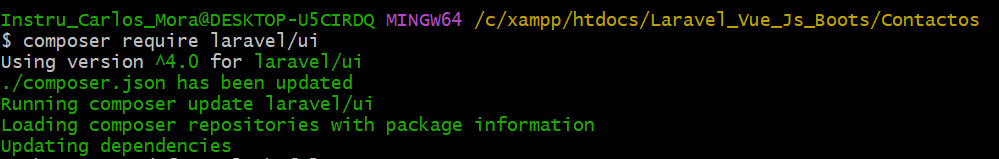
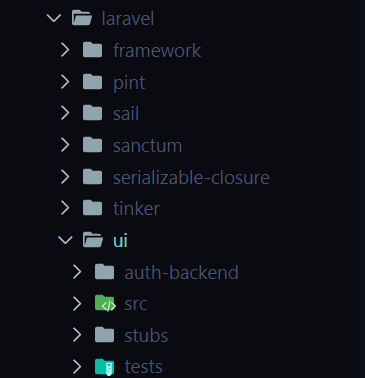
**PROYECTO CONTACTOS CON LARAVEL, VUE Y BOOTSTRAP**

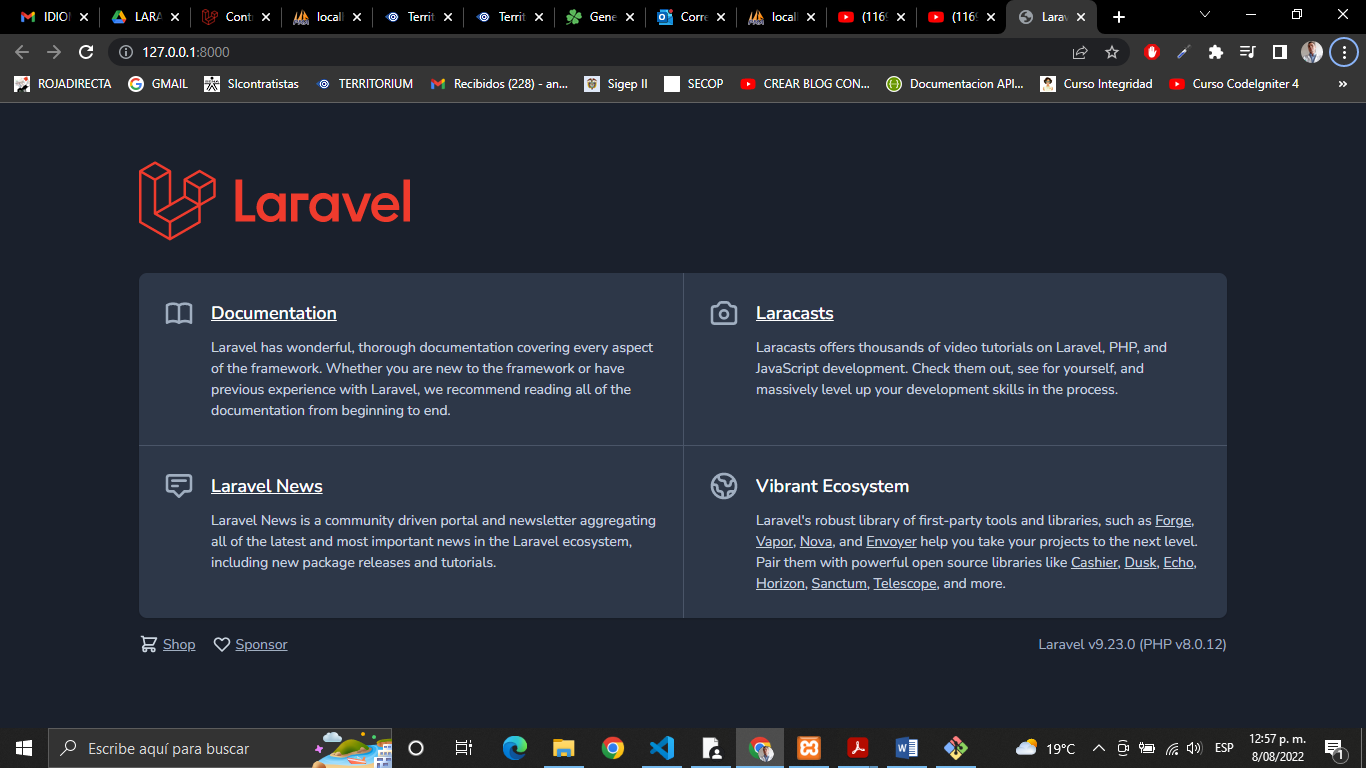
1. Creamos el proyecto Laravel con el nombre Contactos\_Laravel\_vue, en htdocs
2. Verificamos que el Proyecto inicia con normalidad en el servidor de Laravel (php artisan serve).
3. En la terminal ingresamos a la ruta del proyecto con cd Contactos\_Laravel\_vue.
4. Instalamos el módulo de autenticación tradicional de Laravel con el comando composer require laravel/ui



1. Abrimos el proyecto en Visual Studio Code.
2. Verificamos en vendor/laravel que esté la carpeta ui. Esta se creó con el comando del punto 4.



