



TALLER PRÁCTICO

SISVEN

Fecha: 29/03/2023

Creando un nuevo Proyecto

- a. En una carpeta, por ejemplo laravel-projects, crear el proyecto en laravel llamado **sisven** utilizando **composer**

```
C:\WINDOWS\system32\cmd.exe - composer create-project laravel/laravel sisven

c:\laravel-projects>composer create-project laravel/laravel sisven
Creating a "laravel/laravel" project at "./sisven"
Info from https://repo.packagist.org: #StandwithUkraine
Installing laravel/laravel (v8.6.12)
- Downloading laravel/laravel (v8.6.12)
- Installing laravel/laravel (v8.6.12): Extracting archive
Created project in C:\laravel-projects\sisven
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Info from https://repo.packagist.org: #StandwithUkraine
```

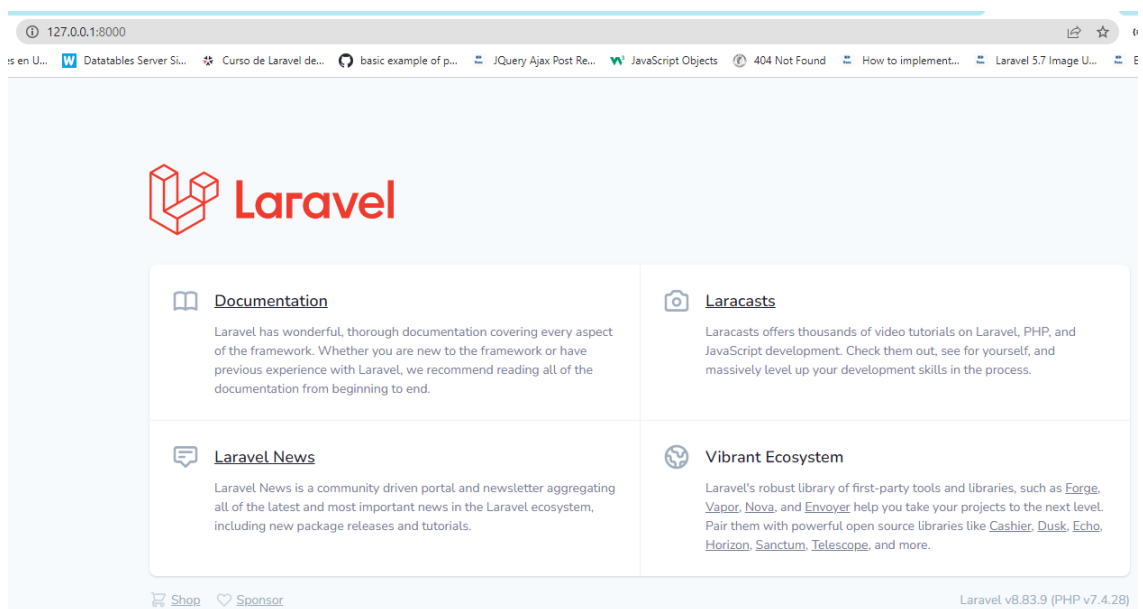
b. Ingresar al proyecto **sisven**

```
C:\WINDOWS\system32\cmd.exe
c:\laravel-projects>cd sisven
c:\laravel-projects\sisven>
```

