



Universidad Católica

Memoria

---

# Manual de Desarrollador

---

SIAP

(Sistema de Agenda e Información para Psicóloga)

Luis Caroca Araya  
David González Elgueta

## Historial de Versiones

---

<b>Versión</b>	<b>Fecha</b>	<b>Autor</b>	<b>Descripción</b>
<b>1.0</b>	13/07/22	Luis Caroca	Bosquejo Inicial del documento
<b>1.1</b>	04/10/22	Luis Caroca	Edición de formato y cabeza de página. Agregada sección de migración de base de datos
<b>1.2</b>	05/10/22	Luis Caroca	Estructura de carpetas agregado
<b>2.0</b>	08/10/22	Luis Caroca y David González	Visto bueno final por el equipo

## Índice

Introducción .....	4
Programación .....	5
Entorno de desarrollo .....	5
XAMPP .....	5
Visual Studio Code .....	5
Git Bash.....	5
Navegador de internet (Browser) .....	5
Laravel.....	5
Vue.js .....	6
Obtención del código fuente .....	6
Que hacer después de clonar el proyecto .....	6
Actualizar Composer .....	7
Instalar dependencias npm .....	7
Copia de archivo .env .....	7
Generar clave de acceso .....	7
Compilar cambios .....	7
Font-Awesome.....	8
Modificar código fuente.....	8
Base de datos .....	10
Diagrama de base de datos.....	10
Modificar base de datos.....	10
Migrar la base de datos.....	11
Migrar base de datos .....	11
Cargando datos predefinidos.....	12
Estructura de carpetas .....	14
Carpeta raíz .....	14
Carpeta app.....	15
Carpeta database .....	15
Carpeta public .....	16
Carpeta resources .....	17
Carpeta routes .....	18
Archivo .env .....	18

## Introducción

El propósito de este manual es servir como guía para aquel programador que quiera modificar el código original o agregar alguna funcionalidad al mismo.

Este documento, en primera instancia, se centra en las tecnologías utilizadas para el desarrollo de SIAP, como así también los softwares utilizados.

Luego se hace presentación de pasos a tener en cuenta para lograr el montar el proyecto, explicando los diferentes comandos a utilizar y el porque estos deben ser utilizados previos a la ejecución de SIAP.

Además, este documento presentará la manipulación de código en una instancia local, ya que fue así él como fue creado. La modificación del código puede hacerse en producción, pero hay que tener los resguardos adecuados para dicha tarea.

Finalmente, se presenta la estructura de carpetas del proyecto, con intención de explicar los distintos directorios presentes dentro de este, dando una explicación de que incorpora cada uno de estos, con intención de que sea una guía de fácil acceso para las personas que tengan acceso al proyecto de desarrollo de SIAP.

## Programación

### Entorno de desarrollo

Para trabajar con SIAP se necesitan los siguientes programas y sus dependencias, se mencionan alternativas usadas en caso de existir.

#### *XAMPP*

Como SIAP es una aplicación web que fundamentalmente corre en PHP y contiene una base de datos MariaDB, se optó por utilizar XAMPP ya que es de fácil instalación y su integración con Windows no presenta problema alguno. Este puede ser descargado desde su [página oficial](#). Existen otras versiones, como WAMP y DAMP, pero cada una presenta una alternativa distinta con entornos de trabajo distintos.

#### *Visual Studio Code*

Visual Studio Code es actualmente uno de los editores de texto de código abierto más utilizados dentro del mercado, debido principalmente a la tremenda cantidad de extensiones que pueden facilitar y acelerar el trabajo en los distintos lenguajes de programación y PHP no es la excepción. Existen extensiones para diversas problemáticas dependiendo de las necesidades del programador o, porque no decirlo, las comodidades para este. Este puede ser descargado desde su [página oficial](#). Otros programas de edición de texto como Sublime y Notepad++ también pueden servir como alternativas de bajos recursos, solo se debe tener en cuenta el mantener la estructura de código para su fácil manejo.

#### *Git Bash*

Durante todo el proceso de diseño y desarrollo, se utilizó Git Bash como alternativa de consola de comandos ya que es de código abierto, utiliza un código de colores amigable a la vista y puede funcionar sin problemas con permisos de administrador desde un inicio. Este puede ser descargado desde su [página oficial](#). Si bien, Visual Studio Code posee una consola de comandos que nos permite hacer lo mismo, esta fue utilizada principalmente para mantener orden al momento de montar la aplicación en el servidor local. Diversos editores de texto también cuentan con este tipo de consolas, pero si así fuese necesario, CMD de Windows es una excelente alternativa nativa del Sistema operativo para manejar archivos y datos dentro de los directorios de SIAP (de forma local).

