🚀 Guide Complet - Installation et Configuration Laravel

📋 Table des matières

---

1. Prérequis

Logiciels requis

- **PHP** >= 8.1
- **Composer** (gestionnaire de dépendances PHP)
- **Node.js** & npm (pour les assets frontend)
- **MySQL** ou **MariaDB**
- **Git**
- Un éditeur de code (VS Code, PHPStorm, etc.)

Vérifier les versions installées

```
php -v
composer -v
node -v
npm -v
mysql --version
git --version
```

Extensions PHP requises

Assurez-vous que ces extensions sont activées dans `php.ini` :

```
extension=pdo_mysql
extension=mbstring
extension=openssl
extension=tokenizer
extension=xml
extension=ctype
extension=json
```

```
extension=bcmath
extension=fileinfo
```

---

2. Installation de Laravel

Méthode 1 : Via Composer (Recommandée)

```
# Créer un nouveau projet Laravel
composer create-project laravel/laravel tp4-ecommerce

# Naviguer dans le dossier
cd tp4-ecommerce
```

Méthode 2 : Via Laravel Installer

```
# Installer Laravel Installer (une seule fois)
composer global require laravel/installer

# Créer un nouveau projet
laravel new tp4-ecommerce

cd tp4-ecommerce
```

Lancer le serveur de développement

```
php artisan serve
```

🌐 Accédez à : `http://localhost:8000`

---

3. Configuration de la base de données

Étape 1 : Créer la base de données

Option A : Via ligne de commande MySQL

```
mysql -u root -p
```

```
CREATE DATABASE tp4_ecommerce CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
EXIT;
```

Option B : Via phpMyAdmin

- Ouvrir phpMyAdmin
- Cliquer sur "Nouvelle base de données"
- Nom : `tp4_ecommerce`
- Interclassement : `utf8mb4_unicode_ci`

Étape 2 : Configurer le fichier `.env`

Ouvrir le fichier `.env` à la racine du projet :

```
APP_NAME="TP4 E-commerce"
APP_ENV=local
APP_KEY=base64:xxx  # Généré automatiquement
APP_DEBUG=true
APP_URL=http://localhost:8000

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tp4_ecommerce
DB_USERNAME=root
DB_PASSWORD=          # Votre mot de passe MySQL (vide si aucun)

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

# Configuration mail (optionnel pour TP)
MAIL_MAILER=log
MAIL_HOST=127.0.0.1
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Étape 3 : Générer la clé d'application

```
php artisan key:generate
```

Étape 4 : Tester la connexion

```
php artisan migrate
```

Si tout fonctionne, vous verrez :

```
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
...
```

---

4. Structure du projet

Arborescence Laravel par défaut

```
tp4-ecommerce/
│
├── app/                    # Logique métier
│   ├── Http/
│   │   ├── Controllers/    # Contrôleurs
│   │   ├── Middleware/     # Middleware
│   │   └── Requests/       # Form Requests
│   ├── Models/             # Modèles Eloquent
│   └── Providers/          # Service Providers
│
├── bootstrap/              # Fichiers de démarrage
│
├── config/                 # Fichiers de configuration
│   ├── app.php
│   ├── database.php
│   └── ...
│
├── database/
│   ├── migrations/         # Migrations BDD
│   ├── seeders/            # Seeders (données test)
│   └── factories/          # Factories (génération données)
│
├── public/                 # Point d'entrée public
│   ├── index.php
│   ├── css/
│   ├── js/
│   └── images/
│
├── resources/
│   ├── css/                # Fichiers CSS sources
```

```
│   ├── js/                 # Fichiers JS sources
│   └── views/              # Vues Blade
│       ├── layouts/        # Templates de base
│       ├── components/     # Composants réutilisables
│       └── pages/          # Pages spécifiques
│
├── routes/
│   ├── web.php             # Routes web
│   ├── api.php             # Routes API
│   └── console.php         # Commandes Artisan
│
├── storage/
│   ├── app/                # Fichiers générés
│   ├── framework/          # Cache, sessions, vues compilées
│   └── logs/               # Logs application
│
├── tests/                  # Tests automatisés
│
├── vendor/                 # Dépendances Composer
│
├── .env                    # Variables d'environnement
├── .env.example            # Exemple de configuration
├── artisan                 # CLI Laravel
├── composer.json           # Dépendances PHP
└── package.json            # Dépendances Node.js
```

---

5. Configuration initiale

1. Installer les dépendances Node.js

```
npm install
```

2. Compiler les assets (CSS/JS)

```
# Développement (avec watch pour recompiler automatiquement)
npm run dev

# OU en mode watch
npm run watch

# Production (minifié)
npm run build
```

3. Créer le lien symbolique pour le stockage

```
php artisan storage:link
```

Cette commande crée un lien symbolique de `public/storage` vers `storage/app/public`.

4. Configuration des permissions (Linux/Mac)

