	Prueba técnica Fullstack para Lirmi.com
@author	Miguel Angel Vázquez Vázquez mvk_12@outlook.com
@version	1.0.0

Escenario

Descripción funcional:

- Se necesita mostrar un listado de usuarios donde se muestre solo su nombre y apellido, luego al hacer click sobre uno de los usuarios se debe abrir un modal donde se muestre su avatar (a la izquierda) y nombre, apellido e email a la derecha del avatar (uno en cada línea respectivamente), luego debajo de todo se debe mostrar un botón "Registrar" que sólo registre el nombre, apellido e email del usuario en una base de datos. Nota: El usuario podría ser registrado más de 1 vez.
- El modal debe contener un botón "Cerrar" o la "X" en la esquina superior derecha para ser cerrado.
- También sobre el listado de usuarios debe existir la opción de encontrar directamente un usuario por su "id", en que si el usuario no existe se abre un modal con el mensaje de error "Usuario no encontrado" y si existe se abre el mismo modal del punto anterior.
- En la primera pantalla, bajo el listado de usuarios obtenidos desde la API, debe existir un enlace a otra vista que muestre los usuarios que se registraron en la base de datos en una tabla que muestre el nombre, apellido e email de cada usuario. Además debe tener un botón "Eliminar" para eliminar el registro del usuario en la base de datos.
    - Si no existen usuarios registrados o se elimina el único usuario de la lista, debe mostrar el mensaje "No existen usuarios registrados".
- Si ocurre un error inesperado, debe mostrar una pantalla con el texto "Lo sentimos, intente más tarde." en medio de la pantalla.

Descripción técnica:

- El proyecto debe ser desarrollado en PHP Laravel 7 y VueJS 2.
- Se deben obtener los 2 primeros listados de usuarios desde la API: "https://reqres.in/api/users?page=X", donde X es el numero de la página a consumir.
- El campo que permite filtrar por ID debe aceptar solo números enteros mayores que 0 y menores que 15.
- Como sugerencia, la base de datos donde se registra cada usuario obtenido desde la API puede ser PostgreSQL, la tabla es "profesores" del esquema "colegio". Solo se almacena su nombre, apellido e email.
- Las pantallas y modales deben ser desarrollados en Blade usando componentes VueJS.

> - En caso de excepción (Error inesperado) debe mostrar la pantalla con el texto definido, sin importar cual sea la excepción.

Información del proyecto

Introducción

El proyecto fue realizado utilizando PHP 7.4.x, Laravel 7.0, VueJS 2 y base de datos Postgres 12.x; tal y como fue solicitado en los requerimientos arriba mencionados.

¿Cómo replicar en ambiente local?

Es necesario contar con las siguientes caracteríctcas:

- PHP 7.x
- Composer
- NodeJS 11 o superior

Pasos a seguir:

1. Acceder al enlace del repositorio https://github.com/mvk12/lirmiTechExam y usar las herramientas preferidas para el clonado del mismo.
2. Realizar la instalación de dependencias:
    1. Ejecutar `composer install` para la instalación de dependencias de Laravel.
    2. Ejecutar `npm i` para la instalación de dependencias de VueJS.
3. Establecer la opciones de configuración colocando los valores de las claves respectivas en los archivos `.env` y `.vue.env`.
    - Si el archivo `.env` no fue creado despues de la instalación de dependencias por composer, copiar el contenido de `.env.example` a `.env`.
    - Para `.vue.env`, el proceso es manual copiando el contenido del archivo `.vue.env.example`.
    - NOTA: El valor de la clave **VUE_APP_BASE_URL** en `.vue.env` debe ser la misma que **APP_URL** en el archivo `.env`.
4. Para ejecutar en modo local el proyecto, debe ser ejecutado el comando `php artisan serve` tal y como lo marca la documentación.
5. Para compilar nuevamente el desarrollo de frontend, debe ser ejecutado el comando `npm run dev`.

Notas sobre el desarrollo

- La API para consumo de usuarios "reqres.in/api/users?page=X" realiza una paginación de los recursos que expone, por lo cual fue añadido un control de selección de página para cumplir el criterio de consumo de las primeras dos páginas.

- El criterio funcional: *...sobre el listado de usuarios debe existir la opción de encontrar directamente un usuario por su "id"...* no puede ser cubierto sin violar otro requerimiento técnico: *El campo que permite filtrar por ID debe aceptar solo números enteros mayores que 0 y menores que 15.*.

La API de consumo otorga la búsqueda de un registro dado su id por lo que se le da prioridad al criterio funcional.

- Todos las las tablas presentadas presentan paginación y carga asíncrona de información, de manera que se observarán leyendas como *Cargando...* mientras se resuelve la descarga de información.
- El criterio técnico: *Las pantallas y modales deben ser desarrollados en Blade usando componentes VueJS* considero que no pude cubrirlo al cien por cierto dado que me causó ambigüedad sobre la implementación.

  Esto es, realizar todos los componentes incrustados como parte de la página usando Blade pero sacrificar el uso amplio del lenguaje javascript, o en este caso realizado, separar completamente del frontend y backend.
- Se realizaron solo unos ajustes visuales para brindar mejor lectura e identificación de cambios en la interfaz de usuario.