Práctica: sistema de toma de pedidos para restaurante

Servidor - Joaquín Lorca Nieto

****

C.E.S. Vega Media

Desarrollo de Aplicaciones Web – 2º D.A.W.

**Índice**

[- **Sprint 1** 2](#_Toc190346639)

[o Instrucciones configuración y ejecución del proyecto en local. 2](#_Toc190346640)

[o Carpeta views. 2](#_Toc190346641)

[o Carpeta controllers. 2](#_Toc190346642)

[o Carpeta models. 2](#_Toc190346643)

[o Archivos de configuración. 3](#_Toc190346644)

[o Vista parte pública. 3](#_Toc190346645)

[o Vista parte de autenticación. 4](#_Toc190346646)

[o Vista parte privada. 4](#_Toc190346647)

[- Sprint 2 5](#_Toc190346648)

[o Seeders. 5](#_Toc190346649)

[o CRUD de categorías. 6](#_Toc190346650)

[o CRUD de productos. 10](#_Toc190346651)

[o Integración con vistas públicas. 15](#_Toc190346652)

[- Sprint 3 18](#_Toc190346653)

[o Modelo y migración de anuncios. 18](#_Toc190346654)

[o CRUD de anuncios. 19](#_Toc190346655)

[o Vista pública de anuncios. 23](#_Toc190346656)

[o Notificaciones por Telegram. 24](#_Toc190346657)

# **Sprint 1**

## Instrucciones configuración y ejecución del proyecto en local.

Primero clonamos el repositorio de GitHub con esta URL: <https://github.com/joaquinln22/DW_EntServidor.git>

Una vez hemos clonado el repositorio extraemos los archivos de la carpeta comprimida **CartaRestaurante.rar**.

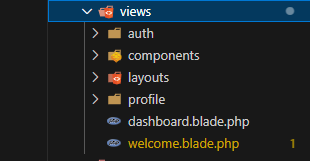
Después, movemos la carpeta que hemos extraído previamente y la situamos dentro de la carpeta de “**/htdocs**” de nuestro Xampp.

Ahora, ejecutamos Xampp y creamos una base de datos vacía desde <http://localhost/phpmyadmin/>. Lo siguiente es asegurar que el archivo **.env** está modificado con el nombre de la base de datos correspondiente, debe aparecer la siguiente línea de código “**DB\_DATABASE=carta\_restaurante**”.

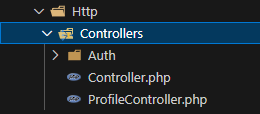
Comprobado lo anterior, abrimos la terminal de Windows PowerShell como administrador e introducimos el siguiente comando “cd C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel” para direccionarnos dentro de dicha carpeta. Entonces, introducimos el comando “php artisan migrate” para ejecutar **las migraciones de base de datos** y crear las tablas de la base de datos.

Con todo esto ya podríamos ejecutar el proyecto debidamente.

## Carpeta views.

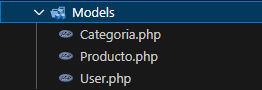


## Carpeta controllers.



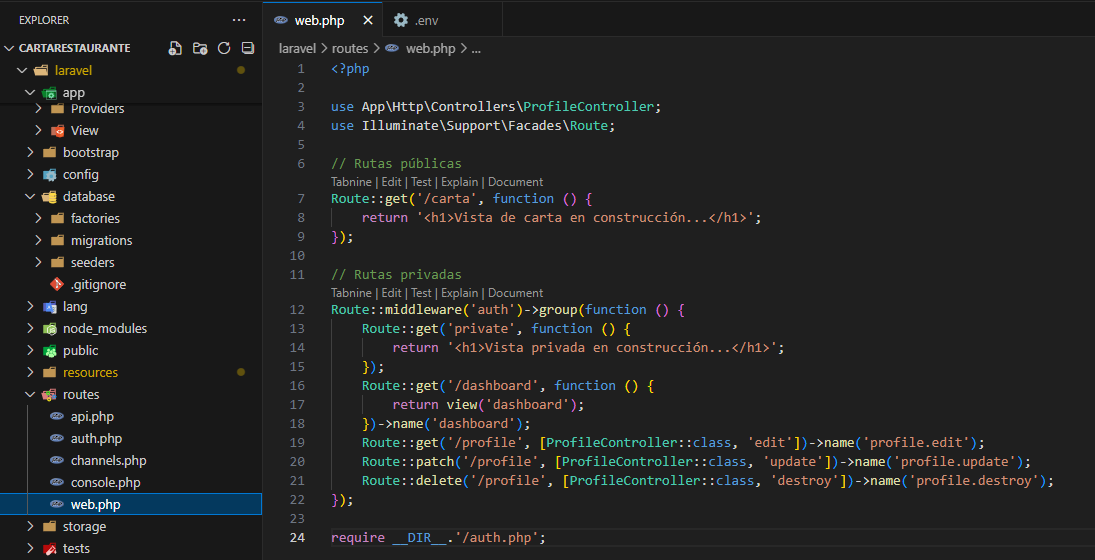
## Carpeta models.

El modelo User.php viene generado por defecto, los otros dos los he creado con el comando “php artisan make:model nombre\_modelo -m”.

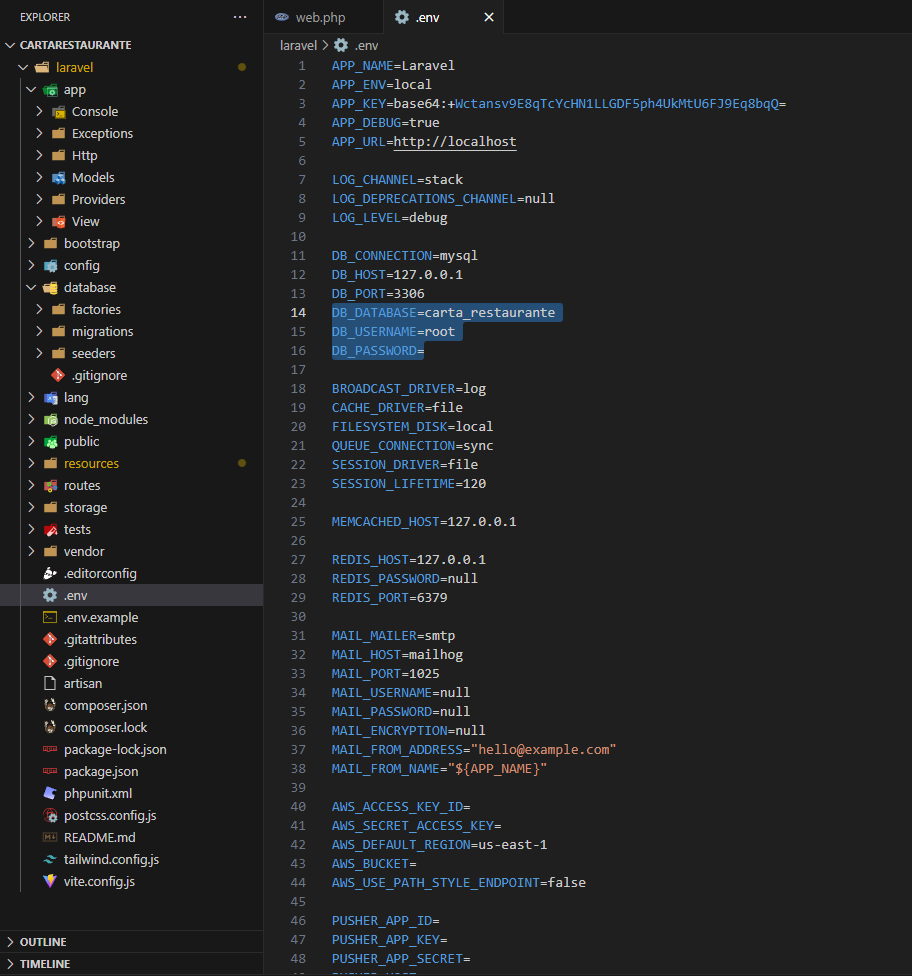


## Archivos de configuración.

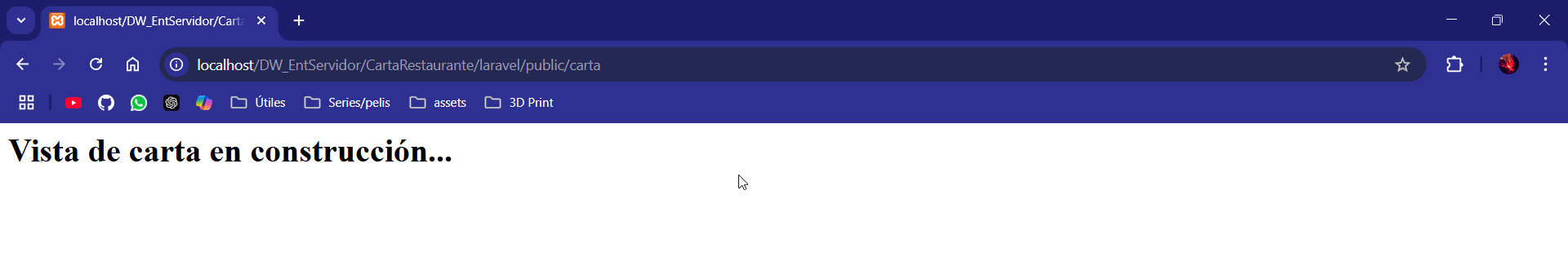
Archivo **web.php**. He creado una ruta pública que nos muestra el texto “Vista de carta en construcción...” y otra ruta de tipo middleware, donde insertar el resto de rutas que quiero dentro de la parte privada de la web. Excepto la ruta “private”, las demás rutas privadas han sido creadas por defecto gracias a Breeze, estas son rutas que nos muestran el panel de perfil.



