

- Proyecto cliente-pedido
 - 1º Creamos el proyecto
 - 2º Creamos modelo, migración y controller de Cliente
 - 3º Actualizamos el contenido de la migración y la ejecutamos
 - 4º Creamos el modelo con sus relaciones
 - 5º Hacemos lo mismo con los pedidos
 - 6º Creamos la migración
 - 7º Ejecutamos la migración
 - 8º Completamos el modelo y hacemos la relación
 - 9º Actualizamos las rutas
 - 10º Creamos clientController
 - 11º Creamos orderController
 - 12º Creamos todas las vistas
 - 13º Creamos las factorías de cliente
 - 12º Creamos factorías de order
 - 13º Creamos las seeders de cliente y de order

Proyecto cliente-pedido

Para esta parte de Laravel crearemos un proyecto donde gestionaremos clientes con sus respectivos pedidos como si fuésemos una tienda online de x producto. Para ello utilizaremos todo lo visto en clase que son: Blade, Migration, Model, Controller, CRUD, SeederFactory y las relaciones.

1º Creamos el proyecto

Primero vamos a crear nuestro proyecto en Laravel con el comando `composer create-project laravel/laravel clientes-pedidos`. Esto nos creará un proyecto básico con todo lo esencial para que funcione.

```
PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal> composer create-project laravel/laravel clientes-pedidos
Creating a "laravel/laravel" project at "./clientes-pedidos"
Installing laravel/laravel (v12.11.2)
- Downloading laravel/laravel (v12.11.2)
- Installing laravel/laravel (v12.11.2): Extracting archive
Created project in C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
```

2º Creamos modelo, migración y controller de Cliente

Lo siguiente que haremos es crear el modelo, migración y controlador del cliente, elegimos el cliente ya que sería como el "padre" con respecto a los pedidos. Para ello usaremos el comando `php artisan make:model Client -mc`, este comando nos crea del tirón todo lo anterior

```
PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:model Client -mc

INFO Model [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Models\Client.php] created successfully.
INFO Migration [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\Migrations\2026_01_24_153515_create_clients_table.php] created successfully.
INFO Controller [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Http\Controllers\ClientController.php] created successfully.
```

3º Actualizamos el contenido de la migración y la ejecutamos

Dentro del archivo de la migración del cliente vamos a modificar los campos que queremos que tenga la tabla de los clientes, como curiosidad, el modelo se creará como `Client` mientras que la tabla irá en minúscula y en plural (`clients`). Nuestro cliente tendrá un `id` con el que se identificará, un email único, un teléfono que puede ser nulo, la dirección que puede ser nula también y un booleano activo que por defecto será `true`, es decir activo, además or supuesto se guardará la hora. Por último ejecutamos la migración con el comando `php artisan migrate` para crear la tabla en la base de datos

```
11  */
12  public function up(): void
13  {
14  Schema::create(table: 'clients', callback: function (Blueprint $table): void {
15  $table->id();
16  $table->string(column: 'name');
17  $table->string(column: 'email')->unique();
18  $table->string(column: 'phone')->nullable();
19  $table->string(column: 'address')->nullable();
20  $table->boolean(column: 'active')->default(value: true);
21  $table->timestamps();
22  });
23  }
24
25  /**
26   * Reverse the migrations.
27   */
28  public function down(): void
29  {
30      Schema::dropIfExists(table: 'clients');
31  }
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS GITLENS powershell - clientes-pedidos + - [] [] ... | [] >

```
6 C:\xampp\htdocs\DWES> cd .\Dwes_Maria\
6 C:\xampp\htdocs\DWES\Dwes_Maria> cd .\UD6Laravel\
6 C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel> cd .\Entrega8\
6 C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8> cd .\ProyectoFinal\
6 C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal> cd .\clientes-pedidos\
6 C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan migrate
```

INFO Running migrations.

```
2026_01_24_153515_create_clients_table ..... 41.79ms DONE
```

4º Creamos el modelo con sus relaciones

Pasamos ahora a crear el modelo, esto nos servirá como bien dice su nombre de modelo para los objetos que vayan a guardarse en la base de datos, todos sus atributos serán protected y fillable lo que quiere decir que podrán ser modificados en la base de datos, el id no se podrá modificar por lo que sería guarded, pero laravel ya lo intuye ya que no está dentro del fillable. También tenemos aquí la relación, en este caso será hasMany con los pedidos. Como punto clave destacar el uso del use HasFactory, lo cual es esencial para que la factoria sepa como tiene que crear los objetos

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  17 references | 0 implementations
9  class Client extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $fillable = [
15         'name',
16         'email',
17         'phone',
18         'address',
19         'active'
20     ];
21
22     0 references | 0 overrides
23     public function orders(): HasMany
24     {
25         return $this->hasMany(related: Order::class);
26     }
27 }

