**PROGRAMACIÓN WEB**

**FASE 4**

**PRESENTADO POR**

**LUIS SALCEDO**

**ANDRES MELENDEZ**

**MARLENE HERNANDEZ**

**BENJAMIN QUINONES**

**ROBERTSON ORTEGA**

**PRESENTADO A**

**FRANCISCO HILARION**

**GRUPO 301127\_1**

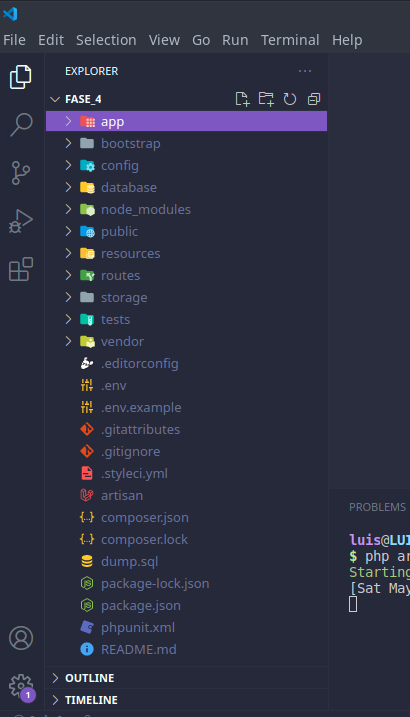
**UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA**

**SANTA MARTA**

**2022**

**INFORME**

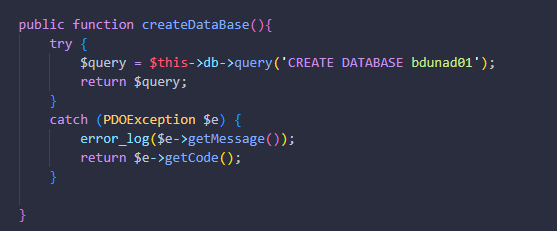
**Estructura básica:**

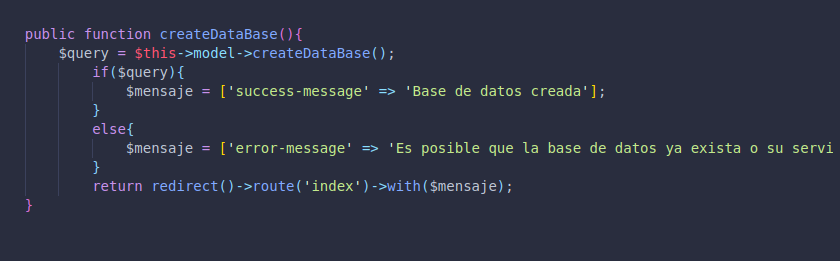


El proyecto se realizó con el framework Laravel por lo que la estructura de carpetas corresponde al de dicho marco de desarrollo.

**Funciones – explicación y descripción**

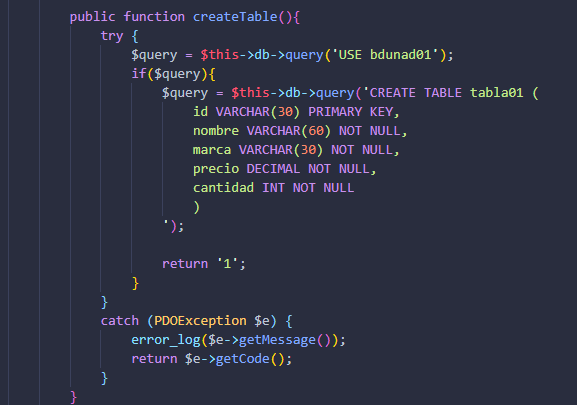
**Crear base de datos:**





La función con el código mysql para crear la base de datos se encuentra en el modelo **admin** y se ejecuta en la función **createDataBase** del controlador **admin**, esto cuando se accede a la ruta pertinente.

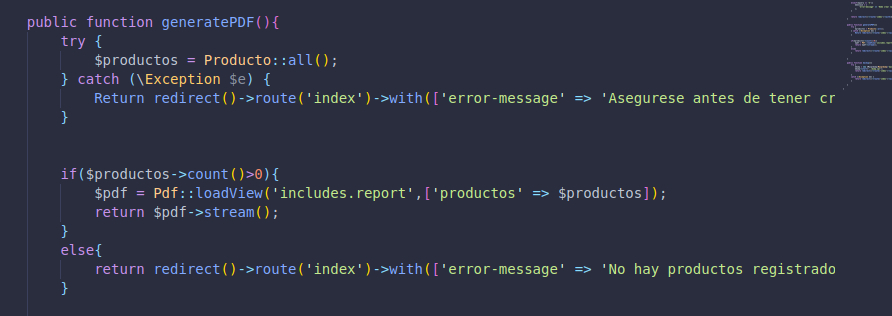
**Crear tabla:**





La función para crear la tabla se encuentra en el modelo **admin** y se usa en el Controlador **admin**, el cual la ejecuta al acceder a la ruta pertinente.

