QCM Larvel

Question 1 :

Quelle commande crée un nouveau projet Laravel ?

```
> composer create-project laravel/laravel nom_du_projet
```

Question 2 :

Comment passer des données à une vue ?

On peut utiliser la méthode ***with()*** sur l'objet de vue ou utiliser la fonction helper ***view()***. Voici deux exemples :

Utilisation de la méthode ***with()*** :

```php
return view('nom-de-la-vue')->with('variable', $valeur);
```

Utilisation de la fonction helper ***view()*** :

```php
return view('nom-de-la-vue', ['variable' => $valeur]);
```

Question 3 :

Quel est le nom du moteur de template utilisé par Laravel ?

Laravel utilise le moteur de template ***Blade.***

Question 4 :

Comment peut-on valider les données de requête dans un contrôleur ?

En utilisant la classe ***Validator*** :

```php
public function mahboub(Request $request){
    $validator = Validator::make($request->all(), [
        'nom' => 'required|string|max:32',
    ]);
    if ($validator->fails()) {
        return redirect('route')->withErrors($validator)->withInput();
    }
}
```

Question 5 :

Quelle commande utilise-t-on pour créer un modèle ?

```
> php artisan make:model NomDuModele
```

Question 6 :

Comment ajoute-t-on une colonne à une table existante via une migration ?

- Créez une nouvelle migration avec la commande :

```
> php artisan make:migration add_column_to_table nom_de_la_table
```

- Modifiez la méthode ***up()*** de la migration pour ajouter la colonne :

```php
public function up()
{
    Schema::table('nom_de_la_table', function (Blueprint $table) {
        $table->string('nouvelle_colonne')->nullable();
    });
}
```

- Exécutez la migration avec la commande :

```
> php artisan migrate
```

Question 7 :

Quelle méthode utilise-t-on pour faire une redirection vers une autre route ?

Avec la méthode ***redirect()***:

```php
return redirect()->route('nom_de_la_route');
```

Avec la fonction helper ***redirect()***:

```php
return redirect('nom_de_la_route');
```

Question 8 :

Comment récupère-t-on toutes les données d'une table avec Eloquent ?

```php
$products = Product::all();
```

Question 9 :

Quelle commande crée un middleware ?

```
> php artisan make:middleware NomDuMiddleware
```

Question 10 :

Comment définit-on une route qui accepte n'importe quelle méthode HTTP ?

```php
Route::any('/nom-de-la-route', [PostController::class ,'methode']);
```

Question 11 :

Comment accède-t-on à l'instance de la requête actuelle dans un contrôleur ?

utiliser la variable ***$request*** :

```php
public function mahboub(Request $request) {
    $data = $request->all();
    // ...
}
```

Question 12 :

Quelle méthode permet de limiter le nombre de requêtes qu'un utilisateur peut faire ?

le middleware ***throttle*** :

```php
Route::middleware('throttle:rate_limit,1')->group(function () {
    // ...
});
```

Question 13 :

Comment envoie-t-on un email en utilisant Laravel ?

utiliser la classe ***Mail*** :

```php
use Illuminate\Support\Facades\Mail;
Mail::to('marouane.ma7boub@gmail.com')->send(new MonMail());
```

Question 14 :

Quelle commande permet de lancer le serveur de développement de Laravel ?

```
> php artisan serve
```

Question 15 :

Comment accède-t-on aux paramètres d'URL dans une route ?

```php
Route::get('/utilisateur/{id}', function ($id) {
    // Utilisez $id pour accéder au paramètre d'URL
});
```

Question 16 :

Comment fait-on pour ajouter une contrainte de clé étrangère dans une migration ?

```php
Schema::table('articles', function (Blueprint $table) {
    $table->foreign('user_id')->references('id')->on('users');
});
```

Question 17 :

Comment peut-on créer une tâche planifiée (scheduled task) dans Laravel ?

Pour créer une tâche planifiée dans Laravel, vous pouvez utiliser la commande artisan **make:schedule**. Ensuite, vous définissez votre tâche planifiée dans le fichier **app/Console/Kernel.php**. Par exemple :

```
protected function schedule(Schedule $schedule) {
    $schedule->command('moncommande')->daily();
}
```

Pour tester la tâche planifiée :

```
> php artisan schedule:run
```

Question 18 :

Quelle méthode permet de valider une requête entrante et rediriger automatiquement en cas d'erreur ?

Vous pouvez utiliser la méthode ***validate*** :

```
public function store(Request $request) {
    $validatedData = $request->validate([
        'nom' => 'required|max:255',
    ]);
    // Suite du code si la validation réussit
}
```

Question 19 :

Comment peut-on créer une relation un-à-plusieurs dans Eloquent ?

On peut définir une méthode dans le modèle parent qui utilise la fonction ***hasMany,*** Par exemple un post a plusiereurs commantaires :

```
class Post extends Model {
    public function commentaires() {
        return $this->hasMany(Commentaire::class);
    }
}
```

Question 20 :

Quelle méthode Eloquent utilise-t-on pour charger les relations d'un modèle de manière conditionnelle ?

On peut utiliser la méthode ***when :***

```
$post = Post::when($condition, function ($query) {
    // Chargement conditionnel des relations
    return $query->with('comments');
})->get();
```

Question 21 :

Comment peut-on ajouter un attribut muté à un modèle Eloquent ?

```php
protected function firstName(): Attribute
{
    return Attribute::make(
        get: fn (string $value) => ucfirst($value),
        set: fn (string $value) => strtolower($value),
    );
}
```

Question 22 :

Qu'est-ce que le Tinker et à quoi sert-il ?

Le Tinker est un outil interactif intégré à Laravel qui permet d'exécuter du code PHP et d'interagir avec l'application. Il est utile pour tester du code rapidement.

Pour lancer le Tinker, il suffit d'exécuter la commande :

```
> php artisan tinker
```

Question 23 :

Comment peut-on publier les assets d'un package ?

```
> php artisan vendor:publish --provider="NomDuProvider"
```

Question 24 :

Comment génère-t-on une clé d'application dans Laravel ?

```
> php artisan key:generate
```

Question 25 :

Quelle commande permet de créer un événement et son listener associé ?

```
> php artisan make:event NomDeLEvenement
```

```
> php artisan make:listener NomDuListener --event=NomDeLEvenement
```

Question 26 :

Comment implémente-t-on une stratégie de cache personnalisée ?

Pour implémenter une stratégie de cache personnalisée dans Laravel, créez un nouveau fichier nommé **MonCacheStrategy.php**, définissez la classe **MonCacheStrategy** à l'intérieur, puis configurez Laravel dans **config/cache.php** pour utiliser cette stratégie.

Question 27 :

Quel fichier de configuration détermine les services externes que l'application utilise ?

Le fichier *config/services.php*

Question 28 :

Comment peut-on utiliser les slots et les composants dans les vues Blade ?

- **Utilisation des slots**

Définition d'un Slot :

```
<div>
    @slot('title')
        Default Title
    @endslot
    <p>Contenu de la vue parente</p>
</div>
```

Utilisation du Slot dans une Vue Enfant :

```
@extends('vue-parente')
@section('title')
    Custom Title
@endsection
@section('content')
    Contenu de la vue enfant
@endsection
```

La directive @parent est utilisée pour inclure le contenu par défaut du slot.

- **Utilisation des Composants :**

Création d'un Composant :

Vous pouvez créer un composant à l'aide de la commande

```
> php artisan make:component custom-button
```

Cela générera une classe de composant dans le répertoire ***View/Components***.

Utilisation d'un Composant dans une Vue :

Vous pouvez utiliser un composant dans une vue à l'aide de la directive **x-component**.

```
<x-custom-button :color="$buttonColor" />
```

Question 29 :

Qu'est-ce qu'un trait dans Laravel et comment peut-on l'utiliser ?

