



Instituto
de Ciencias
Tecnológicas

De la Universidad
San Sebastián

Unidad 2: Actividad de la Unidad

Asignatura: Framework y Programación Web

Docente: Víctor Cofré Farías

Alumnos: Mauricio Alejandro Oxman Oliva

Eliú Martínez Pincheira

Sección y Grupo: Sección 50

Fecha de Entrega: 20/11/2023

Caso a Resolver

Instrucciones:

Se continúa con el proyecto de la actividad calificada 1.

Software Gestión de Stock.

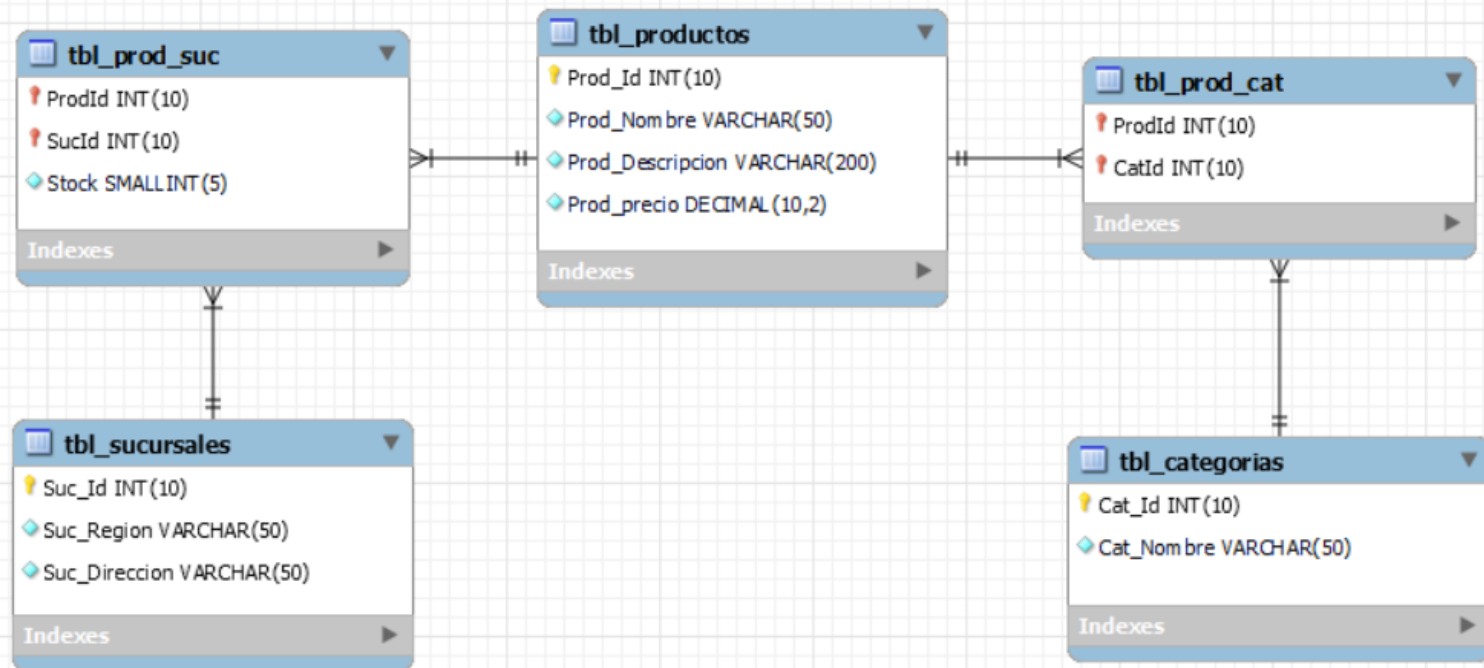
- Registrar un producto con los siguientes datos, id auto incrementable, código único del producto, nombre del producto, categoría, sucursal en la que se encuentra (Suponga 3), y descripción, cantidad, y precio venta.
- Consultar productos por código, nombre, y opcionalmente la sucursal.
- Dar de baja un producto. (Con la opción de eliminar el producto).
- Actualizar nombre, precio y descripción del producto.
- Crear un login para administrador del sistema.

Se solicita:

-

De acuerdo a los requerimientos del software, diseñe e implemente la base de datos que dará soporte al software.

I. Modelo de Base de Datos de Proyecto



Explicación breve de modelo de base de datos

Como podemos ver, el modelo creado consta de 5 tablas, la cual, incluye las 3 tablas principales para el proyecto: **sucursales, productos y categorías**.

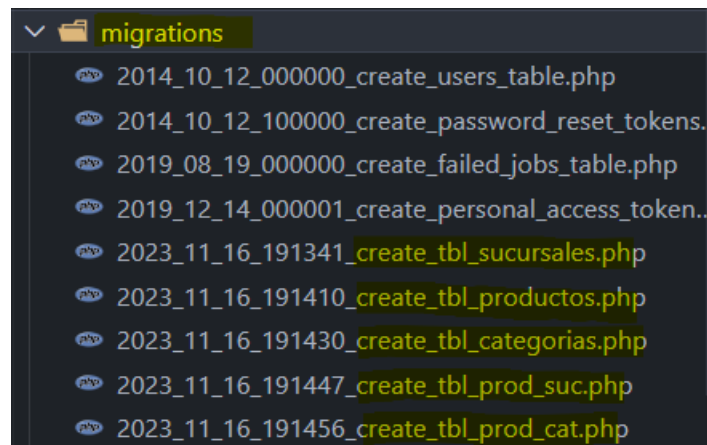
Ahora veremos sus relaciones y explicación del modelo:

En **una sucursal** puede haber **muchos productos** y **un producto** puede estar en **muchas sucursales**, por lo tanto, tenemos una relación de muchos a muchos, por lo tanto, se crea la tabla: **tbl_prod_suc**, en esta agregamos el campo **stock**, la que nos permitirá saber los stocks de un producto en una determina sucursal.

También tenemos la relación de productos y categorías, muy similar a la anterior. **Un producto** puede tener **más de una categoría** y **un tipo de categoría** puede estar en **muchos productos**. Ejemplo: el producto reloj de pared, puede pertenecer a la categoría de hogar y a la categoría de adorno, y viceversa, la categoría electrónica la pueden tener los productos tales como: celular, televisión, audífonos, etc.

Ahora que conocemos el modelo, nos adentramos a la continuación del proyecto, esto incluirá migraciones, seeders y modelos.

II. Migraciones en el Proyecto



Ahora veremos como creamos cada una de las migraciones necesarias, para el modelo.

```
MINGW64:/d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyecto/Proje...
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ code .

Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ php artisan make:migration nombreMigracion --create=nombreTabla
```

Abrimos la terminal que vamos a usar, el IDE (vs code en este caso) y nos debemos posicionar en el proyecto con marco de trabajo Laravel.

Luego ejecutar el comando mostrado en pantalla, cambiándole **nombreMigracion**, por el nombre de nuestra migración y **--create=nombreTabla**. Por el nombre de la tabla a la que se relaciona en nuestro modelo.

De esta manera creamos todas nuestras migraciones que necesitamos.

```

public function up(): void
{
    Schema::create('tbl_sucursales', function (Blueprint $table) {
        $table->increments('Suc_Id');
        $table->string('Suc_Region', 50);
        $table->string('Suc_Direccion', 50);

        $table->timestamps();
    });
}

```

Como primera instancia, veremos nuestra migración que hace referencia a la **tbl_sucursales**. Nos enfocaremos en el método **up()**, ya que, este nos permitirá crear las columnas y datos de nuestra tabla.

Nuestra tabla sucursal consta de 3 columnas, y la timestamps por defecto. **Increments()**, lo que hará es: **crear un id auto-incrementable de dato unsignedInteger**. Luego tenemos los campos strings y con su longitud como segundo parámetro.

```

public function up(): void
{
    Schema::create('tbl_productos', function (Blueprint $table) {
        $table->increments('Prod_Id');
        $table->string('Prod_Nombre', 50);
        $table->string('Prod_Descripcion', 200);
        $table->decimal('Prod_Precio', $precision = 10, $scale = 2);

        $table->timestamps();
    });
}

```

Tenemos la migración de la **tbl_productos**, tenemos 4 columnas y la timestamps.

Una de ellas sería nuestro id '**increments()**', 2 columnas strings con sus longitudes, y la columna decimal, que incluye 2 parámetros, el primero es el total de dígitos numéricos y el segundo es el total de decimales que tendrá.

```

public function up(): void
{
    Schema::create('tbl_categorias', function (Blueprint $table) {
        $table->increments('Cat_Id');
        $table->string('Cat_Nombre', 50);

        $table->timestamps();
    });
}

```

Tenemos la migración de la **tbl_categorias**, tenemos 2 columnas y la timestamps. Tenemos el id **'increments()'** y una columna string con su longitud.

```

public function up(): void
{
    Schema::create('tbl_prod_suc', function (Blueprint $table) {
        $table->unsignedInteger('ProdId');
        $table->unsignedInteger('SucId');
        $table->unsignedSmallInteger('Stock');
        $table->primary(['ProdId', 'SucId']);

        // Relaciones
        $table->foreign('ProdId')->references('Prod_Id')->on('tbl_productos');
        $table->foreign('SucId')->references('Suc_Id')->on('tbl_sucursales');

        $table->timestamps();
    });
}

