

BAB 8 | Laravel MVC dan Database Connection

8.1 Pengenalan MVC (Model-View-Controller)

MVC adalah pola desain arsitektur yang memisahkan aplikasi menjadi tiga komponen utama. Arsitektur ini membantu pengembang mengorganisir kode dengan lebih baik, sehingga aplikasi lebih mudah dikembangkan dan dipelihara.

8.1.1 Komponen MVC

1. Model

- Mengelola data dan logika bisnis aplikasi
- Bertanggung jawab untuk interaksi dengan database
- Memproses aturan bisnis dan validasi data
- **Lokasi:** [app/Models/](#)

2. View

- Menampilkan data kepada pengguna (User Interface)
- Berisi markup HTML, CSS, dan JavaScript
- Menerima data dari Controller untuk ditampilkan
- **Lokasi:** [resources/views/](#)

3. Controller

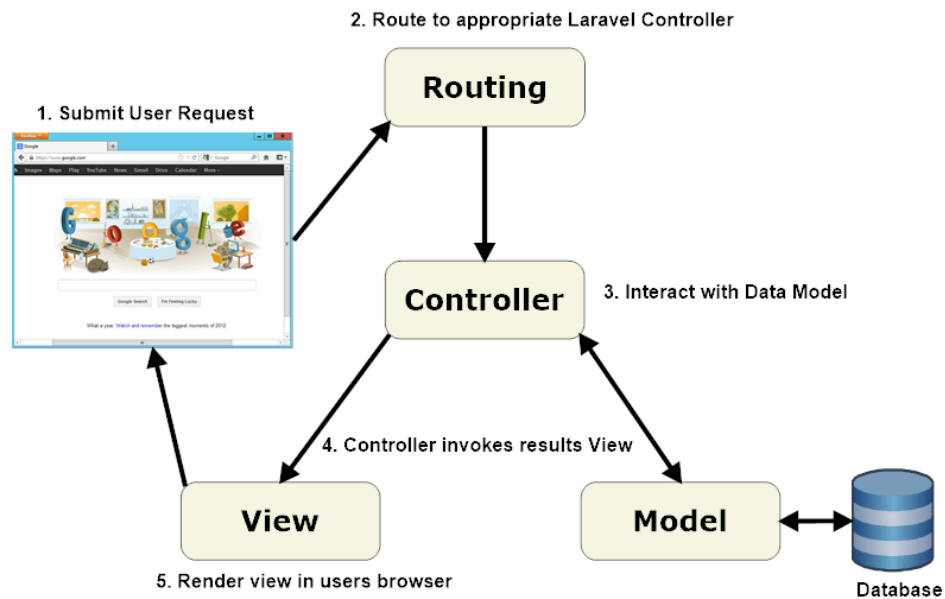
- Menjembatani Model dan View
- Menangani HTTP request dari pengguna
- Memproses input dan menentukan response
- **Lokasi:** [app/Http/Controllers/](#)

8.1.2 Manfaat Arsitektur MVC

Laravel mengadopsi pola MVC karena memberikan beberapa keuntungan penting:

1. **Separation of Concerns** - Setiap komponen memiliki tanggung jawab yang terpisah
2. **Maintainability** - Kode terstruktur dan mudah dipelihara
3. **Scalability** - Aplikasi dapat dikembangkan dengan mudah
4. **Code Reusability** - Komponen dapat digunakan kembali
5. **Team Collaboration** - Tim dapat bekerja secara paralel

8.1.3 Alur Kerja MVC di Laravel



Gambar tersebut menjelaskan alur kerja **MVC** di Laravel dalam 5 langkah sederhana:

1. Ketika pengguna mengakses halaman atau mengirimkan formulir di browser mereka, permintaan dikirim ke server.
2. Laravel menggunakan routing untuk mencari Controller yang tepat untuk menangani permintaan ini.
3. Controller kemudian berkomunikasi dengan Model untuk mengambil atau memproses data dari database.
4. Setelah data diproses, Controller mengirimkannya ke View yang bertugas menampilkan data tersebut.
5. Akhirnya, View menampilkan hasilnya di browser pengguna, seperti halaman web yang diperbarui atau informasi yang diminta.

8.2 Database Connection di Laravel

Laravel mendukung berbagai jenis database:

- MySQL/MariaDB
- PostgreSQL
- SQLite
- SQL Server

8.2.2 Konfigurasi Database

Konfigurasi database dilakukan melalui file `.env`:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=product_management
DB_USERNAME=root
DB_PASSWORD=
```

Penjelasan konfigurasi:

- **DB_CONNECTION** → jenis database yang digunakan
- **DB_HOST** → alamat server database (127.0.0.1 untuk localhost)
- **DB_PORT** → port database (3306 untuk MySQL)
- **DB_DATABASE** → nama database yang akan digunakan
- **DB_USERNAME** → username untuk login ke database
- **DB_PASSWORD** → password untuk login ke database

8.3 Migration

8.3.1 Pengenalan Migration

Migration adalah version control untuk database yang memungkinkan Anda mengelola struktur database dengan lebih terorganisir. Migration menyimpan perubahan struktur database dalam bentuk file PHP, sehingga mudah untuk:

- Membuat dan memodifikasi struktur database melalui kode
- Berbagi perubahan database dengan tim
- Melakukan rollback jika terjadi kesalahan
- Menjaga konsistensi database di berbagai environment

8.4 LANGKAH PRAKTIKUM

Langkah 1: Setup Project Laravel

Buat project Laravel baru:

```
composer create-project laravel/laravel product-management
cd product-management
```

Konfigurasi database di file `.env`:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=product_management
DB_USERNAME=root
```

```
DB_PASSWORD=
```

Buat database di MySQL:

```
CREATE DATABASE product_management;
```

Test koneksi database:

```
php artisan migrate
```

Jika berhasil, Laravel akan membuat tabel-tabel default (users, migrations, dll) di database Anda.

Langkah 2: Membuat Model, Controller, dan Migration

Buat Model Category dengan Migration:

```
php artisan make:model Category -m
```

Buat Model Product dengan Migration:

```
php artisan make:model Product -m
```

Buat Controller untuk Category:

```
php artisan make:controller CategoryController --resource
```

Flag `--resource` akan membuat controller dengan method CRUD lengkap (index, create, store, show, edit, update, destroy).

Buat Controller untuk Product:

```
php artisan make:controller ProductController --resource
```

Langkah 3: Mengedit Migration

Migration mendefinisikan struktur tabel yang akan dibuat di database.

Edit migration categories('database/migrations/xxxx_create_categories_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(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description')->nullable();
            $table->boolean('is_active')->default(true);
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('categories');
    }
};

```

Edit migration products (`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(): void
    {
        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->foreignId('category_id')->constrained()->onDelete('cascade');

        $table->boolean('is_active')->default(true);
        $table->timestamps();

        $table->index(['category_id', 'is_active']);
    });
}

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

```

Penjelasan struktur tabel products:

- `foreignId('category_id')->constrained()` → membuat foreign key ke tabel categories
- `onDelete('cascade')` → jika category dihapus, products yang terkait juga akan terhapus
- `index(['category_id', 'is_active'])` → membuat index untuk mempercepat query

Langkah 4: Menjalankan Migration

```
php artisan migrate
```

Perintah ini akan menjalankan semua migration yang belum dijalankan dan membuat tabel sesuai dengan definisi yang telah dibuat.

Langkah 5: Mengedit Model

Edit Model Category (`app/Models/Category.php`):

```

<?php

namespace App\Models;

```

```

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Category extends Model
{
    protected $fillable = [
        'name',
        'description',
        'is_active'
    ];

    protected $casts = [
        'is_active' => 'boolean'
    ];

    public function products(): HasMany
    {
        return $this->hasMany(Product::class);
    }

    public function scopeActive($query)
    {
        return $query->where('is_active', true);
    }
}

```

Penjelasan:

- `$fillable` → kolom yang boleh diisi secara mass assignment
- `$casts` → konversi tipe data otomatis
- `products()` → relasi one-to-many (satu category memiliki banyak products)
- `scopeActive()` → scope untuk filter data active

Edit Model Product(`app/Models/Product.php`):

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Product extends Model
{
    protected $fillable = [
        'name',
        'description',
        'price',
        'stock',
        'category_id',
        'is_active'
    ];

    protected $casts = [
        'price' => 'decimal:2',
        'is_active' => 'boolean'
    ];

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

    public function scopeActive($query)
    {
        return $query->where('is_active', true);
    }

    public function scopeInStock($query)
    {
        return $query->where('stock', '>', 0);
    }

    public function getFormattedPriceAttribute(): string
```

```

    {
        return 'Rp ' . number_format($this->price, 0, ',', '.');
    }
}

```

Penjelasan tambahan:

- `category()` → relasi belongs-to (product dimiliki oleh satu category)
- `scopeInStock()` → scope untuk filter produk yang tersedia
- `getFormattedPriceAttribute()` → accessor untuk format harga

Langkah 6: Mengedit Controller

Controller bertugas menangani logika aplikasi dan menjembatani antara Model dan View.

Edit CategoryController (`app/Http/Controllers/CategoryController.php`):

```

<?php

namespace App\Http\Controllers;

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

class CategoryController extends Controller
{
    public function index()
    {
        $categories =
Category::withCount('products')->latest()->paginate(10);
        return view('categories.index', compact('categories'));
    }

    public function create()
    {
        return view('categories.create');
    }

    public function store(Request $request)
    {
        $request->validate([

```

```

        'name' => 'required|string|max:255|unique:categories',
        'description' => 'nullable|string',
        'is_active' => 'boolean'
    ]);

