**CODIGO DEL CURSO:**





**Desarrollo de Aplicaciones Empresariales**

**LABORATORIO N°**

**Laravel**

**–**

**Definición de Proyecto y**

**Creación de Modelos**

|  |  |  |
| --- | --- | --- |
|  | ***Alumno(s)*** | ***Nota*** |
|  | **Baldeón Ayala, Willy Alexis** |  |
|  |  |  |
|  |  |  |
| ***Grupo*** | ***C15 – A*** |  |
| ***Ciclo*** | ***IV*** |  |
| ***Fecha de entrega*** |  |  |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 1 de 10 |
| **I.- OBJETIVOS:**    **II.- SEGURIDAD:**   |  |  | | --- | --- | |  | **Advertencia:**  **En este laboratorio está prohibida la manipulación del hardware, conexiones eléctricas o de red; así como la ingestión de alimentos o bebidas.** |     **III.- FUNDAMENTO TEÓRICO:**  Revise sus diapositivas del tema antes del desarrollo del laboratorio.  **IV.- NORMAS EMPLEADAS:**  No aplica **V.- RECURSOS:**   * En este laboratorio cada alumno trabará con un equipo con Windows 8.     **VI.- METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:**   * El desarrollo del laboratorio es individual.     **VII.- PROCEDIMIENTO:**  **Nota**:  Las secciones en cursivas son demostrativas, pero sirven para que usted pueda instalar las herramientas de desarrollo en un equipo externo.  **CREANDO UN DOMINIO LOCAL**   1. Ingresamos a **C:\Windows\System32\drivers\etc** y abrimos el archivo **hosts** con el **block** **de** **notas.** 2. Agregamos un nombre de dominio con resolución de nuestro localhost:        1. Probemos el link creado en nuestro browser, debería abrirse la página de xamp o wamp.   Sin embargo, la página a la que hemos sido re-direccionados, es la página principal de xamp o wamp,     1. Vamos modificar el archivo:    1. En WAMPSERVER: **C:\-wamp64\bin\apache\apache2.4.17\conf\extra\httpd-vhosts.conf**    2. En XAMP: **C:\xamp\apache\conf\extra\httpd-vhosts.conf**       **CREANDO LA BASE DE DATOS PARA EL PROYECTO**  En esta sección vamos a crear la base de datos. Para ello deberíamos tener instalado MySql.   1. Ingresar a la página de **PhpMyAdmin**. 2. Crear una nueva base de datos de nombre: gestorimagenes.          1. Las tablas serán creadas de manera análoga a Symfony2.     Base de datos “gestorimagenes” creada.  **CONFIGURANDO EL ENTORNO Y EL PROYECTO DE LARAVEL**   1. Abra **Sublime** **Text**. 2. Elija la opción **File->OpenFolder** y abra la carpeta del proyecto creado con Composer. 3. Abra la ventana de comandos de Windows y ubíquese en la carpeta del proyecto. 4. Escriba “**php artisan list**”. Visualizará la lista de comandos disponibles que están disponibles en **artisan**      1. Configurar el nombre de la aplicación:    1. Escriba **“php artisan app:name GestorImagenes”.**      * 1. Para corroborar el cambio abra el archivo **app/Http/Controllers/Auth/HomeController.php,** Notará que el nuevo espacio de nombres es el asignado.     **Nombre de la app cambiada por “GestorImagenes”.**     1. Ubique el archivo “.env”.        * 1. Configure el código para que quede de la siguiente manera:   **DEFINIENDO EL PROYECTO**  El proyecto que crearemos en este curso será un gestor de imágenes y álbumes que un usuario registrado podrá realizar.  Algunos casos de Uso:   * El usuario podrá crear, modificar y eliminar fotografías. * El usuario podrá crear, modificar y eliminar álbumes. * Un álbum tiene varias fotografías. * Una fotografía pertenece a un solo álbum. * Un usuario tiene varios álbumes. * La fotografía tiene un identificador, nombre, descripción, una ruta en el servidor, y el álbum al que pertenece. * El álbum tiene un identificador, nombre, descripción y el usuario al que pertenece. * El usuario tiene un identificador, nombre, **email y password.** o **Email y password:** Estos campos, laravel los considera por defecto, es decir que tiene una programación ya dada incluso para enviar un correo y confirmar a un usuario.     **CREANDO LOS MODELOS PARA EL PROYECTO**  1. Ubique el archivo **User.php** que se encuentra dentro de la carpeta **app.** | | |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 2 de 10 |
| 1. Renombre el archivo por: **Usuario.php.** 2. En la línea 9, modifique el nombre de la clase por **Usuario.** 3. En la línea 18, modifique el nombre de la tabla por **usuarios.** 4. En la línea 25, modifique los campos que el usuario tendrá, para que quede de la siguiente manera:       Crearemos el siguiente modelo.   1. Cree un nuevo archivo de nombre **Album.php** en la carpeta en la que se encuentra **Usuario.php.** 2. Copie el código de usuario.php, y modifíquelo para que quede de la siguiente manera:     8. Repita los pasos 6 y 7, solo que ahora para crear el modelo a la entidad **Foto.php.** El código debería quedar de la siguiente manera: | | |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 3 de 10 |
| **DEFINIENDO RELACIONES DE CARDINALIDAD ENTRE MODELOS**   1. Crearemos la relación de **ÁLBUM -> FOTO:** **Un** **álbum** tiene **varias** **fotos**. 2. Crearemos la relación de **ÁLBUM -> USUARIO:** **Un** **álbum** pertenece a un **único** **usuario**. 3. Modificamos el modelo álbum para que quede de la siguiente manera:        1. Crearemos la relación de **FOTO** -> **ÁLBUM**: **Una** **foto** pertenece a **un** **álbum**. 2. Modificaremos el modelo Foto para que quede de la siguiente manera:        1. Crearemos la relación USUARIO -> ALBUM: Un usuario posee varios álbumes. 2. Modificaremos el modelo Usuario para que quede de la siguiente manera:       A este punto, tenemos listos los modelos. Posteriormente realizaremos la migración a la base de datos.  **EJERCICIO:**   **Modifique el modelo USUARIO para que ahora, adicionalmente tenga 2 campos “pregunta” “respuesta”, estos campos servirán para una posible recuperación de contraseña de este.**    **Campos adicionales “pregunta” y “respuesta” para una posible recuperación de la contraseña del usuario.** | | |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 4 de 10 |
