**Módulo 1:**

**Clase 09: Controladores**

**¿Por qué usar Controladores?**

Los controladores nos permiten agrupa código y lógica dentro de archivos individuales, organizando mejor nuestro sistema

**¿Cómo se crean?**

Con el comando php artisan make:controller nombre\_controllador

**¿Dónde se guardan?**

Dentro de la carpeta App/Http/Controllers

Con el comando php artisan make:controller PostsController -–resource podemos crear un controlador con los métodos del crud ya implmentados  
  
**Clase 10: Consola de Lenguaje Tinker**

**¿Qué es Tinker?**  
Tinker es una línea de comandos interactiva, donde podremos ejecutar código php sin tener que hacerlo desde la aplicación, a su vez, tenemos acceso a las clases definidas en el proyecto. Pudiendo crear objetos a partir de ellas.

**¿Cómo ingresamos a dicha consola?**

con el comando php artisan tinker

$user = new App\Models\User: Podemos acceder a las funciones definidas en la clase User

$user->name = ‘Enzo’: definimos un valor en el atributo name.

Si realizamos una actualización al código, debemos cerrar la consola Tinker y volver a acceder.

**Módulo 2:**

**Clase 01: Variables de entorno y conexiones a base de datos**

En este capítulo vamos a hacer uso de base de datos, para recuperar valores de las tablas y visualizarlos en las vistas en este caso.

Para ellos debemos seguir los siguientes pasos:

1. **Especificar Gestor BD:** Debemos decirle a nuestra aplicación con que motor de bd vamos a trabajar, para ello, nos dirigimos al archivo. Env

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Agregar especificaciones: Una vez dentro, vamos al bloque de conexión a bd y detallamos tanto el gestor, url de bd, puerto, tabla, usuario y contraseña.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Nota: el archivo .env no es versionado, es decir no se incluye en el historial de cambios del repositorio de git. Las variables dentro pueden cambiar, dependiendo del entorno donde se encuentre.

1. **Creamos nuestra base de datos:** En este ejemplo, hemos creado un bd llamada “blog”, con una tabla llamada “posts”, esta última tiene 3 atributos (id, title y excpert), con 3 registros

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Accedemos a la BD desde Tinker**: Con el uso de **php artisan Tinker**, ingresamos a la dicha consola, y mediante este comando **DB::table('posts')->get()**, recuperamos los registros.

1. **Comandos:**

DB::table('posts')->get()

* + **BD:** llamados constructor de consulta de Laravel por medio de esta clase
  + **::table('posts'):** accedemos al método de la clase BD y le indicamos el nombre de la tabla
  + **->get():** este método recupera todos los registros que se encuentra dentro de la tabla **“posts”**

Texto

Descripción generada automáticamente

**DB::table('posts')->select('title')->get():** Este commando recupera todos los títulos de la tabla **“posts”**, esto mediante el método ->select('title')->get(), donde indicamos el nombre de la columna, en este caso, ‘title’

Texto

Descripción generada automáticamente

**DB::table('posts')->orderBy('id', 'DESC')->get():** lista los registros por id de forma decreciente, donde en **orderBy(),** indicamos un parámetro para ordenar mediante una columna y otro, que puede ser opcional, el criterio, en este caso de mayor a menor

También podemos almacenar las consultas en variables:

**$posts= DB::table('posts')->get():** en este caso, la variable **$posts** almacena la consulta de recuperar los registros de la tabla **“posts”**

Texto

Descripción generada automáticamente

**$posts[0]:** Recupera el primer elemento, el que se encuentra en posición 0, de la tabla **“posts”**

Texto

Descripción generada automáticamente

**$posts[2]->title:** Recupera el titulo del tercer valor, el que se encuentra en posición 2, de la tabla “posts”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Clase 02: Migraciones**

Crearemos la tabla mediante migraciones de laravel, por la escritura de código php, asegura que todas las personas que descarguen el proyecto tengan el mismo esquema de base de datos.

**¿Dónde se encuentran?**

En la carpeta de databse/migrations, por defecto se generan 4 migraciones cuando creamos un nuevo proyecto.

Texto

Descripción generada automáticamente

**php artisan make:migration create\_nombreTabla\_table**

Con este comando creamos la tabla desde la terminal por medio de una migración

**php artisan make:migration create\_ nombreTabla \_table --create=nuevoNombre**

usamos este comando cuando queremos darle un nombre distinto a la tabla