site\_desc\_to\_settings\_table --table=settings

// عايزه ارفعه لل phpMyAdmin - php artisan migrate

php artisan make:migration change\_site\_desc\_column\_type\_in\_settings\_table

php artisan make:migration modify\_column\_data\_type\_in\_table\_name --table=table\_name

public function up()

{

Schema::table('settings', function (Blueprint $table) {

$table->json('site\_desc')->change();

});

}

public function down()

{

Schema::table('settings', function (Blueprint $table) {

$table->text('site\_desc')->change();

});

}

امتي استخدم sql join و امتي استخدم ORM

في Laravel، يمكنك كتابة استعلامات SQL مباشرة باستخدام **Query Builder**

يسمح بكتابة استعلامات بطريقة برمجية دون الاعتماد على Model.

لنفرض أن لدينا جدولين في قاعدة البيانات:

* **employees** (الموظفين).
* **departments** (الأقسام)

مثال

use Illuminate\Support\Facades\DB;

$employees = DB::table('employees')

->join('departments', 'employees.department\_id', '=', 'departments.department\_id')

->select('employees.name as employee\_name', 'departments.department\_name')

->get();

foreach ($employees as $employee) {

echo $employee->employee\_name . ' works in ' . $employee->department\_name . '<br>';

}

استخدام ORM (Eloquent) في Laravel:

لتسهيل التفاعل مع قاعدة البيانات باستخدام الكائنات. يمكنك تحديد العلاقات بين الجداول بسهولة باستخدام Eloquent، بدلاً من كتابة استعلامات SQL يدويًا.

من ناحيه الكود اقل وتعامل مع العلاقات اسهل

تعريف النماذج (Models):

في Eloquent، سنحتاج إلى إنشاء نموذج (Model) لكل جدول في قاعدة البيانات. على سبيل المثال:

نموذج Employee

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Employee extends Model

{

protected $table = 'employees';

public function department()

{

return $this->belongsTo(Department::class, 'department\_id');

}

$employees = Employee::with('department')->get();

foreach ($employees as $employee) {

echo $employee->name . ' works in ' . $employee->department->department\_name . '<br>';

}

# **Service Container**

-إيه هو الـ Service Container في Laravel؟

-ببساطة :- هو نظام Dependency Injection Container بيستخدمه Laravel لإدارة الاعتمادات (dependencies) وربط الـ classes ببعضها بطريقة مرنة وقابلة للاختبار.

مثلاً لو عندي كلاس بيحتاج Repository معين، بدل ما أعمل new Repository جوه الكلاس، ممكن Laravel يحقنه مباشرة من خلال الـ constructor.

**Dependency Injection**:

 protected $brandService;

    public function \_\_construct(BrandService $brandService)

    {

        $this->brandService = $brandService;

    }

-طب الفرق بين Service Container و Service Provider؟

Service Providers هم المكان الرئيسي في Laravel اللي بنستخدمه علشان نربط الخدمات (services) أو الكلاسات في الـ Service Container.

الـService Container:-

-التعريف:- نظام داخلي لإدارة الاعتمادات وربط الكلاسات.

-الدور:- هو اللي بينفذ عملية الـ binding ويحفظ الربط.

-الاستخدام:- بنستخدمه علشان نربط أو نحقن الكلاسات.

-العلاقة:- بيُستخدم داخل Service Provider أو أي مكان تاني.

الـService Provider:-

-التعريف:- كلاس بيُستخدم لتسجيل الخدمات في التطبيق.

-الدور:- بيحتوي على كود الـ binding أو الإعدادات الخاصة بالخدمات.

-الاستخدام:- بنستخدمه علشان نسجل خدمات أو نجهز إعدادات.

-العلاقة:- بيستخدم الـ Container داخله.

يعني ببساطة:- الـ Service Provider هو المكان اللي بكتب فيه الكود اللي بيستخدم الـ Service Container.  
  
على سبيل المثال، لو عايز أربط Interface بـ Class معين، بكتبه داخل دالة register() في الـ Provider، وبلّغ Laravel من خلاله إن فيه خدمة جاهزة للاستخدام

-ايه الفرق بين الـ bind والـ singleton في Service Container؟  
-bind: كل مرة Laravel بيحقن فيها الخدمة → بيعمل كائن جديد. ينشئ كائن جديد في كل مرة  
-Laravel :singleton بيعمل الكائن مرة واحدة بس، وبيحقنه في كل مكان محتاجه.

💡 الفكرة إن الـ bind والـ singleton بيخدموا نفس الهدف، لكن الفرق في عدد مرات إنشاء الكائن.

        $this->app->bind(DoctorRepositoryInterface::class, DoctorRepository::class);

Façade هو static كلاس

هو معتمد بشكل رئيسي من services container

هو عباره عن كلاس مؤشر ع اوبجيكت موجود ف services container وبتسمحلي ان انا اعمل اكسس بطريقه static

هو مجرد كلاس بيخليني استدعى ميثود بطريقه static موجوده داخل اوبجيكت مخزنه داخل services container

هو طريقه للوصول لنفس الاوبجيكت بس بطريقه static

بيعطيني اكسس لل services container او الوبجيكت اللي ف services container بطريقه static

**Cache**

ايه هو الكاشين ان انا عندي ببدا ان انا اخزن فيه بيانات تمام بشكل مؤقت وببدا ان انا بجيب البيانات دي بسرعه اكبر من ان انا اجيبها من الداتا بيس طب ليه انا بجيب البيانات دي بسرعه ليه الكاشين سريع الكاشين سريع ليه لانه بيخزن البيانات بتاعته او الداتا بتاعته في الرام تمام على عكس طبعا الداتا بيس بتكون البيانات عندي متخزنه على الديسك

resource لاي API في Laravel

ولا الresource هو انك تعمل بتفصل اسامي ال columns الاصلية اللي جاية من Database عن اللي بتطلع للفرونت اند او في API طيب ليه بعمل حاجة زي كده :

• اولا عشان اتحكم في البيانات اللي هتتعرض في الapi و اخفي اي بيانات حساسة او غير ضرورية

• ثانيا عشان لو حصل اي تغيير مستقبلي في اسامي ال columns في الداتا بيز ميأثرش علي التطبيقات

• ثالثا الأمان من خلال فصل أسماء columns ، بتحمي البنية الداخلية لقواعد بياناتك من ان حد ملهوش صلاحية يقدر يشوف اسامي ال columns

PostResource::make بترجع بوست واحد

// new PostResource بترجع بوست واحد

// لو عندي سنجل بوست هستخدم PostResource new , make

//new PostResource::collection بترجع مجموعه

What is throttling and how to implement it in Laravel?

ال throttle هى طريقة بعمل بيها تحكم لعدد ال Requests اللى هتروح للسيرفر فى وقت معين ممكن استخدمها واطبقها على ال route من خلال ال

وبالنسبة لل rate limit يمكن تعريفة فى service provider بالشكل ده

RateLimiter::for('api', function (Request $request) {

            return Limit::perMinute(20)->by($request->user()?->id ?: $request->ip())->response(function(){

                return apiResponse('429','Try After Minnute');

            });

        });

Git hub

Git

Version control هو سوفت وير نوع من انواع ال

Repository : المكان اللي بحط فيه اكواد المشروع بتاعي لكل بروجيكت ريبو واحده

Branch : فيه فرع رئيسي من البروجيكت وفي فروع اخري بتخرج من الفرع الرئيسي

Local Repository : ع جهازي لسه مترفعش السيرفر

Remote Repository : git hup , gitlap , or server

Commite : الكود جاهز لرفع من اللوكال الي السيرفر

Clone : استنساخ يعني في بروجيكت ع السيرفر وانا عايزاه ع جهازي بجيبه من السيرفر الي الوكال عندي ع جهازي

git clone linke

Push : برفع البروجيكت بتاعي من اللوكال جهازي الي السيرفر ريموت ريبو

يوجد 3 اوامر

git add سم الفايل, git commit -m "first commit", git push

لوعايز ارجع بعد ما عملت ادد git reset head واسم الملف

Pull : واحد زميلي رفع تعديلات وانا عايزاها ع جهازي

**Git branching**

git branch بيجبلي عندي كام برانش

create : git branch [branche\_name] انشا برانش جديد

switch : git checkout [branche\_name] اللي عايزه ارحو كده هعمل سويتش

git checkout –b [branche\_name] كده معناه انشات ملف جديد وروحتلو عملت سويتش يعني

merge : git merge [branche\_name]

touch login.php اكريت فايل جديد \اسم الفايل

PHP

Datatype , type casting , typ juggling

Arithmetic Operators => + , - , \* , % , / , \*\* اسس , +$a identity , -$a navigation ,

Assignment Operators => += , -= , \*= , %= , /= . \*\*= ,

Comparison Operators =>

== equal بنقارن بين قيمتين من ناحيه القيمه فقط بغض النظر عن نوع البيانات

=== identical بنقارن بين قيمتين من ناحيه القيمه ونوع البيانات

!= not equal ,

<> not equal ,

!== not identical

>= , > , <= , < ,<=> -1- 0 - 1,

&& , || , ! ,

While تنفذ **طالما الشرط صحيح**، ويتم التحقق من الشرط **قبل** تنفيذ الكود داخل الحلقة.

إذا كان الشرط خاطئًا من البداية، **لن يتم تنفيذ الكود إطلاقًا**.

يتم تنفيذ الكود **مرة واحدة على الأقل**، ثم يتم التحقق من الشرط. إذا كان الشرط صحيحًا do While

وحتى لو كان الشرط خاطئًا من البداية، سيتم تنفيذ الكود **مرة واحدة على الأقل**.

Include

 إذا **فشل** في تضمين الملف (مثلاً الملف غير موجود)، **يظهر تحذير (warning)** ولكن **يستمر تنفيذ السكريبت**.

Include once

لو انا عامله تضمين لملف مش بيخليني اعمل مره اخري

Require

 إذا **فشل** في تضمين الملف، **يظهر خطأ قاتل (fatal error)** و **يتوقف تنفيذ السكريبت بالكامل**.

عدم إدراج نفس الملف أكثر من مرة Require once

**ما الفرق بين echo و print؟**

* echo: أسرع، ويمكن طباعة أكثر من قيمة.
* print: تُرجع قيمة (1)، لذا يمكن استخدامها في التعبيرات.