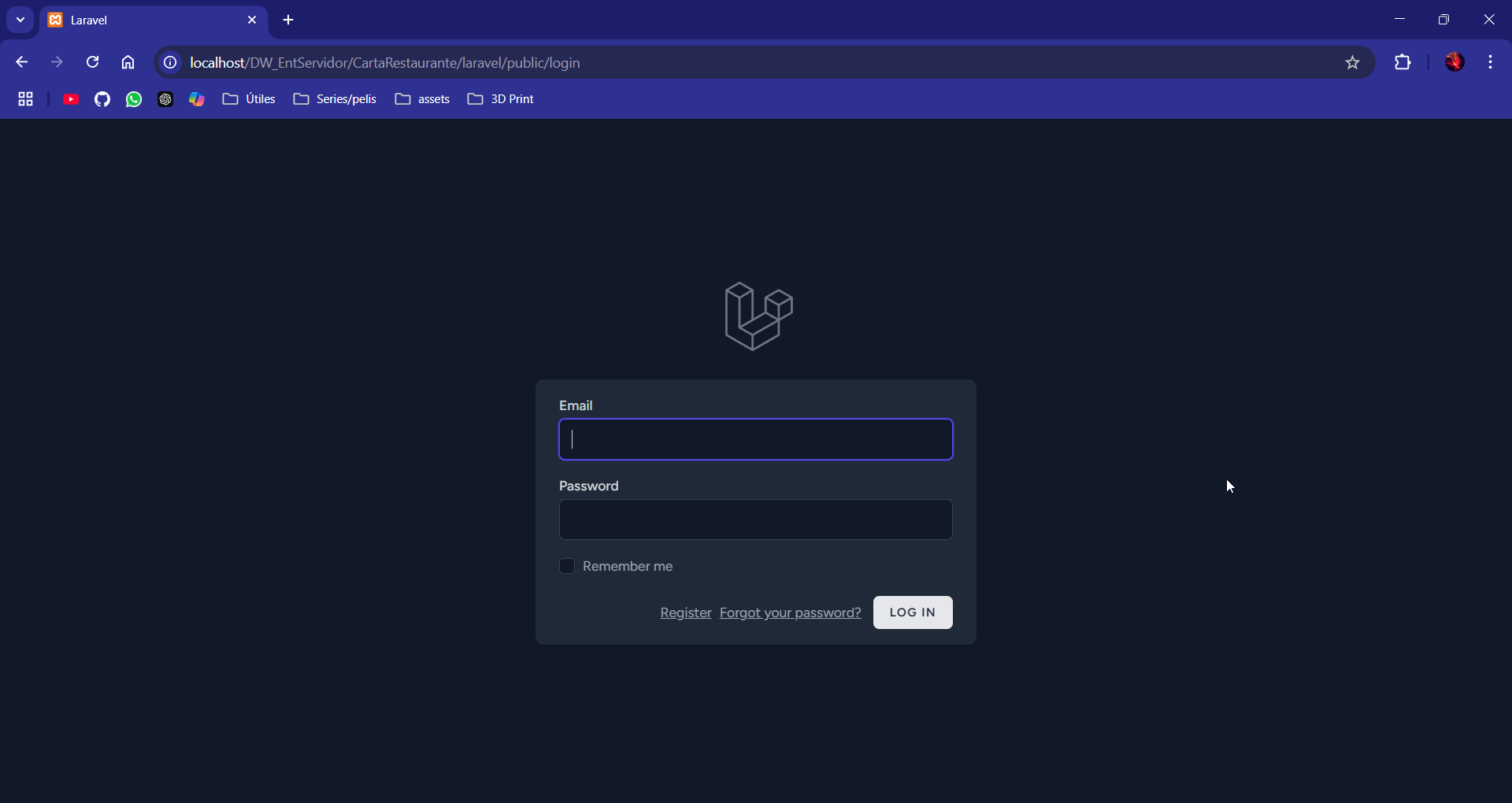
Archivo **.env**. He puesto el nombre de la base de datos en la línea 14 y el resto del archivo lo he dejado por defecto.



## Vista parte pública.



## Vista parte de autenticación.



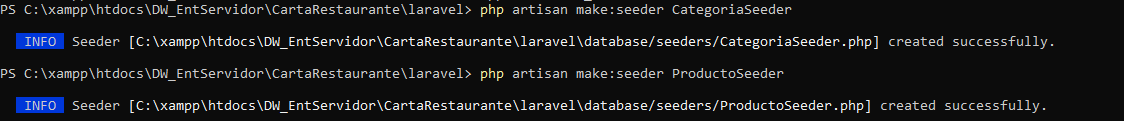
## Vista parte privada.



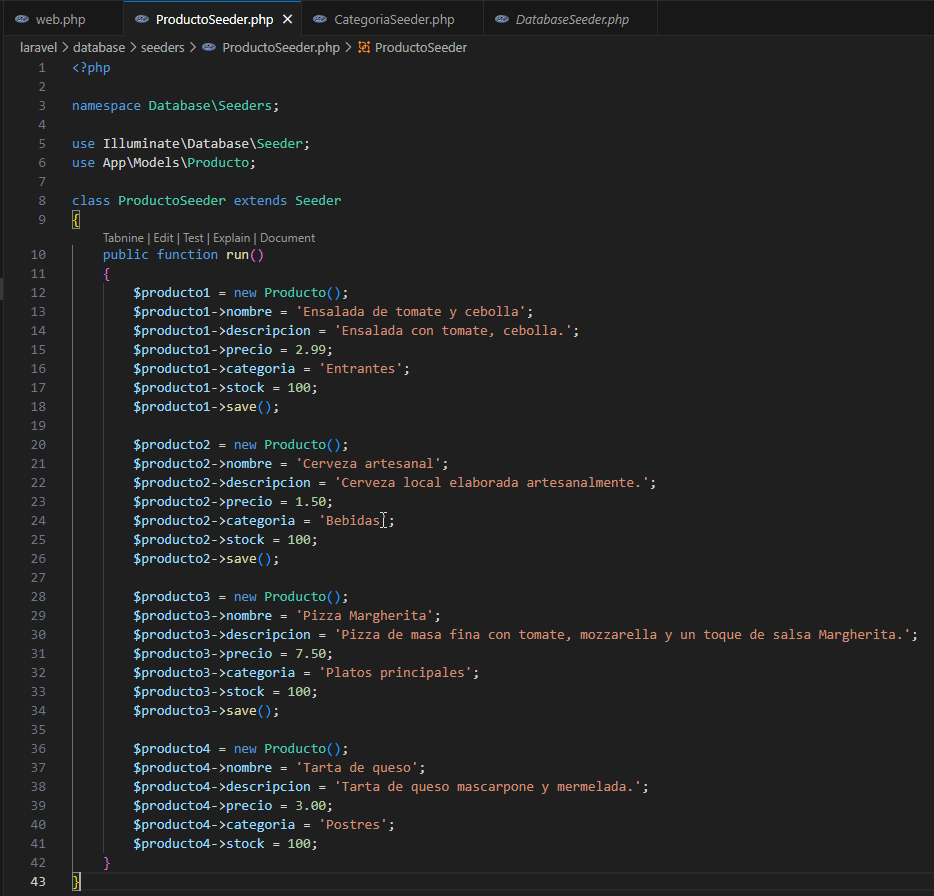
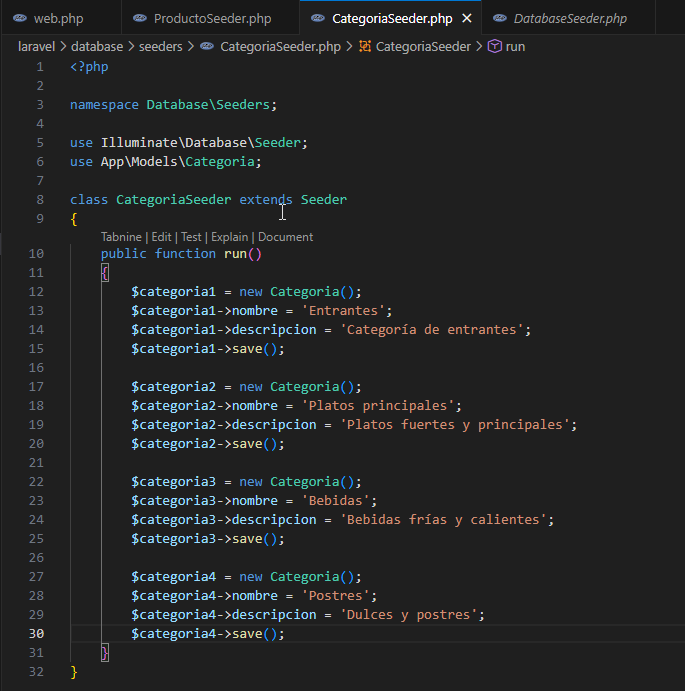
# Sprint 2