```
chmod -R 775 storage bootstrap/cache
```

5. Désactiver le mode maintenance

```
php artisan up
```

---

6. Structure de code recommandée

Organisation des dossiers pour le TP4

```
app/
├── Http/
│   ├── Controllers/
│   │   ├── Auth/
│   │   │   ├── LoginController.php
│   │   │   ├── RegisterController.php
│   │   │   └── LogoutController.php
│   │   │
│   │   ├── Admin/
│   │   │   ├── DashboardController.php
│   │   │   ├── ProductController.php  # CRUD admin
│   │   │   └── OrderController.php    # Gestion commandes admin
│   │   │
│   │   ├── ProductController.php      # Affichage public
│   │   ├── CartController.php         # Gestion panier
│   │   ├── OrderController.php        # Commandes client
│   │   └── PaymentController.php      # Paiement
│   │
│   ├── Middleware/
│   │   ├── IsAdmin.php                # Vérifier si admin
│   │   └── CheckCart.php              # Vérifier panier non vide
│   │
│   └── Requests/
│       ├── LoginRequest.php
│       ├── RegisterRequest.php
│       ├── ProductRequest.php
│       └── OrderRequest.php
│
├── Models/
```

```
│   ├── User.php
│   ├── Product.php
│   ├── Category.php
│   ├── Cart.php
│   ├── CartItem.php
│   ├── Order.php
│   └── OrderItem.php
│
└── Services/                          # Logique métier complexe
(optionnel)
    ├── CartService.php
    ├── OrderService.php
    └── PaymentService.php
```

Structure des vues Blade

```
resources/views/
│
├── layouts/
│   ├── app.blade.php                  # Layout principal
│   ├── admin.blade.php                # Layout admin
│   └── guest.blade.php                # Layout non connecté
│
├── components/
│   ├── navbar.blade.php
│   ├── footer.blade.php
│   ├── product-card.blade.php
│   └── alert.blade.php
│
├── auth/
│   ├── login.blade.php
│   └── register.blade.php
│
├── products/
│   ├── index.blade.php                # Liste produits
│   └── show.blade.php                 # Détail produit
│
├── cart/
│   └── index.blade.php                # Page panier
│
├── orders/
│   ├── index.blade.php                # Historique commandes
│   └── show.blade.php                 # Détail commande
│
├── payment/
│   ├── checkout.blade.php             # Page paiement
│   └── confirmation.blade.php         # Confirmation
│
├── admin/
│   ├── dashboard.blade.php
│   ├── products/
```

```
│   │   ├── index.blade.php
│   │   ├── create.blade.php
│   │   └── edit.blade.php
│   └── orders/
│       ├── index.blade.php
│       └── show.blade.php
│
└── home.blade.php                    # Page d'accueil
```

---

7. Exemple de code de base

7.1 Modèle Product

app/Models/Product.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'description',
        'price',
        'stock',
        'image',
        'category_id'
    ];

    protected $casts = [
        'price' => 'decimal:2',
        'stock' => 'integer'
    ];

    // Relations
    public function category()
    {
        return $this->belongsTo(Category::class);
    }

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

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

    // Méthodes utiles
    public function isInStock($quantity = 1)
    {
        return $this->stock >= $quantity;
    }

    public function decrementStock($quantity)
    {
        $this->stock -= $quantity;
        $this->save();
    }
}
```

7.2 Migration Product

database/migrations/xxxx_create_products_table.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()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description')->nullable();
            $table->decimal('price', 10, 2);
            $table->integer('stock')->default(0);
            $table->string('image')->nullable();
            $table->foreignId('category_id')->constrained()-
>onDelete('cascade');
            $table->timestamps();
        });
    }

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

7.3 Contrôleur ProductController

app/Http/Controllers/ProductController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
    // Afficher tous les produits
    public function index()
    {
        $products = Product::with('category')->paginate(12);
        return view('products.index', compact('products'));
    }

    // Afficher un produit
    public function show($id)
    {
        $product = Product::with('category')->findOrFail($id);
        return view('products.show', compact('product'));
    }

