Laravel Task Management Guide

A Step-by-Step Implementation of a Task Management System in Laravel

Generated by Grok 3, Created by xAI

Date: May 21, 2025

1 Introduction

This document outlines the steps to create a task management system using Laravel. It includes creating a Laravel project, setting up the database, defining models, controllers, migrations, routes, views, and implementing middleware for access control. The system allows users to create, read, update, and delete tasks, with pagination and user authentication.

2 Project Setup

2.1 Creating the Laravel Project

To create a Laravel project named `Gestion_des_Taches`, run the following command:

```
laravel new Gestion_des_Taches
cd Gestion_des_Taches
```

2.2 Database Configuration

Assuming a database named `Gestion_des_Taches` exists, configure the connection in the `.env` file:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=Gestion_des_Taches
DB_USERNAME=your_db_username
DB_PASSWORD=your_db_password
```

Replace `your_db_username` and `your_db_password` with your actual credentials.

3 Model and Migration

3.1 Creating the Tache Model and Migration

Generate the `Tache` model with its migration:

```
php artisan make:model Tache -m
```

The `-m` flag creates a migration file.

3.2 Defining the Taches Table

Modify the migration file (e.g., `database/migrations/YYYY_MM_DD_HHMMSS_create_taches_table.ph` to define the `taches` table:

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

class CreateTachesTable extends Migration
{
    public function up()
    {
        Schema::create('taches', function (Blueprint $table) {
            $table->id();
            $table->string('titre');
            $table->text('description');
            $table->date('date_de_creation');
            $table->date('date_de_limit');
```

```
            $table->foreignId('utilisateur_id')->constrained('utilisateurs');
            $table->timestamps();
        });
    }

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

3.3 Running Migrations

Execute the migrations to create the table:

```
php artisan migrate
```

4 Controller and Routes

4.1 Creating the TacheController

Generate a resourceful controller for `Tache`:

```
php artisan make:controller TacheController --resource
```

4.2 Defining Resource Routes

Add a resource route in `routes/web.php`:

```
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\TacheController;

Route::resource('tache', TacheController::class);
```

5 Views

5.1 Creating the Task Form View

Create `resources/views/AjouteTache.blade.php` for adding tasks:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ajouter une Tache</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min
    .css" rel="stylesheet">
    <style>
        .container {
            margin-top: 50px;
            max-width: 600px;
            border: 1px solid #ddd;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
    </style>
</head>
<body>
```

```
    <div class="container">
        <h2 class="mb-4">Ajouter une Tache</h2>
        <form method="POST" action="{{ route('tache.store') }}">
            @csrf
            <div class="mb-3">
                <label for="titre" class="form-label">Task Title:</label>
                <input type="text" class="form-control" id="titre" name="titre" required>
            </div>
            <div class="mb-3">
                <label for="description" class="form-label">Description:</label>
                <textarea class="form-control" id="description" name="description" rows="3" required></textarea>
            </div>
            <div class="mb-3">
                <label for="date_de_creation" class="form-label">Creation Date:</label>
                <input type="date" class="form-control" id="date_de_creation" name="date_de_creation" required>
            </div>
            <div class="mb-3">
                <label for="date_de_limit" class="form-label">Deadline:</label>
                <input type="date" class="form-control" id="date_de_limit" name="date_de_limit" required>
            </div>
            <div class="mb-3">
                <label for="utilisateur_id" class="form-label">User ID:</label>
                <input type="number" class="form-control" id="utilisateur_id" name="utilisateur_id" required>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
</body>
</html>
```

5.2 Modifying the create() Method

Update the `create` method in `TacheController.php`:

```
public function create()
{
    return view('AjouteTache');
}
```

5.3 Implementing the store() Method

Update the `store` method to save form data:

```
public function store(Request $request)
{
    $request->validate([
        'titre' => 'required|string|max:255',
        'description' => 'required|string',
        'date_de_creation' => 'required|date',
        'date_de_limit' => 'required|date|after_or_equal:date_de_creation',
        'utilisateur_id' => 'required|exists:utilisateurs,id',
    ]);

    Tache::create($request->all());
    return redirect()->route('tache.index')->with('success', 'Tache created successfully!');
}
```

5.4 Listing Tasks

Create `resources/views/ListeTache.blade.php` to display tasks:

```
<!DOCTYPE html>
<html>
<head>
    <title>Liste des tâches</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .container { margin-top: 50px; }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="mb-4">Liste des tâches</h2>
        @if(session('success'))
            <div class="alert alert-success">{{ session('success') }}</div>
        @endif
        <a href="{{ route('tache.create') }}" class="btn btn-primary mb-3">Ajouter une Tache</a>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Titre</th>
                    <th>Description</th>
                    <th>Date de création</th>
                    <th>Date limite</th>
                    <th>Utilisateur ID</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                @forelse ($taches as $tache)
                    <tr>
                        <td>{{ $tache->id }}</td>
                        <td>{{ $tache->titre }}</td>
                        <td>{{ $tache->description }}</td>
                        <td>{{ $tache->date_de_creation }}</td>
                        <td>{{ $tache->date_de_limit }}</td>
                        <td>{{ $tache->utilisateur_id }}</td>
                        <td>
                            <a href="{{ route('tache.edit', $tache->id) }}" class="btn btn-sm btn-info">Éditer</a>
                            <form action="{{ route('tache.destroy', $tache->id) }}" method="POST" style="display:inline-block;">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Êtes-vous sûr de vouloir supprimer cette tâche ?')">Supprimer</button>
                            </form>
                        </td>
                    </tr>
                @empty
                    <tr>
                        <td colspan="7" class="text-center">Aucune tâche trouvée.</td>
                    </tr>
                @endforelse
            </tbody>
        </table>
```

```
        <div class="d-flex justify-content-center">
            {{ $taches->links() }}
        </div>
    </div>
</body>
</html>
```

5.5 Paginating Tasks

Update the `index` method for pagination:

```
public function index()
{
    $taches = Tache::paginate(10);
    return view('ListeTache', compact('taches'));
}
```

5.6 Editing Tasks

Create `resources/views/EditTache.blade.php` for editing tasks:

```
<!DOCTYPE html>
<html>
<head>
    <title>Modifier une Tache</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .container {
            margin-top: 50px;
            max-width: 600px;
            border: 1px solid #ddd;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="mb-4">Modifier une Tache</h2>
        <form method="POST" action="{{ route('tache.update', $tache->id) }}">
            @csrf
            @method('PUT')
            <div class="mb-3">
                <label for="titre" class="form-label">Task Title:</label>
                <input type="text" class="form-control" id="titre" name="titre" value="{{ old('titre', $tache->titre) }}" required>
            </div>
            <div class="mb-3">
                <label for="description" class="form-label">Description:</label>
                <textarea class="form-control" id="description" name="description" rows="3" required>{{ old('description', $tache->description) }}</textarea>
            </div>
            <div class="mb-3">
                <label for="date_de_creation" class="form-label">Creation Date:</label>
                <input type="date" class="form-control" id="date_de_creation" name="date_de_creation" value="{{ old('date_de_creation', $tache->date_de_creation) }}" required>
            </div>
            <div class="mb-3">
```

```
                <label for="date_de_limit" class="form-label">Deadline:</label>
                <input type="date" class="form-control" id="date_de_limit" name="
    date_de_limit" value="{{ old('date_de_limit', $tache->date_de_limit) }}"
    required>
            </div>
            <div class="mb-3">
                <label for="utilisateur_id" class="form-label">User ID:</label>
                <input type="number" class="form-control" id="utilisateur_id" name=
    "utilisateur_id" value="{{ old('utilisateur_id', $tache->utilisateur_id) }}"
    required>
            </div>
            <button type="submit" class="btn btn-primary">Mettre à jour</button>
            <a href="{{ route('tache.index') }}" class="btn btn-secondary">Annuler<
    /a>
        </form>
    </div>
</body>
</html>
```

Update the `edit` method:

```
public function edit(Tache $tache)
{
    return view('EditTache', compact('tache'));
}
```

Update the `update` method:

```
public function update(Request $request, Tache $tache)
{
    $request->validate([
        'titre' => 'required|string|max:255',
        'description' => 'required|string',
        'date_de_creation' => 'required|date',
        'date_de_limit' => 'required|date|after_or_equal:date_de_creation',
        'utilisateur_id' => 'required|exists:utilisateurs,id',
    ]);

    $tache->update($request->all());
    return redirect()->route('tache.index')->with('success', 'Tache mise à jour
    avec succès!');
}
```

5.7 Deleting Tasks

Update the `destroy` method:

```
public function destroy(Tache $tache)
{
    $tache->delete();
    return redirect()->route('tache.index')->with('success', 'Tache supprimée avec
    succès!');
}
```