    Category::create($request->all());

    return redirect()->route('categories.index')
        ->with('success', 'Category created
successfully!');
}

public function show(Category $category)
{
    $category->load('products');
    return view('categories.show', compact('category'));
}

public function edit(Category $category)
{
    return view('categories.edit', compact('category'));
}

public function update(Request $request, Category $category)
{
    $request->validate([
        'name' =>
'required|string|max:255|unique:categories,name,' . $category->id,
        'description' => 'nullable|string',
        'is_active' => 'boolean'
    ]);

    $category->update($request->all());

    return redirect()->route('categories.index')
        ->with('success', 'Category updated
successfully!');
}

public function destroy(Category $category)
{
    if ($category->products()->count() > 0) {

```

```

        return redirect()->route('categories.index')
            ->with('error', 'Cannot delete
category with products!');
    }

    $category->delete();

    return redirect()->route('categories.index')
        ->with('success', 'Category deleted
successfully!');
    }
}

```

Edit ProductController (`app/Http/Controllers/ProductController.php`):

```

<?php

namespace App\Http\Controllers;

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

class ProductController extends Controller
{
    public function index(Request $request)
    {
        $query = Product::with('category');

        if ($request->has('search')) {
            $search = $request->get('search');
            $query->where('name', 'LIKE', "%{$search}%")
                ->orWhere('description', 'LIKE', "%{$search}%");
        }

        if ($request->has('category_id') && $request->category_id
!= '') {
            $query->where('category_id', $request->category_id);
        }

        $products = $query->latest()->paginate(10);
    }
}

```

```

        $categories = Category::active()->get();

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

    public function create()
    {
        $categories = Category::active()->get();
        return view('products.create', compact('categories'));
    }

    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
            'price' => 'required|numeric|min:0',
            'stock' => 'required|integer|min:0',
            'category_id' => 'required|exists:categories,id',
            'is_active' => 'boolean'
        ]);

        Product::create($request->all());

        return redirect()->route('products.index')
            ->with('success', 'Product created
successfully!');
    }

    public function show(Product $product)
    {
        $product->load('category');
        return view('products.show', compact('product'));
    }

    public function edit(Product $product)
    {
        $categories = Category::active()->get();
        return view('products.edit', compact('product',
'categories'));
    }

```

```

public function update(Request $request, Product $product)
{
    $request->validate([
        'name' => 'required|string|max:255',
        'description' => 'nullable|string',
        'price' => 'required|numeric|min:0',
        'stock' => 'required|integer|min:0',
        'category_id' => 'required|exists:categories,id',
        'is_active' => 'boolean'
    ]);

    $product->update($request->all());

    return redirect()->route('products.index')
        ->with('success', 'Product updated
successfully!');
}

public function destroy(Product $product)
{
    $product->delete();

    return redirect()->route('products.index')
        ->with('success', 'Product deleted
successfully!');
}
}

```

Langkah 7: Membuat Routes

Routes mendefinisikan URL yang dapat diakses di aplikasi dan menghubungkannya dengan Controller.

Edit file routes/web.php:

```

<?php

use App\Http\Controllers\CategoryController;
use App\Http\Controllers\ProductController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return redirect()->route('products.index');
});

```

```
});
```

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

Langkah 8: Membuat Views

Views berisi tampilan HTML yang akan dilihat oleh pengguna.

Buat layout utama(resources/views/layouts/app.blade.php`):

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width,  
initial-scale=1.0">  
    <title>@yield('title', 'Product Management')</title>  
    <link  
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootst  
rap.min.css" rel="stylesheet">  
</head>  
<body>  
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">  
        <div class="container">  
            <a class="navbar-brand" href="{{  
route('products.index') }}">Product Management</a>  
            <div class="navbar-nav">  
                <a class="nav-link" href="{{  
route('categories.index') }}">Categories</a>  
                <a class="nav-link" href="{{  
route('products.index') }}">Products</a>  
            </div>  
        </div>  
    </nav>  
  
    <div class="container mt-4">  
        @if(session('success'))  
            <div class="alert alert-success alert-dismissible fade  
show">  
                {{ session('success') }}  
                <button type="button" class="btn-close"  
data-bs-dismiss="alert"></button>  
            </div>
```

```

        @endif

        @if(session('error'))
            <div class="alert alert-danger alert-dismissible fade
show">
                {{ session('error') }}
                <button type="button" class="btn-close"
data-bs-dismiss="alert"></button>
            </div>
        @endif

        @yield('content')
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstra
p.bundle.min.js"></script>
</body>
</html>

```

Buat view index products(resources/views/products/index.blade.php`):

```

@extends('layouts.app')

@section('title', 'Products List')

@section('content')
    <div class="d-flex justify-content-between align-items-center
mb-4">
        <h2>Products Management</h2>
        <a href="{{ route('products.create') }}" class="btn
btn-primary">Add New Product</a>
    </div>

    <div class="row mb-3">
        <div class="col-md-6">
            <form method="GET" class="d-flex">
                <input type="text" name="search" class="form-control
me-2" placeholder="Search products..." value="{{ request('search')
}}">
                <button type="submit" class="btn