#### *Navegador de internet (Browser)*

Google Chrome y Opera fueron los navegadores utilizados para ver los cambios de las vistas en todo momento. No existe ningún factor técnico de por qué utilizar estos navegadores por sobre cualquier otro explorador, simplemente son los browsers de preferencia por parte del equipo de desarrollo, sin embargo, en caso de problemas de compatibilidad con otros navegadores, se recomienda utilizar estos.

#### *Laravel*

Laravel fue el entorno de desarrollo seleccionado, debido a las diversas funcionalidades que nos presenta para trabajar tanto a nivel backend como frontend. Sin embargo, en este caso solo se

utilizó solo para trabajar backend. Para poder utilizar Laravel, primero debemos instalar Composer, que es una herramienta de dependencias para trabajar PHP, el cual puede ser descargado desde su [página oficial](#). Luego podremos acceder a distintos comandos de Composer, como puede ser la creación de un proyecto Laravel con el comando:

```
composer create-project laravel/laravel example-app
```

Para comprender más respecto al funcionamiento de Laravel, sus componentes y su vía de instalación se puede consultar su [documentación oficial](#).

En el caso de este proyecto se trabajó con la versión de Laravel 9.22.1

### *Vue.js*

Para trabajar el frontend, se decidió optar por Vue.js, un framework que nos permite la realización de vistas cómodas y agradables para el desarrollo. Ya que Vue.js está basado en Node, primero debemos realizar la instalación de dicha tecnología, podemos encontrarla en su [página oficial](#). Al momento de realizar esta instalación, se instalará el gestor de paquetes npm, con el cual podremos realizar la instalación de Vue.js con el siguiente comando:

```
npm install vue
```

Esto instalará la última versión de Vue.js en nuestro proyecto. Se puede encontrar más información en su [documentación oficial](#).

En el caso de este proyecto se trabajó con la versión de node 16.13.0, npm 6.14.17 y Vue.js 4.5.15

## **Obtención del código fuente**

El código fuente fue almacenado en GitHub como sistema de control de versiones. Se puede acceder al código con el siguiente comando dentro de una terminal como Git bash:

```
git clone https://github.com/davidgonzalezUCM/siap
```

Esto creará una copia de todos los archivos de SIAP en cualquier lugar donde se desee, listos para modificar.

## **Que hacer después de clonar el proyecto**

Posterior a la obtención del código fuente a través de la clonación de este, debemos considerar que, si bien tenemos el repositorio de manera remota en nuestro equipo, se deben realizar ciertas acciones para que este pueda ser montado de manera correcta.

### *Actualizar Composer*

Como se mencionó anteriormente, Laravel trabaja con los paquetes instalados gracias a Composer. Esto también proporciona la capacidad de precargar los datos de Laravel en el proyecto, sin embargo, las dependencias de este deben ser actualizadas, es por eso por lo que debemos ejecutar el siguiente comando dentro de nuestra terminal:

```
composer update
```

### *Instalar dependencias npm*

De igual forma que con las dependencias de Laravel y Composer, las dependencias de npm deben ser instaladas. Estas dependencias se pueden encontrar de manera visual en el archivo package.json. Para hacer instalación de estas dependencias debemos ejecutar el siguiente comando dentro de nuestra terminal:

```
npm install
```

### *Copia de archivo .env*

El archivo .env es el que se encarga de realizar la conexión tanto con la base de datos como con el servidor de el proyecto, sin embargo, este archivo no se encuentra dentro del proyecto a menos que este sea creado, ya sea a través de terminal o de manera manual. De manera manual podemos copiar el archivo .env.example y eliminar su extensión .example. Si bien nuestro equipo nos puede mostrar una alarma de que este cambio puede dejar el archivo inutilizable, es el paso requerido para el uso de este mismo. Si deseamos realizar una copia a través de terminal, debemos escribir el siguiente comando:

```
cp .env.example .env
```

### *Generar clave de acceso*

Laravel requiere que nuestra aplicación cuente con una clave de acceso para la realización de todas las consultas a base de datos. Generar esta clave es un paso simple, pero importante al momento de trabajar con este poderoso *framework*. Para hacer la creación e implementación automática de esta clave debemos utilizar el siguiente comando en nuestra terminal:

```
php artisan key:generate
```

### *Compilar cambios*

Si bien Laravel no nos exige que debamos realizar una compilación de nuestro proyecto, Vue.js sí, debido a que trabaja en base a Node.js y los paquetes npm. Compilar los cambios nos permitirá que nuestros archivos puedan trabajar de manera correcta y, en caso de estar visualizando nuestra aplicación, permite ver los cambios realizados en las distintas vistas del proyecto. Existen 2 maneras para compilar el proyecto a través de terminal y se explicará la diferencia de estas a continuación:

```
npm run dev
```

Este comando nos permite correr una acción que compilará todo nuestro proyecto en un entorno de desarrollador, lo cual nos permitirá revisar los cambios realizados.

