TP4 - Walkthrough complet

CRUD Clients et Produits, de A a Z

Ce document explique exactement ce qui a ete construit dans mini-crm, comment une action passe du navigateur a SQLite et comment le resultat a ete verifie.

1. Point de depart

• Le TP3 avait deja les tables clients/products, les Model Eloquent, les seeders et une base SQLite initialisees.

• Aucune donnee TP3 n a ete supprimee. Les anciennes routes de demonstration clients ont ete remplacees par le CRUD officiel du TP4.

2. CRUD et Resource Controller

CRUD signifie Create, Read, Update et Delete. Un Resource Controller regroupe index, create, store, show, edit, update et destroy. Chaque methode correspond a une action utilisateur.

Action	Methode	Route
Lister	index	GET /clients
Formulaire creation	create	GET /clients/create
Creer	store	POST /clients
Detail	show	GET /clients/{client}
Formulaire edition	edit	GET /clients/{client}/edit
Modifier	update	PUT/PATCH /clients/{client}
Supprimer	destroy	DELETE /clients/{client}

3. Routes generees par Laravel

```
Route::resource('clients', ClientController::class);
Route::resource('products', ProductController::class);
```

Ces deux lignes generent les 14 routes CRUD : sept Clients et sept Produits. Elles disposent de noms stables comme clients.index ou products.edit, utilises dans les liens et les formulaires.

4. ClientController : chaque etape

• index() lance Client::all() et transmet la collection a clients.index.

• create() retourne la vue du formulaire vide.

• store(Request $request) valide les champs et appelle Client::create(...).

• show(Client $client) et edit(Client $client) recoivent directement le bon client grace au route model binding.

• update(...) appelle $client->update(...), puis destroy(...) appelle $client->delete().

Une validation legere a ete ajoutee : nom et ville obligatoires, email obligatoire, valide et unique. Pendant une modification, l email actuel du client reste autorise.

5. Les vues Clients

• index affiche le total, nom, email, ville, date de creation et les actions Voir/Modifier/Supprimer.

• create utilise POST, route('clients.store') et @csrf.

• show affiche un seul client. edit pre-remplit les champs et utilise @method('PUT').

• La suppression utilise @csrf et @method('DELETE').

6. CRUD Product

ProductController reprend la meme architecture avec name, price et stock. Le Model Product possede deja $fillable, ce qui autorise create() et update().

• products/index liste le total et les actions. Le defi est realise : stock inferieur a 5 affiche « Stock faible », sinon « Disponible ».

• products/create et products/edit utilisent des champs numeriques pour prix et stock.

• products/show affiche le detail. update et destroy modifient/suppriment le produit choisi.

7. Parcours complet : Ajouter un client

1. Le lien route('clients.create') genere GET /clients/create.
2. ClientController@create retourne la View du formulaire.
3. @csrf ajoute un jeton de securite.
4. Le formulaire envoie POST /clients.
5. Laravel donne la Request a ClientController@store.
6. Le Controller valide les donnees, puis Eloquent utilise Client::create() pour ecrire dans SQLite.
7. redirect()->route('clients.index') recharge la liste mise a jour.

8. Navigation et bonnes pratiques

• Le layout emploie les routes nommees clients.index, clients.create et products.index, plutot que des URL fragiles ecrites directement.

• Les formulaires gardent old(...) afin de conserver une saisie si une validation echoue.

• Les seeders TP3 gardent trois clients et cinq produits a montrer.

9. Verification finale

php artisan route:list confirme toutes les routes REST. php artisan test a execute 4 tests et 28 assertions sans echec. Les tests CRUD creent, affichent, modifient et suppriment un client et un produit dans une base de test temporaire, pas dans ta base SQLite personnelle.

10. Comment presenter le TP

Lance php artisan serve. Ouvre /clients puis montre Ajouter, Voir, Modifier et Supprimer. Ouvre ensuite /products et montre le meme parcours, ainsi que le message Stock faible. Les reponses aux questions du sujet sont dans TP4_rempli.pdf.

Conclusion

Le TP4 est complet : Resource Controllers, Route::resource(), Model Binding, formulaires CSRF, PUT/DELETE, Eloquent, CRUD Clients et Produits, exercices et defi ont ete implementes et verifies.