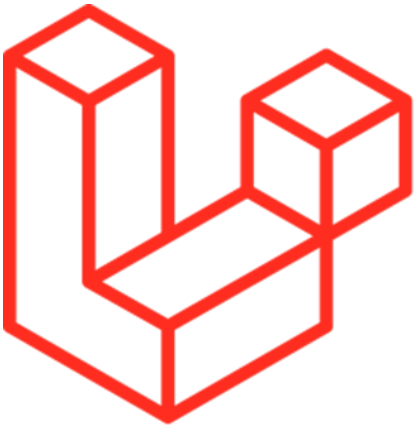
المدرسة العليا للتكنولوجيا الناظور
École Supérieure de Technologie de Nador
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵙⵙⴷⴰⵡⵜ ⵏ ⵜⵓⵜⵔⵉⵙⵜ ⵏⵏⴰⴹⵓⵔ

Développement web avancé

Filière : Ingénierie Logicielle et Cybersécurité (ILCS)

Projet Fin de module



Première année, Semestre S2	**Professeur : RABHI Ouzayr**	**Année universitaire : 2024/2025**

Projet DevProfile

Description du projet

Vous allez développer **DevProfile**, une application web où les utilisateurs **authentifiés** peuvent créer un profil de développeur, ajouter des projets et des compétences, et générer un CV au format PDF. Ce projet montre comment Laravel permet de construire rapidement une application fonctionnelle avec peu de code.

Objectifs pédagogiques :

- Configurer un projet Laravel avec authentification (Breeze).
- Implémenter un CRUD simple pour les projets et compétences.
- Gérer les relations Eloquent (One-to-Many).
- Générer un PDF avec Laravel DomPDF.
- Comprendre la structure MVC de Laravel.

Fonctionnalités requises

1. **Authentification** :
    - Inscription et connexion avec Laravel Breeze.
    - Les utilisateurs non authentifiés ne peuvent pas accéder aux fonctionnalités.
2. **Profil utilisateur** :
    - Modifier le profil : nom, email, titre (ex. : "Développeur Full Stack"), bio courte.
    - Afficher le profil sur une page publique accessible via une URL unique (ex. : /profile/username).
3. **Projets** :
    - Ajouter, modifier, supprimer des projets (titre, description, lien optionnel).
    - Afficher la liste des projets sur le profil.
4. **Compétences** :
    - Ajouter, supprimer des compétences (nom, ex. : "PHP", "Laravel").
    - Afficher la liste des compétences sur le profil.
5. **Génération de CV** :
    - Bouton pour générer un CV PDF avec le nom, titre, bio, projets, et compétences.
    - Le PDF doit être téléchargeable.

Contraintes techniques

- Utiliser **Laravel 11** avec **Breeze** (stack Blade) pour l'authentification.
- Utiliser **Tailwind CSS** (inclus avec Breeze) pour le style.
- Utiliser **Barryvdh/Laravel-DomPDF** pour générer le PDF.
- Base de données : MySQL.
- Structure MVC : modèles, contrôleurs, et vues bien organisés.

- Les projets et compétences sont liés à l'utilisateur connecté (relations Eloquent).

Livrables

- Une application fonctionnelle répondant aux fonctionnalités ci-dessus.
- Une démonstration le jour de l'examen.

Étapes suggérées

1. **Configurer le projet** :
    - Installer Laravel et Breeze.
    - Configurer la base de données MySQL dans .env.
2. **Authentification** :
    - Installer Breeze et tester l'inscription/connexion.
3. **Modèle et migrations** :
    - Étendre la table users pour ajouter title et bio.
    - Créer les tables projects et skills avec les champs nécessaires.
4. **CRUD** :
    - Implémenter les contrôleurs et vues pour les projets et compétences.
    - Sécuriser les routes avec le middleware auth.
5. **Profil public** :
    - Créer une route et une vue pour afficher le profil via un identifiant unique.
6. **PDF** :
    - Installer DomPDF et créer une vue pour le CV.
    - Ajouter une route pour générer et télécharger le PDF.
7. **Tests** :
    - Vérifier que toutes les fonctionnalités fonctionnent.
    - S'assurer que le design est propre avec Tailwind.

Modalités du projet

- **Organisation** : Le projet doit être réalisé en binôme (2 étudiants par groupe).
- **Soumission des groupes** : Envoyez la liste des groupes
- **Présentation** :
    - Date : Jeudi 22 mai 2025, à partir de 12h00.
    - Durée : 10 minutes par groupe.
    - Contenu : Présentation du projet (objectifs, fonctionnalités), démonstration de l'application en direct et explication du code source (structure MVC, choix techniques).
- **Livrables** :
    - Dépôt GitHub avec le code source complet et un README.md expliquant l'installation et l'utilisation.
    - Lien vers le dépôt GitHub à soumettre par email avant la présentation.
- **Directives supplémentaires** :

    - Testez toutes les fonctionnalités avant la présentation pour garantir une démonstration fluide.
    - Utilisez des données fictives pour montrer un profil complet.
    - Documentez les éventuels problèmes rencontrés et leurs solutions dans le README.md.
- **Conseils** :
    - Préparez un support visuel simple pour la présentation.
    - Divisez les tâches équitablement au sein du binôme et mentionnez la contribution de chaque membre lors de la présentation.