`npm run watch`

Este comando nos permite correr una acción que mantendrá activa nuestro compilador del proyecto, lo cual nos permitirá mantener el proceso abierto y que se actualice cada vez que hacemos un cambio y guardemos nuestros cambios sin tener la necesidad de ejecutar nuevamente el comando.

### Font-Awesome

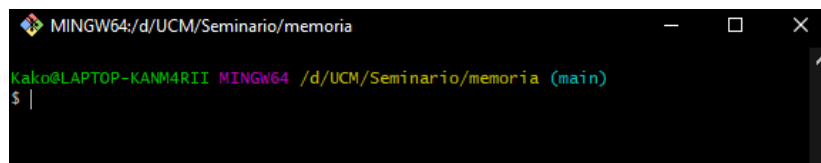
SIAP se alimenta de diversos íconos a través de la librería de íconos *Font-Awesome*. Actualmente, esta se encuentra implementada de manera local a través de su [repositorio GitHub](https://github.com/FortAwesome/Font-Awesome), debido a que esta no se ve clonada directamente en nuestro proyecto debido a que está asociada al repositorio oficial de *Font-Awesome*. Si bien esta no es necesaria para que SIAP pueda ser ejecutado de manera correcta, si no contamos con esta librería nuestro proyecto perderá formato de algunos íconos y no se mostrarán las vistas de manera correcta. Para obtener esta librería debemos realizar una clonación dentro de la carpeta *public* de nuestro proyecto con el siguiente comando dentro de nuestra terminal:

`git clone https://github.com/FortAwesome/Font-Awesome`

### Modificar código fuente

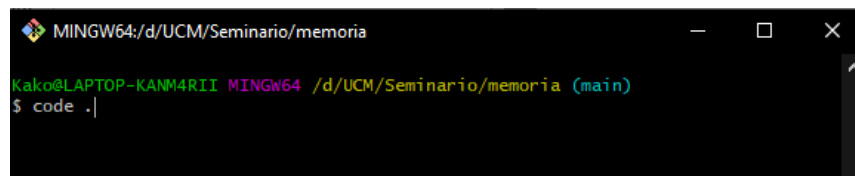
Como se mencionó anteriormente, SIAP puede ser editado en cualquier momento y utilizando cualquier editor de texto. Sin embargo, a continuación, se presenta como acceder al proyecto desde la terminal Git bash:

- En primera instancia, abrimos la terminal Git bash dentro de nuestro proyecto:



```
MINGW64:/d/UCM/Seminario/memoria
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Seminario/memoria (main)
$ |
```

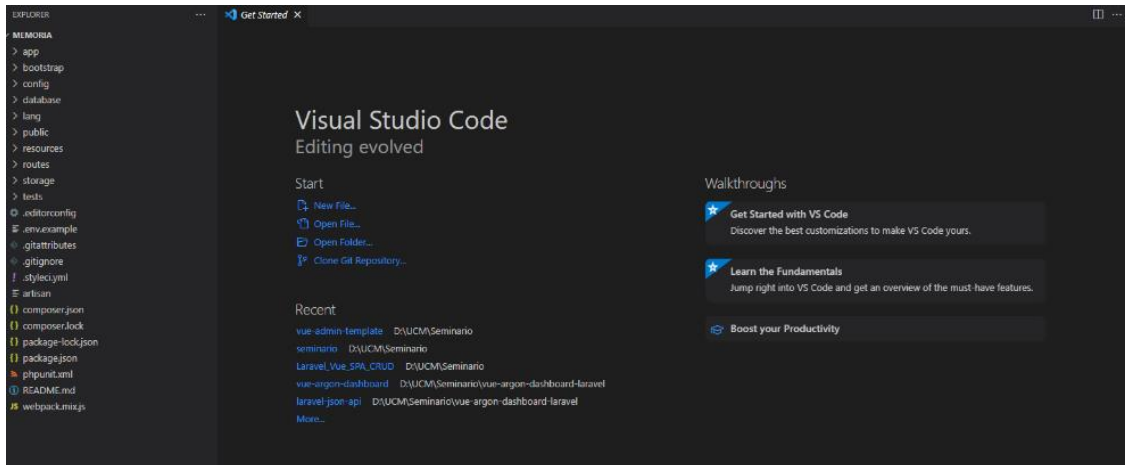
- Luego, teniendo instalado visual studio code, escribimos el comando `"code ."`:



```
MINGW64:/d/UCM/Seminario/memoria
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Seminario/memoria (main)
$ code .|
```



