**Lab : configuration, sessions, gestion des fichiers et envoi d'e-mails**

**1. Configuration**

**Objectif du Lab**

Créer un fichier de configuration personnalisé pour gérer les paramètres d'une application fictive.

**Étapes à suivre**

**a) Créer un fichier de configuration personnalisé**

1. Créez un fichier **config/settings.php** avec le contenu suivant :

php

Copier

return [

'app\_name' => env('APP\_NAME', 'Mon Application'),

'default\_language' => env('DEFAULT\_LANGUAGE', 'fr'),

'max\_users' => env('MAX\_USERS', 100),

];

1. Ajoutez les variables correspondantes dans votre fichier **.env** :

env

Copier

APP\_NAME="Laravel App"

DEFAULT\_LANGUAGE=en

MAX\_USERS=50

1. Accédez aux valeurs de configuration depuis un contrôleur ou une route :

php

Copier

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

$appName = config('settings.app\_name');

$language = config('settings.default\_language');

$maxUsers = config('settings.max\_users');

return "Nom de l'application : $appName<br>Langue par défaut : $language<br>Nombre maximum d'utilisateurs : $maxUsers";

});

1. Testez la route **/settings** dans votre navigateur.

**Résultat attendu**

Affichage des paramètres personnalisés dans le navigateur :

Copier

Nom de l'application : Laravel App

Langue par défaut : en

Nombre maximum d'utilisateurs : 50

**2. Gestion des Sessions**

**Objectif du Lab**

Implémenter un système de session simple pour stocker et afficher un compteur de visites.

**Étapes à suivre**

**a) Configurer le driver de session**

1. Dans votre fichier **.env**, configurez le driver de session sur **file** (ou **database** si vous préférez) :

env

Copier

SESSION\_DRIVER=file

1. Créez une route pour incrémenter et afficher un compteur de visites :

php

Copier

Route::get('/visits', function (\Illuminate\Http\Request $request) {

$visits = $request->session()->get('visits', 0);

$visits++;

$request->session()->put('visits', $visits);

return "Vous avez visité cette page $visits fois.";

});

1. Testez la route **/visits** plusieurs fois dans votre navigateur.

**Résultat attendu**

Chaque fois que vous accédez à **/visits**, le compteur augmente et s'affiche correctement :

Copier

Vous avez visité cette page 1 fois.

Vous avez visité cette page 2 fois.

...

**3. Gestion des Fichiers**

**Objectif du Lab**

Créer un système de téléchargement d'images avec validation et stockage.

**Étapes à suivre**

**a) Préparer la configuration**

1. Créez un fichier **config/images.php** pour définir le chemin de stockage des images :

php

Copier

return [

'path' => 'uploads',

];

1. Configurez le disque **public** dans **config/filesystems.php** pour pointer vers **public/** :

php

Copier

'public' => [

'driver' => 'local',

'root' => public\_path(),

'url' => env('APP\_URL'),

'visibility' => 'public',

],

1. Créez un lien symbolique pour rendre accessible **storage/app/public** :

bash

Copier

php artisan storage:link

**b) Créer une requête de validation**

1. Générez une classe de validation pour les images :

bash

Copier

php artisan make:request ImagesRequest

1. Modifiez le fichier **ImagesRequest.php** pour ajouter les règles suivantes :

php

Copier

public function rules(): array

{

return [

'image' => 'required|image|dimensions:min\_width=100,min\_height=100',

];

}

**c) Implémenter le contrôleur**

1. Générez un contrôleur pour gérer les formulaires :

bash

Copier

php artisan make:controller PhotoController

1. Modifiez le contrôleur **PhotoController.php** :

php

Copier

use App\Http\Requests\ImagesRequest;

use Illuminate\Support\Facades\Storage;

public function create()

{

return view('photo');

}

public function store(ImagesRequest $request)

{

$filePath = $request->image->store(config('images.path'), 'public');

return view('photo\_ok', ['filePath' => $filePath]);