1. La aplicación en el navegador web luce así:



1. Consultamos la documentación oficial de Laravel/ui en GitHub <https://github.com/laravel/ui>
2. De los comandos allí descritos, usaremos los que aparecen resaltados a continuación:

Texto

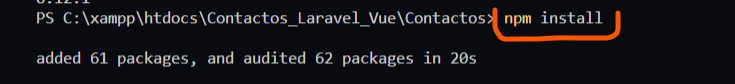
Descripción generada automáticamente

En la terminal se verían más o menos así:

Texto

Descripción generada automáticamente

1. Ejecutamos npm install en la terminal. Advertencia: Se debe tener primero instalado node js para hacer esto. Se puede verificar en la terminal si está instalado con npm -v



1. Por último ejecutamos el comando npm run dev.

Es importante tener dos terminales abiertas, si es posible en el mismo Visual Studio. En una terminal se ejecuta el servidor de backend con el comando php artisan serve y en la otra terminal se ejecuta el comando npm run dev, para que el frontend también esté activo. Si ambos servidores no están corriendo, no se puede usar el módulo de autenticación en Laravel.

1. Ingresamos desde el navegador con la dirección localhost:8000 y damos clic en Login. Deberíamos ver algo como lo siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. En el archivo .env del proyecto definimos cómo queremos que se llame la base de datos. En este ejemplo, se deja el nombre corto Contactos

Texto

Descripción generada automáticamente

1. En phpmyadmin creamos la base de datos con ese nombre.
2. Luego hacemos la migración php artisan migrate, desde la terminal de Visual Studio.
3. Podemos registrar un usuario para comprobar que este módulo funciona.

**Creando el primer Modelo con su migración y controlador**

Vamos a crear un modelo para Contact (ya que como recordará el objetivo es construir una aplicación de Agenda de Contactos, de ahí el nombre Contact), pero además crearemos su respectiva migración (m). Además, crearemos su factory (f) y su controlador de recursos (r). Finalmente, indicaremos que los recursos del controlador sean de tipo API.

1. Para lograrlo, usaremos el siguiente comando:



1. Verificamos que el controlador aparezca en su lugar

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aclaración: Un factory nos permitirá crear datos de prueba o falsos para la base de datos, y así probar con mayor rapidez el funcionamiento de nuestra aplicación.

1. Modificamos el archivo de migración para Contactos, con los siguientes campos:

Texto

Descripción generada automáticamente

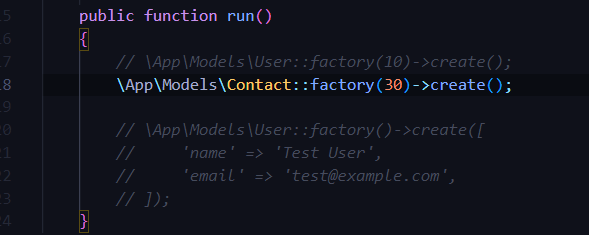
1. Vamos a la ruta database/factories y ubicamos ContactFactory, y especificamos los datos falsos que queremos insertar en la tabla Contatcts así:

Texto

Descripción generada automáticamente

**Aclaración**: Como podemos ver, faker provee una serie de tipos de datos predeterminados (en inglés) que permitirán precisamente llenar con ese tipo de datos de prueba la tabla. Por ejemplo, name() generará nombres al azar, lastName() apellidos al azar, o randomElement ingresará aleatoriamente lo que nosotros especifiquemos, en este caso algunas veces ingresará el teléfono 123456789 y otras veces ingresará 987654321.

1. Debemos llamar estos campos desde el seeder (sembrador), precisamente para lograr insertarlos o sembrarlos en la tabla de la BD. Vamos a la ruta database/seeders y ubicamos DatabaseSeeder.php. Allí codificamos:



El 30 entre paréntesis significa que queremos que se siembren 30 registros de prueba, pero podemos colocar más, sea 40, 50, etc.

1. Guardamos y ejecutamos la migración junto con el proceso de sembrado de datos falsos, con el siguiente comando en la terminal:



El resultado será:

