- Entrega 8: Laravel parte 2
  - Programa 4: Modelos
    - Conceptos básicos:
    - Protección de asignación masiva
    - Relaciones entre modelos
    - Operaciones CRUD
    - Proyecto blog
      - Paso 1: Crear un nuevo proyecto de Laravel
      - Paso 2:Crear el modelo de la publicación y migración
      - Paso 3: Definir la migración
      - Paso 4: Definir la lógica del modelo
      - Paso 5: Crear la ruta
      - Paso 6: creamos la vista de blade
      - Paso 7: servimos la aplicación
  - Programa 5: Controladores
    - Proyecto Controllers
      - Paso 1: Creamos un nuevo proyecto
      - Paso 2: Creamos un controlador
      - Paso 3: Agregamos ruta
      - Paso 4: Definir el index método en UserController.php
      - Paso 5: Crear la vista
  - Programa 6: CRUD
    - Proyecto Clothing store
      - Paso 1: Nuevo proyecto en Laravel
      - Paso 2: Creamos modelo, migración y controlador
      - Paso 3: rutas y vistas básicas
      - Paso 4: Creamos el controlador
      - Paso 5: Añadimos en fillable el modelo
      - Paso 6: Creamos las vistas de cada controlador
    - Actividad

Entrega 8: Laravel parte 2

Programa 4: Modelos

Conceptos básicos:

Cada clase del modelo representa a una sola tabla en la base de datos

La tabla asociada con el con el modelo tiene su nombre con su manera de escribirse específica en la tabla, si el modelo es Post, su nombre en la tabla será en minúscula y en plural -> posts

La clave principal es el id

Las marcas de tiempo se gestionan automáticamente

Protección de asignación masiva

Eloquent protege contra vulnerabilidades de asingación masiva

- $fillable se usa para definir que atributos van a poder ser modificados

  ```
  protected $fillable = ['title', 'body', 'published'];
  ```

- $guarded se usa para definir que atributos no se podrán modificar

  ```
  protected $guarded = ['id'];
  ```

Normalmente solo se debe usar uno de estos por modelo

Relaciones entre modelos

Laravel simplifica la definición de relaciones entre modelos. Estas son las más comunes

- hasOne
- hasMany
- belongsTo
- belongsToMany
- hasManyThrough

Operaciones CRUD

Eloquent ofrece métodos elegantes para crear, leer, actualizar y eliminar registros

Proyecto blog

Paso 1: Crear un nuevo proyecto de Laravel

Para esto usaremos el comando

```
composer create-project laravel/laravel blog
```

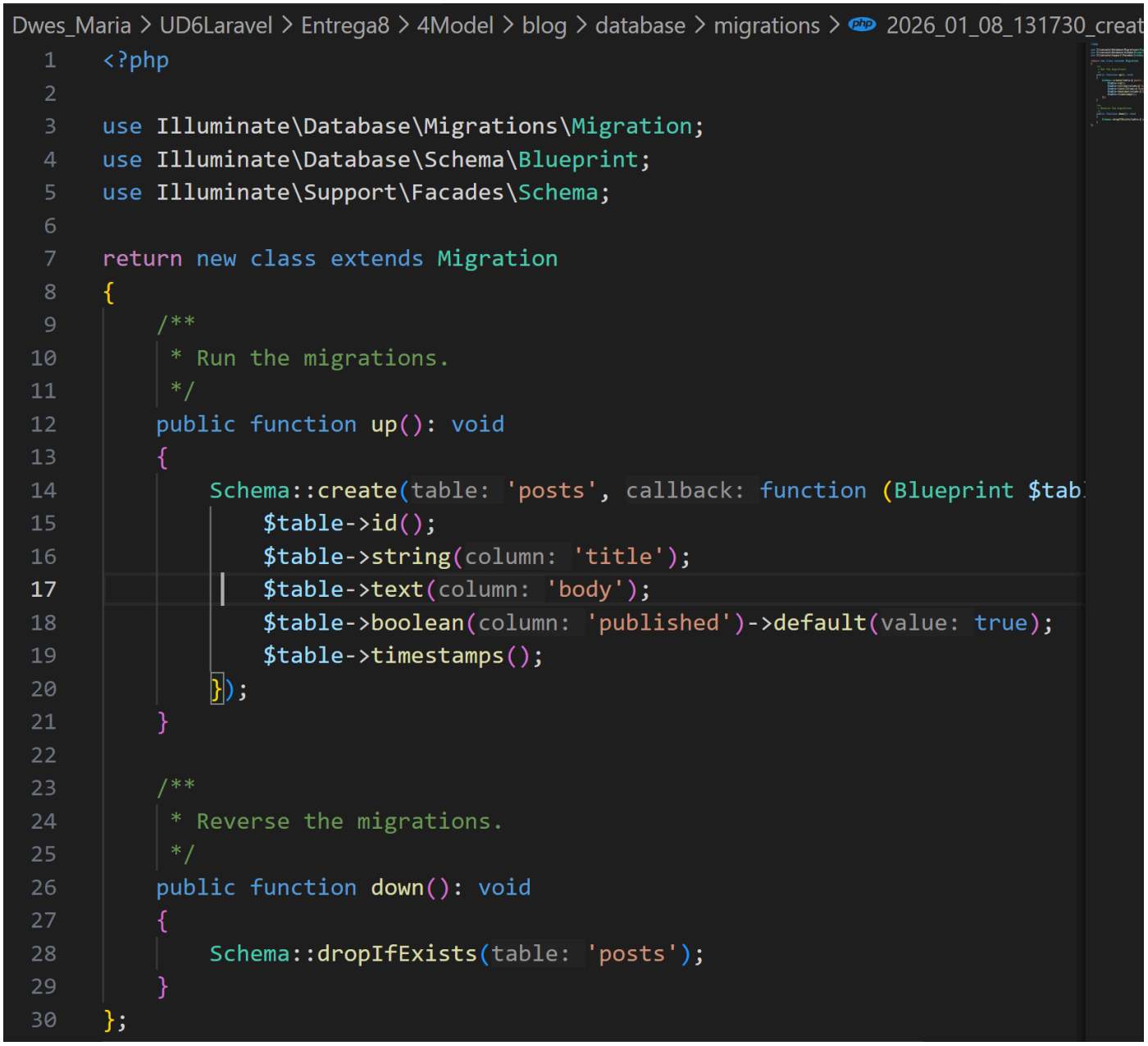
Paso 2:Crear el modelo de la publicación y migración

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

Esto crea el modelo Post y la migración posts

Paso 3: Definir la migración

Editamos el archivo de la migración



y luego ejecutamos la migración:

```
php artisan migrate
```

Paso 4: Definir la lógica del modelo

Dwes_Maria > UD6Laravel > Entrega8 > 4Model > blog > app > Models > Post.php > PHP > Post

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

6 references | 0 implementations
class Post extends Model
{

    0 references
    protected $fillable = ['title', 'body', 'published'];

    // Accessor
    0 references | 0 overrides
    public function getTitleAttribute($value): string
    {
        return ucfirst(string: $value); // Capitalize the first letter of the title
    }

    // Scope
    0 references | 0 overrides
    public function scopePublished($query): mixed
    {
        return $query->where('published', true); // Filter to get only published posts
    }

    // Auto-insert sample posts when the model is first loaded
    0 references | 0 overrides
    protected static function booted(): void
    {
        if (self::count() === 0) { // Check if there are no posts
            // Insert sample posts
            self::create(attributes: [
                'title' => 'welcome post',
                'body' => 'This is a welcome post created automatically.',
                'published' => true,
            ]);

            self::create(attributes: [
                'title' => 'draft example',
                'body' => 'This post is a draft and won't show on the page.',
                'published' => false,
            ]);
        }
    }
}

```

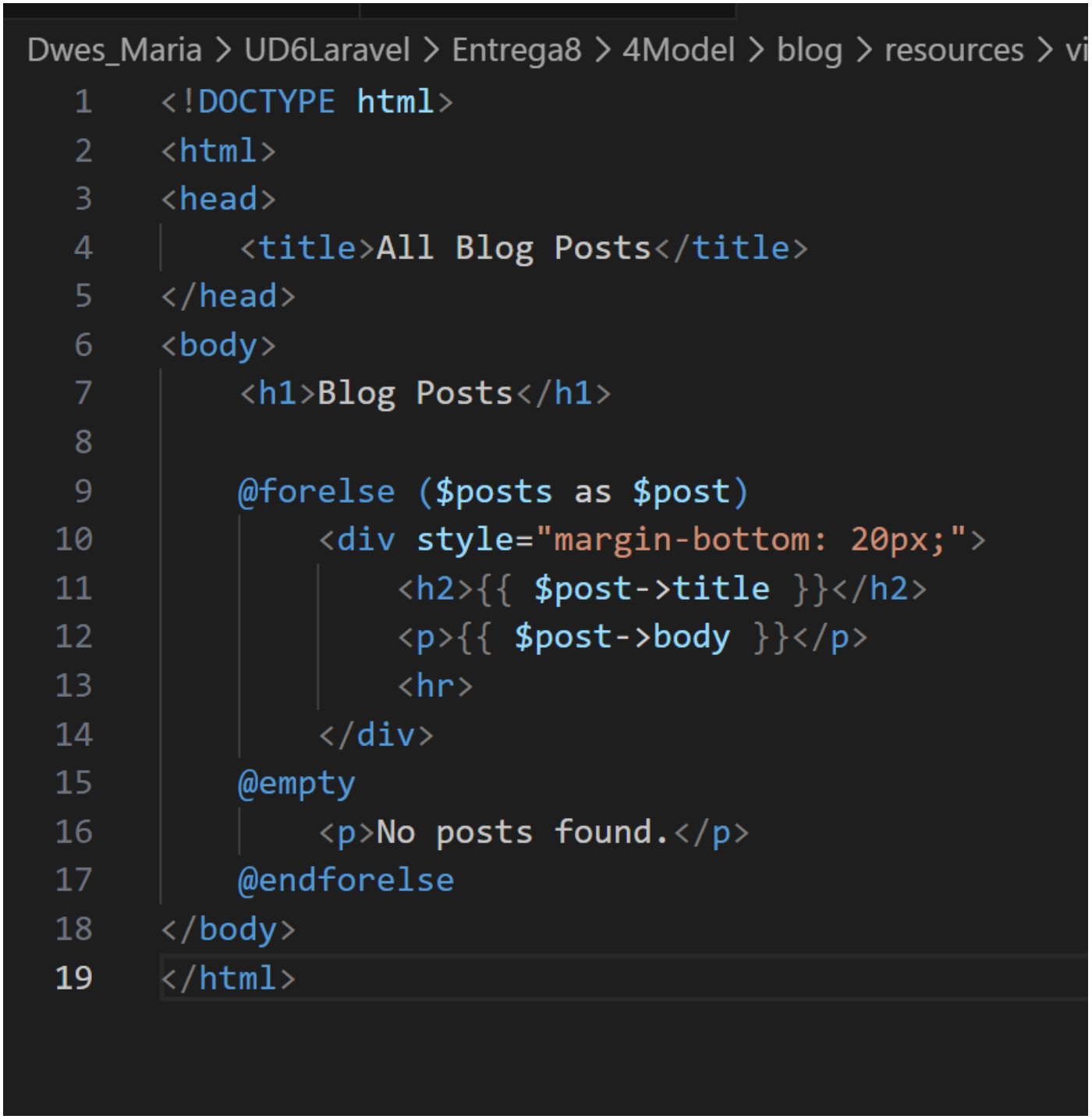
Paso 5: Crear la ruta

Paso 6: creamos la vista de blade

Creamos el archivo de la vista

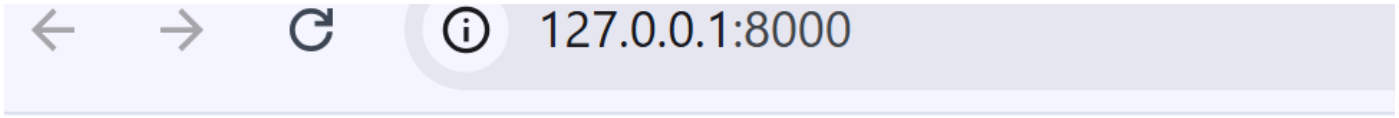
```
php artisan make:view post.index
```

Editamos



Paso 7: servimos la aplicación

```
php artisan serve
```



Blog Posts

Welcome post

This is a welcome post created automatically.

Programa 5: Controladores

En Laravel, los controladores son claseas que agrupan la lógica de manejo de solicitudes relacionada en un solo lugar

Actúan como enlace entre las rutas y la lógica

Proyecto Controllers

Paso 1: Creamos un nuevo proyecto