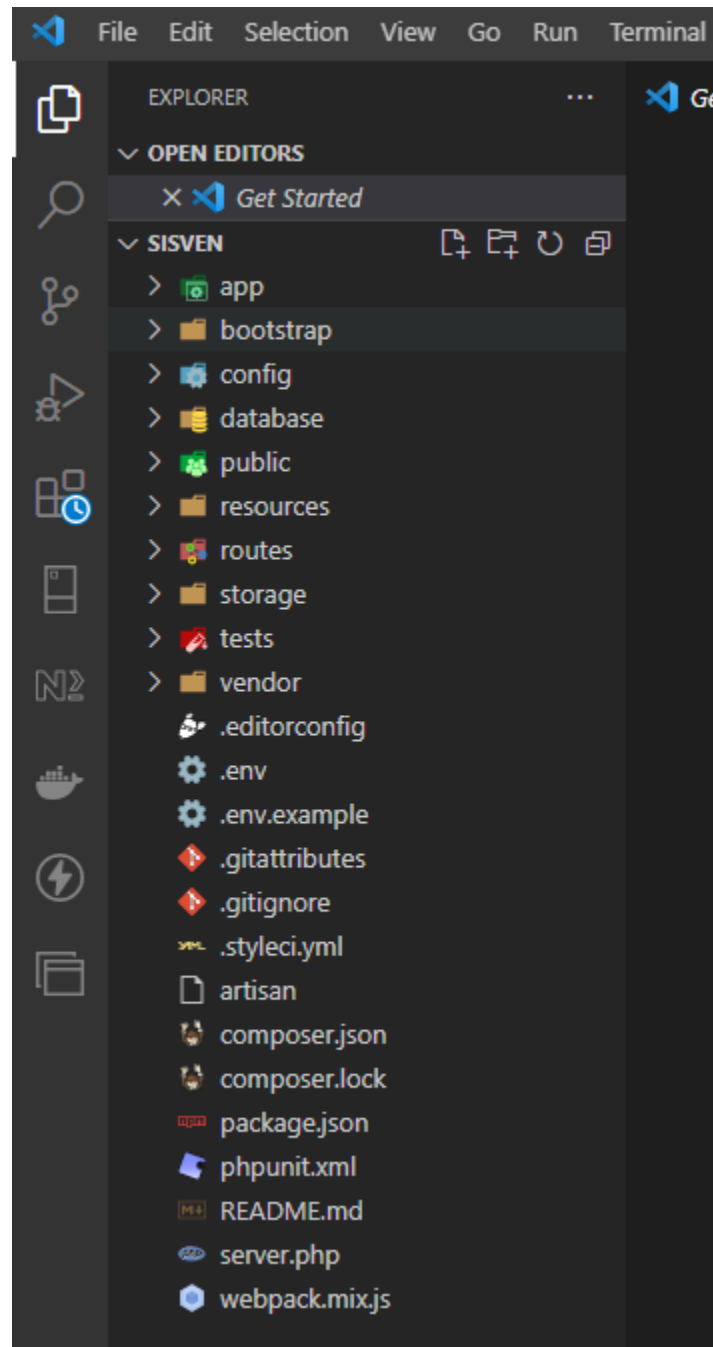
c. Ejecutar el proyecto

```
C:\WINDOWS\system32\cmd.exe - php artisan serve
c:\laravel-projects\sisven>php artisan serve
starting Laravel development server: http://127.0.0.1:8000
[Tue Apr 26 23:16:00 2022] PHP 7.4.28 Development Server (http://127.0.0.1:8000) started
```

d. Abra el navegador y en la barra de dirección escriba <http://127.0.0.1:8000> el navegador deberá levantar su proyecto recientemente creado