```

Tenemos la migración de la **tbl_prod_suc**, incluye 3 columnas y la timestamps. Esta tabla refleja una relación de muchos a muchos, por lo que, contendrá 2 id, que hacen referencia a las tablas involucradas. Primero se debe crear los campos que serán las llaves foráneas, estas deben ser del mismo tipo de dato que las llaves primarias **'unsignedInteger'**, también un campo **'unsignedSmallInteger'** para el Stock de los productos en una sucursal determinada. Creamos los campos foráneos, pero, estos serán **ids**, por lo tanto, se agregarán en la función **'primary()'**. Finalmente debemos crear las relaciones foráneas que, incluye la sintaxis mostrada, primero va el dato de la llave foránea, luego, viene el dato de la llave primaria, y tercero la tabla que contendrá la llave primaria.

```

public function up(): void
{
    Schema::create('tbl_prod_cat', function (Blueprint $table) {
        $table->unsignedInteger('ProdId');
        $table->unsignedInteger('CatId');
        $table->primary(['ProdId', 'CatId']);

        // Relaciones
        $table->foreign('ProdId')->references('Prod_Id')->on('tbl_productos');
        $table->foreign('CatId')->references('Cat_Id')->on('tbl_categorias');

        $table->timestamps();
    });
}

```

Tenemos la migración de la **tbl_prod_cat**, incluye 2 campos y la timestamps. Creamos las llaves foráneas del mismo tipo de dato que las primarias **'unsignedInteger'**, luego, indicamos que son **ids** con **'primary()'** y creamos las relaciones con la sintaxis mostrada.

Luego que creamos las migraciones, podemos inicializarlas, comandos básicos:

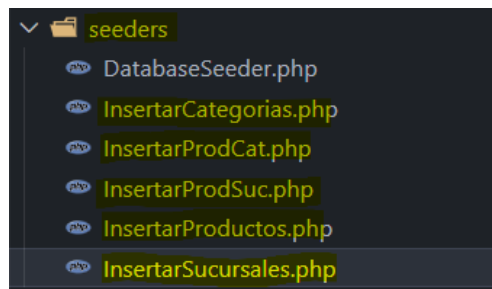
'php artisan migrate o php artisan migrate:install', nos sirve para iniciar la migración.

'php artisan migrate:restart', realiza una limpieza de las migraciones.

'php artisan migrate:refresh', realiza una limpieza y luego inicia las migraciones.

Al final del proyecto, veremos el proceso completo configurando la base de datos, para luego, realizar las migraciones y el ingreso de datos con los seeders.

III. Seeders e Ingreso de Datos



Ahora veremos los seeders, como crear cada uno de ellos, para realizar inserciones a nuestra base de datos.

```
MINGW64:/d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyecto/Proyecto...
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ code .

Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ php artisan make:seeder NombreSeeder
```

Para crear un seeder, se debe realizar la sintaxis mostrada, en la terminal que estemos utilizando y en el directorio de nuestro proyecto Laravel.


```

1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class InsertarSucursales extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         DB::table('tbl_sucursales')->insert(array(
17             ['Suc_Region' => 'Metropolitana', 'Suc_Direccion' => 'Las Ramas 780'],
18             ['Suc_Region' => 'Valparaiso', 'Suc_Direccion' => 'Aguas Claras 500'],
19             ['Suc_Region' => 'Metropolitana', 'Suc_Direccion' => 'San Antonio 690']
20         ));
21
22         /*
23         INSERT INTO tbl_sucursales (Suc_Region, Suc_Direccion) VALUES
24         ('Metropolitana', 'las ramas 780'),
25         ('Valparaiso', 'aguas claras 500'),
26         ('Metropolitana', 'San Antonio 690');
27         */
28     }
29
30
31 }
32

```

Tenemos el seeder **InsertarSucursales**, en este caso usamos una librería, que nos provee Laravel **'use Illuminate\Support\Facades\DB'**, Con este podremos ingresar a la base de datos a través de la función: **'run()'**, la sintaxis para conectarse es la mostrada, donde se indica que nos conectamos a la **DB**, a la tabla sucursales (en este caso), para realizar una inserción de datos, que a la vez se ingresa un arreglo, ya que, un registro de base de datos normalmente contiene más de un dato, luego se ingresa el nombre del campo de la tabla y luego el valor.

```

public function run(): void
{
    \DB::table("tbl_productos")->insert(array(
        ['Prod_Nombre' => 'Lámpara de Mesa', 'Prod_Descripcion' => 'Lámpara moderna'],
        ['Prod_Nombre' => 'Sofá de Cuero', 'Prod_Descripcion' => 'Sofá de cuero, 3 p'],
        ['Prod_Nombre' => 'Cafetera Automática', 'Prod_Descripcion' => 'Cafetera pro'],
        ['Prod_Nombre' => 'Juego de Sábanas', 'Prod_Descripcion' => 'Sábanas de algo'],
        ['Prod_Nombre' => 'Smart TV 55 pulgadas', 'Prod_Descripcion' => 'Televisor 4'],
        ['Prod_Nombre' => 'Mesa de Centro', 'Prod_Descripcion' => 'Mesa de centro co'],
        ['Prod_Nombre' => 'Jarrón de Cerámica', 'Prod_Descripcion' => 'Jarrón decora'],
        ['Prod_Nombre' => 'Aspiradora Robot', 'Prod_Descripcion' => 'Aspiradora robo'],
        ['Prod_Nombre' => 'Reloj de Pared', 'Prod_Descripcion' => 'Reloj de pared co'],
        ['Prod_Nombre' => 'Cortinas Opacas', 'Prod_Descripcion' => 'Cortinas de oscu'],
        ['Prod_Nombre' => 'Refrigerador', 'Prod_Descripcion' => 'Refrigerador de mat'],
        ['Prod_Nombre' => 'Espejo Decorativo', 'Prod_Descripcion' => 'Espejo de pare'],
        ['Prod_Nombre' => 'Set Cocina', 'Prod_Descripcion' => 'Set de ollas y sarten'],
        ['Prod_Nombre' => 'Almohada Sencilla', 'Prod_Descripcion' => 'Almohada de al'],
        ['Prod_Nombre' => 'Audifonos Alámbricos', 'Prod_Descripcion' => 'Audifonos A
    ));
}

```

Tenemos el seeder **InsertarProductos**, esta vez no utilizamos la librería de Laravel y nos enfocamos en la función **'run()'**, debido a que, dentro de esta función nos conectamos a la base de datos y realizamos las inserciones. La única diferencia para conectarnos esta vez a la base de datos es que, debemos agregar el signo **'\'**. Aquí nos conectamos a la tabla **productos** y realizamos las inserciones de sus campos.

```

public function run(): void
{
    \DB::table('tbl_categorias')->insert(array(
        ['Cat_Nombre' => 'Hogar'],
        ['Cat_Nombre' => 'Electrónica'],
        ['Cat_Nombre' => 'Adorno']
    ));
}

```

Tenemos el seeder **InsertarCategorias**, nos conectamos e insertamos datos a la tabla: **tbl_categorias**.

```

public function run(): void
{
    \DB::table('tbl_prod_suc')->insert(array(
        ['ProdId' => 1, 'SucId' => 1, 'Stock' => 8],
        ['ProdId' => 2, 'SucId' => 1, 'Stock' => 3],
        ['ProdId' => 4, 'SucId' => 1, 'Stock' => 12],
        ['ProdId' => 5, 'SucId' => 1, 'Stock' => 7],
        ['ProdId' => 8, 'SucId' => 1, 'Stock' => 4],
        ['ProdId' => 12, 'SucId' => 1, 'Stock' => 14],
        ['ProdId' => 14, 'SucId' => 1, 'Stock' => 23],
        ['ProdId' => 15, 'SucId' => 1, 'Stock' => 17],
        ['ProdId' => 1, 'SucId' => 2, 'Stock' => 15],
        ['ProdId' => 3, 'SucId' => 2, 'Stock' => 5],
        ['ProdId' => 5, 'SucId' => 2, 'Stock' => 8],
        ['ProdId' => 6, 'SucId' => 2, 'Stock' => 3],
        ['ProdId' => 7, 'SucId' => 2, 'Stock' => 9],
        ['ProdId' => 8, 'SucId' => 2, 'Stock' => 14],
        ['ProdId' => 11, 'SucId' => 2, 'Stock' => 2],
        ['ProdId' => 14, 'SucId' => 2, 'Stock' => 12],
        ['ProdId' => 2, 'SucId' => 3, 'Stock' => 15],
        ['ProdId' => 3, 'SucId' => 3, 'Stock' => 24],
        ['ProdId' => 4, 'SucId' => 3, 'Stock' => 5],
        ['ProdId' => 5, 'SucId' => 3, 'Stock' => 16],
        ['ProdId' => 6, 'SucId' => 3, 'Stock' => 8],
        ['ProdId' => 7, 'SucId' => 3, 'Stock' => 3],
        ['ProdId' => 8, 'SucId' => 3, 'Stock' => 15],
        ['ProdId' => 9, 'SucId' => 3, 'Stock' => 5],
        ['ProdId' => 10, 'SucId' => 3, 'Stock' => 7],
        ['ProdId' => 11, 'SucId' => 3, 'Stock' => 16],
        ['ProdId' => 12, 'SucId' => 3, 'Stock' => 3],
        ['ProdId' => 13, 'SucId' => 3, 'Stock' => 9],
        ['ProdId' => 14, 'SucId' => 3, 'Stock' => 7]
    ));
}