    // Recherche
    public function search(Request $request)
    {
        $query = $request->input('query');
        $products = Product::where('name', 'LIKE', "%{$query}%")
                           ->orWhere('description', 'LIKE', "%{$query}%")
                           ->paginate(12);

        return view('products.index', compact('products', 'query'));
    }
}
```

7.4 Routes

routes/web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\ProductController;
```

```php
use App\Http\Controllers\CartController;
use App\Http\Controllers\OrderController;
use App\Http\Controllers\PaymentController;
use App\Http\Controllers\Admin\DashboardController;

// Page d'accueil
Route::get('/', function () {
    return view('home');
})->name('home');

// Authentification
Route::get('/login', [LoginController::class, 'showLoginForm'])-
>name('login');
Route::post('/login', [LoginController::class, 'login']);
Route::get('/register', [RegisterController::class,
'showRegistrationForm'])->name('register');
Route::post('/register', [RegisterController::class, 'register']);
Route::post('/logout', [LoginController::class, 'logout'])->name('logout');

// Produits (public)
Route::get('/products', [ProductController::class, 'index'])-
>name('products.index');
Route::get('/products/{id}', [ProductController::class, 'show'])-
>name('products.show');
Route::get('/search', [ProductController::class, 'search'])-
>name('products.search');

// Routes protégées (authentification requise)
Route::middleware(['auth'])->group(function () {

    // Panier
    Route::get('/cart', [CartController::class, 'index'])-
>name('cart.index');
    Route::post('/cart/add', [CartController::class, 'add'])-
>name('cart.add');
    Route::patch('/cart/{id}', [CartController::class, 'update'])-
>name('cart.update');
    Route::delete('/cart/{id}', [CartController::class, 'remove'])-
>name('cart.remove');
    Route::delete('/cart', [CartController::class, 'clear'])-
>name('cart.clear');

    // Commandes
    Route::get('/orders', [OrderController::class, 'index'])-
>name('orders.index');
    Route::get('/orders/{id}', [OrderController::class, 'show'])-
>name('orders.show');
    Route::post('/orders', [OrderController::class, 'store'])-
>name('orders.store');

    // Paiement
    Route::get('/checkout', [PaymentController::class, 'showCheckout'])-
>name('checkout');
    Route::post('/payment/process', [PaymentController::class,
```

```
'processPayment'])->name('payment.process');
    Route::get('/payment/confirmation/{order}', [PaymentController::class,
'confirmation'])->name('payment.confirmation');
});

// Routes Admin (authentification + rôle admin)
Route::middleware(['auth', 'admin'])->prefix('admin')->name('admin.')-
>group(function () {

    // Dashboard
    Route::get('/', [DashboardController::class, 'index'])-
>name('dashboard');

    // Gestion produits
    Route::resource('products',
\App\Http\Controllers\Admin\ProductController::class);

    // Gestion commandes
    Route::get('/orders',
[\App\Http\Controllers\Admin\OrderController::class, 'index'])-
>name('orders.index');
    Route::get('/orders/{id}',
[\App\Http\Controllers\Admin\OrderController::class, 'show'])-
>name('orders.show');
    Route::patch('/orders/{id}/status',
[\App\Http\Controllers\Admin\OrderController::class, 'updateStatus'])-
>name('orders.updateStatus');
});
```

7.5 Layout principal

resources/views/layouts/app.blade.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>@yield('title', 'E-commerce TP4')</title>

    <!-- CSS -->
    @vite(['resources/css/app.css', 'resources/js/app.js'])
    <link rel="stylesheet" href="{{ asset('css/style.css') }}">
</head>
<body>

    <!-- Header / Navigation -->
    @include('components.navbar')

    <!-- Messages flash -->
```

```
    @if(session('success'))
        <div class="alert alert-success">
            {{ session('success') }}
        </div>
    @endif

    @if(session('error'))
        <div class="alert alert-error">
            {{ session('error') }}
        </div>
    @endif

    <!-- Contenu principal -->
    <main>
        @yield('content')
    </main>

    <!-- Footer -->
    @include('components.footer')

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}"></script>
    @stack('scripts')
</body>
</html>
```

7.6 Composant Navbar

resources/views/components/navbar.blade.php

```
<nav class="navbar">
    <div class="container">
        <a href="{{ route('home') }}" class="logo">
            TP4 E-commerce
        </a>

        <ul class="nav-links">
            <li><a href="{{ route('home') }}">Accueil</a></li>
            <li><a href="{{ route('products.index') }}">Produits</a></li>

            @auth
                <li><a href="{{ route('cart.index') }}">
                    Panier
                    @if(auth()->user()->cart && auth()->user()->cart-
>items->count() > 0)
                        <span class="badge">{{ auth()->user()->cart->items-
>count() }}</span>
                    @endif
                </a></li>
                <li><a href="{{ route('orders.index') }}">Mes commandes</a>
</li>
```

```
                @if(auth()->user()->is_admin)
                    <li><a href="{{ route('admin.dashboard') }}">Admin</a>
</li>
                @endif

                <li>
                    <form action="{{ route('logout') }}" method="POST"
style="display: inline;">
                        @csrf
                        <button type="submit">Déconnexion</button>
                    </form>
                </li>
            @else
                <li><a href="{{ route('login') }}">Connexion</a></li>
                <li><a href="{{ route('register') }}">Inscription</a></li>
            @endauth
        </ul>
    </div>
</nav>
```

---

8. Commandes Artisan utiles

Migrations