```

5º Hacemos los mismo con los pedidos

Usando el mismo comando que en el cliente vamos a hacer los pedidos para crear así su modelo su migración y su controlador

```

PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:model Order -mc

[INFO] Model [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Models\Order.php] created successfully.

[INFO] Migration [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\Migrations\2026_01_25_113116_create_orders_table.php] created successfully.

[INFO] Controller [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Http\Controllers\OrderController.php] created successfully.

```

6º Creamos la migración

Actualizamos el archivo de migración de los pedidos, como punto más importante aquí destacar el uso del enum, y el atributo `client_id`, que es el id que pasa de la tabla cliente, es decir de la 1 a la N. También decir que tiene un `on delete cascade`, en caso de que el id del cliente se borre, todos los pedidos asociados a este id serán eliminados

```
es_maria / UD6Laravel / Entrega8 / ProyectoFinal / clientes-pedidos / database / migrations / 2026_01_25_113116_create_orders_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration {
8      /**
9       * Run the migrations.
10      */
11      public function up(): void
12      {
13          Schema::create(table: 'orders', callback: function (Blueprint $table): void {
14              $table->id();
15              $table->string(column: 'order_number')->unique();
16              $table->date(column: 'date');
17              $table->enum(column: 'status', allowed: ['pendiente', 'enviado', 'entregado', 'cancelado']);
18              $table->decimal(column: 'total', total: 8, places: 2);
19              $table->text(column: 'notes')->nullable();
20
21              $table->foreignId(column: 'client_id')
22                  ->constrained()
23                  ->onDelete(action: 'cascade');
24
25              $table->timestamps();
26          });
27      }
28
29      /**
30       * Reverse the migrations.
31       */
32      public function down(): void
33      {
34          Schema::dropIfExists(table: 'orders');
35      }
36  };
37
```


7º Ejecutamos la migración

Ejecutamos la migración para crear la tabla de pedidos

```
PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS  GITLENS  powershell - clientes-pedidos + v [ ] [ ] ... [ ] [ ]
INFO Model [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Models\Order.php] c
INFO Controller [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\app\Http\Controllers\OrderController.php] created successfully.
PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan migrate
INFO Running migrations.
2026_01_25_113116 create orders table ..... 40.52ms DONE
```

8º Completamos el modelo y hacemos la relación

Vamos ahora a rellenar el modelo al igual que el cliente, el pedido hace uso del HasFactory, del fillable y en este caso la relación es belongsTo

```
Dwes_Maria / UD6Laravel / Entregas / ProyectoFinal / clientes-pedidos / app / Models / 
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  10 references | 0 implementations
9  class Order extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $fillable = [
15         'order_number',
16         'date',
17         'status',
18         'total',
19         'notes',
20         'client_id'
21     ];
22
23     0 references | 0 overrides
24     public function client(): BelongsTo
25     {
26         return $this->belongsTo(related: Client::class);
27     }
28 }
```

9º Actualizamos las rutas

Una vez tenemos ya los modelos y las migraciones necesarias vamos a crear las rutas a través de las cuales navegaremos entre los diferentes CRUD, tenemos el welcome

tipico de laravel, el clients y el orders

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ClientController;
use App\Http\Controllers\OrderController;

Route::get(uri: '/', action: function (): View {
    return view(view: 'welcome');
});
Route::resource(name: 'clients', controller: ClientController::class);
Route::resource(name: 'orders', controller: OrderController::class);
```

10 ° Creamos clientController

Vamos a pasar ahora con la creación del controller del cliente que contará con con las funciones de index, create, store, show,edit, update y destroy

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Models\Client;
6 use Illuminate\Http\Request;
7
```

2 references | 0 implementations

```
8 class ClientController extends Controller
9 {
```

0 references | 0 overrides

```
10 public function index(): View
11 {
12     $clients = Client::all();
13     return view(view: 'clients.index', data: compact(var_name: 'clients'));
14 }
15
```

0 references | 0 overrides

```
16 public function create(): View
17 {
18     return view(view: 'clients.create');
19 }
20
```

