**ESCUELA POLITÉCNICA NACIONAL**

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**DESARROLLO DE APLICACIONES WEB**

|  |  |
| --- | --- |
|  | |
| ASIGNATURA: | Desarrollo de Aplicaciones Web |
| PROFESOR: | Ing. Loarte Byron |
| PERÍODO ACADÉMICO: |  |
|  | |

**LABORATORIO - 08**

|  |  |
| --- | --- |
|  | |
| TÍTULO:  **RELACIONES** | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  |  |
|  |  |
| Como instalar Laravel 6 en mi servidor |  |
|  |  |
|  | |

**PROPÓSITO DE LA PRÁCTICA**

Familiarizar al estudiante con el manejo relaciones en Laravel.

**OBJETIVO GENERAL**

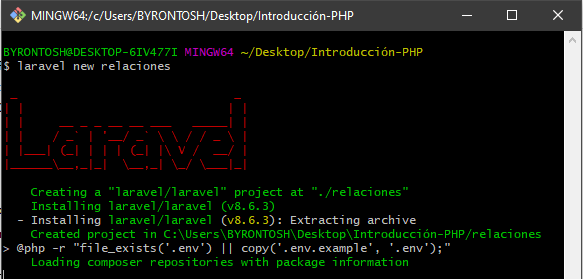
Determinar el manejo de relaciones.

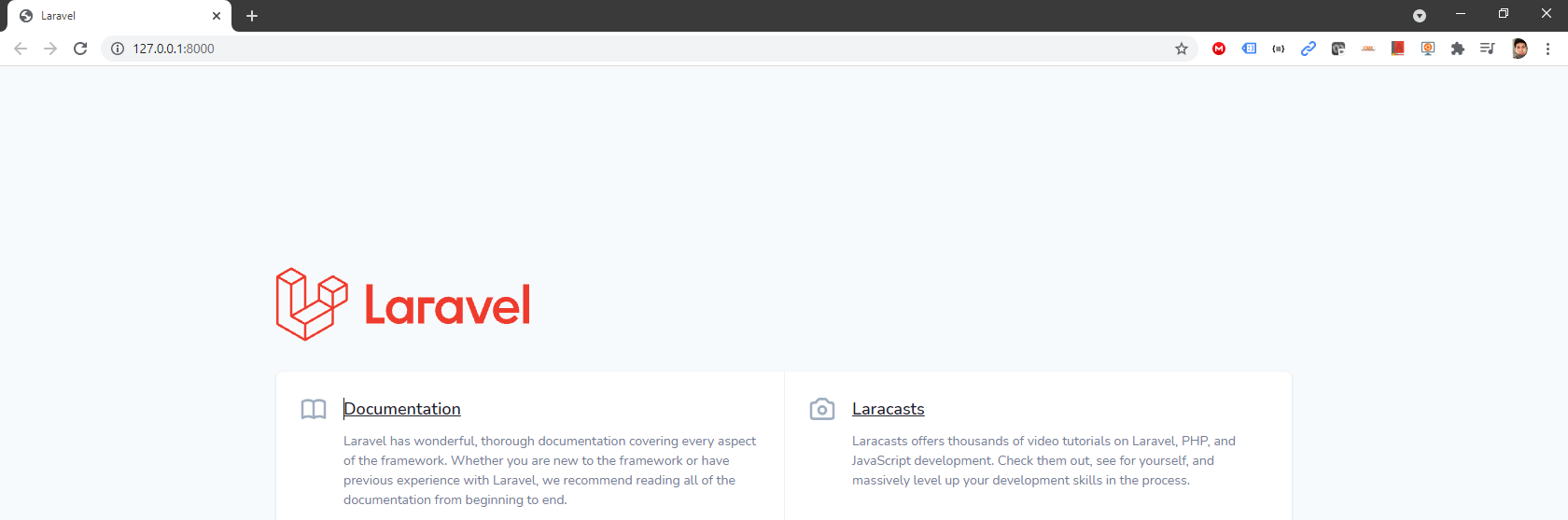
**OBJETIVOS ESPECÍFICOS**

* Configurar las herramientas para el laboratorio.
* Crear un proyecto en Laravel.
* Comprender los archivos y directorios más importantes en Laravel.
* Comprender como funciona las relaciones.
* Comprender como funciona las relaciones polimórficas.
* Comprender como visualizar los resultados de las relaciones en un proyecto.
* Visualizar los resultados obtenidos.