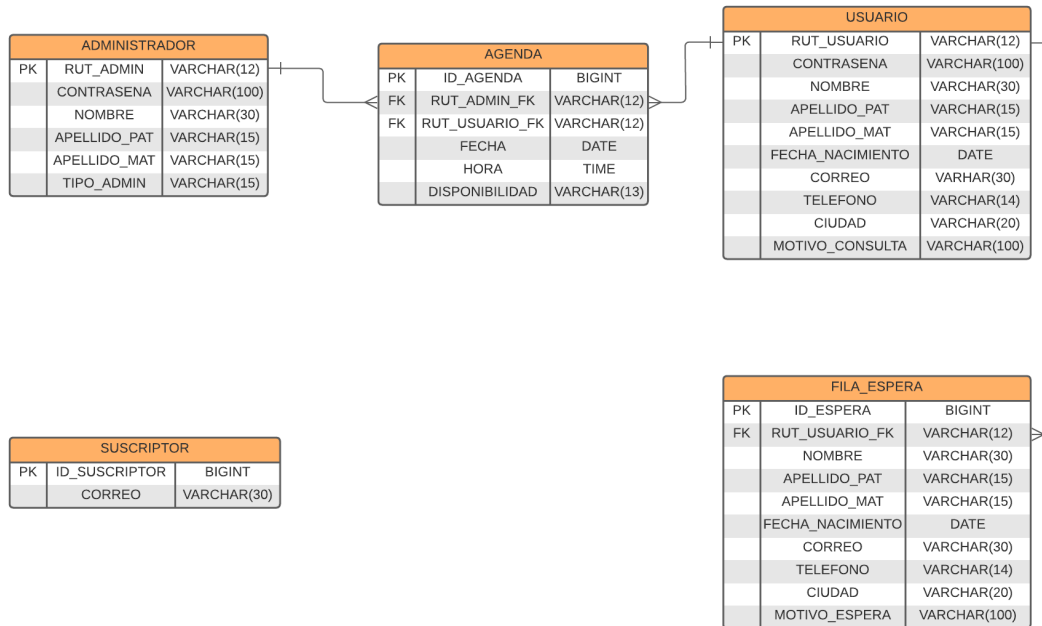
- Presionamos ENTER y esperamos a que se abra visual studio code con nuestro proyecto completo cargado:



De esta manera nuestra IDE creará un árbol de directorios y podrá alterar los datos que estime convenientes.

## Base de datos

### Diagrama de base de datos



### Modificar base de datos

SIAP durante su concepción fue diseñada utilizando el gestor de base de datos MariaDB (aún referenciado como MySQL por parte de Laravel) en una instancia local. Al momento de ser puesto en producción se debe tener en cuenta que los datos de conexión deben ser editados para no ocasionar problemas, sin embargo, gracias a Laravel, todos los datos importantes de conexión están almacenados dentro del archivo `.env`, el cual se encuentra almacenado en la carpeta raíz del proyecto.

```

> vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
  
```

Los datos que se deben tener en cuenta al momento de realizar los cambios son los siguientes:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_psicologa
DB_USERNAME=root
DB_PASSWORD=
```

## Migrar la base de datos

El proyecto cuenta con migraciones de las distintas tablas que posee la base de datos, además de sus campos, restricciones y relaciones entre tablas. Esto es para que no existan problemas con la base de datos creada originalmente, presentada anteriormente. Sin embargo, estas migraciones pueden ser encontradas en la carpeta migrations, dentro de la carpeta database, tal como se muestra a continuación.

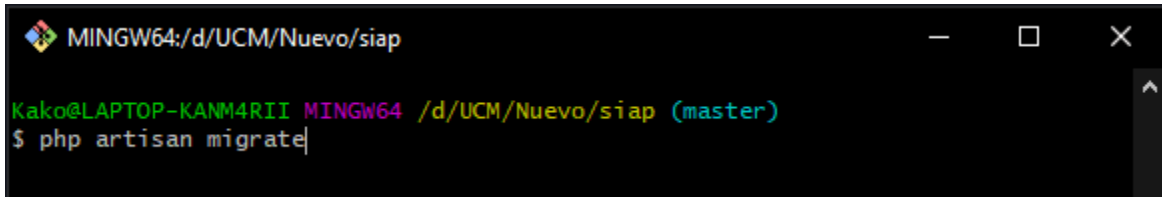
```
▼ database
  > factories
  ▼ migrations
    🐘 2022_10_10_214433_create_administrador...
    🐘 2022_10_10_214433_create_agenda_table...
    🐘 2022_10_10_214433_create_fila_espera_ta...
    🐘 2022_10_10_214433_create_suscriptor_ta...
    🐘 2022_10_10_214433_create_usuario_table...
    🐘 2022_10_10_214434_add_foreign_keys_to...
    🐘 2022_10_10_214434_add_foreign_keys_to...
```

Estas migraciones pueden ser modificadas individualmente dentro de cualquier editor de texto.

De igual forma, estas migraciones pueden ser montadas dentro de la base de datos a través de la terminal de comandos git bash. A continuación, se presentan imágenes mostrando cómo proceder, considerando que ya se configuró y creó la base de datos a utilizar.

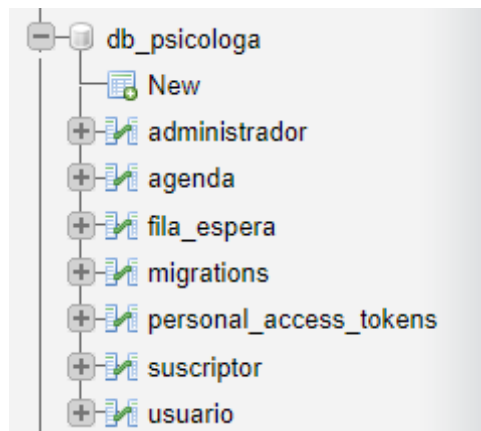
### *Migrar base de datos*

Con el comando `php artisan migrate` (tal como se presenta en la imagen) cargaremos todos los archivos que existan dentro de la carpeta migrations.

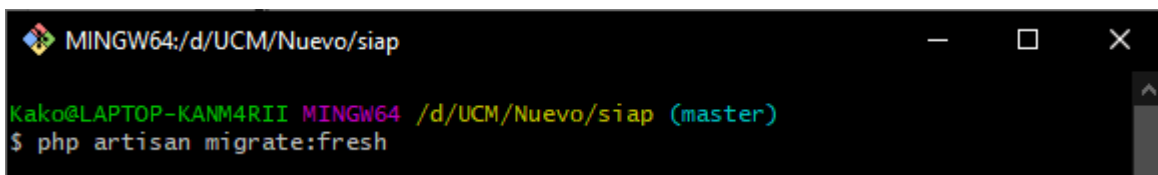


```
MINGW64:/d/UCM/Nuevo/siap
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Nuevo/siap (master)
$ php artisan migrate
```

Esto provocará que nuestra base de datos reciba todas nuestras tablas especificadas en dichos archivos y sus diversas restricciones. Se debe mencionar que las tablas *migrations* y *personal\_access\_tokens* son generadas automáticamente gracias a Laravel. La tabla *migrations* almacena los datos relacionados a las tablas migradas y la cantidad de veces que se ha realizado la migración, mientras que la tabla *personal\_access\_tokens* se genera a través del paquete *Sanctum* de Laravel, el cual es utilizado para el almacenado de *tokens* o credenciales de usuarios en la aplicación. Para más información sobre este paquete se puede visitar su [documentación oficial](#), en donde se detallan sus distintos usos y su implementación.



En caso de realizar un cambio dentro de las migraciones de la base de datos, para que esta se vea reflejado en nuestro gestor y que no ocurran problemas, se debe utilizar el siguiente comando:



```
MINGW64:/d/UCM/Nuevo/siap
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Nuevo/siap (master)
$ php artisan migrate:fresh
```

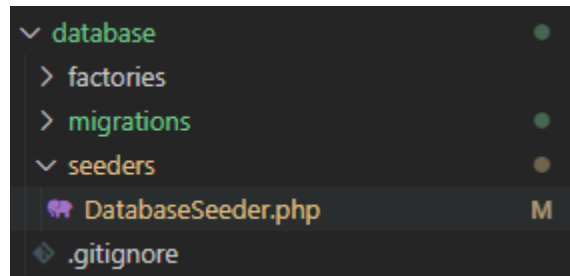
Esto provocará un drop en todas nuestras tablas, para luego volver a montarlas en el gestor, esto evitará problemas de tablas duplicadas.

### Cargando datos predefinidos

En toda base de datos, sobre todo en las que registran datos para inicio de sesión de usuarios, se recomienda generar un usuario base, para poder hacer ingreso a la aplicación desde el

momento en que se monta la aplicación. Este usuario puede ser utilizado para pruebas o como super usuario o administrador general dentro de la aplicación a futuro.

Estos datos se encuentran en archivos dentro de la carpeta seeders, la cual también se encuentra almacenada dentro de database.



En este caso, de momento solo se desea generar una carga de solo el dato de super usuario, sin embargo, se pueden crear mas archivos para la carga de datos a diversas tablas.

Al momento de desear realizar una carga de datos se debe utilizar el siguiente comando:

```
MINGW64:/d/UCM/Nuevo/siap
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Nuevo/siap (master)
$ php artisan db:seed
INFO Seeding database.
```

Esto cargará los datos definidos dentro de nuestro archivo a nuestra base de datos.

En caso de haber realizado cambios dentro de las migraciones de la base de datos y se desee montar con sus nuevos cambios y cargar los datos de esta, se pueden mezclar ambos comandos de la siguiente manera:

```
MINGW64:/d/UCM/Nuevo/siap
Kako@LAPTOP-KANM4RII MINGW64 /d/UCM/Nuevo/siap (master)
$ php artisan migrate:fresh --seed
```

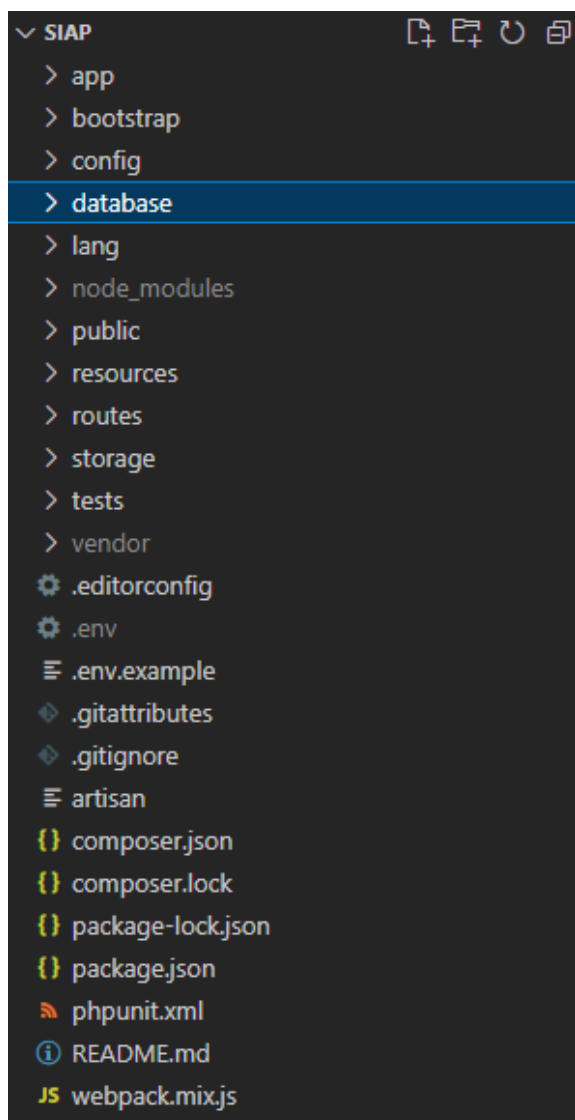
Este comando primero hará un drop a todas nuestras tablas (eliminándolas), para luego proceder a cargar las migraciones, volviendo a montar la base de datos y finalmente, procederá a cargar los datos definidos.

## Estructura de carpetas

A continuación, se presentará la estructura de carpetas dentro del proyecto, acto seguido se presentarán y explicarán las distintas subcarpetas, con intención de facilitar el entendimiento de la distribución del proyecto para su posible edición futura.

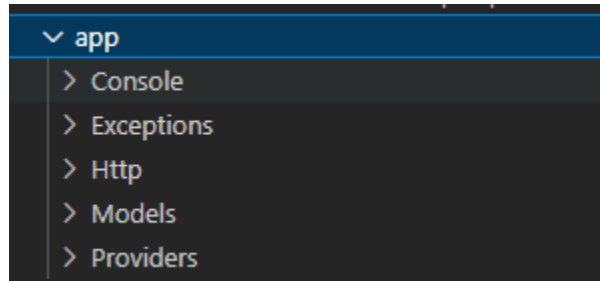
### Carpeta raíz

Esta carpeta es la carpeta que contiene todo el proyecto, esta carpeta será creada al momento de clonar el proyecto desde el repositorio GitHub. Se debe destacar que en esta carpeta se encuentra el archivo .env (con la información de la base de datos) y el archivo package.json, el cual presenta todas las dependencias del proyecto.



## Carpeta app

App es un directorio utilizado por Laravel para alojar todo el código relacionado con el funcionamiento de los componentes críticos de la aplicación web, como lo son sus controladores y los modelos. En muchos casos este directorio es considerado como el más importante de la estructura.

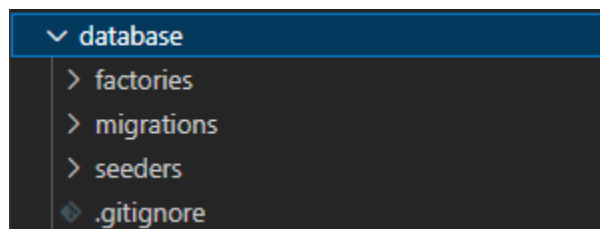


A pesar de que, al utilizar Vue.js, SIAP no hace uso de control de vistas a través de los modelos y controladores como se haría a través de solo la implementación de Laravel, a continuación, se explicarán la funcionalidad de los subdirectorios principales que se encuentran dentro de app.

- app/http/Controllers: Directorio donde se guardan los controladores que servirán de punto medio entre las vistas y el modelo.
- app/http/Middleware: son las reglas de navegación, permisos de usuario y control de roles.
- app/Models: Directorio donde se guardan los modelos ya sea los creados por Laravel como los creados por el programador.

## Carpeta database

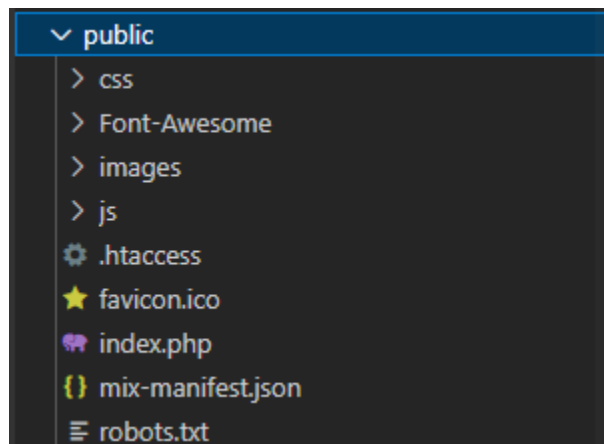
Database es un directorio utilizado por Laravel para alojar todo código relacionado con la estructura de la base de datos.



- database/migrations: Directorio donde se guardan todos los archivos utilizados para migrar la base de datos, para una implementación más veloz y confiable de esta.
- database/seeders: Directorio donde se guardan todos los archivos utilizados para cargar datos automáticamente a la base de datos.

## Carpeta public

Public es un directorio que almacena todos los recursos públicos a utilizar por SIAP.

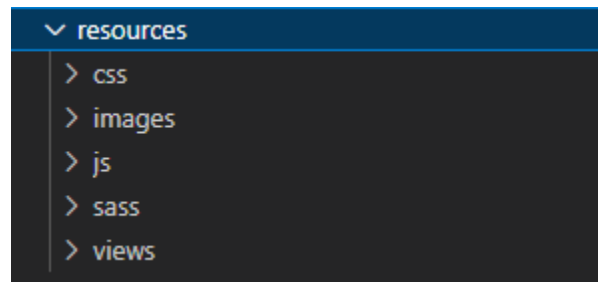


- public/css: Este subdirectorio almacena todo el css (Cascade Style Sheets o, en español, hojas de estilo en cascada), el cual es un archivo que aplica los diversos estilos a SIAP. En este caso, al estar utilizando scss (Sass – Syntactically Awesome Stylesheet) este archivo replica lo definido en otro archivo, dentro de otro directorio, que será presentado más adelante.
- public/Font-Awesome: Este subdirectorio almacena todo lo relacionado con la librería Font Awesome, la cual presenta diversos íconos para utilizar. En caso de que este subdirectorio no se vea incluido en su proyecto base, puede clonarlo directamente desde su [repositorio GitHub](#).