```
folder: UD6 Laravel / Sesion5

composer create-project laravel/laravel controller-demo

cd controller-demo php artisan serve
```

Paso 2: Creamos un controlador

```
php artisan make:controller UserController
```

Este comando genera un nuevo archivo controlador

Paso 3: Agregamos ruta

Dwes_Maria > UD6Laravel > Entrega8 > 5Controllers > ControllerDemo > routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;

// Import the UserController
use App\Http\Controllers\UserController;

// Define a route for the UserController index method
Route::get(uri: '/users', action: [UserController::class, 'index']);
```

Paso 4: Definir el index método en UserController.php

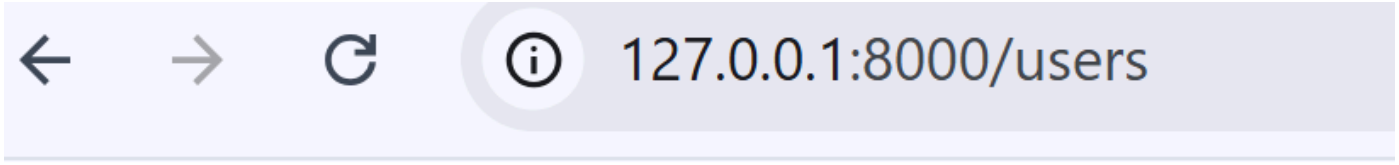
vel > Entrega8 > 5Controllers > ControllerDemo > app > Http > Controllers > UserController.php > PHP

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): View
    {
        $users = [
            ['name' => 'Maria', 'email' => 'maria@example.com'],
            ['name' => 'Juan', 'email' => 'juan@example.com'],
        ];

        return view(view: 'users.index', data: ['users' => $users]);
    }
}

```

Paso 5: Crear la vista

Dwes_Maria > UD6Laravel > Entrega8 > 5Controllers > ControllerDemo > resources > views > users >

```blade
<!DOCTYPE html>
<html>
<head>
    <title>User List</title>
</head>
<body>
    <h1>Users</h1>
    <ul>
        @foreach ($users as $user)
            <li>{{ $user['name'] }} - {{ $user['email'] }}</li>
        @endforeach
    </ul>
</body>
</html>
```



Users

- Maria - maria@example.com
- Juan - juan@example.com

Programa 6: CRUD

Proyecto Clothing store

Paso 1: Nuevo proyecto en Laravel

```
folder Sesion6 composer create-project laravel/laravel clothing_store
OR // laravel new clothing_store cd clothing_store code .
```

Paso 2: Creamos modelo, migración y controlador

```
php artisan make:model ClothingItem -mc
```

Actualizamos la migración

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > database > migrations > 2026_01_12_122212_create_clothing_items_table.ph

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'clothing_items', callback: function (Blueprint $table): void {
            $table->id();
            $table->string(column: 'name');
            $table->string(column: 'size');
            $table->decimal(column: 'price', total: 8, places: 2);
            $table->string(column: 'color');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists(table: 'clothing_items');
    }
};
```

Ejecutamos la migración

```
php artisan migrate
```

Paso 3: rutas y vistas básicas

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ClothingItemController;

//you can define a route for each method instead
Route::resource(name: 'clothing-items', controller: ClothingItemController::class);

```

Podemos mostrar las rutas con

```
php artisan route:list
```

Paso 4: Creamos el controlador

```php
<?php

namespace App\Http\Controllers;

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

2 references | 0 implementations
class ClothingItemController extends Controller
{
    0 references | 0 overrides
    public function index(): View
    {
        // Fetch all clothing items from the database
        // and pass them to the view
        // This will be used to display the list of items
        $items = ClothingItem::all();
        return view(view: 'clothing-items.index', data: compact(var_name: 'items'));
    }

    0 references | 0 overrides
    public function show(ClothingItem $clothingItem): View
    {
        // Fetch a single clothing item by its ID
        // and pass it to the view
        return view(view: 'clothing-items.show', data: compact(var_name: 'clothingItem'));
    }
     0 references | 0 overrides
     public function create(): View
    {
        // Show the form to create a new clothing item
        // This will be used to display the form for adding a new item
        return view(view: 'clothing-items.create');
    }

