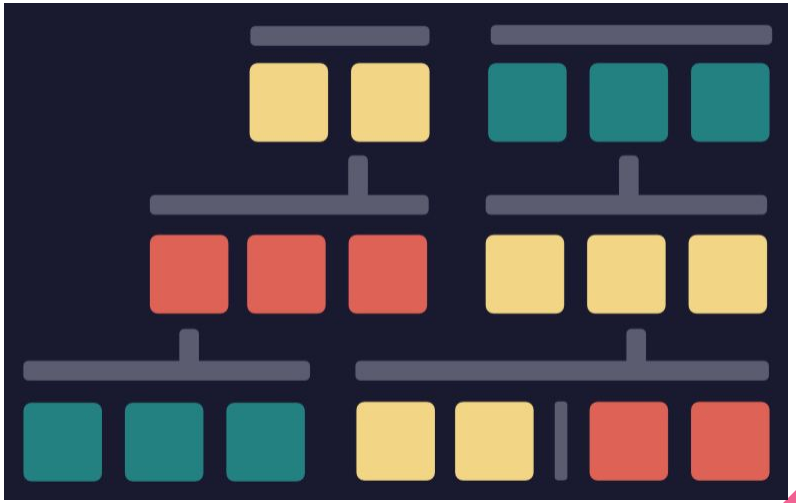
# Collections

Estructura de dato Collections usando con ORM Eloquent

# Clase Collection base de Laravel

La clase **Collection** de Laravel, definida en ***Illuminate\Support\Collection***, proporciona una metodología fluida y conveniente para **trabajar fácilmente con arrays de datos**.

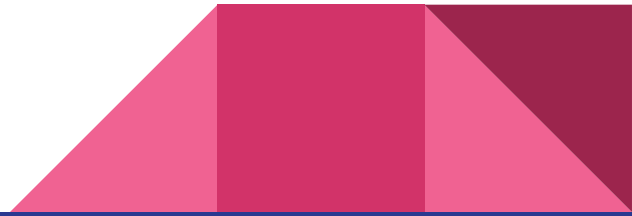


# Crear una colección - collect()

Utilizando el helper ***collect()*** se genera una nueva instancia de colección a partir de un array.

Por ejemplo:

*$coleccion = **collect([1, 2, 3])**;*



# Métodos para procesar colecciones de datos

Esta clase “Collection” ofrece una a**mplia variedad de métodos** para manipular datos del array contenido.

En la documentación oficial se puede ver la lista completa de métodos existentes: https://laravel.com/docs/11.x/collections#available-methods

Todos estos métodos **pueden encadenarse** para manipular de manera fluida el array de datos.

Además, casi todos los métodos **devuelven una nueva instancia** de “Collection”, lo que **permite conservar la versión original** de la colección.

# Métodos para procesar colecciones - Ejemplos

- **Selección**: all(), get(), search(), before(), after(), select(), unique(), random()...
- **Comprobación**: isEmpty(), isNotEmpty(), has(), hasAny()...
- **Transformación**: map(), flatten(), dot(), concat(), merge(), union(), filter()...
  - **Añadir**: put(), push(), prepend()...
  - **Eliminar**: forget(), pull(), reject()...
- **Cálculos/Agregación**: count(), sum(), average(), max(), min()...
- **Ordenamiento**: sort(), sortBy(), shuffle()...
- **Iteración**: each(), every(), first()...

La categorización no es exacta; muchos métodos pueden realizar múltiples procesos.

Por ejemplo; *pull()* selecciona y elimina un dato.

# Dump collection - dd()

Un caso particular: el método ***dd()*** hace un **dump** de la colección y **detiene la ejecución** del script. Empleado para debuggear.

# Encadenamiento de métodos

Como se ha indicado previamente, los métodos de las colecciones permiten el **encadenamiento**, facilitando operaciones complejas de manera legible.

Por ejemplo:

```
$collection = collect([5, 3, 8, 1]);

// sort() ordena los valores del array (pero mantiene las claves)
// values() restablece la numeració de las claves del array
$result = $collection->sort()->values();

print_r($result->all());        // Imprime la colección completa
```

Si solo se realiza un sort() el resultado sería:
Array ( [3] => 1 [1] => 3 [0] => 5 [2] => 8 )

Pero, al encadenar value() tras sort(), el resultado es:

Array ( [0] => 1 [1] => 3 [2] => 5 [3] => 8 )

# Extracción de datos de Collection - get() - only()

Ejemplo para **un único dato**:

*$collection = collect(['nombre' => 'Carlos', 'edad' => 30]);*

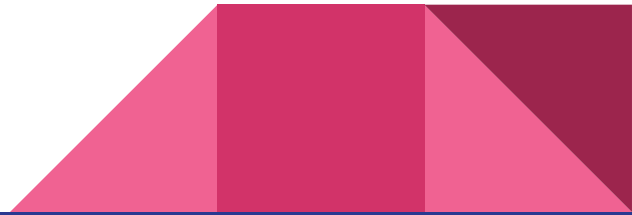
*$nombre = $collection**->get('nombre')**;*

*echo $nombre; // imprime Carlos*

Ejemplo para **múltiples datos**:

*$datos = $collection**->only(['nombre', 'edad'])**;*

*print_r($datos->all());  // ['nombre' => 'Carlos', 'edad' => '30']*



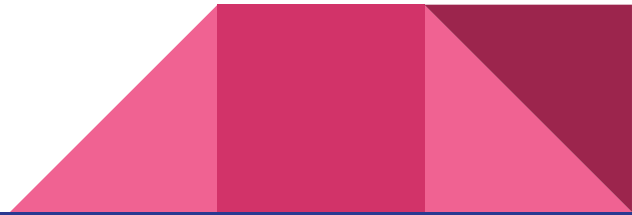
# Inserción de dato en Collection - put()

Ejemplo de **inserción** de dato en una colección de datos existente:

```
$collection = collect(['nombre' => 'Carlos']);

$collection->put('edad', 30);

print_r($collection->all());  // ['nombre' => 'Carlos', 'edad' => 30]
```



# Actualización de un dato en Collection - put()

Mediante el método put() podemos **actualizar valores** existentes en una colección. No es el único método que permite modificar la información.

*$collection = collect(['nombre' => 'Carlos', **'edad' => 30**]);*

*$collection**->put('edad', 31)**; // Modifica la edad*

*print_r($collection->all()); // ['nombre' => 'Carlos', **'edad' => 31**]*

# Convertir collection a array o json - toArray() y toJson()

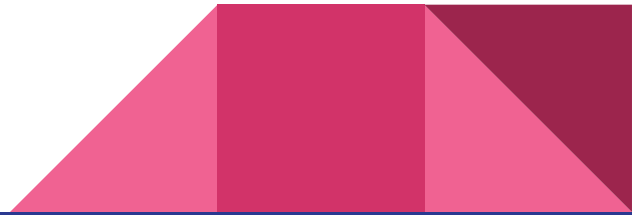
***toArray()*** convierte la colección a un array normal de PHP.

*$collection = collect(['name' => 'Desk', 'price' => 200]);*

*$collection**->toArray();** // Devuelve: [ ['name' => 'Desk', 'price' => 200] ]*

Por otro lado, ***toJson()*** convierte la colección al formato JSON.

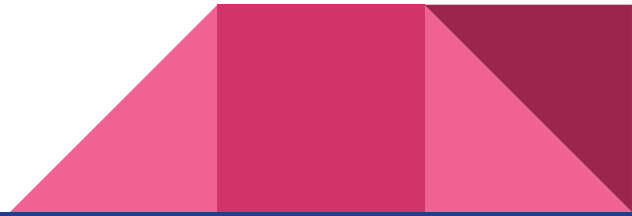
*$collection**->toJson();** // Devuelve: '{"name":"Desk", "price":200}'*



# Clase Collection de Eloquent

Todos los métodos de Eloquent que devuelven más de un resultado del modelo serán instancias de la clase “***Illuminate\Database\Eloquent\Collection***”, incluyendo los resultados obtenidos mediante el método “get()” o accedidos a través de una relación.

El objeto de colección de Eloquent extiende la colección base de Laravel.



# Métodos específicos para Collections del Modelo

La clase Collection de Eloquent está enlazada con datos del modelo (base de datos), por lo que ofrece métodos adicionales para procesar la información.

De estos métodos a encadenar sobre una colección de datos existentes, se pueden destacar:

- **contains($key, $value = null)** → Verifica si la colección contiene un elemento con una clave o valor específico.
- **diff($items)** → Devuelve los elementos de la colección que no están en otra colección dada.
- **find($key)** → Busca un modelo por su clave primaria.
- **findOrFail($key)** → Busca un modelo por su clave primaria y lanza una excepción si no lo encuentra.
- **load($relations)** → Carga relaciones adicionales en los modelos de la colección.
- **only($keys)** → Devuelve solo los elementos con las claves especificadas.
- **unique($key = null)** → Filtra los valores duplicados en la colección.
- **toQuery()** → Convierte la colección en una consulta de base de datos (Query).