- public/images: Este subdirectorio almacena todas las imágenes utilizadas por el proyecto. De igual manera que con public/css, los archivos en este subdirectorio son una copia de archivos utilizados en otro directorio, que será presentamos más adelante.
- public/js: Este subdirectorio almacena todos los archivos .js presentes en el proyecto, la mayoría de estos están asociados a las vistas con extensión .vue, por lo que muchos de estos archivos se crearán de forma automática al compilar el proyecto.

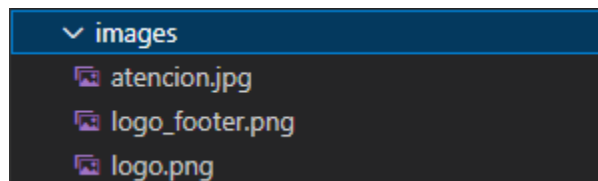


## Carpeta resources

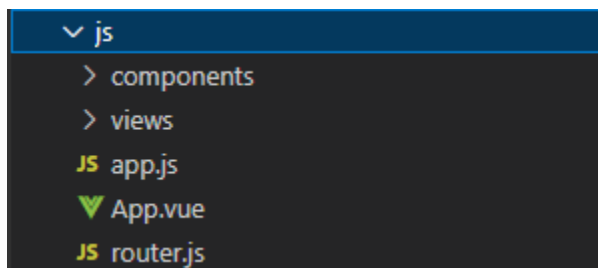
Resources, como su nombre lo indica, es un directorio que almacena todos los recursos utilizados por SIAP. Esta es la carpeta principal para el control de vistas a través de Vue.js, por lo cual se hará explicación por parte de cada uno de sus subdirectorios y archivos más importantes.



- resources/images: Este subdirectorio almacena todas las imágenes utilizadas por SIAP. Al no estar referenciadas a través de la carpeta public, se genera un mayor grado de seguridad para la aplicación. Estos se verán reflejados después en public/images al momento de compilar.

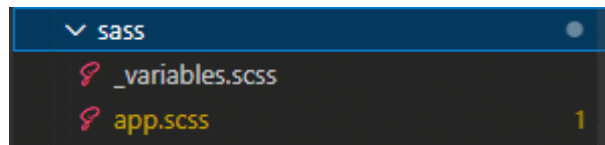


- resources/js: Este subdirectorio almacena los archivos principales para la parte visual y control de Vue.js. A continuación, se describirán los subdirectorios y archivos que se encuentran dentro.

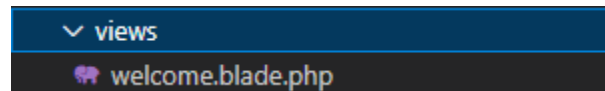


- resources/js/components: En este subdirectorio se encuentran los componentes utilizados por todas las vistas de SIAP, como puede ser el HEADER y el FOOTER.
- resources/js/views: En este subdirectorio se encuentran las vistas que se pueden apreciar en SIAP.
- resources/js/app.js: Este archivo es el que se utiliza para montar SIAP, es decir, acá se debe declarar todo *import* que se desee utilizar, como así también montar la aplicación.

- resources/js/App.vue: Este archivo se puede presentar como la vista principal de SIAP. Es en donde se declara el orden del proyecto, considerando los distintos componentes y vistas.
- resources/js/router.js: Este archivo es el que genera las rutas para las diferentes vistas creadas.
- resources/sass: Este subdirectorío es el que almacena todo el scss que se debe escribir para dar estilo a SIAP. Este se verá reflejado después en public/css al momento de compilar.

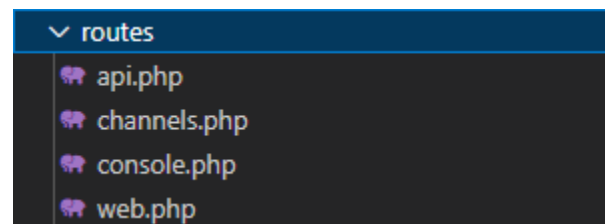


- resources/views: Este subdirectorío almacena lo que se podría llamar la vista principal por parte de Laravel. Es en este archivo donde podemos hacer los llamados a las distintas librerías de fuentes a utilizar, cargar distintos estilos y es donde se debe hacer llamado al archivo app.js.



## Carpeta routes

Routes es un directorio creado por Laravel, el cual sirve para controlar las distintas rutas del proyecto. Dentro del archivo web.php se debe hacer la declaración de las distintas rutas a utilizar. Sin embargo, en el caso de SIAP, al hacer uso de Vue.js, las rutas fueron trabajadas utilizando la propiedad **Router-Link**.



## Archivo .env

El antes mencionado archivo .env, ubicado en la carpeta raíz, se encarga de almacenar la información de conexión con la base de datos.