    0 references | 0 overrides
    public function store(Request $request): RedirectResponse
    {
        // Validate the request data will be needed
        // and create a new clothing item in the database
        // This will be used to save the new item
        ClothingItem::create(attributes: $request->validate(rules: [
            'name' => 'required',
            'size' => 'required',
            'price' => 'required|numeric',
            'color' => 'required',
        ]));

        return redirect()->route(route: 'clothing-items.index');


    }

    0 references | 0 overrides
    public function edit(ClothingItem $clothingItem): View
{
        // Show the form to edit an existing clothing item
        // The $clothingItem is automatically resolved by the route model binding
        return view(view: 'clothing-items.edit', data: compact(var_name: 'clothingItem'));
}

0 references | 0 overrides
public function update(Request $request, ClothingItem $clothingItem): RedirectResponse
{
        // Validate the request data and update the clothing item in the database
        // This will be used to save the changes made to an existing item
        // The $clothingItem is automatically resolved by the route model binding
```

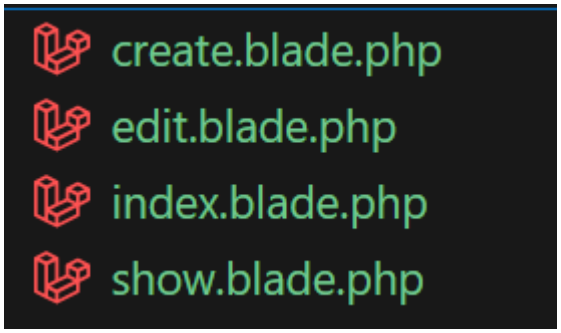
```php
        $clothingItem->update(attributes: $request->validate(rules: [
            'name' => 'required',
            'size' => 'required',
            'price' => 'required|numeric',
            'color' => 'required',
        ]));

        return redirect()->route(route: 'clothing-items.index');
    }
    0 references | 0 overrides
    public function destroy(ClothingItem $clothingItem): RedirectResponse
    {
        $clothingItem->delete();
        return redirect()->route(route: 'clothing-items.index');
    }
}
```

Paso 5: Añadimos en fillable el modelo

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > app > Models > ClothingItem.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

7 references | 0 implementations
class ClothingItem extends Model
{
    0 references
    protected $fillable = ['name', 'size', 'price', 'color'];
}
```

Paso 6: Creamos las vistas de cada controlador



Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > resources > views > clothing-items > create.blade.php

```blade
<form method="POST" action="{{ route(name: 'clothing-items.store') }}">
    @csrf
    <input name="name" placeholder="Name">
    <input name="size" placeholder="Size">
    <input name="price" placeholder="Price">
    <input name="color" placeholder="Color">
    <button type="submit">Save</button>
</form>
```

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > resources > views > clothing-items > edit.blade.php

```blade
<h1>Edit Clothing Item</h1>

<!-- Back to List Button (at the top) -->
<div style="margin-bottom: 15px;">
    <a href="{{ route(name: 'clothing-items.index') }}"
       style="padding: 8px 16px; background-color: #6c757d; color: white; text-decoration: none; border-radius: 4px;">
        Back to List
    </a>
</div>

<form method="POST" action="{{ route(name: 'clothing-items.update', parameters: $clothingItem->id) }}">
    @csrf
    @method('PUT')

    <table border="0" cellpadding="8">
        <tr>
            <td><label for="name">Name:</label></td>
            <td><input type="text" name="name" id="name" value="{{ old(key: 'name', default: $clothingItem->name) }}" required></td>
        </tr>
        <tr>
            <td><label for="size">Size:</label></td>
            <td><input type="text" name="size" id="size" value="{{ old(key: 'size', default: $clothingItem->size) }}" required></td>
        </tr>
        <tr>
            <td><label for="price">Price ($):</label></td>
            <td><input type="text" name="price" id="price" value="{{ old(key: 'price', default: $clothingItem->price) }}" required></td>
        </tr>
        <tr>
            <td><label for="color">Color:</label></td>
            <td><input type="text" name="color" id="color" value="{{ old(key: 'color', default: $clothingItem->color) }}" required></td>
        </tr>
        <tr>
            <td colspan="2" style="text-align: right;">
                <button type="submit" style="padding: 8px 16px; background-color: #4CAF50; color: white; border: none; border-radius: 4px;">
                    ✅ Update
                </button>
                <a href="{{ route(name: 'clothing-items.index') }}"
                   style="padding: 8px 16px; background-color: #f44336; color: white; border: none; border-radius: 4px; text-decoration: none; margin-left: 8px;">
                    ❌ Cancel
                </a>
            </td>
        </tr>
    </table>
</form>
```