**INSTRUCCIONES**

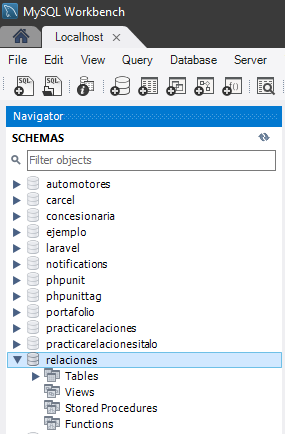
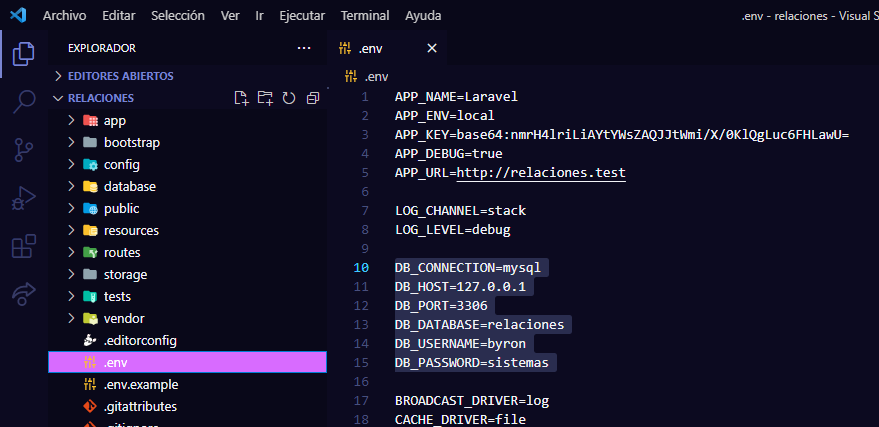
1. Se procede con la creación de un nuevo proyecto y verificar la ejecución del mismo.





1. Ahora se procede a realizar la conexión con la base de datos

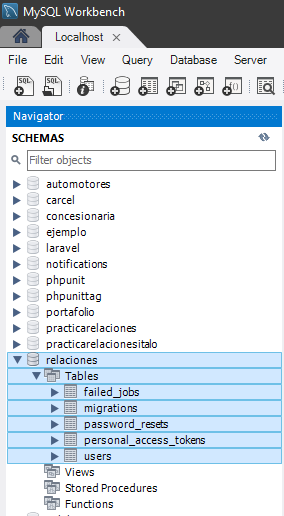
<https://laravel.com/docs/9.x/database#introduction>



1. Para verificar si la conexión esta configurada de manera correcta, ejecutar las migraciones con el siguiente comando.

<https://laravel.com/docs/9.x/migrations#running-migrations>

**php atisan migrate**



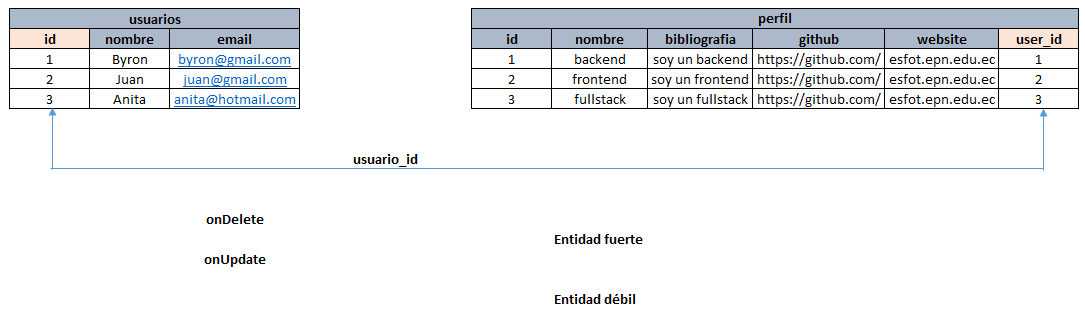
1. Ahora se procede a trabajar con las relaciones

