MINI-CRM

Walkthrough complet - TP7

Gestion des utilisateurs et authentification Laravel

Projet : mini-crm de BUT2

Objectif : passer d'un CRM ouvert à une application où les ressources sont accessibles uniquement après connexion.

1. Objectifs et résultat attendu

Le TP7 met en place l'authentification : création d'utilisateurs, mots de passe hachés, formulaire de connexion, session, déconnexion et middleware `auth`. Après le TP, un invité est redirigé vers `/login` et un utilisateur connecté arrive sur le tableau de bord.

État	Accès
Invité	/login uniquement pour l'interface CRM
Utilisateur connecté	Tableau de bord, clients, produits et contacts
Compte administrateur	Même accès + page Administration (TP8)

La distinction importante est la suivante : l'authentification répond à « qui es-tu ? », alors que l'autorisation répond à « as-tu le droit ? ». L'autorisation est traitée au TP8.

2. Vérifier la table users

Laravel fournit déjà la migration des utilisateurs. La colonne `password` contient un hash, jamais le mot de passe en clair. La migration TP8 ajoute ensuite `is_admin`.

Extrait / commande

```
php artisan migrate:status
```

Dans le projet final, la migration `2026_10_20_223959_add_is_admin_to_users_table.php` ajoute un booléen à false par défaut.

3. Créer et comprendre les utilisateurs

Le seeder `UserSeeder` prépare deux comptes reproductibles. Le mot de passe est produit par `Hash::make` et le modèle User possède le cast booléen de `is_admin`.

Compte	E-mail	Mot de passe	Rôle
Jean Dupont	jean@example.com	password123	Administrateur
Marie Martin	marie@example.com	password123	Utilisateur

Extrait / commande

```
php artisan db:seed --class=UserSeeder --force
```

4. Formulaire et traitement de connexion

Le contrôleur `AuthController` expose trois actions : `create` affiche le formulaire, `store` valide puis appelle `Auth::attempt`, et `destroy` ferme la session.

Extrait / commande

```
if (Auth::attempt($credentials)) {
    $request->session()->regenerate();
    return redirect()->intended(route('dashboard'));
}
```

Le formulaire Blade contient `@csrf`, conserve l'adresse e-mail en cas d'erreur et affiche le message « Les identifiants sont incorrects. ». Les tentatives POST sont limitées par `throttle:5,1`.

La déconnexion utilise `Auth::logout()`, invalide la session et régénère le jeton CSRF. Elle est volontairement en POST car elle modifie l'état de la session.

5. Protéger toutes les routes CRM

Les routes de connexion sont dans le groupe `guest`. Les routes applicatives sont dans le groupe `auth` : dashboard, clients, produits, contacts et pages d'exemple.

Extrait / commande

```
Route::middleware('auth')->group(function (): void {
    Route::resource('clients', ClientController::class);
    Route::resource('products', ProductController::class);
});
```

Le layout adapte son affichage avec `@auth` et `@else` : le nom de l'utilisateur connecté et le bouton de déconnexion apparaissent après connexion.

6. Tests réalisés

Les tests d'authentification couvrent les cas attendus : invité redirigé, connexion réussie, identifiants invalides, conservation de l'e-mail et déconnexion.

Extrait / commande

```
php artisan test --compact
PHPUnit: 20 tests, 175 assertions, tous réussis
```

Pour vérifier manuellement : ouvrir `/login`, utiliser Jean ou Marie, constater la redirection vers `/dashboard`, puis se déconnecter et retenter `/clients`.

Conclusion : le Mini-CRM dispose maintenant d'un cycle complet de session et d'une frontière claire entre pages publiques et pages protégées.