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > resources > views > clothing-items > index.blade.php

```blade
<h1>Clothing Items</h1>

<a href="{{ route(name: 'clothing-items.create') }}">➕ Add New Item</a>

<table border="1" cellpadding="8" cellspacing="0">
    <thead>
        <tr>
            <th>Name</th>
            <th>Size</th>
            <th>Price</th>
            <th>Color</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($items as $item)
        <tr>
            <td>{{ $item->name }}</td>
            <td>{{ $item->size }}</td>
            <td>${{ number_format(num: $item->price, decimals: 2) }}</td>
            <td>{{ $item->color }}</td>
            <td>
                <form method="GET" action="{{ route(name: 'clothing-items.show', parameters: $item) }}" style="display:inline;">
                    <button type="submit">👁 View</button>
                </form>

                <form method="GET" action="{{ route(name: 'clothing-items.edit', parameters: $item) }}" style="display:inline;">
                    <button type="submit">✏️ Edit</button>
                </form>

                <form method="POST" action="{{ route(name: 'clothing-items.destroy', parameters: $item) }}" style="display:inline;">
                    @csrf
                    @method('DELETE')
                    <button type="submit" onclick="return confirm('Are you sure you want to delete this item?')">🗑 Delete</button>
                </form>
            </td>
        </tr>
        @endforeach
    </tbody>
</table>
```

Entrega7.md U | Entrega8.md U | show.blade.php U | web.php U

Dwes_Maria > UD6Laravel > Entrega8 > 6CRUD > clothing_store > resources > views > clothing-items > show.blade.php

```blade
<h1>Clothing Item Details</h1>

<p><strong>Name:</strong> {{ $clothingItem->name }}</p>
<p><strong>Size:</strong> {{ $clothingItem->size }}</p>
<p><strong>Price:</strong> ${{ number_format(num: $clothingItem->price, decimals: 2) }}</p>
<p><strong>Color:</strong> {{ $clothingItem->color }}</p>

<a href="{{ route(name: 'clothing-items.edit', parameters: $clothingItem->id) }}">Edit</a>

<form method="POST" action="{{ route(name: 'clothing-items.destroy', parameters: $clothingItem->id) }}" style="display:inline;">
    @csrf
    @method('DELETE')
    <button type="submit" onclick="return confirm('Are you sure you want to delete this item?')">Delete</button>
</form>

<br><br>
<a href="{{ route(name: 'clothing-items.index') }}">Back to List</a>
```

Clothing Items

➕ Add New Item

Name	Size	Price	Color	Actions
camiseta darkrai	M	$340.00	negro	👁 View ✏️ Edit 🗑 Delete