Annexe : Ressources fournies

1. Documentation recommandée

- **Laravel** :
  - Installation : Guide pour installer Laravel.
  - Breeze : Configurer l'authentification.
  - Eloquent : Gérer les modèles et relations.
  - Routing : Créer des routes.
  - Blade : Créer des vues.
  - Validation : Valider les formulaires.
- **Laravel DomPDF** :
  - Barryvdh/Laravel-DomPDF : Guide pour générer des PDFs.
- **Tailwind CSS** :
  - Tailwind Guide : Utiliser Tailwind avec Laravel.
  - Tailwind Play : Tester des styles rapidement.
- **MySQL** :
  - MySQL Workbench : Visualiser et gérer la base.

2. Scripts et templates

Script 1 : Migration pour étendre la table users

```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string('title')->nullable();
            $table->text('bio')->nullable();
            $table->string('username')->unique()->nullable();
        });
    }

    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn(['title', 'bio', 'username']);
        });
    }
};
```

Script 2 : Migration pour la table projects

```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('projects', function (Blueprint $table) {
            $table->id();

$table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->string('title');
            $table->text('description');
            $table->string('link')->nullable();
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('projects');
    }
};
```

Script 3 : Migration pour la table skills

```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('skills', function (Blueprint $table) {
            $table->id();

$table->foreignId('user_id')->constrained()->onDelete('cascade');
            $table->string('name');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('skills');
    }
};
```

Script 4 : Modèle User

```php
<?php
namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    protected $fillable = [
        'name', 'email', 'password', 'title', 'bio', 'username',
    ];

    public function projects()
    {
        return $this->hasMany(Project::class);
    }

    public function skills()
    {
        return $this->hasMany(Skill::class);
    }
}
```

Script 5 : Modèle Project

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Project extends Model
{
    protected $fillable = ['user_id', 'title', 'description', 'link'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

Script 6 : Modèle Skill

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Skill extends Model
{
    protected $fillable = ['user_id', 'name'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

Script 7 : Contrôleur DashboardController

```php
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DashboardController extends Controller
{
    public function index()
    {
        return view('dashboard');
    }
}
```

Script 8 : Template Blade pour la vue du profil public

```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            Profil de {{ $user->name }}
        </h2>
    </x-slot>

    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
                <div class="p-6 bg-white border-b border-gray-200">
                    <h3 class="text-lg font-bold">{{ $user->title }}</h3>
                    <p>{{ $user->bio }}</p>
                    <h4 class="mt-4 font-semibold">Projets</h4>
                    <ul>
                        @foreach ($user->projects as $project)
                            <li>{{ $project->title }} - {{
$project->description }}</li>
                        @endforeach
                    </ul>
                    <h4 class="mt-4 font-semibold">Compétences</h4>
                    <ul>
                        @foreach ($user->skills as $skill)
                            <li>{{ $skill->name }}</li>
                        @endforeach
                    </ul>
                    <a href="{{ route('pdf.generate', $user->username) }}"
class="mt-4 inline-block bg-blue-500 text-white px-4 py-2
rounded">Télécharger CV</a>
                </div>
            </div>
        </div>
    </div>
</x-app-layout>
```

Script 9 : Template Blade pour le CV PDF

```
<!DOCTYPE html>
<html>
<head>
    <title>CV de {{ $user->name }}</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        h1 { color: #333; }
        h2 { color: #555; }
        ul { list-style-type: none; padding: 0; }
        li { margin: 5px 0; }
    </style>
</head>
<body>
    <h1>{{ $user->name }}</h1>
    <h2>{{ $user->title }}</h2>
    <p>{{ $user->bio }}</p>
    <h3>Projets</h3>
    <ul>
        @foreach ($user->projects as $project)
            <li><strong>{{ $project->title }}</strong>: {{
$project->description }}</li>
        @endforeach
    </ul>
    <h3>Compétences</h3>
    <ul>
        @foreach ($user->skills as $skill)
            <li>{{ $skill->name }}</li>
        @endforeach
    </ul>
</body>
</html>
```

Script 10 : Routes suggérées (routes/web.php)

```php
<?php
use App\Http\Controllers\DashboardController;
use App\Http\Controllers\PDFController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\ProjectController;
use App\Http\Controllers\PublicProfileController;
use App\Http\Controllers\SkillController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});

Route::middleware(['auth'])->group(function () {
    Route::get('/dashboard', [DashboardController::class,
'index'])->name('dashboard');
    Route::get('/profile', [ProfileController::class,
'edit'])->name('profile.edit');
    Route::put('/profile', [ProfileController::class,
'update'])->name('profile.update');
    Route::resource('projects', ProjectController::class);
    Route::resource('skills', SkillController::class);
    Route::get('/pdf/{username}', [PDFController::class,
'generate'])->name('pdf.generate');
});

Route::get('/profile/{username}', [PublicProfileController::class,
'show'])->name('profile.show');
```

Afficher dans la barre latérale

Tâches à réaliser

- Implémentez les fonctionnalités décrites dans l'énoncé.
- Utilisez les scripts fournis pour les migrations, modèles, et templates.
- Testez toutes les fonctionnalités avant de soumettre.

Ressources

- Voir l'annexe de l'énoncé pour les scripts, templates, et documentation.