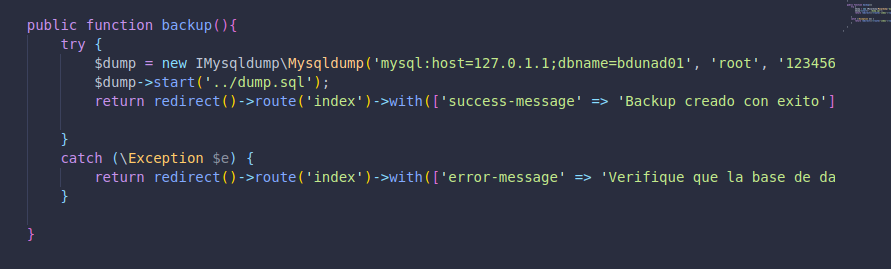
**Reporte PDF**

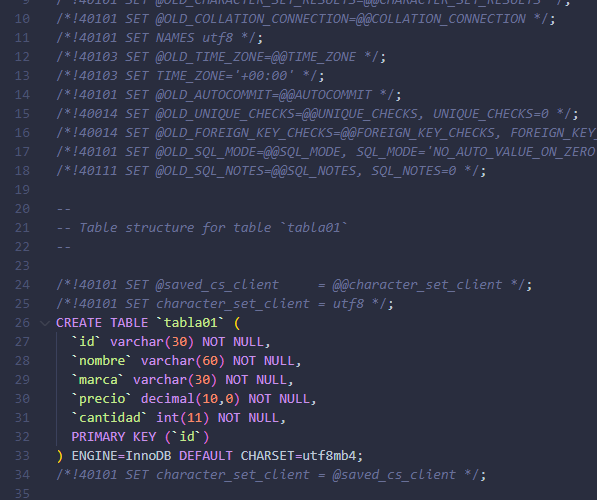
****

****

Esta función se encuentra en el controlador **admin**. Como primer paso hace una consulta a la base de datos, si se consigue conectar a la base de datos y encuentra registros en la tabla generará el PDF, de lo contrario redirigirá a la página del inventario con el mensaje de error pertinente.