```

Tenemos el seeder **InsertarProdSuc**, nos conectamos e insertamos datos a la tabla: **tbl_prod_suc**.

```

public function run(): void
{
    \DB::table('tbl_prod_cat')->insert(array(
        ['ProdId' => 1, 'CatId' => 1],
        ['ProdId' => 1, 'CatId' => 3],
        ['ProdId' => 2, 'CatId' => 1],
        ['ProdId' => 3, 'CatId' => 1],
        ['ProdId' => 3, 'CatId' => 2],
        ['ProdId' => 4, 'CatId' => 1],
        ['ProdId' => 5, 'CatId' => 2],
        ['ProdId' => 6, 'CatId' => 1],
        ['ProdId' => 6, 'CatId' => 3],
        ['ProdId' => 7, 'CatId' => 1],
        ['ProdId' => 7, 'CatId' => 3],
        ['ProdId' => 8, 'CatId' => 1],
        ['ProdId' => 8, 'CatId' => 2],
        ['ProdId' => 9, 'CatId' => 1],
        ['ProdId' => 9, 'CatId' => 3],
        ['ProdId' => 10, 'CatId' => 1],
        ['ProdId' => 11, 'CatId' => 1],
        ['ProdId' => 12, 'CatId' => 1],
        ['ProdId' => 12, 'CatId' => 3],
        ['ProdId' => 13, 'CatId' => 1],
        ['ProdId' => 14, 'CatId' => 1],
        ['ProdId' => 15, 'CatId' => 2]
    ));
}

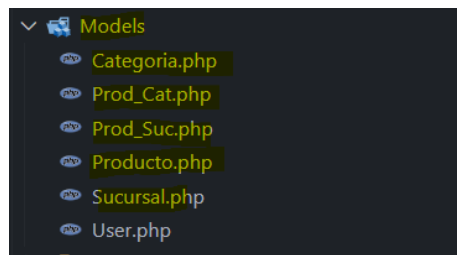
```

Tenemos el seeder **InsertarProdCat**, nos conectamos e insertamos datos a la tabla: **tbl_prod_cat**.

Finalmente tenemos los comandos para realizar las inserciones que los veremos al final:

php artisan db:seed --class=NombreSeeder, este comando nos permitirá realizar la inserción a la base de datos desde un seeder.

IV. Modelos del Proyecto



A continuación, veremos los modelos creados para el proyecto.

```
MINGW64:/d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyecto/Proyecto...
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ code .
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
to/ProyectoLaravel (master)
$ php artisan make:model NombreModelo
```

Con la sintaxis anterior, podemos crear los modelos que nos permitirán comunicarnos con la base de datos que, se utilizará para el proyecto.

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Sucursal extends Model
9  {
10     use HasFactory;
11
12     protected $table = "tbl_sucursales";
13     protected $primaryKey = "Suc_Id";
14
15     public $timestamps = true;
16
17     // Relaciones de (1:n)
18     public function prodSuc(){
19         return $this->hasMany(Prod_Suc::class, "SucId", "Suc_Id");
20     }
21 }
22
23
```

Tenemos el modelo **Sucursal**, en este creamos el nombre de la tabla, su id, y la función que, hace referencia de a la relación (1:n): **una sucursal puede tener muchos productos**, relacionándose con el modelo **Prod_Suc**.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Producto extends Model
9  {
10     use HasFactory;
11
12     protected $table = "tbl_productos";
13     protected $primaryKey = "Prod_Id";
14
15     public $timestamps = true;
16
17     // Relaciones de (1:n)
18     public function prodSuc() {
19         return $this->hasMany(Prod_Suc::class, "ProdId", "Prod_Id");
20     }
21
22     public function prodCat() {
23         return $this->hasMany(Prod_Cat::class, "ProdId", "Prod_Id");
24     }
25
26 }
27

```

Tenemos el modelo **Producto**, creamos el nombre de la tabla, el id de la tabla, y luego creamos las funciones que hacen referencias a las relaciones que tiene nuestra tabla. Tiene la relación con sucursal, donde, **un producto** puede estar en **muchas sucursales**, es por eso que se relaciona con el modelo **Prod_Suc**, y también, tiene la relación con categorías donde, **un producto** puede tener **más de una categoría**, es por eso que se relaciona con el modelo **Prod_Cat**.

```

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Categoria extends Model
9  {
10     use HasFactory;
11
12     protected $table = "tbl_categorias";
13     protected $primaryKey = "Cat_Id";
14
15     public $timestamps = true;
16
17     // Relaciones de (1:n)
18     public function prodCat() {
19         return $this->hasMany(Prod_Cat::class, "CatId", "Cat_Id");
20     }
21
22 }
23

```

Tenemos el modelo **Categoria**, creamos el nombre de la tabla, el id de la tabla y la función que hace referencia a la relación de la tabla, en este caso, nuevamente tenemos una relación de (1:n) donde, **un tipo de categoría** puede estar en **muchos productos**, es por eso la relación con el modelo **Prod_Cat**.