| **CREANDO LOS CONTROLADORES**   1. Ubique la carpeta **app** -> **http** -> **controllers**. 2. Dentro de ella encontrará una carpeta que agrupa a los controladores de la autenticación: **Auth**. 3. Renombre la carpeta “**Auth**” por “**validacion**”.      1. Dentro de la carpeta “**validacion**”, tendremos 2 archivos, **elimine** el archivo “**Password**”. Borraremos el archivo porque nosotros haremos la recuperación de contraseña por medio de la pregunta y respuesta secreta. 2. En la carpeta “**validacion**”, queda un archivo “**AuthController.php**”. Renómbrelo a “**ValidacionController.php**”.      1. Este archivo se encargará de controlar funciones como iniciar sesión, crear cuentas, validar al usuario que va a iniciar sesión, etc. 2. Modifique el **namespace** de **ValidacionController.php**. la última ruta debería ser: **Validacion (en lugar de “Auth”, ya que modificamos el nombre de la carpeta)**       Si usamos la combinación de teclas **Alt + clic** Sobre “**AuthenticatesAndRegostersUsers**”, iremos a la definición del Trait. | | |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 5 de 10 |
| **EJERCICIO**  **Investigue y averigüe: ¿Cuál es la utilidad, características y funcionalidad de un TRAIT en PHP?**  **TRAITS**  Un TRAIT es un mecanismo de reutilización de código en lenguajes de herencia simple, principalmente PHP. Su objetivo es el de reducir las limitaciones propias de la herencia simple permitiendo que los desarrolladores reutilicen a voluntad conjuntos de métodos sobre varias clases independientes y pertenecientes a clases jerárquicas distintas.  Son similares a una clase, pero con el único objetivo de agrupar funcionalidades muy específicas y de una manera coherente. Los TRAITS se inyectan en la clase de como si sus métodos formaran parte de ella, por lo que cualquier método o propiedad ‘private’ o ‘protected’ podrá ser accedido desde un TRAIT.  Resultado de imagen para caracteristicas trait php  **Características**   * Un TRAIT no puede ser instanciado directamente. * Los TRAITS permiten combinar miembros de clases sin tener que usar herencia. * Los TRAITS integran un orden de precedencia para la sobre escritura de métodos, donde los métodos de la clase actual sobre escriben los métodos del TRAIT, que a su vez sobre escriben los métodos de la clase padre. * Un TRAIT puede estar compuesto de otros TRAITS.  1. Lo que tiene la definición del Trait es la obtención del registro de un usuario, login del mismo, etc., Aunque todas las funciones ya están declaradas en inglés, algunas no son lo suficientemente óptimas y adecuadas para la aplicación que deseamos desarrollar. Entonces lo que haremos será crear una copia del código del trait, de modo que irá incluido en el controlador de validación y obviaremos la sentencia “USE”, para omitir el uso del archivo. 2. **Copie** el código de la declaración del **Trait** y **péguelo** en el controlador de **validación**, (Todo el código copiado irá en vez del “**use**” que se encuentra en la **fila 21).** 3. Para ordenar el código, ubique el **constructor** del **controlador** y **muévalo** inmediatamente después de la declaración de la variable protegida **$registrar.**   El resultado debería tener cierta similitud respecto de la imagen a continuación       1. Ubique la función **“getRegister”** y modifique el nombre a **“getRegistro”**. 2. Dentro de la misma función, **modifique** los valores retornados, y retorne el texto: **“formulario creación cuenta”.**      1. Ubique la función “**postRegister**” y modifique el nombre a “**postRegistro**”. 2. Ubique la función “**getLogin**” y modifique el nombre a “**getInicio**” 3. Ubique la función “**postLogin**” y modifique el nombre a “**postInicio**” 4. Ubique la función “**getLogout**” y modifique el nombre a “**getSalida**” 5. Ubique la función “**loginPath**”. En la ruta, al final de la sentencia **return**, modifique “ **‘/auth/Login’** ” por “ **‘/validacion/inicio’** ” 6. Ubique la función **getInicio** y modifique el valor de retorno de la misma por el texto: “**mostrando formulario inicio sesión”** 7. Ubique la función **getFailedLoginMessage,** es la función que se ejecuta en caso de algún dato incorrecto al momento de realizar la validación. Modifique su valor de retorno por el texto: **“email o contraseña incorrectos.”** 8. Ubique la función **redirectPath,** modifique la ruta en el valor de retorno: **‘/home’** por **‘/inicio’.**   Estamos realizando cambios de lenguaje únicamente, más adelante realizaremos cambios funcionales.  Crearemos una función para recuperar la cuenta en caso el usuario no recuerde sus credenciales. Agregue una       1. Ahora abra el archivo **HomeController.php.** 2. Modifique el nombre del archivo a: “**InicioController.php**” 3. Modifique el nombre de la clase de “**HomerController**” a “**InicioController**” 4. Ubique la función **Index,** modifique el nombre de la función por “**getIndex**”. Modifique su valor de retorno por el texto: “**página de inicio validado**”.        1. Abra el archivo **WelcomeController.php. //**Esta es una página que se les muestra a los invitados, vale decir a los que aún no han iniciado sesión. 2. Modifique el nombre del archivo por: “**BienvenidaController.php**” 3. Así mismo, modifique el nombre de la clase “**WelcomeController**” por “**BienvenidaController**” 4. Ubique la función **Index,** modifique el nombre de la función por “**getIndex**”. Modifique su valor de retorno por el texto: “**bienvenida a la aplicación**”. | | |

|  |  |  |
| --- | --- | --- |
|  | **Desarrollo de Aplicaciones Empresariales – Laravel** | Nro. DD-106 |
| Página 9 de 10 |
| **OBSERVACIONES**     * Los TRAITS no son un nuevo concepto de PHP, otras implementaciones parecidas están disponibles en otros lenguajes de programación. * PHP es uno de los principales lenguajes con licencia de software libre utilizados para el desarrollo web. Una de sus características principales es que puede ser incrustado en HTML. * Si creamos una base de datos utilizando MySql, debemos saber que las tablas pueden crearse de manera análoga a Symfony2. * PHP cuenta con numerosas librerías que facilitan el desarrollo de aplicaciones web.     **CONCLUSIONES**   * Agregamos un nombre de dominio a nuestro Localhost: “GestorLaravel”. Al hacer esto, la página de inicio de Wamp redirige automáticamente a la dirección del Localhost establecida. * **Manipulamos los ficheros de nuestro proyecto a través del editor de código “SublemeText”**. Esta herramienta facilita la navegación entre múltiples archivos a través de pestañas. * **Comprendimos que los TRAITS son mecanismos que planean sofisticar el proceso de herencia**. PHP está muy relacionado con ellos por su lenguaje de herencia simple. * Implementamos controladores en nuestro proyecto de PHP. Ellos nos permiten la obtención de datos del modelo y establece la vista correspondiente según sea el caso. * En el transcurso del laboratorio, desarrollamos un proyecto web basado en PHP el cual interactúa con múltiple información y enlaces internos. PHP nos permite trabajar de una manera muy cómoda y gracias a la exhaustiva documentación que posee, es sencillo comprender ciertos aspectos confusos del lenguaje. | | |