5.8 Confirmation View for Deletion

Create `resources/views/ConfirmDeleteTache.blade.php`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Confirmer Suppression</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min
    .css" rel="stylesheet">
```

```
    <style>
        .container {
            margin-top: 50px;
            max-width: 500px;
            border: 1px solid #ddd;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="mb-4">Confirmer la suppression</h2>
        <p>Êtes-vous sûr de vouloir supprimer la tâche : <strong>"{{ $tache->titre }}"</strong> (ID: {{ $tache->id }}) ?</p>
        <form action="{{ route('tache.destroy', $tache->id) }}" method="POST" style="display:inline;">
            @csrf
            @method('DELETE')
            <button type="submit" class="btn btn-danger me-2">Oui, Supprimer</button>
        </form>
        <a href="{{ route('tache.index') }}" class="btn btn-secondary">Annuler</a>
    </div>
</body>
</html>
```

Add a route for confirmation in `routes/web.php`:

```
Route::get('tache/{tache}/confirm-delete', [TacheController::class, 'confirmDelete'])->name('tache.confirm_delete');
```

Add the `confirmDelete` method:

```
public function confirmDelete(Tache $tache)
{
    return view('ConfirmDeleteTache', compact('tache'));
}
```

6 Database Seeding

6.1 Creating and Running a Seeder

Create a seeder for `taches`:

```
php artisan make:seeder TacheSeeder
```

Define the seeder in `database/seeders/TacheSeeder.php`:

```
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class TacheSeeder extends Seeder
{
    public function run()
    {
        DB::table('taches')->insert([
            [
                'titre' => 'Titre de la tâche 1',
```

```
                'description' => 'Description de la tâche 1',
                'date_de_creation' => Carbon::now()->subDays(5)->toDateString(),
                'date_de_limit' => Carbon::now()->addDays(10)->toDateString(),
                'utilisateur_id' => 1,
                'created_at' => Carbon::now(),
                'updated_at' => Carbon::now(),
            ],
            [
                'titre' => 'Titre de la tâche 2',
                'description' => 'Description de la tâche 2',
                'date_de_creation' => Carbon::now()->subDays(2)->toDateString(),
                'date_de_limit' => Carbon::now()->addDays(15)->toDateString(),
                'utilisateur_id' => 2,
                'created_at' => Carbon::now(),
                'updated_at' => Carbon::now(),
            ],
        ]);
    }
}
```

Update `database/seeders/DatabaseSeeder.php`:

```
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        $this->call([TacheSeeder::class]);
    }
}
```

Run the seeder:

```
php artisan db:seed
```

To refresh and seed:

```
php artisan migrate:fresh --seed
```

7 Middleware for Access Control

7.1 Creating the Middleware

Generate a middleware to restrict task modification:

```
php artisan make:middleware CheckTachePermission
```

Define the middleware in `app/Http/Middleware/CheckTachePermission.php`:

```
<?php
namespace App\Http\Middleware;
use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Tache;

class CheckTachePermission
{
    public function handle(Request $request, Closure $next)
    {
        $tacheId = $request->route('tache');
        $tache = Tache::find($tacheId);
```

```
        if (!$tache || (Auth::check() && Auth::id() !== $tache->utilisateur_id)) {
            return redirect()->route('tache.index')->with('error', 'Vous n\'êtes pas autorisé à modifier cette tâche.');
        }
        return $next($request);
    }
}
```

Register it in `app/Http/Kernel.php`:

```
protected $routeMiddleware = [
    // ... other middleware ...
    'tache.permission' => \App\Http\Middleware\CheckTachePermission::class,
];
```

Apply to routes in `routes/web.php`:

```
Route::middleware('tache.permission')->group(function () {
    Route::get('tache/{tache}/edit', [TacheController::class, 'edit'])->name('tache.edit');
    Route::put('tache/{tache}', [TacheController::class, 'update'])->name('tache.update');
    Route::delete('tache/{tache}', [TacheController::class, 'destroy'])->name('tache.destroy');
    Route::get('tache/{tache}/confirm-delete', [TacheController::class, 'confirmDelete'])->name('tache.confirm_delete');
});
```

8 Conclusion

This guide provides a complete implementation of a task management system in Laravel, including project setup, database configuration, CRUD operations, pagination, and access control via middleware. Ensure the `utilisateurs` table exists and is populated for the foreign key to work. Authentication setup (e.g., via Laravel Breeze) is required for the middleware.