```

```

btn-outline-secondary">Search</button>
    </form>
</div>
<div class="col-md-3">
    <form method="GET">
        <select name="category_id" class="form-select"
onchange="this.form.submit()">
            <option value="">All Categories</option>
            @foreach($categories as $category)
                <option value="{{ $category->id }}" {{
request('category_id') == $category->id ? 'selected' : '' }}>
                    {{ $category->name }}
                </option>
            @endforeach
        </select>
    </form>
</div>
</div>

<div class="card">
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Category</th>
                        <th>Price</th>
                        <th>Stock</th>
                        <th>Status</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach($products as $product)
                        <tr>
                            <td>{{ $product->id }}</td>
                            <td>{{ $product->name }}</td>
                            <td>{{ $product->category->name
}}</td>
                            <td>{{ $product->formatted_price

```

```

}}</td>

        <td>
            <span class="badge {{
$product->stock > 0 ? 'bg-success' : 'bg-danger' }}">
                {{ $product->stock }}
            </span>
        </td>
        <td>
            <span class="badge {{
$product->is_active ? 'bg-success' : 'bg-secondary' }}">
                {{ $product->is_active ?
'Active' : 'Inactive' }}
            </span>
        </td>
        <td>
            <div class="btn-group
btn-group-sm">
                <a href="{{
route('products.show', $product) }}" class="btn btn-info">View</a>
                <a href="{{
route('products.edit', $product) }}" class="btn
btn-warning">Edit</a>
                <form method="POST" action="{{
route('products.destroy', $product) }}" class="d-inline">
                    @csrf
                    @method('DELETE')
                    <button type="submit"
class="btn btn-danger" onclick="return confirm('Are you
sure?')">Delete</button>
                </form>
            </div>
        </td>
    </tr>
@empty
    <tr>
        <td colspan="7" class="text-center">No
products found</td>
    </tr>
@endforelse
</tbody>
</table>
</div>

```

```

        </div>
    </div>

    <div class="d-flex justify-content-center mt-4">
        {{ $products->links() }}
    </div>
@endsection

```

Buat view create products(`resources/views/products/create.blade.php`):

```

@extends('layouts.app')

@section('title', 'Add New Product')

@section('content')
    <div class="mb-4">
        <h2>Add New Product</h2>
    </div>

    <div class="card">
        <div class="card-body">
            <form method="POST" action="{{ route('products.store') }}">
                @csrf

                <div class="row">
                    <div class="col-md-6 mb-3">
                        <label class="form-label">Product Name <span
class="text-danger">*</span></label>
                        <input type="text" class="form-control
@error('name') is-invalid @enderror"
                            name="name" value="{{ old('name') }}"
required>
                            @error('name')
                                <div class="invalid-feedback">{{ $message
}}</div>
                            @enderror
                    </div>

                    <div class="col-md-6 mb-3">
                        <label class="form-label">Category <span
class="text-danger">*</span></label>

```

```

        <select class="form-select
@error('category_id') is-invalid @enderror"
            name="category_id" required>
            <option value="">Select Category</option>
            @foreach($categories as $category)
                <option value="{{ $category->id }}"
                    {{ old('category_id') ==
$category->id ? 'selected' : '' }}>
                    {{ $category->name }}
                </option>
            @endforeach
        </select>
        @error('category_id')
            <div class="invalid-feedback">{{ $message
}}</div>
        @enderror
    </div>
</div>

<div class="row">
    <div class="col-md-6 mb-3">
        <label class="form-label">Price <span
class="text-danger">*</span></label>
        <input type="number" step="0.01"
            class="form-control @error('price')
is-invalid @enderror"
            name="price" value="{{ old('price') }}"
required>
        @error('price')
            <div class="invalid-feedback">{{ $message
}}</div>
        @enderror
    </div>

    <div class="col-md-6 mb-3">
        <label class="form-label">Stock <span
class="text-danger">*</span></label>
        <input type="number"
            class="form-control @error('stock')
is-invalid @enderror"
            name="stock" value="{{ old('stock', 0)
}}" required>

```

```

        @error('stock')
        <div class="invalid-feedback">{{ $message
    }}</div>

    @enderror
</div>
</div>

<div class="mb-3">
    <label class="form-label">Description</label>
    <textarea class="form-control
@error('description') is-invalid @enderror"
        name="description" rows="3">{{
old('description') }}</textarea>
    @error('description')
        <div class="invalid-feedback">{{ $message
    }}</div>

    @enderror
</div>

<div class="mb-3">
    <div class="form-check">
        <input class="form-check-input"
type="checkbox" name="is_active"
            id="is_active" value="1" {{
old('is_active', true) ? 'checked' : '' }}>
        <label class="form-check-label"
for="is_active">
            Active
        </label>
    </div>
</div>

<div class="d-flex gap-2">
    <button type="submit" class="btn btn-primary">Save
Product</button>
    <a href="{{ route('products.index') }}" class="btn
btn-secondary">Cancel</a>
</div>
</form>
</div>
</div>
@endsection