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Prod_Suc extends Model
{
    use HasFactory;

    protected $table = "tbl_prod_suc";
    protected $primaryKey = "ProdId";
    protected $primaryKey2 = "SucId";

    public $timestamps = true;

    // Relaciones de (n:1)
    public function producto() {
        return $this->belongsTo(Producto::class, "Prod_Id");
    }

    public function sucursal() {
        return $this->belongsTo(Sucursal::class, "Suc_Id");
    }
}

```

Aquí tenemos el modelo de **Prod_Suc**, este incluye el nombre de la tabla, los ids de las tablas, ya que, es una tabla intermedia que representa una relación (n:n), por lo tanto, la clave primaria es compuesta, y luego tenemos las funciones que hacen referencias a estas relaciones. Primero (n:1), tenemos que **muchos productos** le pueden pertenecer **a una misma sucursal**, por eso la relación con el modelo **Sucursal**. Segundo (n:1), que **muchas sucursales** pueden tener **el mismo producto** por eso la relación con el modelo **Producto**.


```

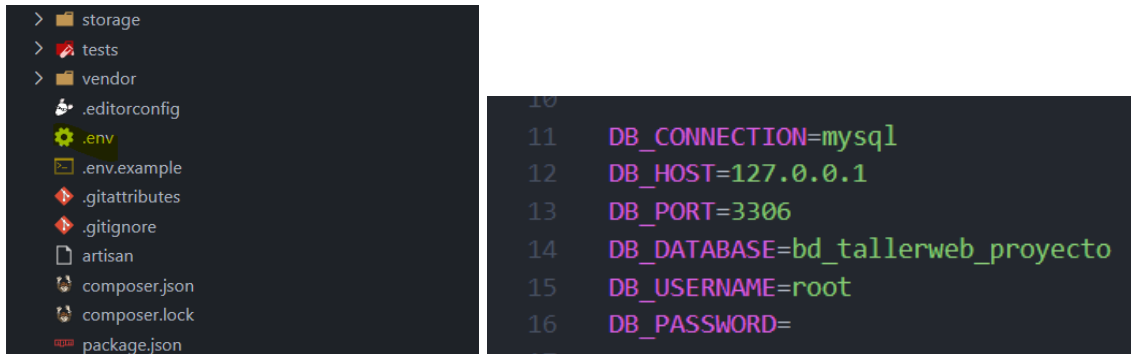
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Prod_Cat extends Model
9  {
10     use HasFactory;
11
12     protected $table = "tbl_prod_cat";
13     protected $primaryKey1 = "ProdId";
14     protected $primaryKey2 = "CatId";
15
16     public $timestamps = true;
17
18     // Relaciones de (n:1)
19     public function producto() {
20         return $this->belongsTo(Producto::class, "Prod_Id");
21     }
22
23     public function categoria() {
24         return $this->belongsTo(Categoria::class, "Cat_Id");
25     }
26
27
28 }
29

```

Para concluir tenemos el modelo **Prod_Cat**, este incluye el nombre de la tabla, los ids de las tablas, ya que, es una tabla intermedia con relación (n:n) y las funciones que hacen referencias a estas relaciones. Primero (n:1), **un conjunto de categorías** las puede tener **un mismo producto**, por eso la relación con el modelo **Producto**. Segundo (n:1), **un conjunto de productos**, pueden compartir **una misma categoría**, por eso la relación con el modelo de **Categoria**.

V. Prueba de Migraciones/Seeders

Vamos a realizar una prueba con la arquitectura que tenemos creada, para eso configuramos nuestras variables de entornos y configuraciones de la base de datos.



The image shows a file explorer on the left with a list of files and folders: storage, tests, vendor, .editorconfig, .env, .env.example, .gitattributes, .gitignore, artisan, composer.json, composer.lock, and package.json. The .env file is highlighted. To the right, a code editor shows the contents of the .env file, with lines 11 through 16 highlighted in green. The code is as follows:

```
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=bd_tallerweb_proyecto
15 DB_USERNAME=root
16 DB_PASSWORD=
```

Para configurar las variables de entorno nos dirigimos al archivo `.env` y luego nos enfocamos en las líneas 11-16, esta será nuestra configuración inicial de la base de datos, donde, veremos la **conexión, host, puerto, base de datos, nombre de usuario y contraseña**, normalmente tenemos las conexiones por defecto, y nos enfocamos en el nombre de la base de datos, que creamos para realizar las migraciones e inserciones.



The image shows a code editor with a PHP array configuration for a database connection. The code is as follows:

```
'mysql' => [
    'driver' => 'mysql',
    'url' => env('DATABASE_URL'),
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'forge'),
    'username' => env('DB_USERNAME', 'forge'),
    'password' => env('DB_PASSWORD', ''),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => 'utf8',
    'collation' => 'utf8_general_ci',
    'prefix' => '',
    'prefix_indexes' => true,
    'strict' => true,
    'engine' => 'InnoDB',
    'options' => extension_loaded('pdo_mysql') ? array_filter([
        PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
    ]) : [],
],
```

También nos dirigimos a nuestro archivo `database.php`, que se encuentra en nuestro directorio `config`, para realizar los ajustes en las propiedades: **charset, collation y engine**. De esta manera, nos encontramos en las condiciones para realizar los comandos de migraciones e inserciones a la base de datos creada.

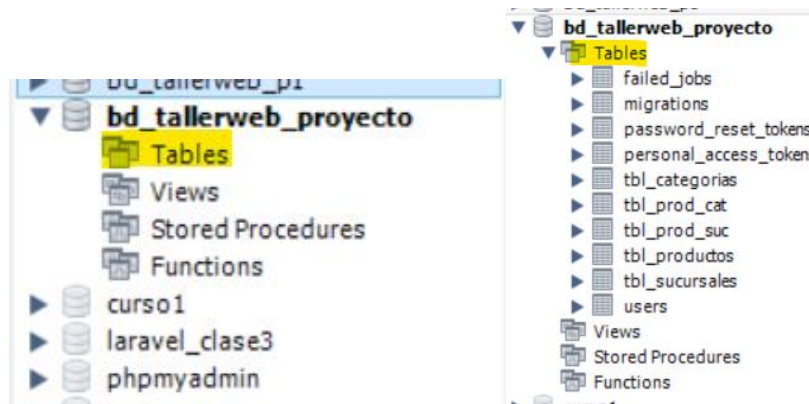
```
MINGW64:/d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyecto/Proje...
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyec
toLaravel (master)
$ php artisan migrate

INFO Preparing database.

Creating migration table ..... 88ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table ..... 30ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 41ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 55ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 43ms DONE
2023_11_16_191341_create_tbl_sucursales ..... 12ms DONE
2023_11_16_191410_create_tbl_productos ..... 12ms DONE
2023_11_16_191430_create_tbl_categorias ..... 12ms DONE
2023_11_16_191447_create_tbl_prod_suc ..... 176ms DONE
2023_11_16_191456_create_tbl_prod_cat ..... 142ms DONE
```



Realizamos el comando para ejecutar las migraciones y vemos nuestra base de datos antes de las migraciones sin ninguna tabla y después de las migraciones con las tablas creadas.

```
Eliulson@emartinez MINGW64 /d/Tec-P-C-A-S/Bime7/F-PW/proyecto_incremental/proyecto/ProyectoLaravel (master)
$ php artisan db:seed --class=InsertarProductos