## Seeders.

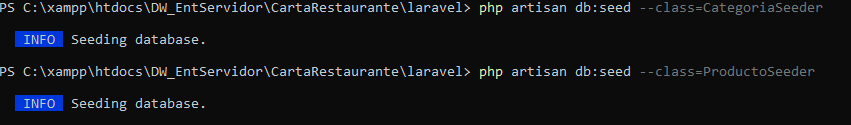
Primero creamos los seeders:



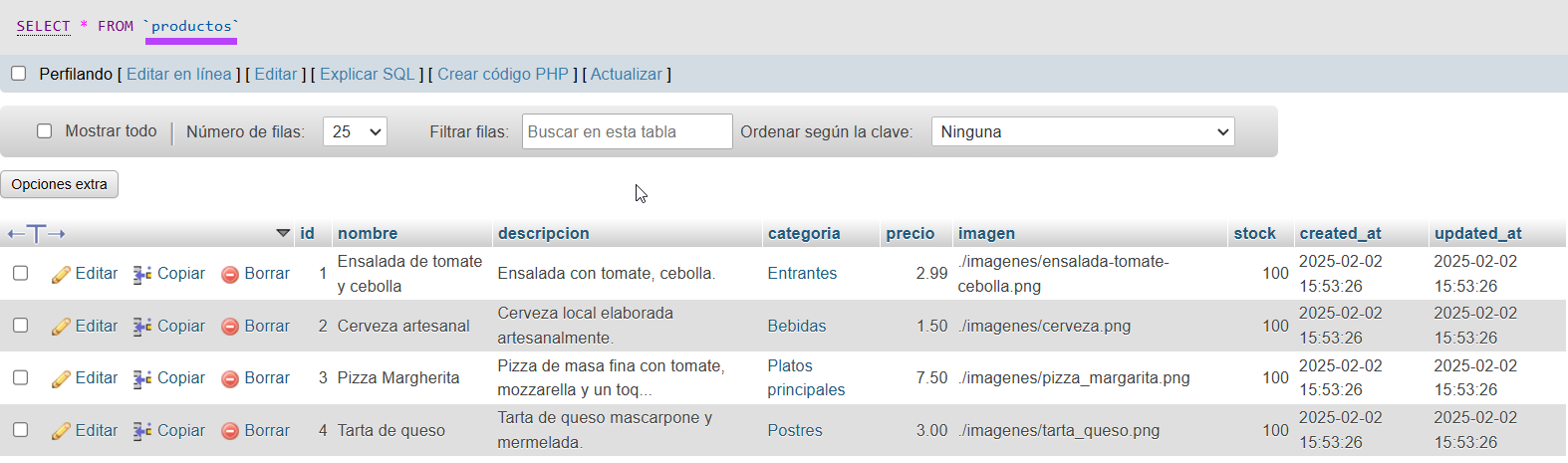
Después editamos los archivos CategoriaSeeder.php y ProductoSeeder.php:



Y ahora ejecutamos los seeders:



Los seeders nos permiten insertar datos a nuestras tablas en la base de datos para, durante el desarrollo de la web, facilitar la ejecución de pruebas y la configuración de datos predefinidos. Así se verían nuestras tablas una vez configurados los seeders:

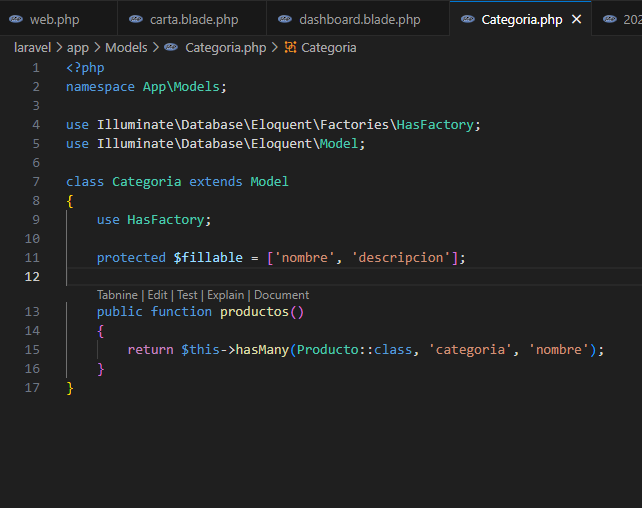
 

## CRUD de categorías.

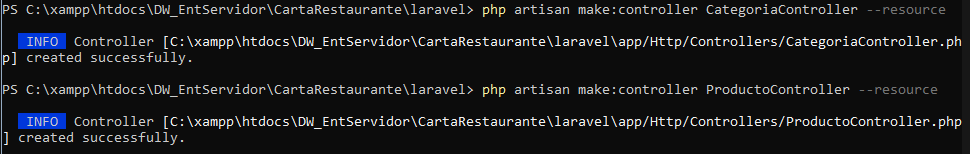
Primero configuramos la ruta de la vista en el archivo web.php:



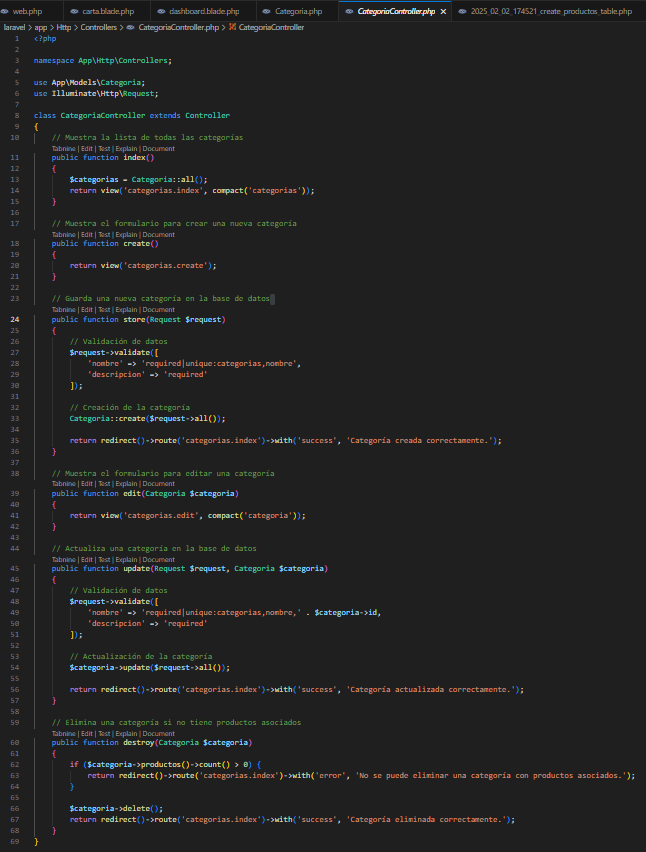
Después editamos el modelo de Categoría.



Después generamos el controlador con el siguiente comando:

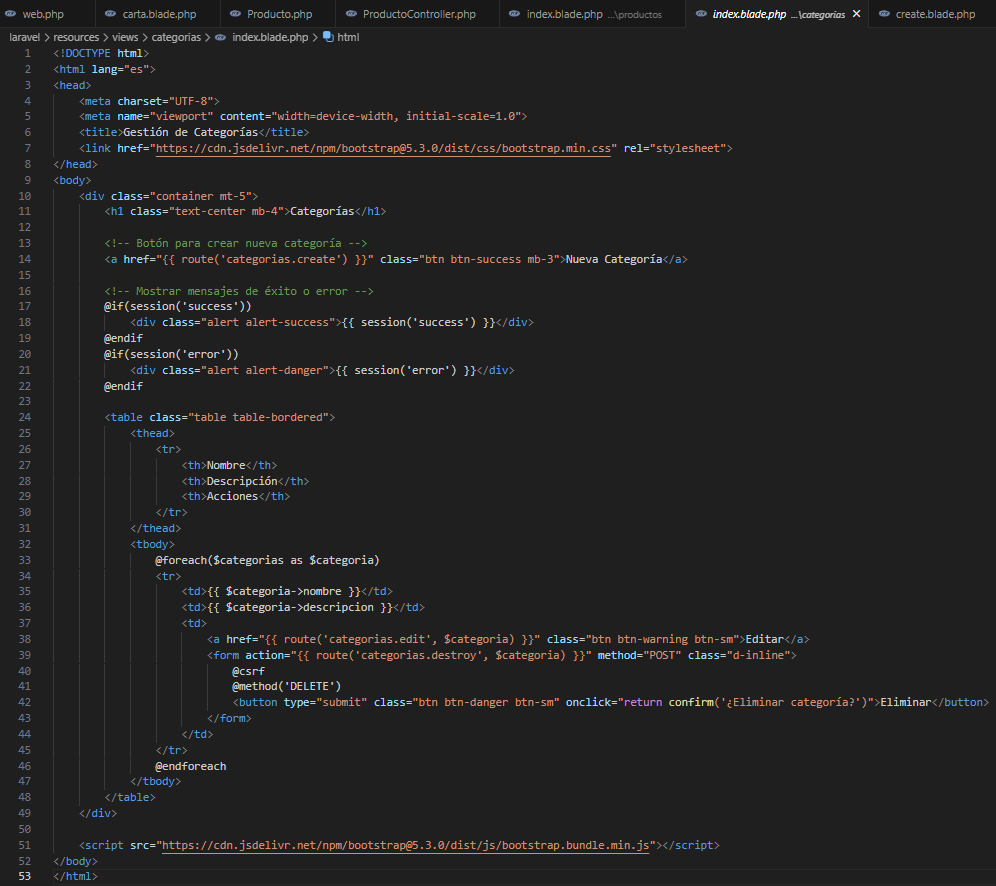


Editamos el archivo CategoriaController.php:



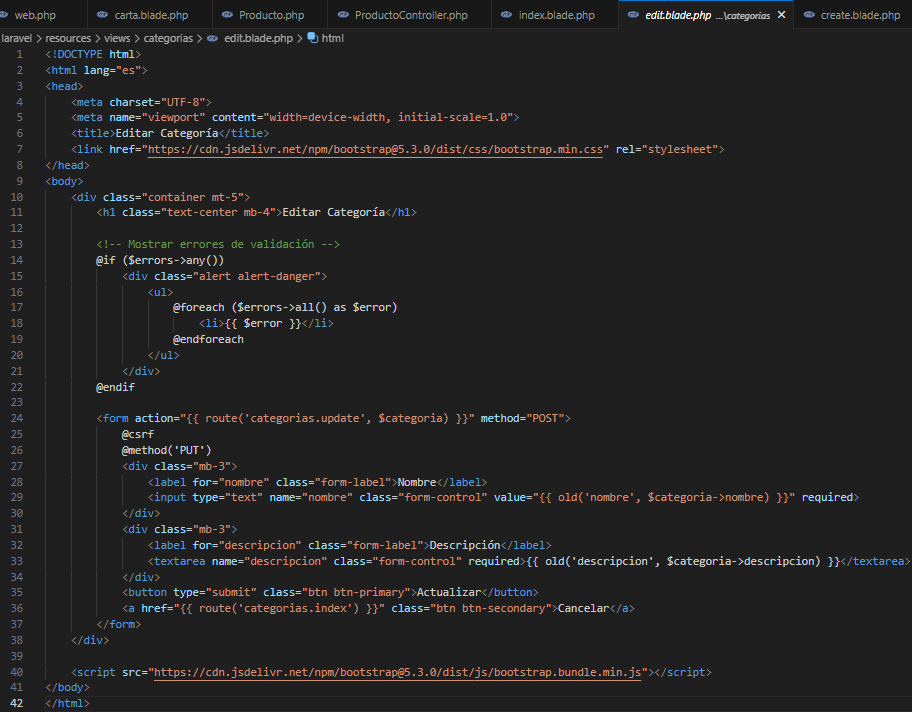
Y, por último, generamos las vistas del CRUD, que serán tres. Para ello creamos la carpeta categorías en la ruta “C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel\resources\views\”. Y dentro de esta, creamos tres archivos:

-index.blade.php (código y vista)



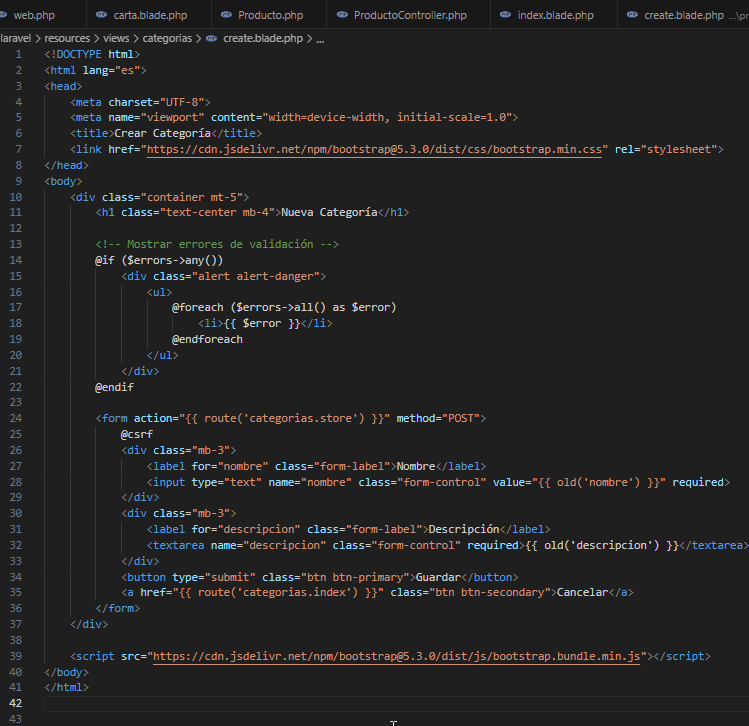


-edit.blade.php (código y vista)





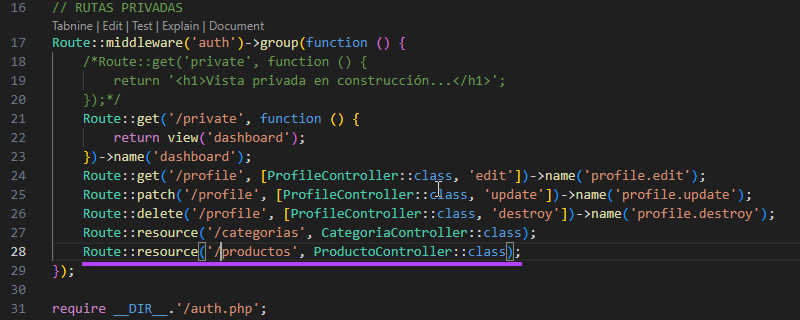
-create.blade.php (código y vista)



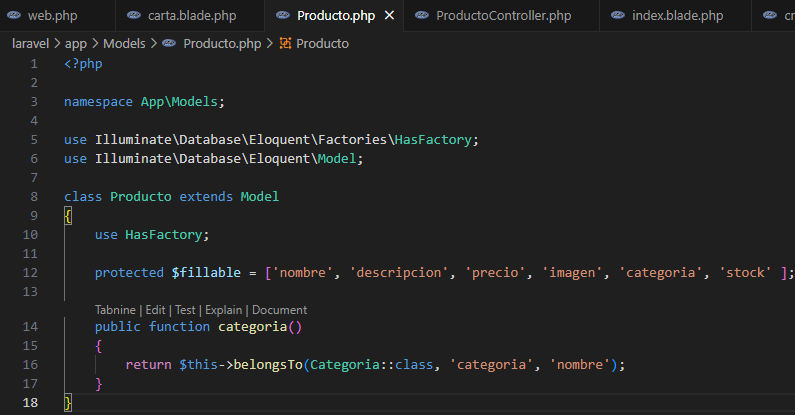


## CRUD de productos.

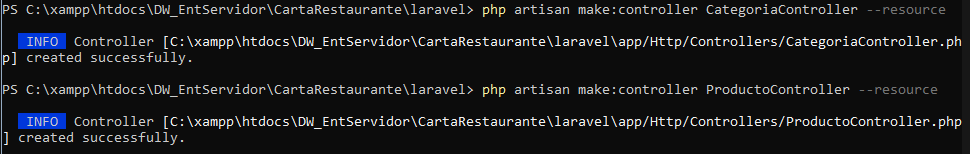
Al igual que antes, primero configuramos la ruta de la vista en el archivo web.php:



Editamos el modelo de Producto.



Después generamos el controlador con el siguiente comando:



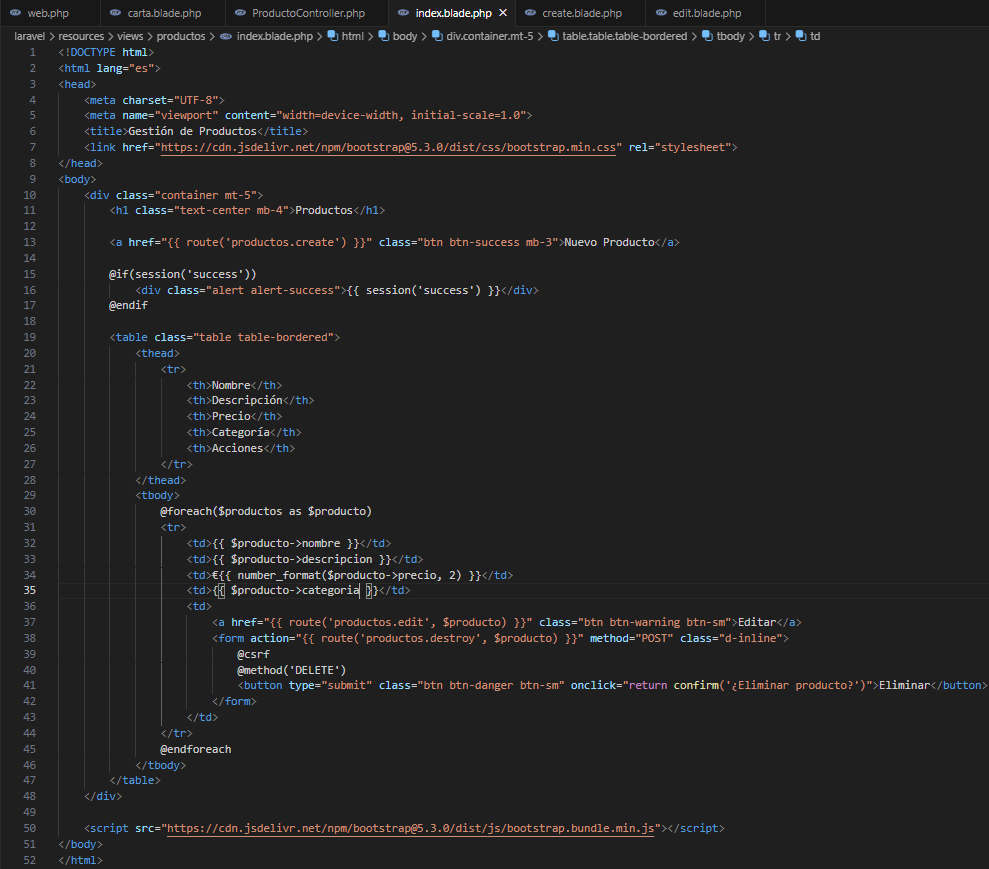
Editamos el archivo ProductoController.php:





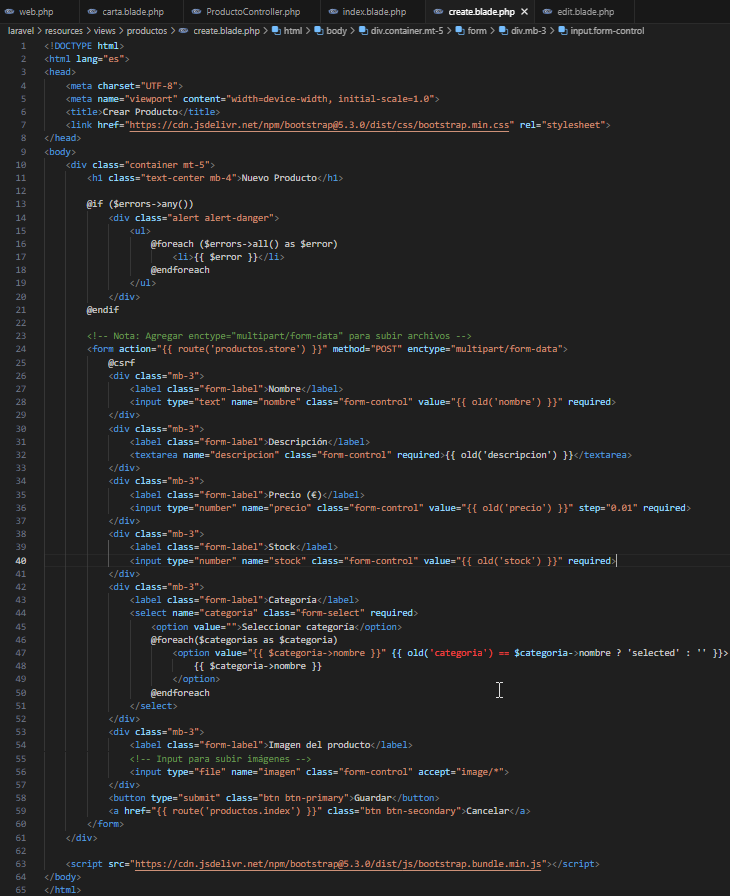
Y, por último, generamos las vistas del CRUD, que serán tres. Para ello creamos la carpeta productos en la ruta “C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel\resources\views\”. Y dentro de esta, creamos tres archivos:

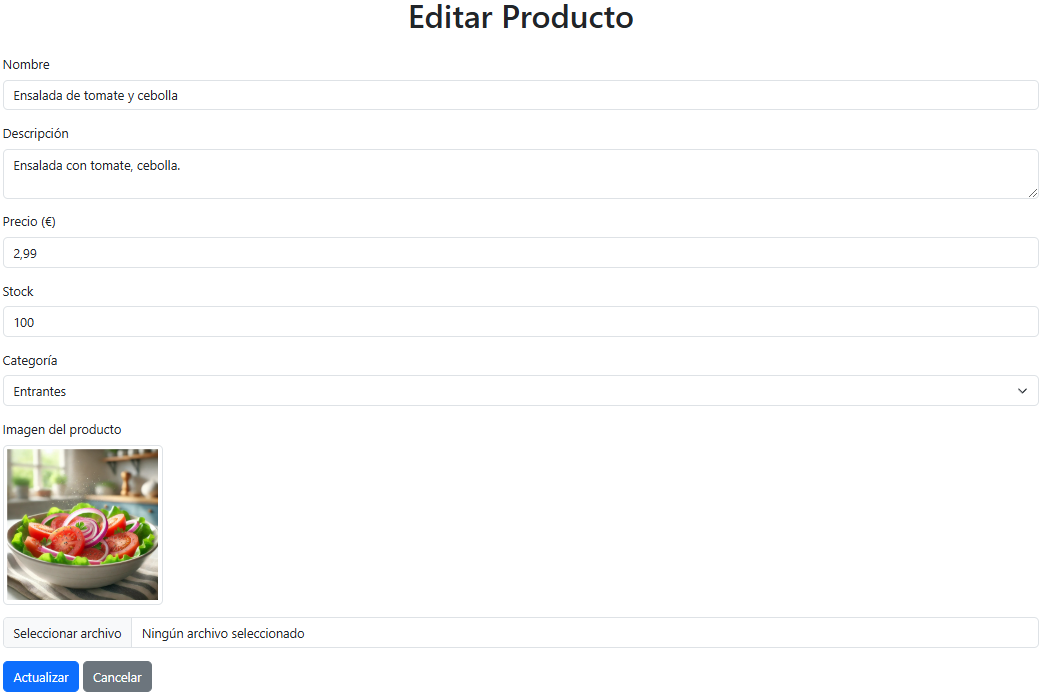
-index.blade.php (código y vista)



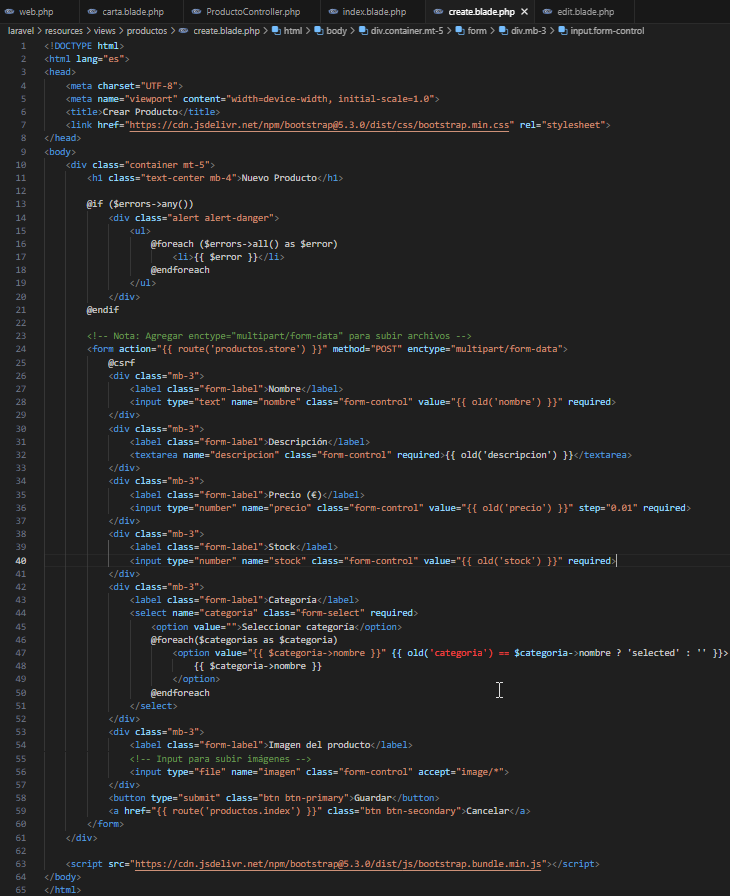


-edit.blade.php (código y vista)





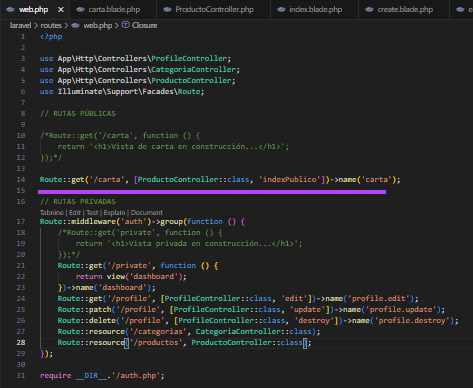
-create.blade.php (código y vista)





## Integración con vistas públicas.

Para mostrar la carta en la vista pública, primero definimos la ruta en web.php:

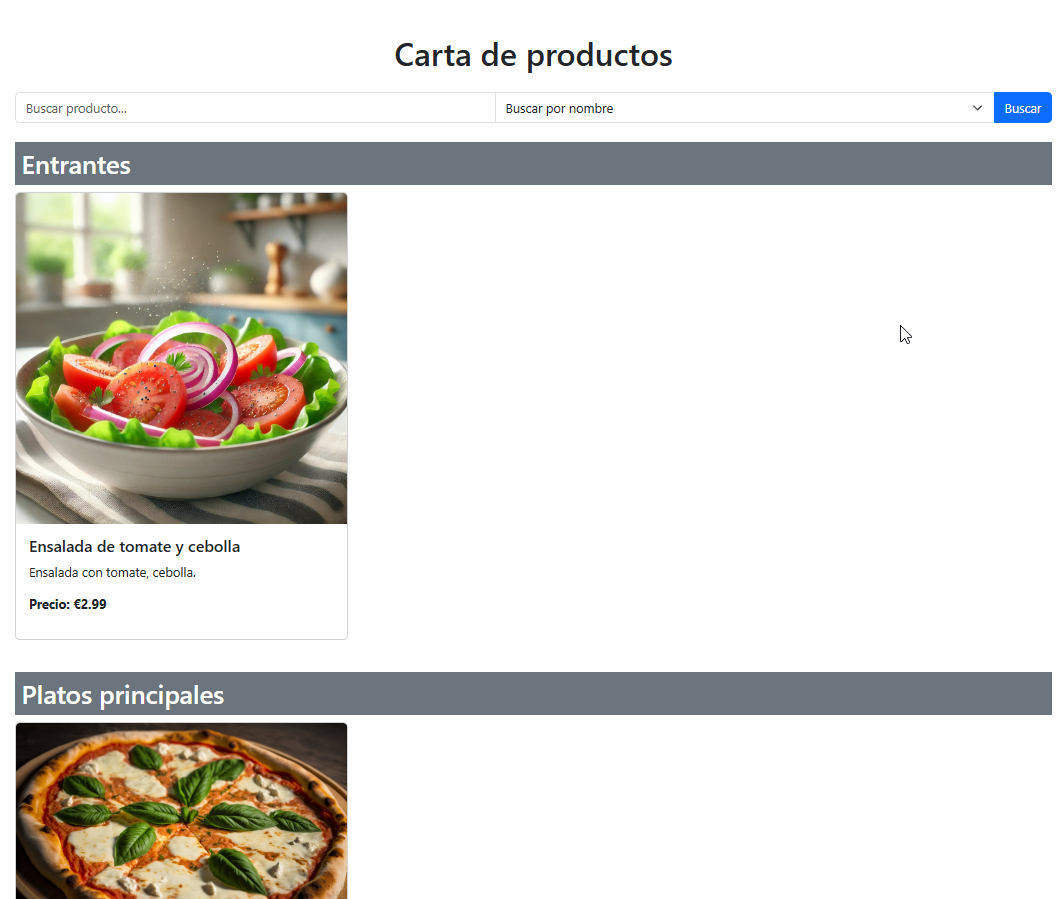


Después, editamos nuestro archivo ProductoController.php y dentro de la clase ProductoController escribimos la siguiente función:



Acto seguido, creamos la vista pública, carta.blade.php, que se vería así (código y vista):

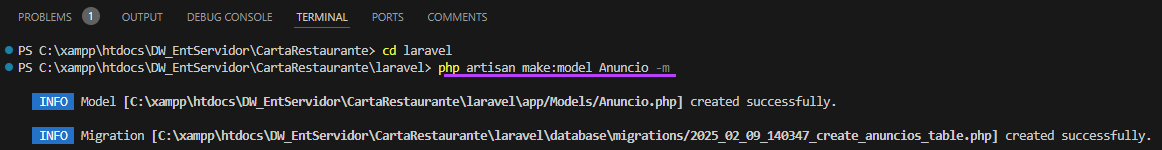




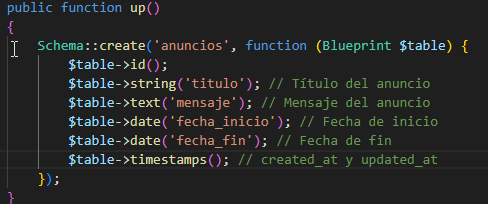
# Sprint 3

## Modelo y migración de anuncios.

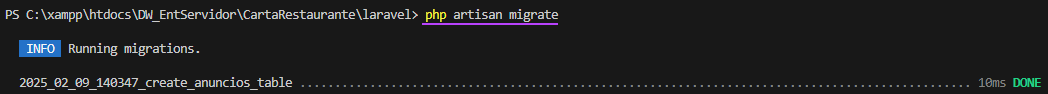
Ejecutamos “php artisan make:model Anuncio -m” en la terminal desde la ruta “C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel” para crear el modelo Anuncio con la correspondiente migración. Deberiamos ver algo así:



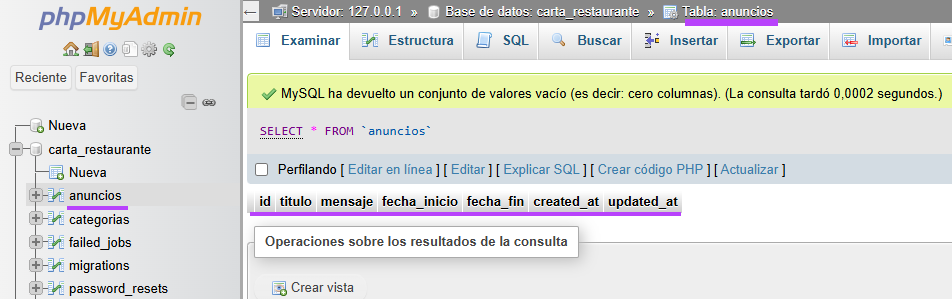
Ahora creamos la tabla en la migración. Modificamos solo la función “up()”:



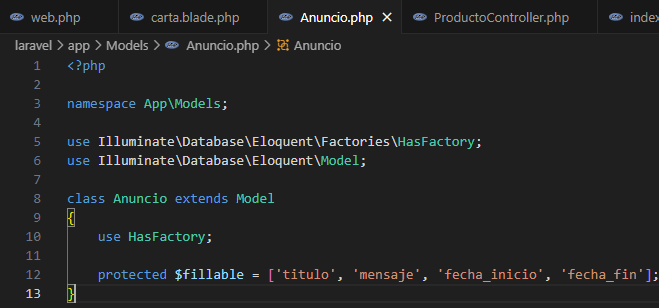
Entonces, ejecutamos la migración:



En este punto comprobamos que se ha creado la tabla adecuadamente:



Y, por último, configuramos el modelo “Anuncio.php” para poder interactuar con la base de datos:

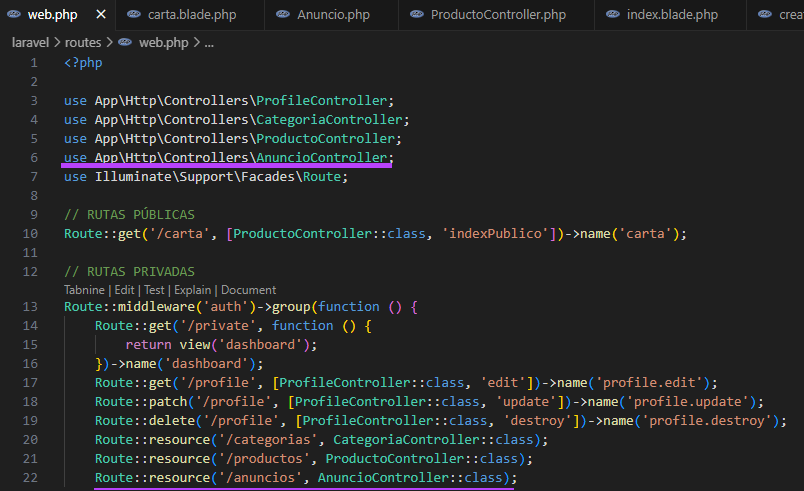


## CRUD de anuncios.

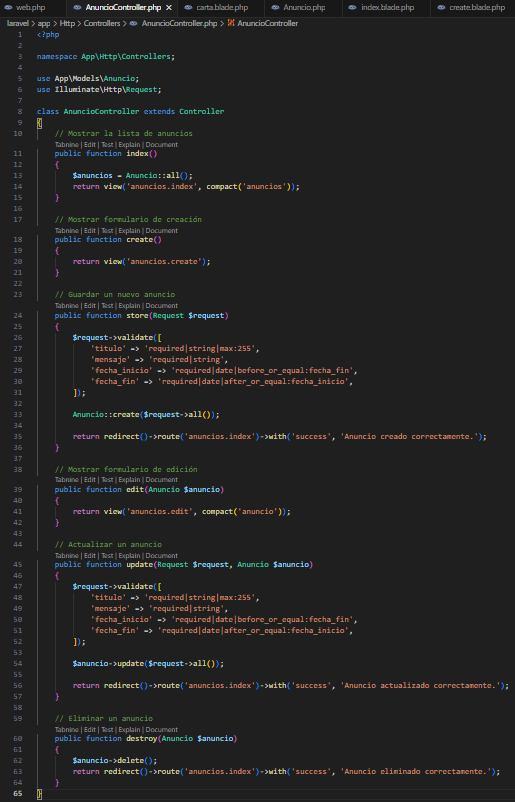
Primero creamos el controlador con las funciones básicas:

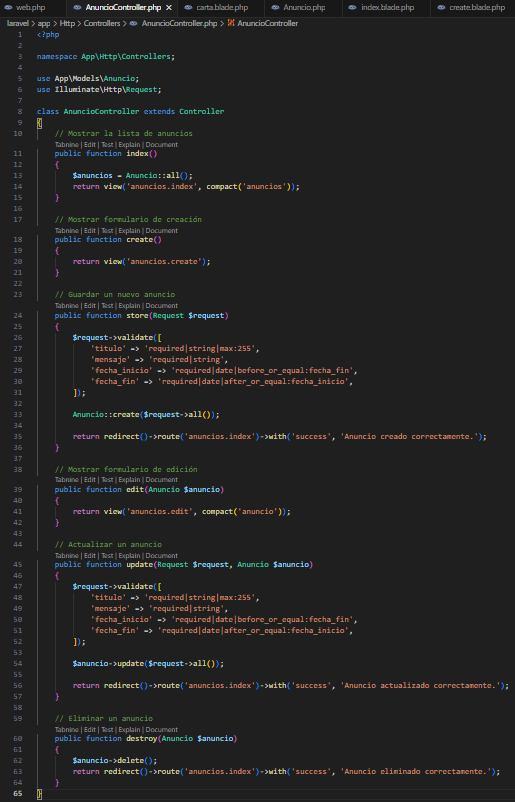


Ahora, agregamos las rutas en “web.php” para gestionar los anuncios:



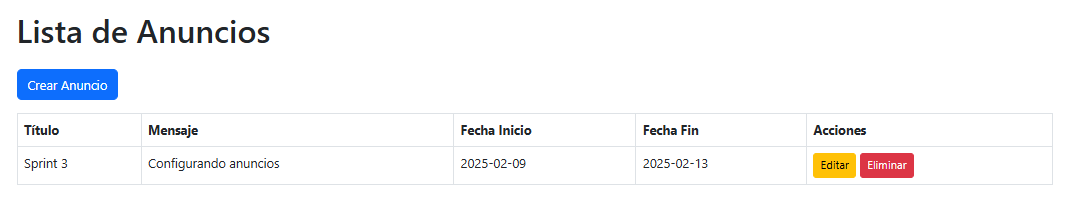
Ahora, abrimos el controlador de anuncios “AnuncioController.php” y editamos el código para implementar el CRUD:

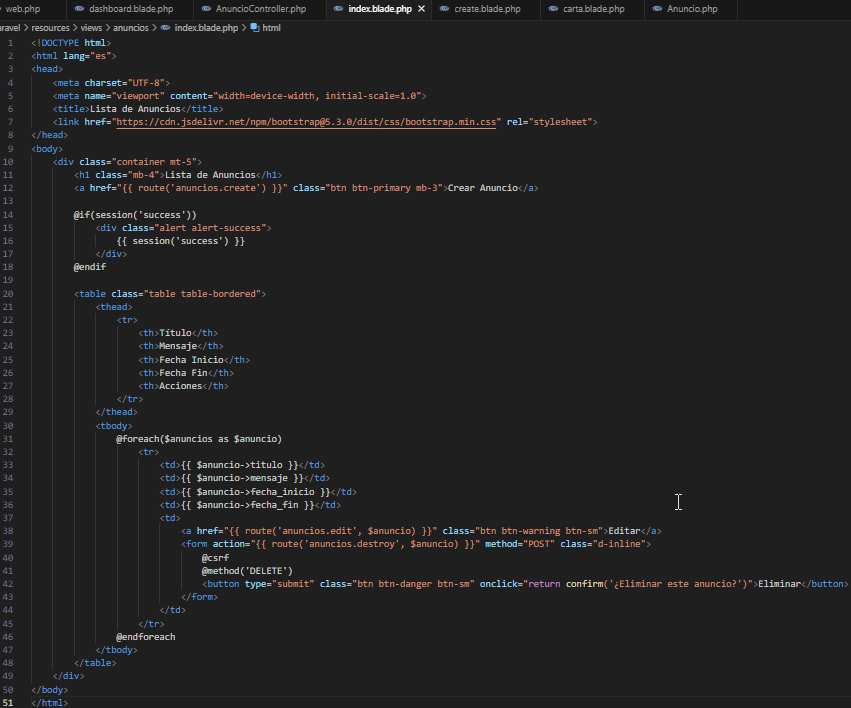




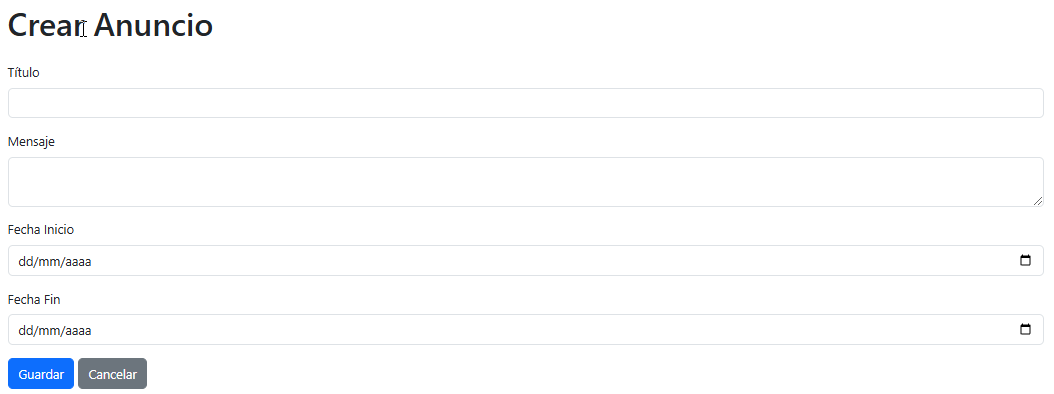
Por último, creamos las vistas privadas necesarias para el CRUD. Estas vistas se guardan en “C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel\resources\views\anuncios”:

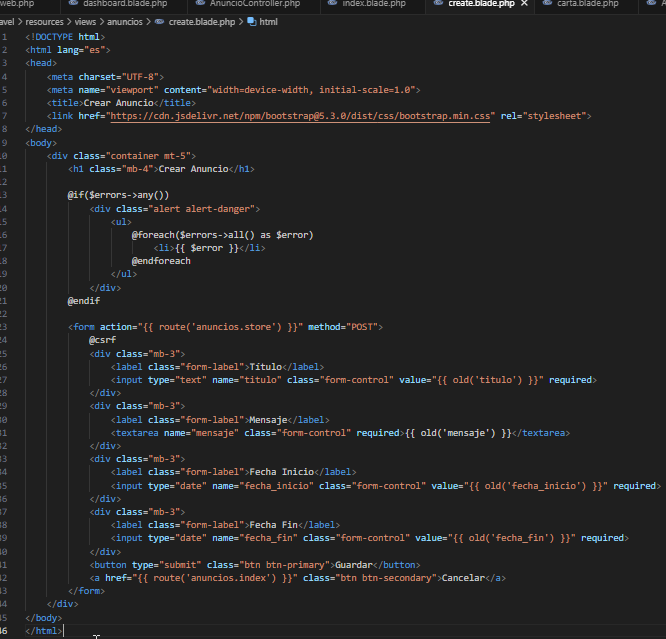
-index.blade.php (vista y código):

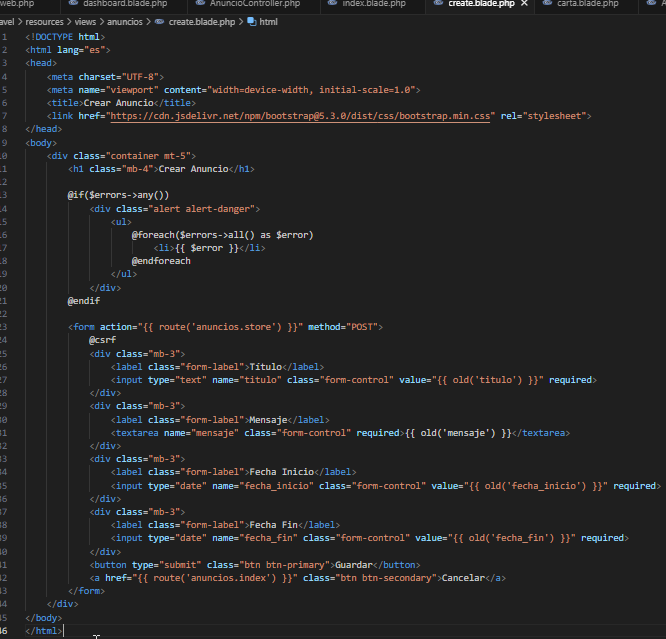




-create.blade.php (vista y cógido):