Définition d'un trait :

```
// MonTrait.php
trait MonTrait {
    public function methodeDuTrait() {
        // Logique du trait
    }
}
```

Utilisation d'un trait dans une classe :

```
// MaClasse.php
```

```php
use MonTrait;
class MaClasse {
    use MonTrait;
    public function uneAutreMethode() {
        // Utilisation de la méthode du trait
        $this->methodeDuTrait();
    }
}
```

Question 30 :

Comment peut-on définir un accesseur dans un modèle Eloquent ?

```php
public function getPrixAttribute($value)
{
    return $value * 100; // Manipulez et retournez la valeur modifiée
}
```

Question 31 :

Comment peut-on utiliser les transactions de base de données dans Laravel ?

Vous pouvez utiliser la facade *DB* pour démarrer une transaction, exécuter vos requêtes et les commiter ou les annuler.

```php
DB::beginTransaction();
try {
    DB::commit();
} catch (\Exception $e) {
    DB::rollBack();
}
```

Question 32 :

Quelle méthode permet de crypter les données avant de les sauvegarder dans la base de données ?

Cryptage à l'aide de la classe *Crypt* :

```php
$valeurCryptee = Crypt::encryptString($valeur);
$valeurDecryptee = Crypt::decryptString($valeurCryptee);
```

Question 33 :

Comment peut-on utiliser Laravel Mix pour compiler les assets ?

Pour utiliser Laravel Mix afin de compiler les assets, vous devez définir vos tâches de compilation dans le fichier ***webpack.mix.js*** et exécuter la commande ***npm run dev*** ou ***npm run production***. Exemple :

```php
const mix = require('laravel-mix');
mix.js('resources/js/app.js', 'public/js')
    .sass('resources/sass/app.scss', 'public/css');
```

Question 34 :

Quelle méthode permet de créer une réponse JSON dans un contrôleur ?

Pour créer une réponse JSON dans un contrôleur, vous pouvez utiliser la méthode ***json*** de la classe ***response*** de Laravel. Par exemple :

```php
return response()->json(['key' => 'value']);
```

Question 35 :

Comment peut-on utiliser les événements de modèle dans Laravel ?

Les événements de modèle permettent d'exécuter du code avant ou après des événements CRUD (création, lecture, mise à jour, suppression) sur un modèle.

Exemple:

```php
use App\Models\User;
User::created(function (User $user) {
    // Envoyer un email de bienvenue à l'utilisateur
});
```

Question 36 :

Comment peut-on implémenter les policies d'autorisation dans Laravel ?

Créez des politiques pour définir les autorisations. Par la commande :

```
> php artisan make:policy UserPolicy
```

Les policies d'autorisation permettent de définir des règles d'accès aux ressources de l'application.

```php
class UserPolicy {
    public function show(User $user) {
        return $user->id === Auth::user()->id;
    }
}
```

Question 37 :

Comment configure-t-on Laravel pour utiliser un système de files d'attente différent ?

Modifiez ***config/queue.php*** :

```php
'connections' => [
    'redis' => [...],
],
'default' => 'redis',
```

Question 38 :

Quelle est la méthode pour créer un canal de diffusion d'événements personnalisé ?

Créez un canal personnalisé pour diffuser des événements à des clients spécifiques.

```
class MyBroadcaster implements Broadcaster {
    public function broadcast(array $channels, $event, $payload) {
        // ...
    }
}
```

Question 39 :

Comment peut-on étendre un service provider existant pour y ajouter des fonctionnalités supplémentaires ?

Créez une nouvelle classe, étendez l'existant et ajoutez les fonctionnalités.

```
class MonServiceProvider extends ServiceProvider
{
    // Fonctionnalités supplémentaires
}
```

Question 40 :

Comment utilise-t-on les packages Laravel pour modulariser une application ?

Créez un package Composer avec une structure définie. Exemple de dossier src :

```
namespace MonPackage;
class MonClasse { /* ... */ }
```

Ou la commande :

```
composer require laravel/cashier
```