0 references | 0 overrides

```
21 public function store(Request $request): RedirectResponse
22 {
23     $request->validate(rules: [
24         'name' => 'required|string|max:255',
25         'email' => 'required|email|unique:clients,email',
26         'phone' => 'nullable|string|max:20',
27         'address' => 'nullable|string|max:255',
28         'active' => 'boolean'
29     ]);
30
31     Client::create(attributes: $request->all());
32
33     return redirect()->route(route: 'clients.index')
34         ->with(key: 'success', value: 'Cliente creado correctamente');
35 }
36
```

0 references | 0 overrides

```
37 public function show($id): View
38 {
39     $client = Client::findOrFail(id: $id);
40     return view(view: 'clients.show', data: compact(var_name: 'client'));
41 }
42
```

0 references | 0 overrides

```
43 public function edit($id): View
44 {
45     $client = Client::findOrFail(id: $id);
46     return view(view: 'clients.edit', data: compact(var_name: 'client'));
47 }
48
```

0 references | 0 overrides

```
49 public function update(Request $request, $id): RedirectResponse
50 {
51     $client = Client::findOrFail(id: $id);
52
53     $request->validate(rules: [
54         'name' => 'required|string|max:255',
55         'email' => 'required|email|unique:clients,email,' . $client->id,
56         'phone' => 'nullable|string|max:20',
57         'address' => 'nullable|string|max:255',
58         'active' => 'boolean'
59     ]);
60
61     $client->update(attributes: $request->all());
62
63     return redirect()->route(route: 'clients.index')
64         ->with(key: 'success', value: 'Cliente actualizado correctamente');
65 }
66
```

0 references | 0 overrides

```
67 public function destroy($id): RedirectResponse
68 {
69     $client = Client::findOrFail(id: $id);
70     $client->delete();
71
72     return redirect()->route(route: 'clients.index')
73 }
```



```
73         ->with(key: 'success', value: 'Cliente eliminado correctamente');  
74     }  
75 }  
76
```

11º Creamos orderController

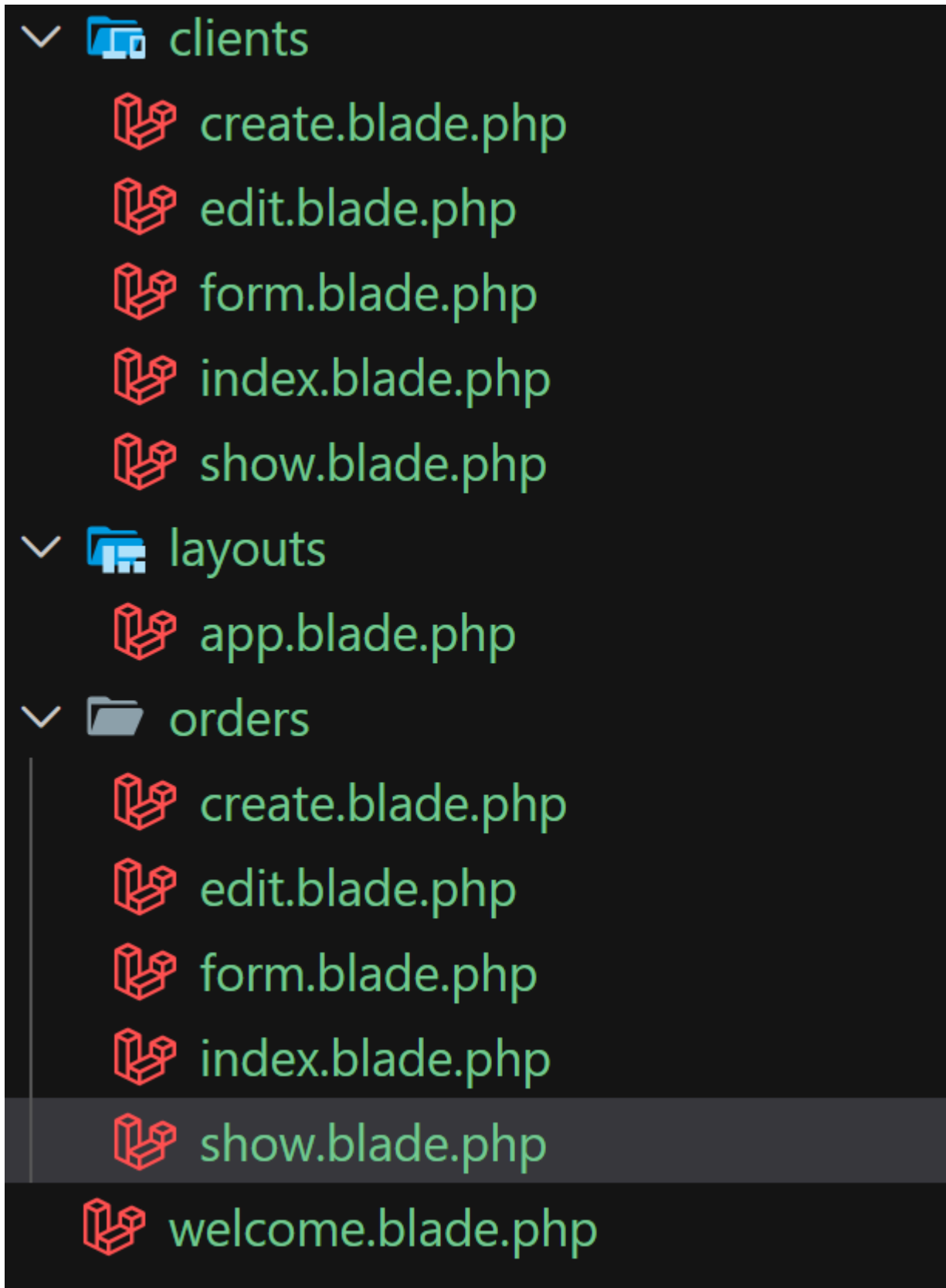
Al igual que acabamos de hacer en ClienteController hacemos en order controller con todas las funciones típicas del CRUD