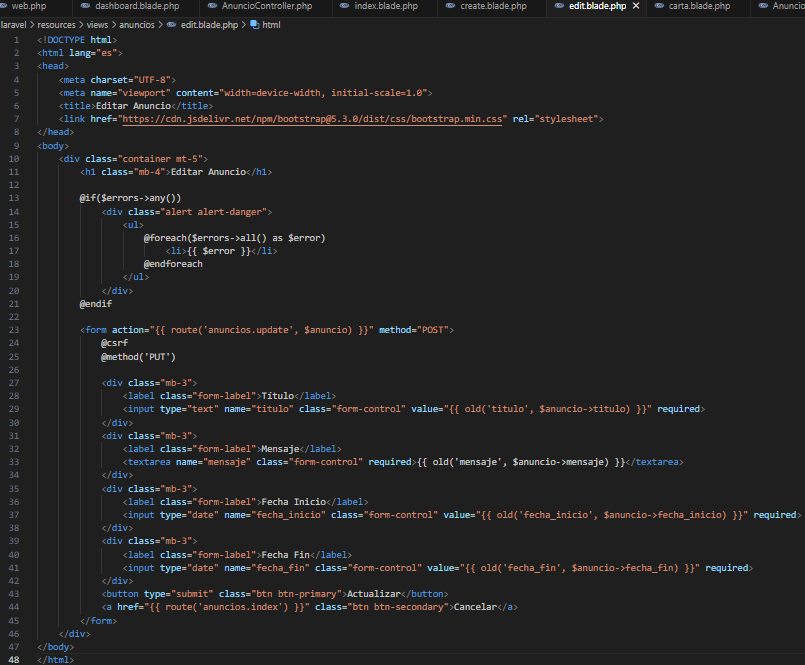






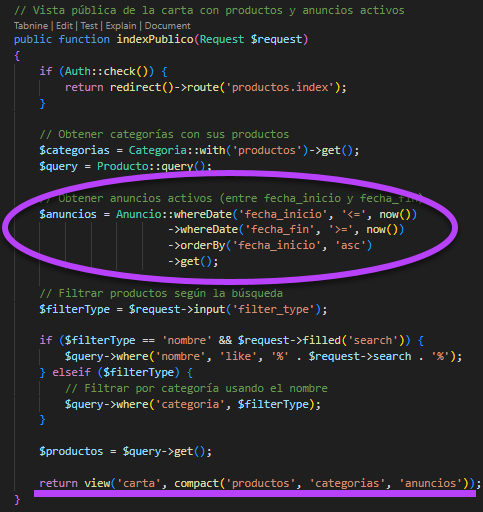
-edit.blade.php (vista y código):



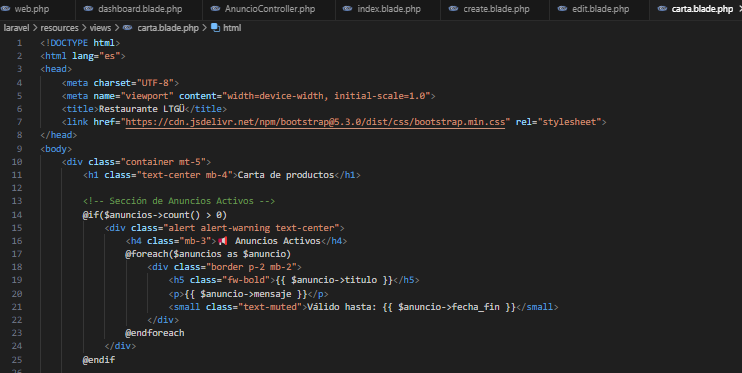


## Vista pública de anuncios.

Para implementar la vista pública para mostrar los anuncios activos en la parte superior de la carta, comenzaremos con modificar el archivo “ProductoController.php” para implementarla siguiente función que nos permitirá obtener anuncios activos filtrados según fecha\_inicio y fecha\_fin, en relación a la fecha actual:



Ahora modificamos “carta.blade.php” para editar la vista pública:

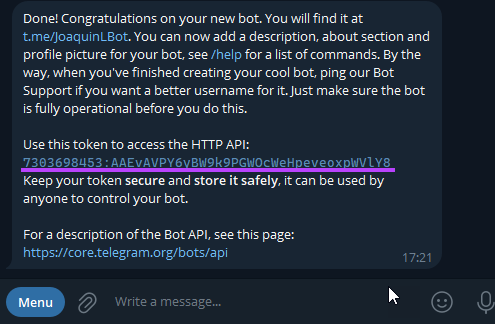


Así se vería la vista de carta:



## Notificaciones por Telegram.

El primer paso es crear y configurar el bot de Telegram. Buscaremos BotFather en el buscador de Telegram, y en el chat de este bot escribiremos /newbot. Seguimos los pasos indicados para crear el bot y copiamos el token proporcionado para acceder a la API:



Ahora instalamos los paquetes necesarios para implementar el uso del bot de Telegram en nuestro proyecto:





El token creado por BotFather lo usaremos para nuestro archivo .env:



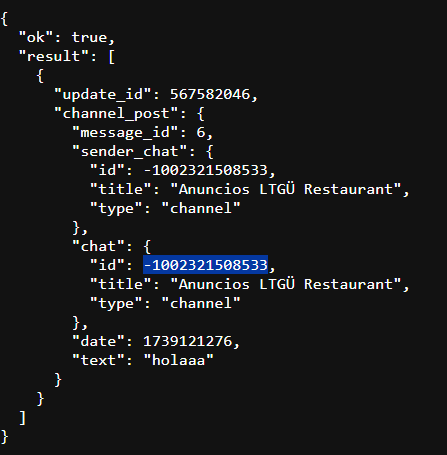
Lo siguiente es crear un canal de Telegram para recibir ahí las notificaciones. En el añadimos como admin al bot JoaquinLBot:



Ahora editamos el archivo de configuración de telegram.php en la siguiente ruta “C:\xampp\htdocs\DW\_EntServidor\CartaRestaurante\laravel\vendor\irazasyed\telegram-bot-sdk\src\Laravel\config”. Añadimos el username y el token generado:



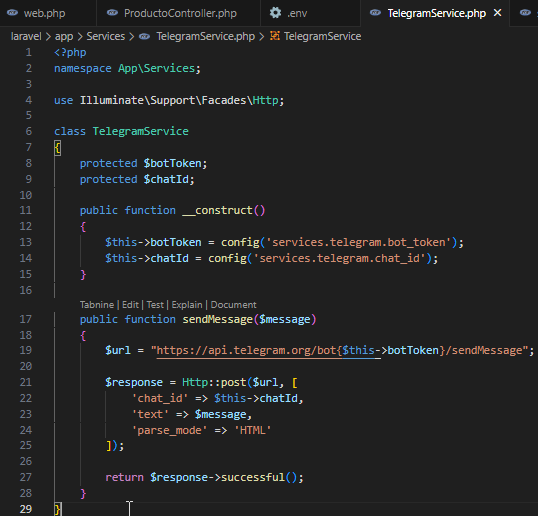
Continuamos con la configuración del bot para que pueda enviar mensajes por el canal creado, para ello necesitamos el ID de chat. Para conseguirlo escribiremos esta URL en el navegador <https://api.telegram.org/bot7303698453:AAEvAVPY6yBW9k9PGWOcWeHpeveoxpWVlY8/getUpdates> y mostrará lo siguiente:



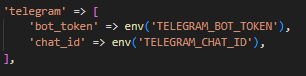
Después, añadimos el siguiente chat\_id al archivo .env y ejecutamos “php artisan config:clear” en la terminal:



Lo siguiente es crear un archivo en “app/Services/TelegramService.php”, creamos la carpeta “Services”, también. Y dentro de dicho archivo escribimos el siguiente código:

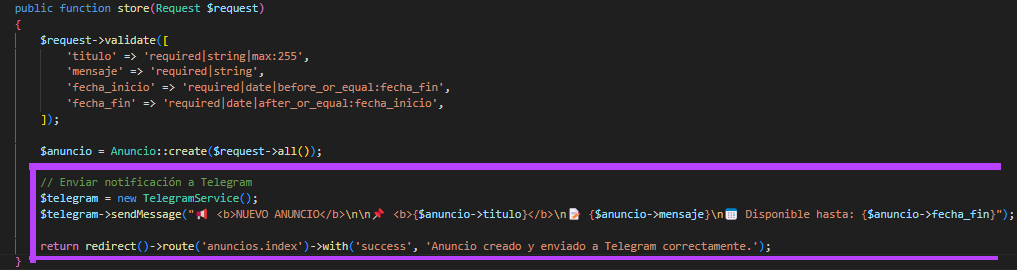


Después, tendremos que configurar el archivo services.php dentro de “config/”, añadiremos lo siguiente:



Por último, en AnuncioController.php, para importar el servicio y enviar el mensaje, agregamos lo siguiente:





Ahora, solo queda probar y verificar que nuestro bot envía un mensaje con nuestro anuncio al canal de Telegram. Creamos el anuncio y automáticamente se envía a nuestro canal.