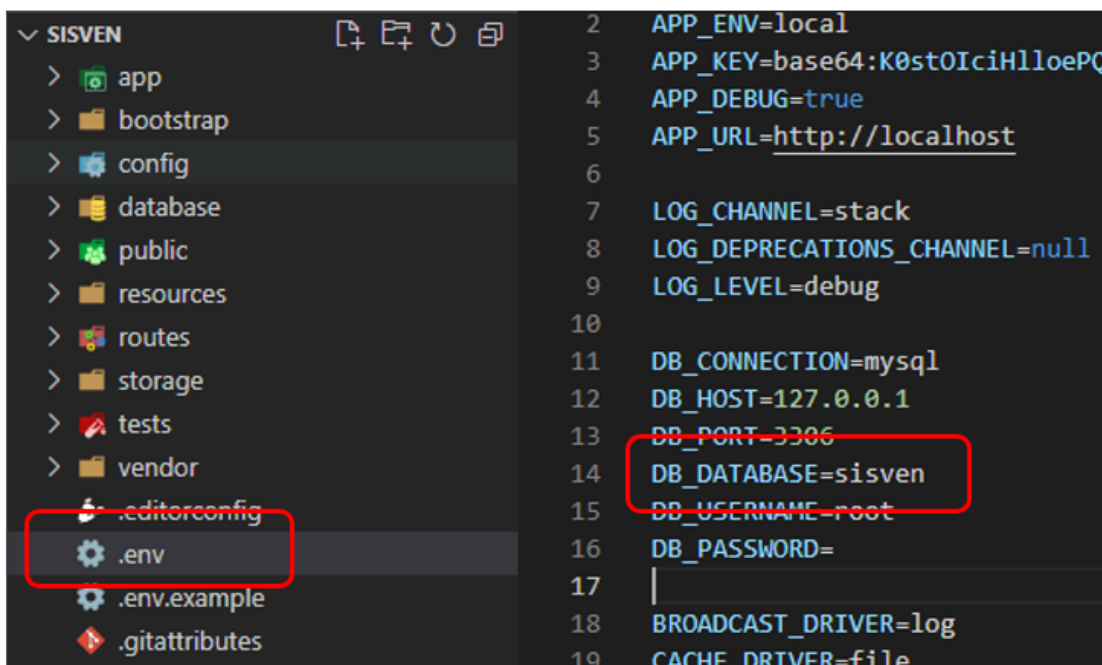


- e. Abra visual studio code y abra la carpeta **sisven**, en el cual verá toda la estructura de directorios del proyecto



- f. Abra xampp y levante los servidores apache y MySQL

- g. En el navegador abra phpMyAdmin y cree la base de datos **sisven**, o si prefiere lo puede realizar en workbench o DBEaver
- h. Configuración de la conexión a la Base de datos **sisven** en el proyecto de laravel Abra el archivo **.env** y modifique las llaves como se observa en la imagen.



- i. Observe que en la carpeta **database/migrations** se encuentran 4 archivos que se crearon con el proyecto. Cada uno de ellos es una migración, que corresponde a la creación o modificación en código para las tablas de la base de datos configurada, ofreciendo así un versionamiento de la BD.

Abra el archivo **2014_10_12_000000_create_users_table.php** y observe que contiene una clase llamada `class CreateUsersTable extends Migration` la cual contiene dos métodos `up()` y `down()`.

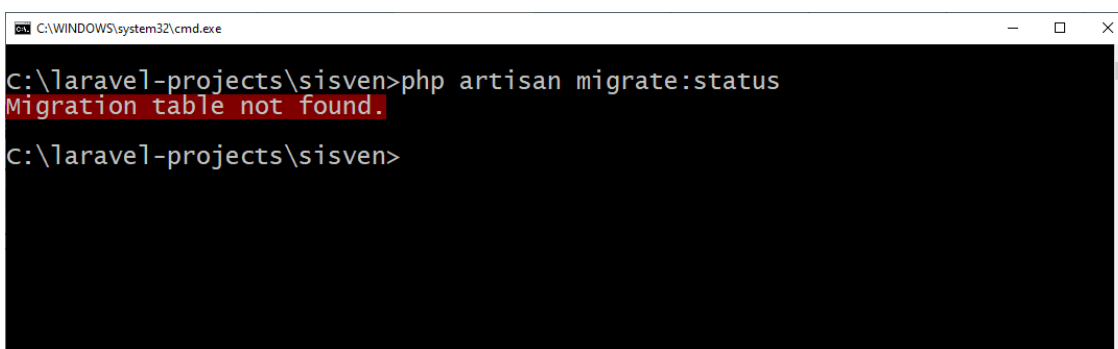
El método `up()` se ejecuta cuando se corre una migración y el método `down()` se ejecuta cuando se reversa una migración.

El método `up()` ejecutará un método de la clase Schema que creará o modificará mediante código la estructura de una tabla. Para el archivo que tiene abierto creará una tabla llamada **users** con los atributos id (Llave primaria por defecto en laravel), name de tipo string, mail de tipo string y con valor único, email_verified_at de tipo timestamp y que permitirá

almacenar valores nulos, password de tipo string, el método `rememberToken` que generará un atributo `remember_token` de tipo `varchar(100)` y el método `timestamps()` que generará los atributos `created_at` y `updated_at` de tipo timestamp.

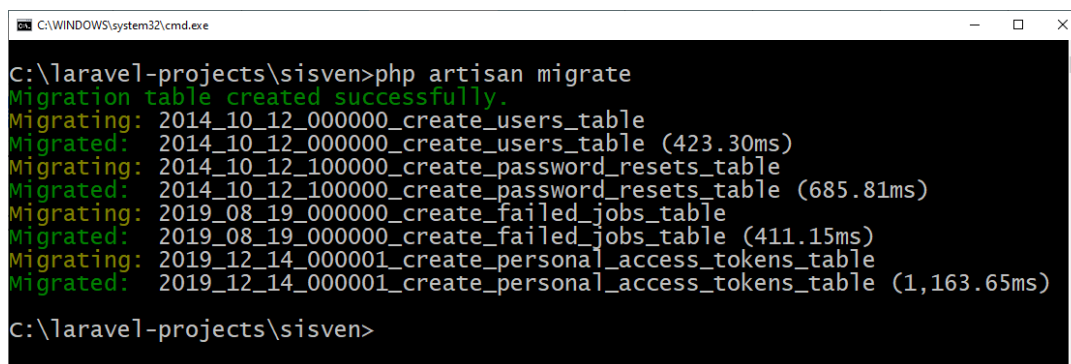
Observe los tres archivos restantes y analice lo que realizará esos archivos al momento de ejecutar por primera vez las migraciones.

- j. Vaya a la consola estando en el directorio del proyecto y ejecute el comando **php artisan migrate:status**. Observe el resultado, laravel nos informa que la tabla migration no existe, porque aún no se ejecuta la primera migración



```
C:\WINDOWS\system32\cmd.exe
C:\laravel-projects\sisven>php artisan migrate:status
Migration table not found.
C:\laravel-projects\sisven>
```

- k. Ejecutemos la primera migración, escriba el comando **php artisan migrate**. Observe que el comando informa que se creó la tabla **migration** exitosamente y procede a migrar los archivos de migración que aún no se habían realizado, en nuestro caso las cuatro migraciones de la carpeta `database/migrations`



```
C:\WINDOWS\system32\cmd.exe
C:\laravel-projects\sisven>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (423.30ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (685.81ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (411.15ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated: 2019_12_14_000001_create_personal_access_tokens_table (1,163.65ms)
C:\laravel-projects\sisven>
```

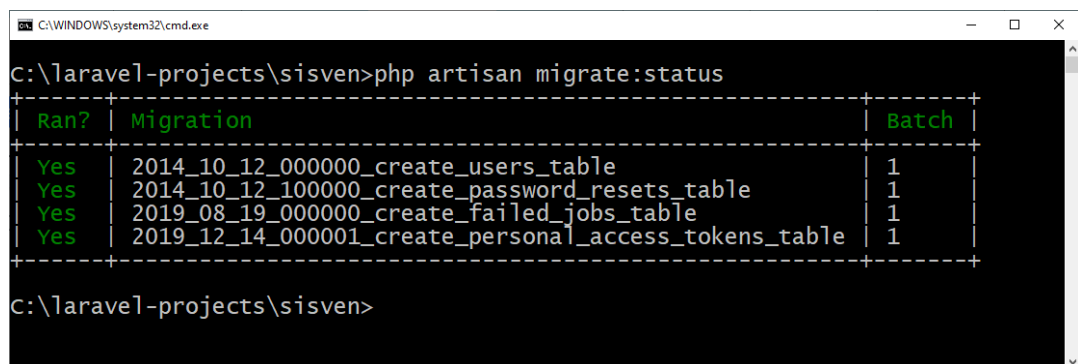
- l. Vaya a phpMyAdmin y actualice la vista de la base de datos **sisven**, ahora la base de datos tiene 5 tablas, como se observa en la imagen. La tabla **migrations** tiene el versionamiento de la migraciones ejecutadas, y las otras tablas correspondientes a los archivos de migración, entre ellas la tabla **users**.