**Backup:**

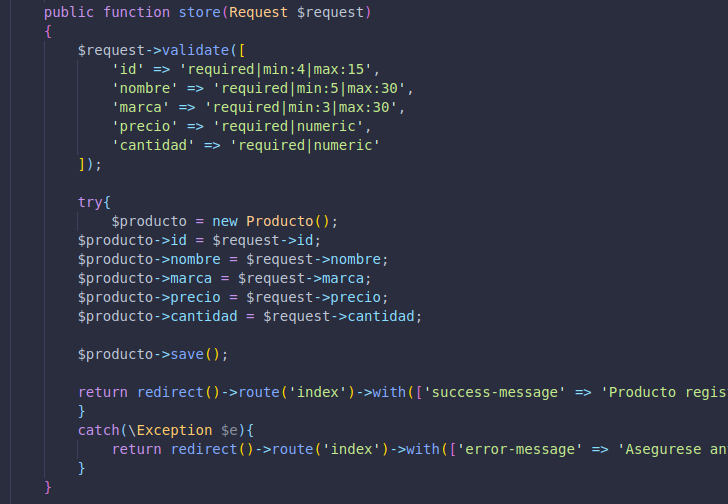


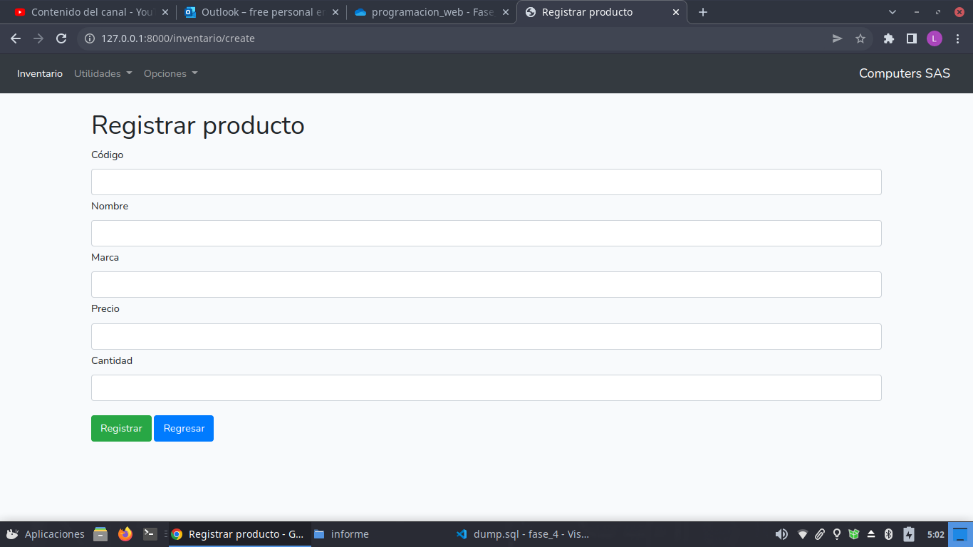


Esta función se encuentra en el controlador **admin** y cuenta con un bloque **try/catch** en donde haciendo uso de la clase **Mysqldumb** intenta conectarse a la base de datos especificada para luego con el método start crear el archivo backup en la carpeta raíz del proyecto con el nombre de **dump.sql**.

En caso de algún error se atrapará en el bloque catch, se redirige a la vista inventario con un mensaje de error.

**Insert:**

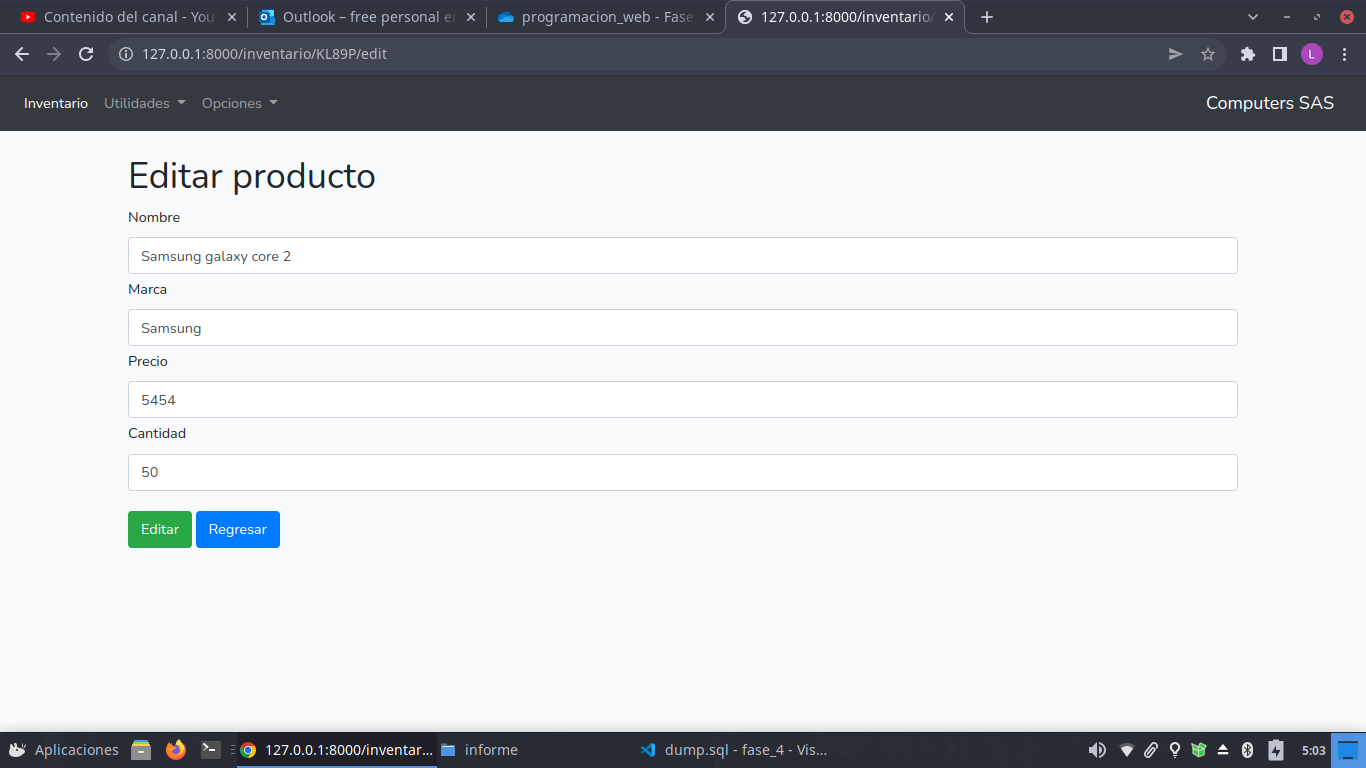
****

****

La función para insertar productos se encuentra en el controlador **ProductoController** y cuenta con validaciones para verificar que todos los campos necesarios fueron pasados en el formulario para luego instanciar el modelo Producto, cargarle los datos y ejecutar su método **save()** y redirigir a la página principal.