```
# Créer une migration
php artisan make:migration create_products_table

# Exécuter les migrations
php artisan migrate

# Réinitialiser et relancer les migrations
php artisan migrate:fresh

# Réinitialiser + seeders
php artisan migrate:fresh --seed

# Rollback dernière migration
php artisan migrate:rollback

# Voir le statut des migrations
php artisan migrate:status
```

Modèles

```
# Créer un modèle
php artisan make:model Product
```

```
# Créer modèle + migration
php artisan make:model Product -m

# Créer modèle + migration + controller + seeder + factory
php artisan make:model Product -mcfs
```

Contrôleurs

```
# Créer un contrôleur
php artisan make:controller ProductController

# Créer un contrôleur avec méthodes CRUD
php artisan make:controller ProductController --resource

# Créer un contrôleur API
php artisan make:controller API/ProductController --api
```

Seeders

```
# Créer un seeder
php artisan make:seeder ProductSeeder

# Exécuter tous les seeders
php artisan db:seed

# Exécuter un seeder spécifique
php artisan db:seed --class=ProductSeeder
```

Middleware

```
# Créer un middleware
php artisan make:middleware IsAdmin
```

Form Requests

```
# Créer un Form Request
php artisan make:request ProductRequest
```

Cache

```
# Vider le cache
php artisan cache:clear
```

```
# Vider le cache de configuration
php artisan config:clear

# Vider le cache de routes
php artisan route:clear

# Vider le cache des vues
php artisan view:clear

# Vider tous les caches
php artisan optimize:clear
```

Autres commandes utiles

```
# Lister toutes les routes
php artisan route:list

# Entrer en mode maintenance
php artisan down

# Sortir du mode maintenance
php artisan up

# Générer la clé d'application
php artisan key:generate

# Créer un lien symbolique storage
php artisan storage:link

# Ouvrir Tinker (REPL PHP/Laravel)
php artisan tinker
```

---

9. Intégration des assets du TP1

Copier les fichiers CSS/JS

```
# Créer les dossiers nécessaires
mkdir -p public/css
mkdir -p public/js
mkdir -p public/images

# Copier vos fichiers du TP1
cp /chemin/vers/tp1/assets/css/style.css public/css/
cp /chemin/vers/tp1/assets/js/script.js public/js/
cp -r /chemin/vers/tp1/assets/images/* public/images/
```

Inclure dans les vues Blade

```
<!-- Dans resources/views/layouts/app.blade.php -->
<link rel="stylesheet" href="{{ asset('css/style.css') }}">
<script src="{{ asset('js/script.js') }}"></script>
```

---

10. Configuration Git

Initialiser le dépôt

```
git init
git add .
git commit -m "Initial commit - Installation Laravel"
```

Créer le dépôt distant

```
# Sur GitHub, créer un nouveau repository puis :
git remote add origin https://github.com/votre-username/tp4-ecommerce.git
git branch -M main
git push -u origin main
```

Fichier `.gitignore` (déjà présent)

```
/node_modules
/public/hot
/public/storage
/storage/*.key
/vendor
.env
.env.backup
.phpunit.result.cache
Homestead.json
Homestead.yaml
npm-debug.log
yarn-error.log
```

---

11. Dépannage

Erreur "Class not found"

```
composer dump-autoload
```

Erreur "Permission denied" (Linux/Mac)

```
sudo chmod -R 775 storage bootstrap/cache
sudo chown -R www-data:www-data storage bootstrap/cache
```

Erreur "SQLSTATE[HY000] [2002] Connection refused"

- Vérifier que MySQL est démarré
- Vérifier les identifiants dans `.env`
- Tester la connexion : `mysql -u root -p`

Page blanche après installation

```
php artisan config:clear
php artisan cache:clear
php artisan view:clear
```

npm install échoue

```
rm -rf node_modules package-lock.json
npm install
```

---

12. Ressources utiles

- 📚 **Documentation Laravel** : https://laravel.com/docs
- 🎓 **Laracasts** : https://laracasts.com
- 💬 **Forum Laravel** : https://laracasts.com/discuss
- 📦 **Packages Laravel** : https://packagist.org

✅ Checklist de validation

- ☐ Laravel installé et serveur démarre
- ☐ Base de données créée et connectée
- ☐ Migrations exécutées avec succès
- ☐ Assets compilés (npm run dev)
- ☐ Git initialisé et premier commit effectué
- ☐ `.env` configuré correctement
- ☐ Permissions correctes sur storage/
- ☐ Lien symbolique storage créé

Votre environnement Laravel est maintenant prêt pour le développement du TP4 ! 🎉

📞 Support

En cas de problème :

1. Vérifier les logs : `storage/logs/laravel.log`
2. Consulter la documentation Laravel
3. Demander de l'aide à vos collègues
4. Créer une issue sur le dépôt Git du groupe

Bon développement ! 💻