```

Buat view edit products(`resources/views/products/edit.blade.php`):

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

@section('title', 'Edit Product')

@section('content')


## Edit Product



<form method="POST" action="{{ route('products.update', $product) }}">
        @csrf
        @method('PUT')

        <div class="row">
            <div class="col-md-6 mb-3">
                <label class="form-label">Product Name <span
class="text-danger">*</span></label>
                <input type="text" class="form-control
@error('name') is-invalid @enderror"
                    name="name" value="{{ old('name',
$product->name) }}" required>
                @error('name')
                    <div class="invalid-feedback">{{ $message
}}</div>
                @enderror
            </div>

            <div class="col-md-6 mb-3">
                <label class="form-label">Category <span
class="text-danger">*</span></label>
                <select class="form-select
@error('category_id') is-invalid @enderror"
                    name="category_id" required>
                    <option value="">Select Category</option>
                    @foreach($categories as $category)


```

```

        <option value="{{ $category->id }}"
            {{ old('category_id',
$product->category_id) == $category->id ? 'selected' : '' }}>
            {{ $category->name }}
        </option>
    @endforeach
</select>
    @error('category_id')
        <div class="invalid-feedback">{{ $message
    }}</div>

    @enderror
</div>

</div>

<div class="row">
    <div class="col-md-6 mb-3">
        <label class="form-label">Price <span
class="text-danger">*</span></label>
        <input type="number" step="0.01"
            class="form-control @error('price')
is-invalid @enderror"
            name="price" value="{{ old('price',
$product->price) }}" required>
        @error('price')
            <div class="invalid-feedback">{{ $message
    }}</div>

        @enderror
    </div>

    <div class="col-md-6 mb-3">
        <label class="form-label">Stock <span
class="text-danger">*</span></label>
        <input type="number"
            class="form-control @error('stock')
is-invalid @enderror"
            name="stock" value="{{ old('stock',
$product->stock) }}" required>
        @error('stock')
            <div class="invalid-feedback">{{ $message
    }}</div>

        @enderror
    </div>

```

```

    </div>

    <div class="mb-3">
        <label class="form-label">Description</label>
        <textarea class="form-control
@error('description') is-invalid @enderror"
            name="description" rows="3">{{
old('description', $product->description) }}</textarea>
        @error('description')
            <div class="invalid-feedback">{{ $message
        }}</div>

        @enderror
    </div>

    <div class="mb-3">
        <div class="form-check">
            <input class="form-check-input"
type="checkbox" name="is_active"
            id="is_active" value="1" {{
old('is_active', $product->is_active) ? 'checked' : '' }}>
            <label class="form-check-label"
for="is_active">
                Active
            </label>
        </div>
    </div>

    <div class="d-flex gap-2">
        <button type="submit" class="btn
btn-primary">Update Product</button>
        <a href="{{ route('products.index') }}" class="btn
btn-secondary">Cancel</a>
    </div>
</form>
</div>
</div>
@endsection

```

Buat view show products(`resources/views/products/show.blade.php`):

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

@section('title', 'Product Details')

@section('content')
<div class="mb-4">
    <a href="{{ route('products.index') }}" class="btn btn-secondary">
        <i class="bi bi-arrow-left"></i> Back
    </a>
</div>

<div class="card">
    <div class="card-header bg-primary text-white">
        <h3 class="mb-0">Product Information</h3>
    </div>
    <div class="card-body">
        <table class="table table-bordered">
            <tr>
                <th width="30%">ID</th>
                <td>{{ $product->id }}</td>
            </tr>
            <tr>
                <th>Product Name</th>
                <td><strong>{{ $product->name }}</strong></td>
            </tr>
            <tr>
                <th>Category</th>
                <td>
                    <span class="badge bg-info">{{
$product->category->name }}</span>
                </td>
            </tr>
            <tr>
                <th>Price</th>
                <td>{{ $product->formatted_price }}</td>
            </tr>
            <tr>
                <th>Stock</th>
                <td>
                    <span class="badge {{ $product->stock > 0 ?
```

```

'bg-success' : 'bg-danger' }}">
        {{ $product->stock }} items
    </span>
</td>
</tr>
<tr>
    <th>Description</th>
    <td>{{ $product->description ?? '-' }}</td>
</tr>
<tr>
    <th>Status</th>
    <td>
        <span class="badge {{ $product->is_active ?
'bg-success' : 'bg-secondary' }}">
            {{ $product->is_active ? 'Active' :
'Inactive' }}
        </span>
    </td>
</tr>
<tr>
    <th>Created At</th>
    <td>{{ $product->created_at->format('d M Y H:i')
}}</td>
</tr>
<tr>
    <th>Updated At</th>
    <td>{{ $product->updated_at->format('d M Y H:i')
}}</td>
</tr>
</table>

<div class="d-flex gap-2 mt-3">
    <a href="{{ route('products.edit', $product) }}"
class="btn btn-warning">Edit</a>
    <form method="POST" action="{{
route('products.destroy', $product) }}" class="d-inline">
        @csrf
        @method('DELETE')
        <button type="submit" class="btn btn-danger"
onclick="return confirm('Are you
sure?')">Delete</button>
    </form>