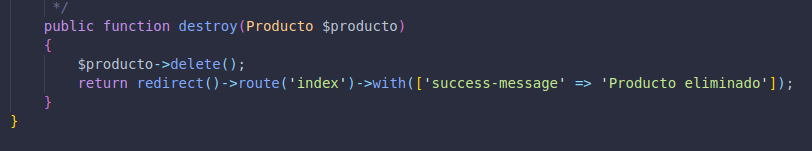
**Update:**





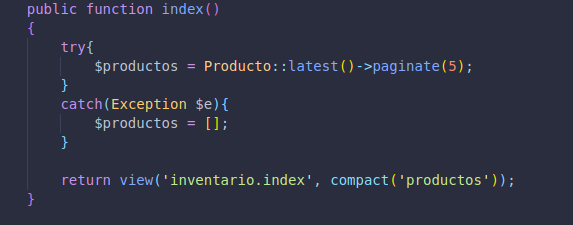
El método **update** se encuentra en el controlador **ProductoController** y lo primero que hace es validar que todos los datos necesarios fueron pasados a través del formulario, luego carga dicho datos en una instancia del modelo Producto y ejecuta su método **save()** lo cual actualizará el registro.

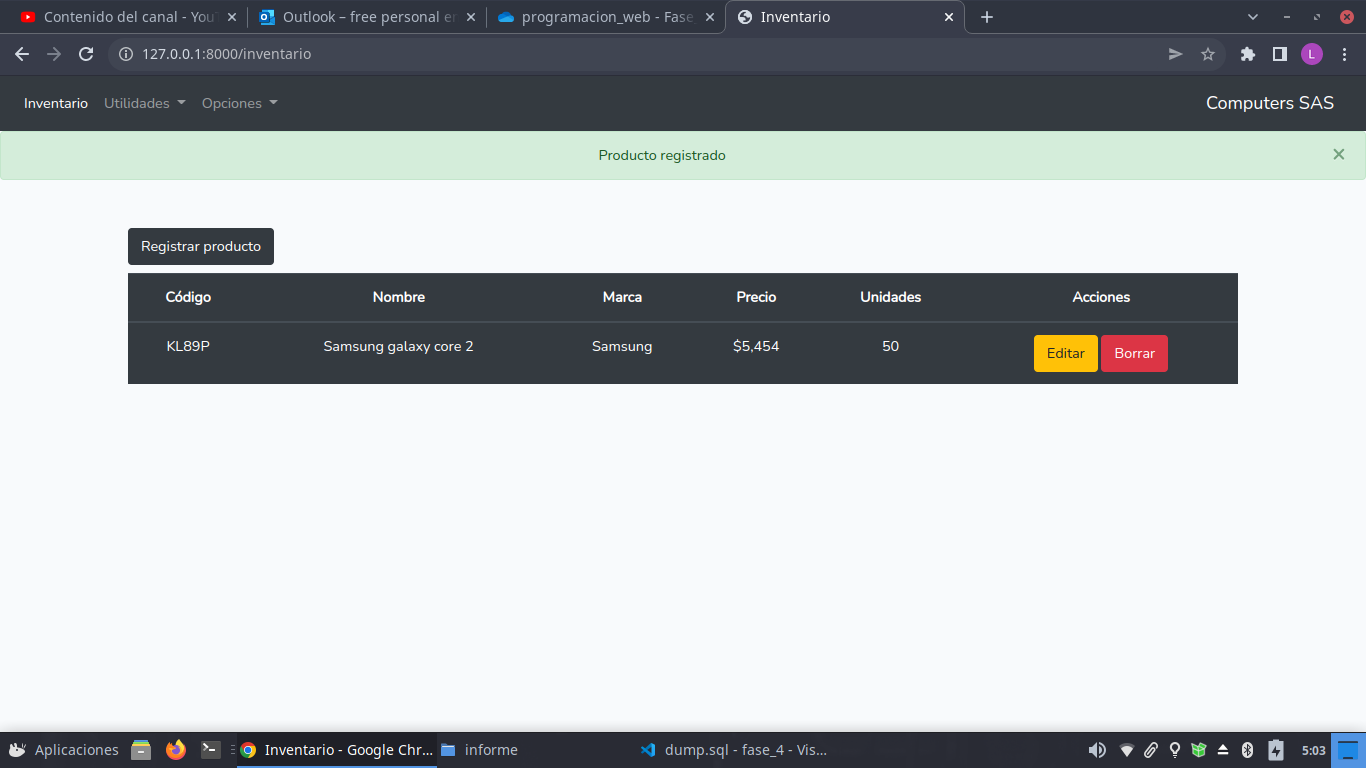
**Delete:**

****

Para eliminar productos se hace desde la tabla que los muestra en el inventario, ahí se encuentra el botón eliminar el cual al llamar a cierta ruta ejecutará el método **destroy** del controlador **ProductoController**. Este recibe el id del registro lo busca en la tabla y crea una instancia del modelo producto con los datos de dicho registro para por ultimo ejecutar su método **delete()** y redirigir a la página principal con el mensaje de éxito.

**Select:**

****

****

El método **select** se usa en el método **index** del controlador **ProductoController**, en este se hace uso del modelo Producto para seleccionar tandas de 5 registros ordenador por fecha de manera descendente para luego cargar la vista principal pasándole dichos registros para que los muestre.

**Resumen de aspectos de seguridad:**

Una de los aspectos de seguridad aplicados fue el uso de la función **prepare** de la clase **PDO** para manipular la base de datos. Dicha función es la recomendada para evitar inyecciones **sql.**

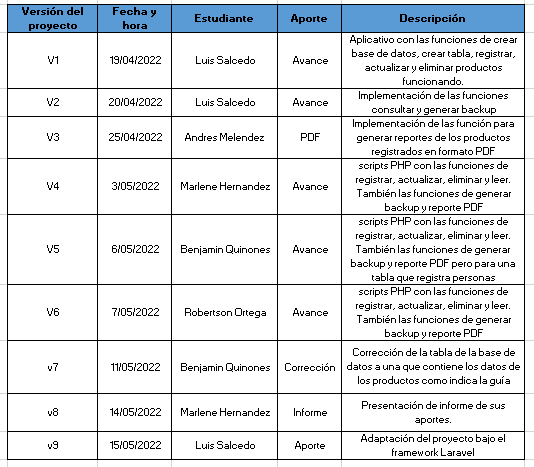
Otro aspecto es el de enviar la información de los formularios a través del método **POST** ya que la información a enviar es sensible y es un fallo de seguridad hacerlo mediante un método como **GET.**

El controlador inventario hace uso de la función **existPost** la cual verifica que se haya recibido la información de los campos de los formularios, necesaria para llevar a cabo sus tareas así como también valida que los campos numéricos recibidos de los formularios sean realmente números, en caso de no cumplirse con los criterios antes mencionados se redirige al usuario a la página principal y se le muestra un mensaje de error informándole porque no se pudo completar su petición.