Tabla	Acción
<input type="checkbox"/> failed_jobs	★ Examinar
<input type="checkbox"/> migrations	★ Examinar
<input type="checkbox"/> password_resets	★ Examinar
<input type="checkbox"/> personal_access_tokens	★ Examinar
<input type="checkbox"/> users	★ Examinar
5 tablas	Número de filas

Observe la estructura y los registros creados en la tabla **migrations**, y observe la estructura de las otras tablas, en particular la tabla **users**

- m. Ahora vuelva a ejecutar en la ventana de comandos **php artisan migrate:status**, observe que el resultado ahora informa el estado de las migraciones, información que corresponde a lo almacenado en la tabla **migrations**

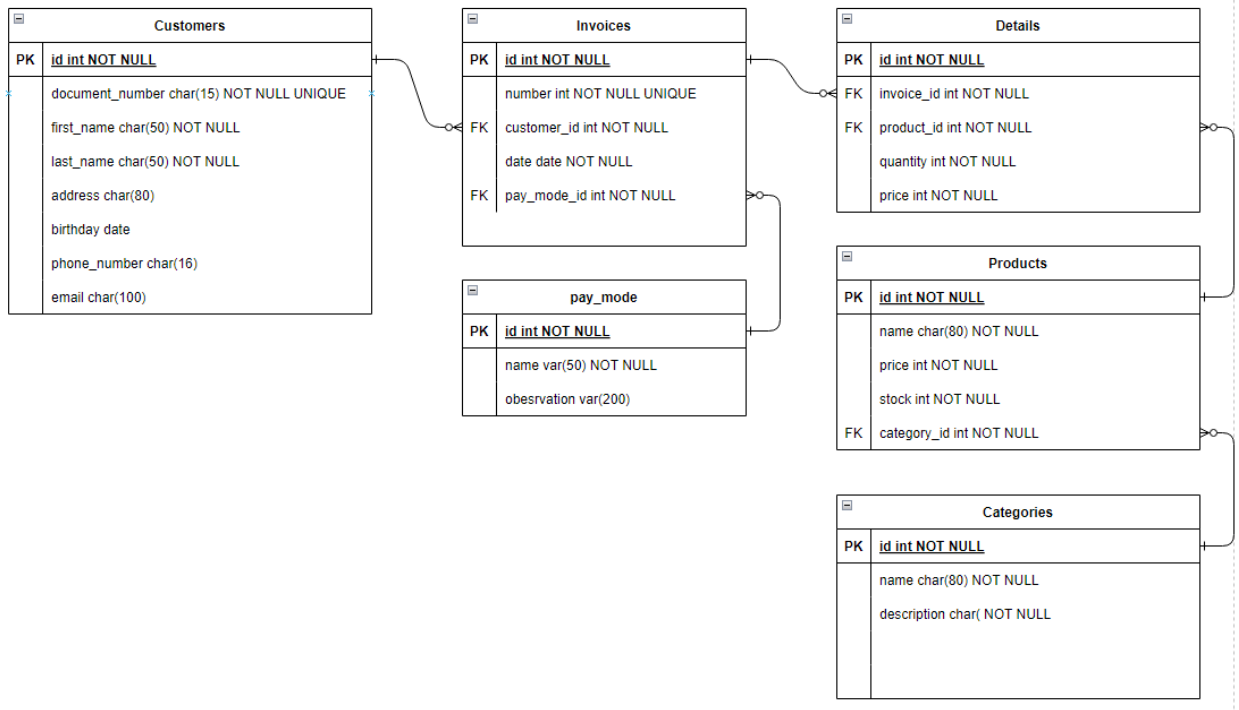


```
C:\laravel-projects\sisven>php artisan migrate:status
```