}

**d) Créer les vues**

1. Créez une vue pour le formulaire (**resources/views/photo.blade.php**) :

blade

Copier

@extends('template')

@section('contenu')

<h1>Envoyer une photo</h1>

<form action="{{ route('photo.store') }}" method="POST" enctype="multipart/form-data">

@csrf

<input type="file" name="image">

@error('image')

<p>{{ $message }}</p>

@enderror

<button type="submit">Envoyer</button>

</form>

@endsection

1. Créez une vue pour la confirmation (**resources/views/photo\_ok.blade.php**) :

blade

Copier

@extends('template')

@section('contenu')

<h1>Envoi réussi</h1>

<p>Votre photo a été enregistrée sous : {{ $filePath }}</p>

@endsection

**e) Définir les routes**

Ajoutez les routes suivantes dans **web.php** :

php

Copier

use App\Http\Controllers\PhotoController;

Route::get('photo', [PhotoController::class, 'create'])->name('photo.create');

Route::post('photo', [PhotoController::class, 'store'])->name('photo.store');

**Résultat attendu**

1. Un utilisateur peut télécharger une image via un formulaire.
2. L'image est validée et sauvegardée dans le dossier **public/uploads**.
3. Une page de confirmation affiche le chemin de l'image sauvegardée.

**4. Envoi d'E-mails**

**Objectif du Lab**

Configurer et tester l'envoi d'un e-mail de contact.

**Étapes à suivre**

**a) Configurer le pilote de messagerie**

1. Configurez le pilote **log** dans votre fichier **.env** pour tester localement :

env

Copier

MAIL\_MAILER=log

MAIL\_FROM\_ADDRESS="hello@example.com"

MAIL\_FROM\_NAME="${APP\_NAME}"

**b) Créer une classe Mailable**

1. Générez une classe Mailable pour les contacts :

bash

Copier

php artisan make:mail Contact

1. Modifiez le fichier **Contact.php** (**app/Mail/Contact.php**) :

php

Copier

use Illuminate\Mail\Mailables\Address;

public $contact;

public function \_\_construct($contact)

{

$this->contact = $contact;

}

public function envelope(): Envelope

{

return new Envelope(

from: new Address('admin@contact.com', 'John Lennon'),

subject: 'Nouveau message de contact',

);

}

public function content(): Content

{

return new Content(

view: 'emails.contact',

);

}

**c) Créer la vue de l'e-mail**

1. Créez un fichier **resources/views/emails/contact.blade.php** :

blade

Copier

<h1>Nouveau message de contact</h1>

<ul>

<li><strong>Nom :</strong> {{ $contact['nom'] }}</li>

<li><strong>Email :</strong> {{ $contact['email'] }}</li>

<li><strong>Message :</strong> {{ $contact['message'] }}</li>

</ul>

**d) Implémenter le contrôleur**

1. Modifiez le contrôleur pour envoyer l'e-mail :

php

Copier

use Illuminate\Support\Facades\Mail;

use App\Mail\Contact;

public function sendEmail(Request $request)

{

$data = $request->validate([

'nom' => 'required|string',

'email' => 'required|email',

'message' => 'required|string',

]);

Mail::to('administrateur@chezmoi.com')->send(new Contact($data));

return view('confirm');

}

**e) Tester l'e-mail**

1. Créez une route temporaire pour tester l'apparence de l'e-mail :

php

Copier

Route::get('/test-contact', function () {

return new \App\Mail\Contact([

'nom' => 'Dupont',

'email' => 'dupont@example.com',

'message' => 'Super site !',

]);

});

1. Accédez à **/test-contact** pour voir le rendu de l'e-mail dans votre navigateur.

**Résultat attendu**

1. Lorsque vous soumettez le formulaire, un e-mail est envoyé à l'administrateur (en mode log).
2. Vous pouvez vérifier le contenu de l'e-mail dans les logs Laravel (**storage/logs/laravel.log**).