Se trabajará con los siguientes ejemplos:

- categories: la tabla de la base de datos.
- Category.php: la clase modelo.
- CategoriesController.php: la clase controlador.
- resources > views > categories > index.blade.php.
- $category: representa una entrada de la tabla categories.
- $categories: representa todas las entradas de la tabla categories.

Estructura del proyecto [Modelo - Vista - Controlador]:

- **app**
  - **Http**
    - **Controllers [Controlador]:** para unir una **vista** con un **modelo**.
      - **CategoriesController.php**
  - **Models [Modelo]:** contiene archivos .php que representa modelos de datos almacenados en una tabla de una db.
    - **Category.php**
- **database**
  - **migrations**
    - Contiene clases php de "control de versiones" que registran todos los cambios realizados sobre una db, permite aplicar nuevos cambios con su función **up()** y deshacer cambios con su función **down()**.
- **public**:
  - Contiene archivos públicos como **favicon.ico**.
- **resources**
  - **css**: archivos css
  - **js**: archivos js
  - **views [Vista]:** archivos con código de php y html que el cliente verá en su navegador
    - **app.blade.php**:
    - **categories**
      - **index.blade.php**
- **routes**
  - **web.php**: Aquí se registran los HTTP Request que recibirá el server y qué respuesta debemos dar.

Explicaciones:

- Para crear el proyecto, ir a la carpeta xampp/htdocs y tipear: **composer create-project laravel/laravel [nombre]**
- Para iniciar el servidor, ir a la carpeta [nombre] del proyecto y tipear: **php artisan serve**
- Plantillas de vistas: en el archivo **app.blade.php** se inserta el código que estará presente en todas las vistas (por ejemplo, un nav bar). Y se agrega **@yield ('nombre de la plantilla')** en la parte del código

donde cada vista agregará su propio código. El archivo **index.blade.php** comenzará con **@extends('app')** y luego utilizará **@section('nombre de la plantilla) … @endsection** para insertar el código propio de la vista **index.blade.php**.

- Agregar **@csrf** en el código de un formulario, es una técnica de seguridad.
- Crear modelo:
  php artisan make:model Category –m
  El parámetro **-m** es para crear las migraciones automáticamente.
  Si el modelo representa una tabla que tiene una relación "uno a muchos" con otro modelo (por ejemplo **Tasks**), la clase Category contendrá la función:
  public function tasks() {
  return $this->hasMany(Task::class);
  }
- Ejecutar código de las clases migraciones:
  php artisan migrate
- Migraciones – chequear estado:
  php artisan migrate:status
- Migraciones – deshacer último cambio:
  php artisan migrate:rollback
- Crear controlador:
  php artisan make:controller CategoriesController --resource
  El parámetro --resource crea automáticamente todos los métodos de CRUD.
  Dentro del controlador se definen las funciones de CRUD que, por defecto, tienen estos nombres:
  index(): Esta función buscará todas las filas de la tabla categories y luego devolverá una vista (es decir, un archivo php con código html que verá el cliente) junto a esa vista, se pasará la variable **$categories** que contiene todas las filas de la tabla, para que el código .php de la vista pueda mostrar esa información con un
  @foreach ($categories as $category)
  @endforeach
  El código de este ejemplo es:

```
public function index() {
    $categories = Category::all();
    return view('categories.index', ['categories' => $categories]
    // categories.index refiere a la vista resources > views > categories
> index.blade.php
}
```

  create()
  store()
  show()
  edit()
  update()
  destroy()