```

```

        </div>
    </div>
</div>
@endsection

```

Buat view index categories(`resources/views/categories/index.blade.php`):

```

@extends('layouts.app')

@section('title', 'Categories List')

@section('content')
<div class="d-flex justify-content-between align-items-center mb-4">
    <h2>Categories Management</h2>
    <a href="{{ route('categories.create') }}" class="btn btn-primary">Add New Category</a>
</div>

<div class="card">
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Description</th>
                        <th>Products Count</th>
                        <th>Status</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    @forelse($categories as $category)
                        <tr>
                            <td>{{ $category->id }}</td>
                            <td><strong>{{ $category->name }}</strong></td>
                            <td>{{
Str::limit($category->description, 50) ?? '-' }}</td>
                            <td>

```

```

        <span class="badge bg-primary">
            {{ $category->products_count
}} products
        </span>
    </td>
    <td>
        <span class="badge {{
$category->is_active ? 'bg-success' : 'bg-secondary' }}">
            {{ $category->is_active ?
'Active' : 'Inactive' }}
        </span>
    </td>
    <td>
        <div class="btn-group
btn-group-sm">
            <a href="{{
route('categories.show', $category) }}" class="btn
btn-info">View</a>
            <a href="{{
route('categories.edit', $category) }}" class="btn
btn-warning">Edit</a>
            <form method="POST" action="{{
route('categories.destroy', $category) }}" class="d-inline">
                @csrf
                @method('DELETE')
                <button type="submit"
class="btn btn-danger"
                    onclick="return
confirm('Are you sure? This will also delete all products in this
category!')">
                    Delete
                </button>
            </form>
        </div>
    </td>
</tr>
@empty
<tr>
    <td colspan="6" class="text-center">No
categories found</td>
</tr>
@endforelse

```

```

        </tbody>
    </table>
</div>
</div>
</div>
</div>

<div class="d-flex justify-content-center mt-4">
    {{ $categories->links() }}
</div>
@endsection

```

Buat view create categories(`resources/views/categories/create.blade.php`):

```

blade@extends('layouts.app')

@section('title', 'Add New Category')

@section('content')
<div class="mb-4">
    <h2>Add New Category</h2>
</div>

<div class="card">
    <div class="card-body">
        <form method="POST" action="{{ route('categories.store') }}">
            @csrf

            <div class="mb-3">
                <label class="form-label">Category Name <span
class="text-danger">*</span></label>
                <input type="text" class="form-control
@error('name') is-invalid @enderror"
                    name="name" value="{{ old('name') }}"
required autofocus>
                @error('name')
                    <div class="invalid-feedback">{{ $message
}}</div>
                @enderror
            </div>

            <div class="mb-3">
                <label class="form-label">Description</label>

```

```

        <textarea class="form-control
@error('description') is-invalid @enderror"
            name="description" rows="4"
placeholder="Enter category description...">{{ old('description')
}}</textarea>

        @error('description')
            <div class="invalid-feedback">{{ $message
}}</div>

        @enderror
    </div>

    <div class="mb-3">
        <div class="form-check">
            <input class="form-check-input"
type="checkbox" name="is_active"
                id="is_active" value="1" {{
old('is_active', true) ? 'checked' : '' }}>
            <label class="form-check-label"
for="is_active">
                Active
            </label>
        </div>
    </div>

    <div class="d-flex gap-2">
        <button type="submit" class="btn btn-primary">Save
Category</button>
        <a href="{{ route('categories.index') }}"
class="btn btn-secondary">Cancel</a>
    </div>
</form>
</div>
</div>
@endsection

```

Buat view edit categories(`resources/views/categories/edit.blade.php`):

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

@section('title', 'Edit Category')
@section('content')
<div class="mb-4">
    <h2>Edit Category</h2>
</div>

<div class="card">
    <div class="card-body">
        <form method="POST" action="{{ route('categories.update',
$category) }}">
            @csrf
            @method('PUT')

            <div class="mb-3">
                <label class="form-label">Category Name <span
class="text-danger">*</span></label>
                <input type="text" class="form-control
@error('name') is-invalid @enderror"
                    name="name" value="{{ old('name',
$category->name) }}" required autofocus>
                @error('name')
                    <div class="invalid-feedback">{{ $message
}}</div>
                @enderror
            </div>

            <div class="mb-3">
                <label class="form-label">Description</label>
                <textarea class="form-control
@error('description') is-invalid @enderror"
                    name="description" rows="4"
placeholder="Enter category description...">{{ old('description',
$category->description) }}</textarea>
                @error('description')
                    <div class="invalid-feedback">{{ $message
}}</div>
                @enderror
            </div>
        </form>
    </div>
</div>
```