Clothing Item Details

Name: camiseta darkrai

Size: M

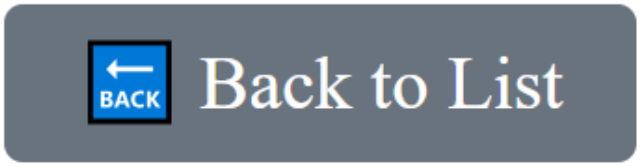
Price: $340.00

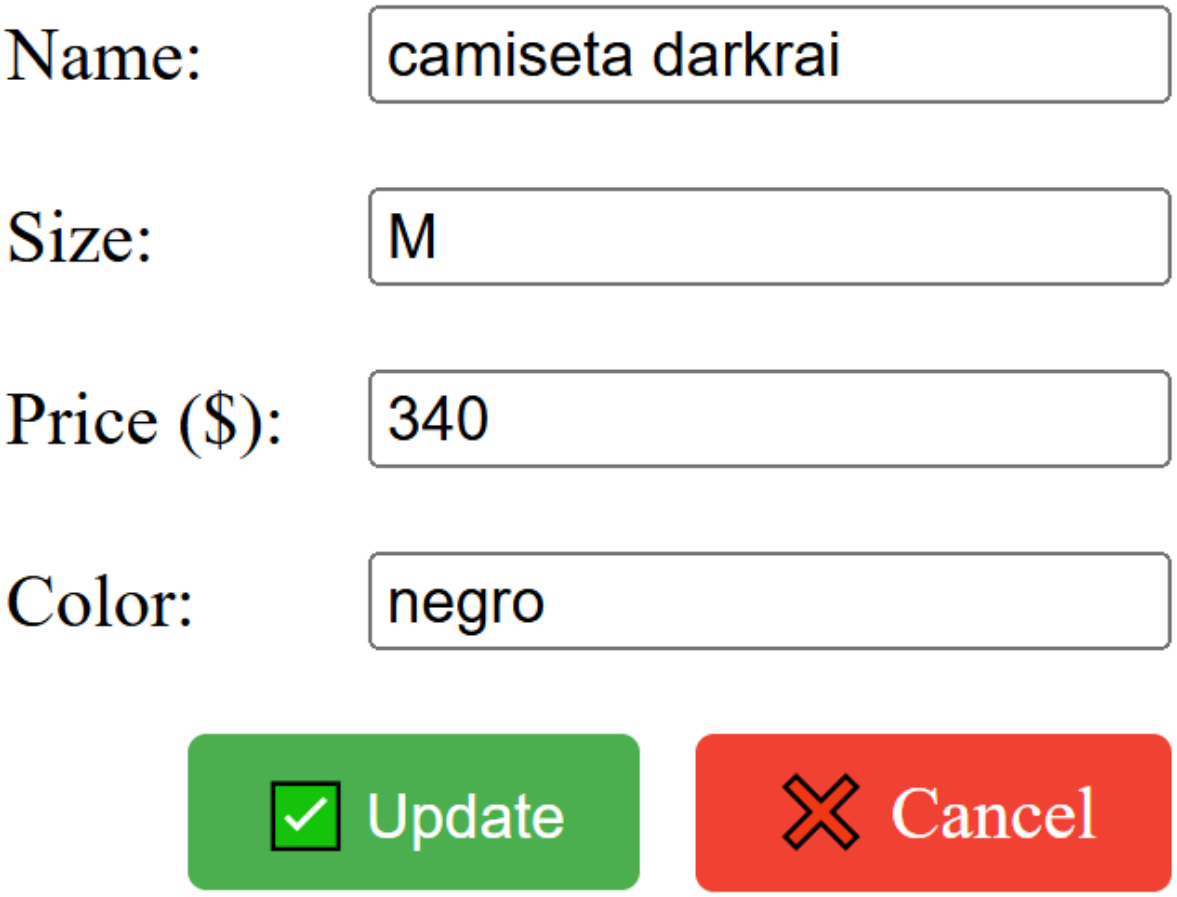
Color: negro

Edit Delete

Back to List

Edit Clothing Item





Actividad

1. **Is the code readable and well-organized?** Si, la estructura sigue el patrón MVC con controladores, modelos y vistas separados correctamente
2. Are variable and method names descriptive? Lamayoría si que lo son aunque yo algunos los cambiaria por otros
3. Are the views easy to understand? si
4. **Are forms protected?** si, se usa @csrf

5. Did you include `@csrf` and `@method('PUT'/'DELETE')` in all appropriate forms?
   Sí, se usan correctamente
6. **Validation:**
7. Is there server-side validation for user inputs in `store()` and `update()` methods?
   si
8. Is there any form of client-side validation (like `required`, `type="number"`)?

   si
9. **Routes:**
10. Are you using `Route::resource()` properly?

    SI
11. Are any unused routes present?

    No se detectan rutas inncesarias
12. **Reusability:**
13. Could you reuse code by creating partials for forms (`_form.blade.php`)?

    Si, los formularios son miu parecidos por lo que se podria extraer a un archivo y reutilizarlo

Yo como elementos de mejora mostraria mensajes como "elemento actualizado" con flash para mejorar la experiencia. Además le añadiría estilos con Bootstrap o CSS para lograr una presentación más profesional y como último usaría la paginación en los controladores para manejar listas largas de registros de forma más eficiente