```
$categories = Category::all(); // Collection de tous les enregistrements
```

Trouver par ID

```
$category = Category::find(1); // null si non trouvé
$category = Category::findOrFail(1); // Exception 404 si non trouvé
```

Premier enregistrement

```
$category = Category::first(); // Premier de la table
$category = Category::firstOrFail(); // Exception si vide
```

Filtres avec WHERE

Condition simple

```
$categories = Category::where('name', 'Technologie')->get();
$categories = Category::where('id', '>', 10)->get();
```

Conditions multiples

```
// AND
$categories = Category::where('name', 'Tech')
                ->where('is_active', true)
                ->get();

// OR
$categories = Category::where('name', 'Tech')
                ->orWhere('name', 'Sport')
                ->get();
```

Opérateurs disponibles

```
->where('price', '>', 100)
->where('name', 'like', '%tech%')
->where('created_at', '>=', now()->subDays(7))
->whereIn('status', ['active', 'pending'])
```

```
->whereNotNull('description')
->whereBetween('price', [10, 50])
```

Tri

```
Category::orderBy('name', 'asc')->get();
Category::orderBy('created_at', 'desc')->get();
Category::latest()->get(); // Par created_at desc
Category::oldest()->get(); // Par created_at asc
```

Limitation

```
Category::take(5)->get(); // 5 premiers
Category::limit(10)->get(); // Alias de take
Category::skip(10)->take(5)->get(); // Sauter 10, prendre 5
```

Pagination

```
$categories = Category::paginate(15); // 15 par page
$categories = Category::simplePaginate(15); // Sans numéros de page
```

Agrégations

```
$count = Category::count();
$max = Category::max('price');
$min = Category::min('price');
$avg = Category::avg('price');
$sum = Category::sum('price');
```

Sélection de colonnes

```
Category::select('id', 'name')->get();
Category::select('name as category_name')->get();
```

Requêtes avancées

Groupement

```
Category::where('is_active', true)
        ->where(function($query) {
            $query->where('price', '<', 100)
                  ->orWhere('discount', '>', 20);
        })
        ->get();
```

Existence

```
if (Category::where('name', 'Tech')->exists()) {
    // Existe
}
```

Récupérer une seule valeur

```
$name = Category::where('id', 1)->value('name');
```

Avec relations (eager loading)

```
// Charger les produits liés
$categories = Category::with('products')->get();

// Conditions sur les relations
$categories = Category::whereHas('products', function($query) {
    $query->where('price', '>', 100);
})->get();
```

Requêtes brutes (si besoin)

```
Category::whereRaw('YEAR(created_at) = ?', [2025])->get();
```

Debug

```
// Voir la requête SQL
Category::where('name', 'Tech')->toSql();
```

```
// Avec les bindings
Category::where('name', 'Tech')->dd(); // dump and die
```

Astuce : Enchaîne les méthodes pour construire des requêtes complexes pas à pas ! 🎯

Commandes CLI Laravel

Commandes de Migration

Créer une migration

```
# Migration simple
php artisan make:migration create_categories_table

# Modèle + migration
php artisan make:model Category -m

# Modèle + migration + controller + resource
php artisan make:model Category -mcr

# Toutes les options combinées
php artisan make:model Category -mcfrs
```

Options disponibles

- `-m` : crée la migration
- `-c` : crée le controller
- `-r` : ajoute les méthodes resource au controller
- `-f` : crée la factory
- `-s` : crée le seeder

Exécuter les migrations

```
# Lancer toutes les migrations en attente
php artisan migrate

# Rollback de la dernière batch
php artisan migrate:rollback

# Rollback de toutes les migrations
php artisan migrate:reset

# Reset complet et re-migration
php artisan migrate:fresh

# Fresh + exécution des seeders
php artisan migrate:fresh --seed

# Rollback et re-migration
php artisan migrate:refresh
```

Vérifier le statut

```
# Voir quelles migrations ont été exécutées
php artisan migrate:status
```

Partie Pratique (2h30)

Exercice 1 : Setup du projet (20 min)

Étapes

1. Créer le modèle avec migration, controller et resource
2. Configurer la migration
3. Exécuter la migration
4. Définir les propriétés `$fillable` dans le modèle
5. Créer les routes resource dans `web.php`

Routes

```
Route::resource('categories', CategoryController::class);
```

Exercice 2 : Validation (20 min)

1. Créer `StoreCategoryRequest`
2. Créer `UpdateCategoryRequest`
3. Définir les règles de validation
4. Personnaliser les messages d'erreur
5. Intégrer dans le controller

Exercice 3 : Vues Blade (45 min)

Layout principal

Créer `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">
    <title>@yield('title', 'Laravel App')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstr
ap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">Laravel CRUD</a>
        </div>
    </nav>

    <div class="container mt-4">
        @if(session('success'))
```

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

    @yield('content')
  </div>
</body>
</html>
```

Index (liste des catégories)

Créer `resources/views/categories/index.blade.php`

```
@extends('layouts.app')

@section('title', 'Catégories')

@section('content')
  <div class="d-flex justify-content-between align-items-center mb-4">
    <h1>Catégories</h1>
    <a href="{{ route('categories.create') }}" class="btn btn-primary">No
uvelle catégorie</a>
  </div>

  <table class="table table-striped">
    <thead>
      <tr>
        <th>ID</th>
        <th>Nom</th>
        <th>Description</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      @foreach($categories as $category)
        <tr>
          <td>{{ $category->id }}</td>
          <td>{{ $category->name }}</td>
          <td>{{ Str::limit($category->description, 50) }}</td>
```

```
                <td>
                    <a href="{{ route('categories.show', $category) }}" class="b
tn btn-sm btn-info">Voir</a>
                    <a href="{{ route('categories.edit', $category) }}" class="btn
btn-sm btn-warning">Modifier</a>
                    <form action="{{ route('categories.destroy', $category) }}"
method="POST" class="d-inline">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-sm btn-danger" on
click="return confirm('Êtes-vous sûr ?')">Supprimer</button>
                    </form>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>

{{ $categories->links() }}
@endsection
```

Exercice 4 : Factories et Seeders (25 min)

Factory

```
php artisan make:factory CategoryFactory
```

```
namespace Database\Factories;
use App\Models\Category;
use Illuminate\Database\Eloquent\Factories\Factory;
{
    protected $model = Category::class;

        return [
            'name' => $this->faker->unique()->words(2, true),
            'description' => $this->faker->paragraph(),
        ];
```

```
    }
}
```

Seeder

```
php artisan make:seeder CategorySeeder
```

```
namespace Database\Seeders;
use App\Models\Category;
use Illuminate\Database\Seeder;
{
    public function run()
    {
        Category::factory()->count(20)->create();
    }
}
```

Exécution

```
php artisan db:seed --class=CategorySeeder
```

TP Final : CRUD Complet (40 min)

Objectifs

1. Implémenter toutes les fonctionnalités CRUD
2. Ajouter la pagination
3. Gérer les messages flash
4. Ajouter une fonctionnalité de recherche (bonus)
5. Tester toutes les opérations

Bonus : Recherche

```
public function index(Request $request)
{
```

```
$query = Category::query();
if ($request->has('search')) {
    $query->where('name', 'like', '%' . $request->search . '%')
          ->orWhere('description', 'like', '%' . $request->search . '%');
}
return view('categories.index', compact('categories'));
```

Checklist de validation

- [ ] Liste des catégories affichée avec pagination
- [ ] Création d'une catégorie fonctionnelle
- [ ] Modification d'une catégorie fonctionnelle
- [ ] Suppression d'une catégorie fonctionnelle
- [ ] Validation des données en place
- [ ] Messages flash affichés correctement
- [ ] Factory et seeder fonctionnels
- [ ] Design responsive avec Bootstrap