**Vulnerabilidad Cross Site Request Forgery:** Este ataque fuerza al navegador web de su víctima, validado en algún servicio (como por ejemplo correo o home banking) a enviar una petición a una aplicación web vulnerable.

Esta aplicación se encarga de realizar la acción elegida a través de la víctima, debido que la actividad maliciosa será procesada en nombre del usuario logueado. Al contrario de los ataques conocidos como Cross Site Scripting (su traducción sería ordenes en sitios cruzados – XSS) los cuales explotan la confianza del usuario para con un sitio particular; el Cross Site Request Forgery explota la confianza que un sitio web tiene en un usuario particular. Para evitar esta vulnerabilidad cada petición de un usuario debería estar firmada con un token único el cual se verifica al momento de recibir la petición del usuario y si todo concuerda se procesa, sino se ignora.

**Tabla de control de cambios:**



**Demostración del funcionamiento**

<https://youtu.be/VXuMobhKUaE>

**REFERENCIAS**

Hilarion, F. J. (2016). Conexión Base de datos php. [Objeto\_virtual\_de\_Informacion\_OVI]. Repositorio Institucional UNAD. <https://repository.unad.edu.co/handle/10596/9096>.

Hilarión, F. J. (2020). Librería reportes pdf. [Objeto\_virtual\_de\_Informacion\_OVI]. Repositorio Institucional UNAD. <https://repository.unad.edu.co/handle/10596/38624>.

Hilarion, F. J. (2016). Funciones para el manejo de archivos planos en php. [Objeto\_virtual\_de\_Informacion\_OVI]. Repositorio Institucional UNAD. <https://repository.unad.edu.co/handle/10596/9097>.

Hilarion, F. J. (2018). Lenguajes de bases de datos. [Objeto\_virtual\_de\_Informacion\_OVI]. Repositorio Institucional UNAD. <https://repository.unad.edu.co/handle/10596/21625>.