```

        <div class="mb-3">
            <div class="form-check">
                <input class="form-check-input"
type="checkbox" name="is_active"
                    id="is_active" value="1" {{
old('is_active', $category->is_active) ? 'checked' : '' }}>
                <label class="form-check-label"
for="is_active">
                    Active
                </label>
            </div>
        </div>

        <div class="d-flex gap-2">
            <button type="submit" class="btn
btn-primary">Update Category</button>
            <a href="{{ route('categories.index') }}"
class="btn btn-secondary">Cancel</a>
        </div>
    </form>
</div>
</div>
@endsection

```

Buat view show categories(`resources/views/categories/show.blade.php`):

```

@extends('layouts.app')
@section('title', 'Category Details')
@section('content')
    <div class="mb-4">
        <a href="{{ route('categories.index') }}" class="btn
btn-secondary">
            <i class="bi bi-arrow-left"></i> Back
        </a>
    </div>

    <div class="card mb-4">
        <div class="card-header bg-primary text-white">
            <h3 class="mb-0">Category Information</h3>
        </div>
        <div class="card-body">

```

```

<table class="table table-bordered">
  <tr>
    <th width="30%">ID</th>
    <td>{{ $category->id }}</td>
  </tr>
  <tr>
    <th>Category Name</th>
    <td><strong>{{ $category->name }}</strong></td>
  </tr>
  <tr>
    <th>Description</th>
    <td>{{ $category->description ?? '-' }}</td>
  </tr>
  <tr>
    <th>Total Products</th>
    <td>
      <span class="badge bg-primary">{{
$category->products->count() }} products</span>
    </td>
  </tr>
  <tr>
    <th>Status</th>
    <td>
      <span class="badge {{ $category->is_active ?
'bg-success' : 'bg-secondary' }}">
        {{ $category->is_active ? 'Active' :
'Inactive' }}
      </span>
    </td>
  </tr>
  <tr>
    <th>Created At</th>
    <td>{{ $category->created_at->format('d M Y H:i')
}}</td>
  </tr>
  <tr>
    <th>Updated At</th>
    <td>{{ $category->updated_at->format('d M Y H:i')
}}</td>
  </tr>
</table>

```

```

        <div class="d-flex gap-2 mt-3">
            <a href="{{ route('categories.edit', $category) }}"
class="btn btn-warning">Edit</a>
            <form method="POST" action="{{
route('categories.destroy', $category) }}" class="d-inline">
                @csrf
                @method('DELETE')
                <button type="submit" class="btn btn-danger"
                    onclick="return confirm('Are you sure?
This will also delete all products in this category!')">
                    Delete
                </button>
            </form>
        </div>
    </div>
</div>

<!-- Products List in This Category -->
<div class="card">
    <div class="card-header bg-info text-white d-flex
justify-content-between align-items-center">
        <h4 class="mb-0">Products in {{ $category->name }}</h4>
        <a href="{{ route('products.create') }}" class="btn
btn-light btn-sm">Add Product</a>
    </div>
    <div class="card-body">
        @if($category->products->count() > 0)
            <div class="table-responsive">
                <table class="table table-hover">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Product Name</th>
                            <th>Price</th>
                            <th>Stock</th>
                            <th>Status</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach($category->products as $product)
                            <tr>

```

```

<td>{{ $product->id }}</td>
<td><strong>{{ $product->name
}}</strong></td>

<td>{{ $product->formatted_price
}}</td>

<td>
    <span class="badge {{
$product->stock > 0 ? 'bg-success' : 'bg-danger' }}">
        {{ $product->stock }}
    </span>
</td>
<td>
    <span class="badge {{
$product->is_active ? 'bg-success' : 'bg-secondary' }}">
        {{ $product->is_active ?
'Active' : 'Inactive' }}
    </span>
</td>
<td>
    <div class="btn-group
btn-group-sm">
        <a href="{{
route('products.show', $product) }}" class="btn btn-info">View</a>
        <a href="{{
route('products.edit', $product) }}" class="btn
btn-warning">Edit</a>
    </div>
</td>
</tr>
@endforeach
</tbody>
</table>
</div>
@else
    <div class="alert alert-info mb-0">
        <i class="bi bi-info-circle"></i> No products in
this category yet.
    </div>
@endif
</div>
</div>
@endsection

```

Langkah 9: Testing dan Debugging

Setelah semua komponen selesai dibuat, saatnya menjalankan dan menguji aplikasi.

Jalankan server development:

```
php artisan serve
```

Akses aplikasi di browser:

```
http://localhost:8000
```

Test semua fungsi CRUD:

- **Create:** Tambah kategori dan produk baru
- **Read:** Lihat daftar dan detail
- **Update:** Edit data yang ada
- **Delete:** Hapus data

8.5 OPERASI CRUD

Laravel menyediakan beberapa cara untuk berinteraksi dengan database. Berikut adalah perbandingan tiga metode utama.

8.5.1 Raw Query

Raw Query memungkinkan Anda menulis SQL secara langsung. Metode ini cocok untuk query kompleks yang sulit dilakukan dengan Query Builder atau Eloquent.

