



# LARAVEL: CAPA DE SERVICIOS

DISEÑO DE SISTEMAS SOFTWARE

# Contenido

1. Transacciones
2. Capa de servicios

Laravel: Capa de servicios

# TRANSACCIONES

# Transacciones

- Cuando realicemos una operación que modifique el contenido de varias tablas tenemos que meterla dentro de una transacción.
- De esta forma si la operación falla la base de datos no se quedará en un estado inconsistente.
- Por ejemplo, al guardar una nueva factura tenemos que almacenar los datos de la cabecera de la factura y cada una de las líneas de la factura.

<https://laravel.com/docs/5.5/database#database-transactions>

# Transacciones

- Para realizar una transacción en Laravel utilizamos el método “transaction” de la clase DB:

```
DB::transaction(function () {  
    DB::table('users')->update(['votes' => 1]);  
    DB::table('posts')->delete();  
});
```

- En caso de que falle cualquiera de las operaciones que contiene se capturará la excepción y se desharán todas las operaciones que se hubieran realizado hasta el momento.
- Este método también funciona con Eloquent ORM.

# Transacciones manuales

- También podemos realizar la transacción manualmente mediante los métodos:

```
DB::beginTransaction();  
DB::commit();  
DB::rollback();
```

- No debemos olvidar realizar el “commit” al finalizar la transacción y de llamar a “rollback” para deshacerla.
- Este tipo de transacciones nos pueden ayudar cuando queremos tener un mayor control sobre cuándo se realiza el “rollback” o qué hacer en este caso.

Laravel: Capa de servicios

## **CAPA DE SERVICIOS**

# Capa de servicios

- Cuando realicemos operaciones que trabajen con varias tablas de la base de datos (operaciones de alto nivel) tendremos que separarlas en una capa adicional de servicios.
- Para esto podemos crear una carpeta dentro de “app” y definir una nueva clase con métodos estáticos con las operaciones que queramos realizar.
- En el siguiente repositorio tenéis un proyecto de ejemplo:  
<https://github.com/cperezs/dss-business-logic-patterns/blob/master/app/ServiceLayer/ActiveRecordServices.php>



# Capa de servicios

- Podemos crear la carpeta “app/ServiceLayer” para guardar todas las clases PHP con nuestros servicios.
- Y dentro de ella tener las clases PHP con los servicios como métodos estáticos de la clase.
- Por ejemplo, en el fichero “OrderServices.php” tendríamos:

```
<?php
namespace App\ServiceLayer;
use Illuminate\Support\Facades\DB;

class OrderServices {
    public static function processOrder($products) {
        // ...
    }
}
```

# Capa de servicios

```
public static function processOrder($products) {  
    $rollback = false;  
    DB::beginTransaction();  
    $order = new Order();  
    $order->insert();  
    foreach ($products as $productId => $quantity) {  
        $product = Product::find($productId);  
        if ($product->getStock() >= $quantity) {  
            $product->setStock($product->getStock() - $quantity);  
            $product->update();  
  
            $line = new OrderLine();  
            $line->setQuantity($quantity);  
            $line->setOrderId($order->getId());  
            $line->setProductId($productId);  
            $line->insert();  
        } else {  
            $rollback = true;  
            break;  
        }  
    }  
    if ($rollback) {  
        DB::rollBack();  
        return null;  
    }  
    DB::commit();  
    return $order;  
}
```