Ran?	Migration	Batch
Yes	2014_10_12_000000_create_users_table	1
Yes	2014_10_12_100000_create_password_resets_table	1
Yes	2019_08_19_000000_create_failed_jobs_table	1
Yes	2019_12_14_000001_create_personal_access_tokens_table	1

```
C:\laravel-projects\sisven>
```

- n. Vamos a crear las migraciones correspondientes a las tablas del siguiente modelo



- o. Creemos la migración para la tabla categories, en la ventana de comandos ejecute **php artisan make:migration create_categories_table**

```

C:\WINDOWS\system32\cmd.exe

c:\laravel-projects\sisven>php artisan make:migration create_categories_table
Created Migration: 2022_04_27_051558_create_categories_table
c:\laravel-projects\sisven>_
  
```

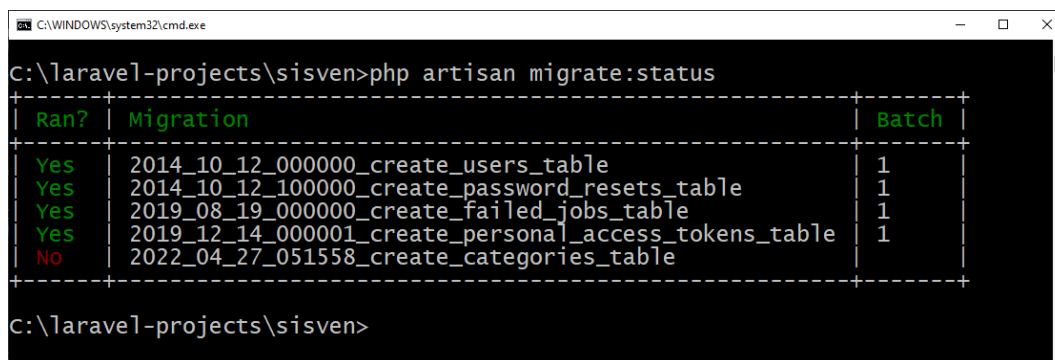
- p. Observe en la carpeta **database/migrations** y ahora aparece un nuevo archivo llamado **2022_04_27_045826_create_categories_table.php** (el inicio del nombre del archivo es la fecha de creación) el cual contiene una clase llamada **CreateCategoriesTable**, con los métodos **up()** y **down()**.
- q. El método **up()** tiene **Schema::create('categories'** que indica que creará una tabla llamada **categories**, con dos métodos: **id()** que genera un atributo **id** como llave primaria y **timestamps()** que genera los dos atributos **created_at** y **updated_at** de tipo timestamp.

A dicha estructura se debe agregar los otros campos de la tabla **categories**, de acuerdo al diagrama.

Agregue después de **\$table->id()**, lo que observa en la siguiente imagen, y luego grabe el archivo

```
public function up()
{
    Schema::create('categories', function (Blueprint $table) {
        $table->id();
        $table->string('name', 64)->unique();
        $table->text('description')->nullable();
        $table->timestamps();
    });
}
```

- r. Vaya a la ventana de comandos y consulte nuevamente el estado de las migraciones, con el comando **php artisan migrate:status**, observe ahora que el resultado informa que hay una migración que no se ha ejecutado

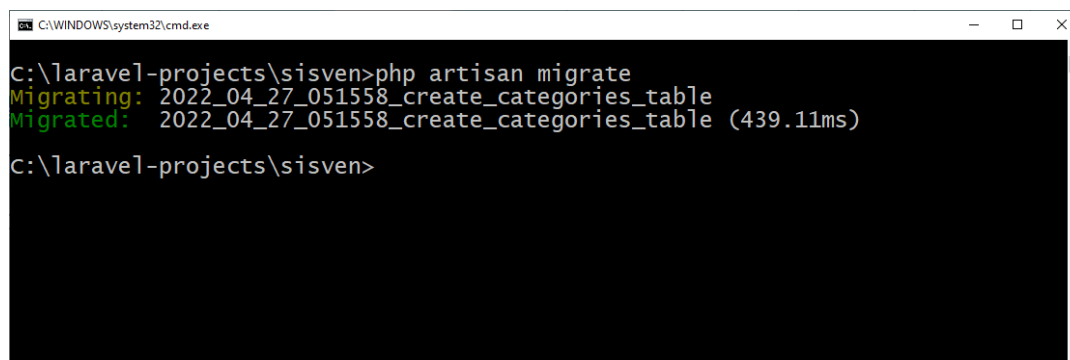


```
C:\laravel-projects\sisven>php artisan migrate:status
```

Ran?	Migration	Batch
Yes	2014_10_12_000000_create_users_table	1
Yes	2014_10_12_100000_create_password_resets_table	1
Yes	2019_08_19_000000_create_failed_jobs_table	1
Yes	2019_12_14_000001_create_personal_access_tokens_table	1
No	2022_04_27_051558_create_categories_table	

```
C:\laravel-projects\sisven>
```

- s. Ahora ejecute la migración, con el comando **php artisan migrate**, el resultado será que la migración se ejecutó con éxito

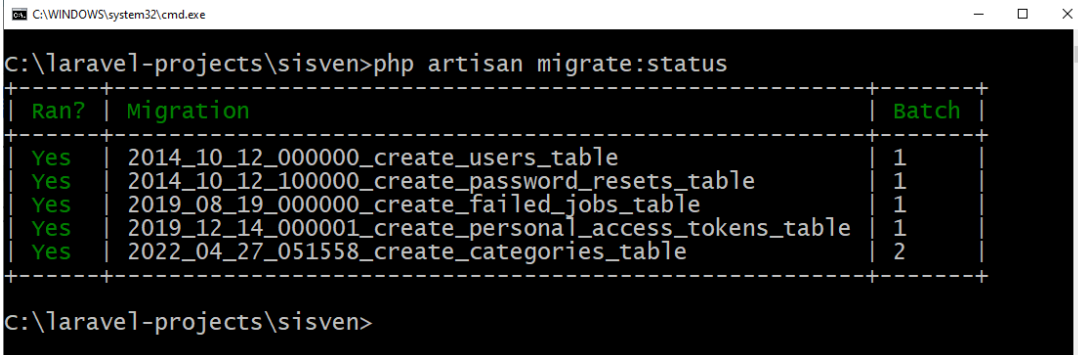


```
C:\laravel-projects\sisven>php artisan migrate
```

```
Migrating: 2022_04_27_051558_create_categories_table
Migrated: 2022_04_27_051558_create_categories_table (439.11ms)
```

```
C:\laravel-projects\sisven>
```


- t. Consulte nuevamente el estado de la migración **php artisan migrate:status**, el resultado informa que todas las migraciones han sido ejecutadas



```
C:\WINDOWS\system32\cmd.exe
C:\laravel-projects\sisven>php artisan migrate:status
```

Ran?	Migration	Batch
Yes	2014_10_12_000000_create_users_table	1
Yes	2014_10_12_100000_create_password_resets_table	1
Yes	2019_08_19_000000_create_failed_jobs_table	1
Yes	2019_12_14_000001_create_personal_access_tokens_table	1
Yes	2022_04_27_051558_create_categories_table	2

```
C:\laravel-projects\sisven>
```