```
use Illuminate\Support\Facades\DB;

// Create
DB::insert('INSERT INTO products (name, price, stock, category_id,
created_at, updated_at) VALUES (?, ?, ?, ?, NOW(), NOW())',
    ['Laptop Gaming', 15000000, 10, 1]);

// Read
$products = DB::select('SELECT p.*, c.name as category_name FROM
products p JOIN categories c ON p.category_id = c.id WHERE
p.is_active = ?', [1]);

// Update
DB::update('UPDATE products SET price = ?, updated_at = NOW()
WHERE id = ?', [14000000, 1]);
```

```
// Delete
DB::delete('DELETE FROM products WHERE id = ?', [1]);
```

Kelebihan:

- Fleksibel untuk query kompleks
- Performa optimal untuk operasi khusus

Kekurangan:

- Rentan SQL injection jika tidak hati-hati
- Tidak typesafe
- Sulit di-maintain

8.5.2 Query Builder

Query Builder menyediakan interface yang lebih bersih dan aman untuk membangun query database.

```
use Illuminate\Support\Facades\DB;

// Create
DB::table('products')->insert([
    'name' => 'Smartphone',
    'price' => 8000000,
    'stock' => 20,
    'category_id' => 2,
    'created_at' => now(),
    'updated_at' => now()
]);

// Read
$products = DB::table('products')
    ->join('categories', 'products.category_id', '=',
    'categories.id')
    ->select('products.*', 'categories.name as category_name')
    ->where('products.is_active', true)
    ->orderBy('products.created_at', 'desc')
    ->get();

// Update
DB::table('products')
    ->where('id', 1)
    ->update([
```

```

        'price' => 7500000,
        'updated_at' => now()
    ]);

// Delete
DB::table('products')->where('id', 1)->delete();

```

Kelebihan:

- Lebih aman dari SQL injection
- Sintaks lebih bersih dan mudah dibaca
- Mendukung method chaining

Kekurangan:

- Tidak memiliki fitur ORM lengkap
- Tidak ada relasi model

8.5.3 Eloquent ORM (Recommended)

Eloquent adalah **ORM (Object-Relational Mapping)** Laravel yang paling powerful dan direkomendasikan untuk digunakan. Eloquent memungkinkan Anda bekerja dengan database menggunakan model dan object-oriented approach.

```

php
use App\Models\Product;

// Create
$product = Product::create([
    'name' => 'Tablet',
    'price' => 5000000,
    'stock' => 15,
    'category_id' => 3
]);

// Read
$products = Product::with('category')
    ->where('is_active', true)
    ->orderBy('created_at', 'desc')
    ->get();

// Read Single
$product = Product::find(1);

```

```
// atau
$product = Product::where('name', 'Tablet')->first();

// Update
$product = Product::find(1);
$product->update(['price' => 4500000]);

// atau update langsung
Product::where('id', 1)->update(['price' => 4500000]);

// Delete
$product = Product::find(1);
$product->delete();

// atau delete langsung
Product::destroy(1);
// atau delete multiple
Product::destroy([1, 2, 3]);
```

Kelebihan:

- Sintaks paling bersih dan elegan
- Mendukung relasi antar model
- Fitur lengkap (casting, accessor, mutator, scope, dll)
- Mudah di-maintain dan test

Kekurangan:

- Sedikit overhead performa untuk query sangat kompleks
- Learning curve lebih tinggi

8.6 Kesimpulan

Dalam bab ini, Anda telah mempelajari:

1. **Konsep MVC** - Arsitektur yang memisahkan logika aplikasi menjadi Model, View, dan Controller
2. **Database Connection** - Cara mengkonfigurasi dan menghubungkan Laravel dengan database
3. **Migration** - Version control untuk database yang memudahkan pengelolaan struktur tabel
4. **Eloquent ORM** - Cara berinteraksi dengan database menggunakan model yang object-oriented
5. **CRUD Operations** - Implementasi Create, Read, Update, Delete menggunakan Laravel
6. **Routing dan Controller** - Menghubungkan URL dengan logika aplikasi
7. **Blade Templating** - Membuat tampilan yang dinamis dan reusable

Dengan memahami konsep-konsep ini, Anda sudah dapat membuat aplikasi web CRUD lengkap menggunakan Laravel. Praktikkan terus dan eksplorasi fitur-fitur Laravel lainnya untuk meningkatkan kemampuan Anda.

"Setiap baris kode yang kamu tulis adalah langkah menuju puncak. Teruslah mendaki, karena pemandangan terindah ada di 3726 mdpl."— 3726 mdpl