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Order;
6  use App\Models\Client;
7  use Illuminate\Http\Request;
8
9  2 references | 0 implementations
10 class OrderController extends Controller
11 {
12     0 references | 0 overrides
13     public function index(): View
14     {
15         $orders = Order::with(relations: 'client')->get();
16         return view(view: 'orders.index', data: compact(var_name: 'orders'));
17     }
18
19     0 references | 0 overrides
20     public function create(): View
21     {
22         $clients = Client::all();
23         return view(view: 'orders.create', data: compact(var_name: 'clients'));
24     }
25
26     0 references | 0 overrides
27     public function store(Request $request): RedirectResponse
28     {
29         $request->validate(rules: [
30             'order_number' => 'required|unique:orders,order_number',
31             'date' => 'required|date',
32             'status' => 'required|in:pendiente,enviado,entregado,cancelado',
33             'total' => 'required|numeric|min:0',
34             'notes' => 'nullable|string',
35             'client_id' => 'required|exists:clients,id'
36         ]);
37
38         Order::create(attributes: $request->all());
39
40         return redirect()->route(route: 'orders.index')
41             ->with(key: 'success', value: 'Pedido creado correctamente');
42     }
43
44     0 references | 0 overrides
45     public function show($id): View
46     {
47         $order = Order::with(relations: 'client')->findOrFail(id: $id);
48         return view(view: 'orders.show', data: compact(var_name: 'order'));
49     }
50
51     0 references | 0 overrides
52     public function edit($id): View
53     {
54         $order = Order::findOrFail(id: $id);
55         $clients = Client::all();
56
57         return view(view: 'orders.edit', data: compact(var_name: 'order', var_names: 'clients'));
58     }
59
60     0 references | 0 overrides
61     public function update(Request $request, $id): RedirectResponse
62     {
63         $order = Order::findOrFail(id: $id);
64
65         $request->validate(rules: [
66             'order_number' => 'required|unique:orders,order_number,' . $order->id,
67             'date' => 'required|date',
68             'status' => 'required|in:pendiente,enviado,entregado,cancelado',
69             'total' => 'required|numeric|min:0',
70             'notes' => 'nullable|string',
71             'client_id' => 'required|exists:clients,id'
72         ]);
73
74         $order->update(attributes: $request->all());
75
76         return redirect()->route(route: 'orders.index')
77             ->with(key: 'success', value: 'Pedido actualizado correctamente');
78     }
79
80     0 references | 0 overrides
81     public function destroy($id): RedirectResponse
```

```
74 {  
75     $order = Order::findOrFail(id: $id);  
76     $order->delete();  
77   
78     return redirect()->route(route: 'orders.index')  
79         ->with(key: 'success', value: 'Pedido eliminado correctamente');  
80 }  
81 }  
82
```

12º Creamos todas las vistas

A continuación una vez creados los controllers deberemos de crear las vistas por las que el usuario irá navegando, habrá una vista principal app.blade y luego tanto clients como orders tendrán las vistas de create, edit, form, index, y show



13° Creamos las factorias de cliente

Pasamos ahora a crear las factorias. Las factorias se usan para poder probar los programas con datos falsos pero en grandes cantidades por ejemplo, en este caso, vamos a crear a 10 clientes donde cada uno de ellos tendrá por defecto 3 pedidos

```
Directorio: C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\resources\views
PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:factory ClientFactory
INFO Factory [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\factories\ClientFactory.php] created successfully.
```

```
Dwes_Maria > UD6Laravel > Entrega8 > ProyectoFinal > clientes-pedidos > database > factories > ClientFactory.php > ...
1  <?php
2
3  namespace Database\Factories;
4
5  use Illuminate\Database\Eloquent\Factories\Factory;
6
7  0 references | 0 implementations
8  class ClientFactory extends Factory
9  {
10     0 references | 0 overrides
11     public function definition(): array
12     {
13         return [
14             'name' => $this->faker->name(),
15             'email' => $this->faker->unique()->safeEmail(),
16             'phone' => $this->faker->phoneNumber(),
17             'address' => $this->faker->address(),
18             'active' => $this->faker->boolean(chanceOfGettingTrue: 90),
19         ];
20     }
21 }
```

12º Creamos factorias de order

Usamos el random para que la factoria elija aleatoriamente el valor que ella decida entre las opciones que les damos

```
PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:factory OrderFactory
INFO Factory [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\factories\OrderFactory.php] created successfully.
```

```

Dwes_Maria > UD6Laravel > Entrega8 > ProyectoFinal > clientes-pedidos > database > factories > OrderFactory.php > ...
1 namespace Database\Factories;
2
3
4
5 use Illuminate\Database\Eloquent\Factories\Factory;
6
7 0 references | 0 implementations
8 class OrderFactory extends Factory
9 {
10     0 references | 0 overrides
11     public function definition(): array
12     {
13         return [
14             'order_number' => 'ORD-' . $this->faker->unique()->numberBetween(int1: 1000, int2: 9999),
15             'date' => $this->faker->date(),
16             'status' => $this->faker->randomElement(array: [
17                 'pendiente',
18                 'enviado',
19                 'entregado',
20                 'cancelado'
21             ]),
22             'total' => $this->faker->randomFloat(nbMaxDecimals: 2, min: 10, max: 500),
23             'notes' => $this->faker->optional()->sentence(),
24         ];
25     }
26 }

```

13° Creamos las seeders de cliente y de order

Pasamos ahora a crear las seeders que son como los datos a fuego que va a llevar nuestra tabla en la base de datos tanto clientes como pedidos en este caso uno de cada

```

PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:seeder ClientSeeder
INFO Seeder [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\seeders\ClientSeeder.php] created successfully.

PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos> php artisan make:seeder OrderSeeder
INFO Seeder [C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos\database\seeders\OrderSeeder.php] created successfully.

PS C:\xampp\htdocs\DWES\Dwes_Maria\UD6Laravel\Entrega8\ProyectoFinal\clientes-pedidos>

```

```
README.md ...\clientes-pedidos 0 ClientSeeder.php 0 OrderSeeder.php 0 database.sq
Dwes_Maria > UD6Laravel > Entrega8 > ProyectoFinal > clientes-pedidos > database > seeders > ClientSeeder.ph
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use App\Models\Client;
7
8  1 reference | 0 implementations
9  class ClientSeeder extends Seeder
10 {
11     0 references | 0 overrides
12     public function run(): void
13     {
14         Client::create(attributes: [
15             'name' => 'Juan Pérez',
16             'email' => 'juan.perez@example.com',
17             'phone' => '123456789',
18             'address' => 'Calle Falsa 123, Ciudad',
19             'active' => true
20         ]);
21     }
22 }
```

```
4
5  use Illuminate\Database\Seeder;
6  use App\Models\Order;
7  use App\Models\Client;
8
9  1 reference | 0 implementations
10 class OrderSeeder extends Seeder
11 {
12     0 references | 0 overrides
13     public function run(): void
14     {
15         $client = Client::first();
16
17         Order::create(attributes: [
18             'order_number' => 'ORD-1001',
19             'date' => now(),
20             'status' => 'pendiente',
21             'total' => 150.75,
22             'notes' => 'First order notes',
23             'client_id' => $client->id
24         ]);
25     }
26 }
```



```
1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\Client;
6  use Illuminate\Database\Seeder;
7
8  0 references | 0 implementations
9  class DatabaseSeeder extends Seeder
10 {
11     0 references | 0 overrides
12     public function run(): void
13     {
14         $this->call(class: [
15             ClientSeeder::class,
16             OrderSeeder::class,
17         ]);
18
19         Client::factory()
20             ->count(10)
21             ->hasOrders(3)
22             ->create();
23     }
24 }
```