- u. Ahora vaya a phpMyAdmin y consulte los registros de la tabla migrations y observe la estructura de la tabla categories
- v. Es momento de que usted cree y ejecute las migraciones de las demás entidades del modelo. Cualquier duda con la creación de la estructura de la migración puede consultar los siguientes enlaces:

<https://laravel.com/docs/8.x/migrations>

<https://laravel.com/docs/8.x/migrations#creating-columns>

GENERANDO UN SISTEMA DE AUTENTICACIÓN

Laravel cuenta con varios paquetes que permiten implementar un sistema de autenticación, entre ellos: Laravel Breeze, Laravel Fortify, Laravel Jetstream, Laravel Passport, cada uno con sus propias características.

Vamos a trabajar con el más simple de ellos, Laravel Breeze. Laravel Breeze es una implementación mínima y simple de todas las funciones de autenticación de Laravel, incluido el inicio de sesión, el registro, el restablecimiento de contraseña, la verificación de correo electrónico y la confirmación de contraseña. Además, Breeze incluye una página de "perfil" simple donde el usuario puede actualizar su nombre, dirección de correo electrónico y contraseña.

La capa de vista predeterminada de Laravel Breeze se compone de plantillas Blade simples diseñadas con Tailwind CSS. O bien, Breeze puede montar su aplicación usando Vue o React and Inertia.

1. Antes de iniciar la instalación de Breeze, versione su proyecto y cree el commit Initial Project
2. Para realizarla instalación de Laravel Breeze, primero se debe correr la migración que contiene la creación de la tabla user, o crear manualmente la tabla con sus correspondientes campos.

Luego de contar con la tabla user, **se debe descargar el paquete:**

`composer require laravel/breeze --dev`

Una vez descargado el paquete verifique que archivos se modificaron.

Instalar el paquete

`php artisan breeze:install`

Si el instalador pregunta:

Which stack would you like to install? seleccione **0** (blade)

Would you like to install dark mode support? (yes/no) **[no]**

Would you prefer Pest tests instead of PHPUnit? (yes/no) **[no]**

Una vez instalado el paquete verifique que archivos se modificaron.

Correr de nuevo las migraciones

php artisan migrate

Instalar las librerías Javascript

npm install

npm run dev

Al correr el último comando si el proyecto tiene configurado **webpack.mix** como herramienta compiladora de javascript, se generará un error. Esto lo puede corroborar si en la raíz del proyecto aparece el archivo `webpack.mix.js`. Por tanto, deberá migrar el proyecto a Vite, dado que Breeze opera con esta herramienta.

<https://laravel.com/docs/10.x/vite>

3. Realice una nueva versión de su proyecto.
4. Se debe ejecutar la guía de migración de webpack a vite.

- Se debe instalar Vite y el Plugin de Larvel Vite

npm install --save-dev vite laravel-vite-plugin

Una vez instalados los paquetes verifique que archivos se modificaron.

- Verifique que en la raíz del proyecto está el archivo `vite.config.js`, si no está creelo y pegue el siguiente código

```
import laravel from 'laravel-vite-plugin';
import { defineConfig } from 'vite';

export default defineConfig({
  plugins: [
    laravel({
      input: ["resources/css/app.css",
"resources/js/app.js"],
      refresh: true,
    }),
  ],
  resolve: {
    alias: {
      "@": "/resources/js",
    },
  },
});
```

```
    },
  });
```

- Actualizar los scripts del archivo package.json, el archivo debe quedar como se observa el siguiente código

```
"scripts": {
  "dev": "vite",
  "build": "vite build"
},
```

- Agregar al final del archivo .env, las llaves de VITE_PUSHER, como se observa

```
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Puede quitar las de **MIX_PUSHER**

- Agregar al archivo resources/js/app.js

```
import './bootstrap';
import '../css/app.css';
```

- <https://github.com/laravel/vite-plugin/blob/main/UPGRADE.md#migrating-from-laravel-mix-to-vite>