

Architecture Modulaire Laravel

Installation et Configuration

Installation du package :

```
composer require nwidart/laravel-modules "^9.0"
```

Publication de la configuration :

```
php artisan vendor:publish --  
provider="Nwidart\Modules\LaravelModulesServiceProvider"
```

Pour publier les fichiers de configuration nécessaires pour gérer les modules.

Créer un nouveau module

```
php artisan module:make Stage
```

Configuration Composer

Autoloading des modules

Ajouter dans `composer.json` du projet principal :

```
{  
    "autoload": {  
        "psr-4": {  
            "App\\": "app/",  
            "Database\\Factories\\": "database/factories/",  
            "Database\\Seeders\\": "database/seeders/",  
            "Modules\\Stage\\": "Modules/Stage/"  
        },  
        "files": [  
            "app/Helpers/FieldHelper.php",  
            "app/Helpers/MenuNavigationLogic.php"  
        ]  
    }  
}
```

Recharger l'autoloader

```
composer dump-autoload
```

Séparation des Modules avec Git

Objectif

Séparer le module Stage en repository Git indépendant pour permettre sa réutilisation dans d'autres projets.

Étape 1 : Initialiser un repository Git pour le module :

```
cd Modules/Stage  
git init  
git add .  
git commit -m "Initial commit - Stage module "
```

Étape 2 : Créer et lier le repository distant :

-Lier le repository local au distant :

```
git remote add origin git@github.com:votre-compte/module-stage.git  
git push -u origin main
```

Étape 3 : Configurer .gitignore du projet principal

dans le fichier `.gitignore` du projet principal :

```
Modules/Stage/
```

Cela évite que les fichiers du module soient trackés par le repository principal.

Commandes Utiles

Génération de composants

```
# Créer un contrôleur  
php artisan module:make-controller StageController Stage  
  
# Créer un modèle  
php artisan module:make-model Stage Stage  
  
# Créer une migration  
php artisan module:make-migration create_stages_table Stage  
  
# Créer un seeder  
php artisan module:make-seeder StageSeeder Stage  
  
# Créer une factory  
php artisan module:make-factory StageFactory Stage
```

Gestion des modules

```
# Activer un module  
php artisan module:enable Stage  
  
# Désactiver un module  
php artisan module:disable Stage  
  
# Lister tous les modules  
php artisan module:list  
  
# Publier les assets d'un module  
php artisan module:publish Stage  
  
# Migrer un module spécifique  
php artisan module:migrate Stage
```

Gestion des Repositories Séparés

Workflow de développement

1. **Travailler sur le module :**
2. cd Modules/Stage
3. # Faire vos modifications
4. git add .
5. git commit -m "Ajout nouvelle fonctionnalité"
6. git push origin main
7. **Synchroniser avec subtree (optionnel) :**
8. git subtree push --prefix=Modules/Stage stage-module main

Avantages de cette approche

- **Réutilisabilité** : Le module peut être utilisé dans d'autres projets
- **Indépendance** : Développement et versioning séparés
- **Maintenance** : Corrections de bugs centralisées
- **Collaboration**

Utilisation dans d'Autres Projets

Méthode 1 : Git Submodule

```
# Dans le nouveau projet  
git submodule add https://github.com/votre-compte/module-stage.git  
Modules/Stage  
git submodule update --init --recursive
```

Méthode 2 : Package Composer (Recommandé)

1. **Publier le module comme package Composer** sur Packagist
2. **Installer dans le nouveau projet :**

3. composer require votre-organisation/stage-module

Méthode 3 : Repository Composer privé

```
// Dans composer.json du nouveau projet
{
  "repositories": [
    {
      "type": "vcs",
      "url": "https://github.com/votre-compte/module-stage.git"
    }
  ],
  "require": {
    "votre-organisation/stage-module": "dev-main"
  }
}
```

Dépannage

Problèmes courants

Module non reconnu :

```
composer dump-autoload
php artisan config:clear
php artisan cache:clear
```

Conflits de routes :

```
php artisan route:clear
php artisan route:cache
```

Migrations :

```
php artisan module:migrate-refresh Stage
```

Cette architecture modulaire permet de créer des applications Laravel scalables et maintenables, avec des modules réutilisables et indépendants.