INFO Seeding database.
```

Realizamos el comando para insertar datos, a través de los seeders, un recordatorio importante, las inserciones con los seeders, al igual que la creación de las migraciones, deben ser en orden, o de otra manera, nos dará error, ya que, no podemos relacionar datos entre sucursales y productos, si no hemos realizado el ingreso de productos y sucursales antes.

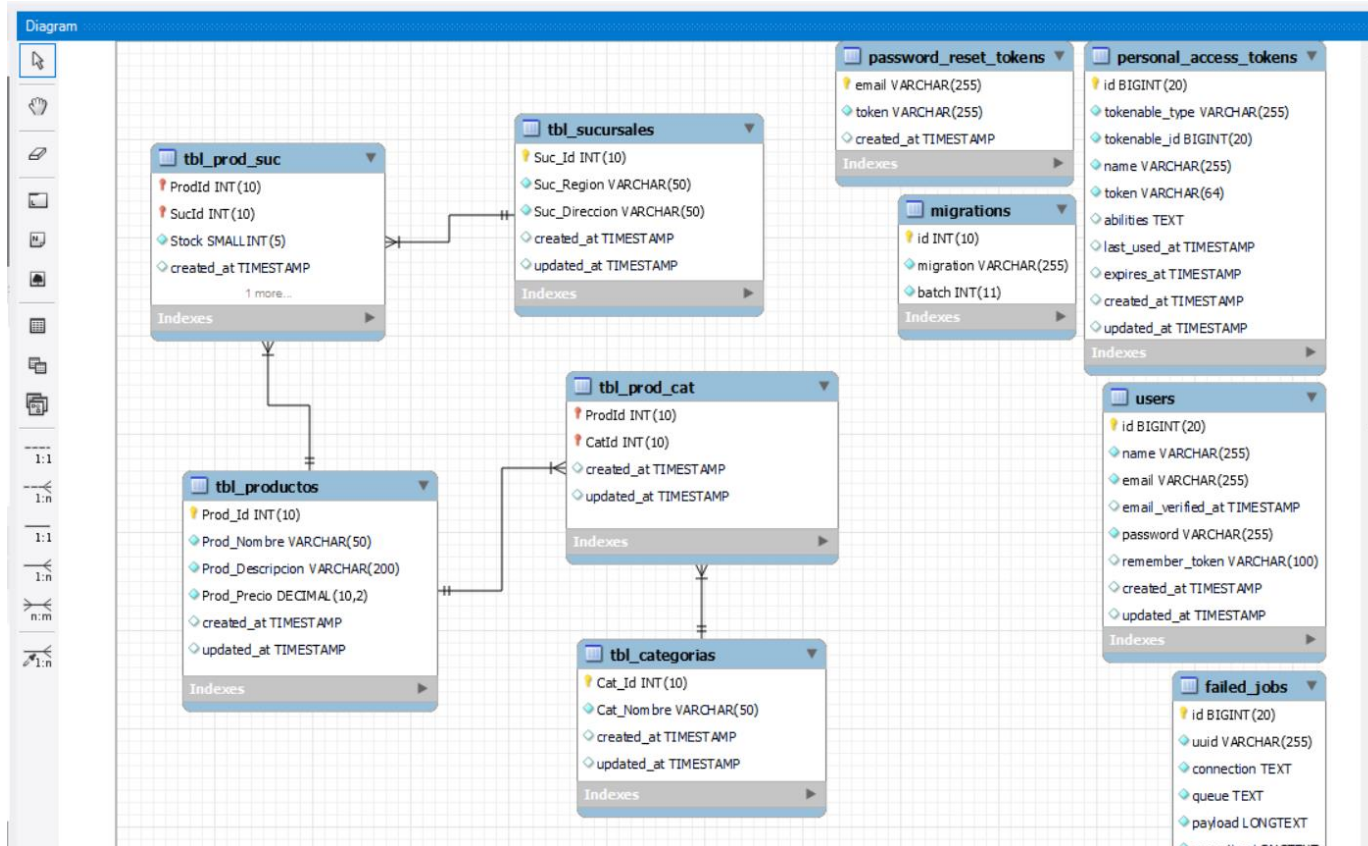
Aquí tenemos un ejemplo de los registros de la base de datos antes y después del comando que nos permite hacer la inserción:

1 • SELECT * FROM tbl_sucursales;	1 • SELECT * FROM tbl_sucursales;
2 • SELECT * FROM tbl_productos;	2 • SELECT * FROM tbl_productos;
3 • SELECT * FROM tbl_categorias;	3 • SELECT * FROM tbl_categorias;
4 • SELECT * FROM tbl_prod_suc;	4 • SELECT * FROM tbl_prod_suc;
5 • SELECT * FROM tbl_prod_cat;	5 • SELECT * FROM tbl_prod_cat;

Prod_Id	Prod_Nombre	Prod_Descripcion	Prod_Precio	created_at	updated_at
1	Lámpara de Mesa	Lámpara moderna para iluminación en el hogar	3200.00	2024-01-01 12:00:00	2024-01-01 12:00:00
2	Sofá de Cuero	Sofá de cuero, 3 plazas	250000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
3	Cafetera Automática	Cafetera programable para café fresco	30000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
4	Juego de Sábanas	Sábanas de algodón suave, juego de 4 piezas	22000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
5	Smart TV 55 pulgadas	Televisor 4K Smart	210000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
6	Mesa de Centro	Mesa de centro con diseño especial	130000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
7	Jarrón de Cerámica	Jarrón decorativo de cerámica en blanco	4500.00	2024-01-01 12:00:00	2024-01-01 12:00:00
8	Aspiradora Robot	Aspiradora robotizada para limpieza	14000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
9	Reloj de Pared	Reloj de pared con diseño rústico	7000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
10	Cortinas Opacas	Cortinas de oscurecimiento para habitaciones	26000.00	2024-01-01 12:00:00	2024-01-01 12:00:00
11	Refrigerador	Refrigerador de 3 puertas inoxidable	28000.00	2024-01-01 12:00:00	2024-01-01 12:00:00

Una vez ingresamos los comandos de las migraciones y seeder, con **MySQL workbench**, podremos crear el modelo de la base de datos que será similar a la inicial de este informe.

VI. Modelo Creado con Migraciones y Seeders



VII. Conclusiones

Hemos podido realizar con éxito la implementación de la base de datos: desde la integración de migraciones, seeders y modelos, para posteriormente darle el uso en el proyecto final, se adjunta archivos y enlace de Git.

Enlace de Git:

<https://github.com/gasdar/LaravelMyLogo>