<https://laravel.com/docs/9.x/eloquent#introduction>

<https://laravel.com/docs/9.x/eloquent-relationships#introduction>

https://laravel.com/docs/9.x/eloquent-relationships#one-to-one

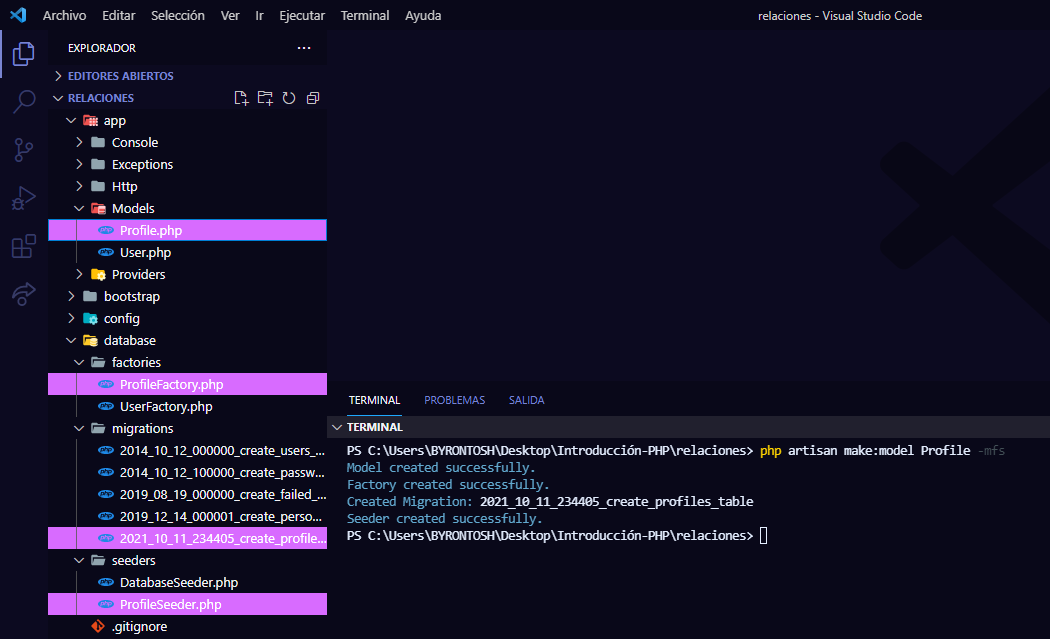
**RELACIÓN DE UNO A UNO**



1. Se procede con la creación de un nuevo modelo llamado profile y adicional a ello se crea su migración, factory y sedder con el siguiente comando.

<https://laravel.com/docs/9.x/eloquent#generating-model-classes>

**php artisan make:model Profile -mfs**

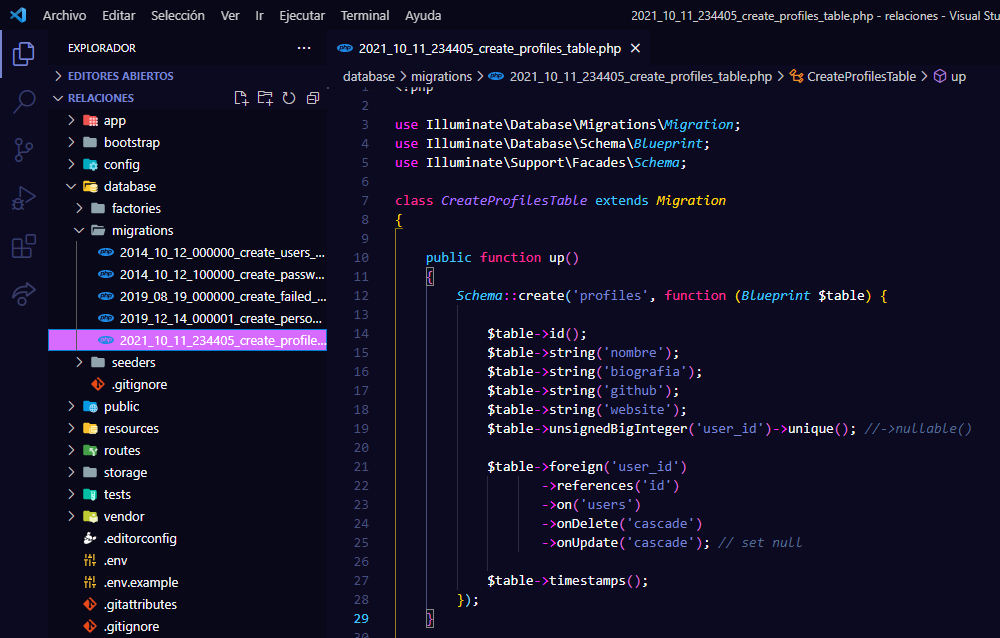


1. Ahora se procede a definir los campos (nombre de las columnas) para la tabla de la base de datos, esto se lo realiza en la migración del modelo respectivo, recordando que:

**Recordar que un usuario tiene un perfil y que el perfil le pertenece a un único usuario**

Con este procedimiento se realizó la relación a nivel de tablas de la **Base de Datos** únicamente.

<https://laravel.com/docs/9.x/migrations#tables>



            $table->id();

            $table->string('nombre');

            $table->string('biografia');

            $table->string('github');

            $table->string('website');

            $table->unsignedBigInteger('user\_id')->unique(); //->nullable()

            $table->foreign('user\_id')

                   ->references('id')

                   ->on('users')

                   ->onDelete('cascade')

                   ->onUpdate('cascade'); // set null

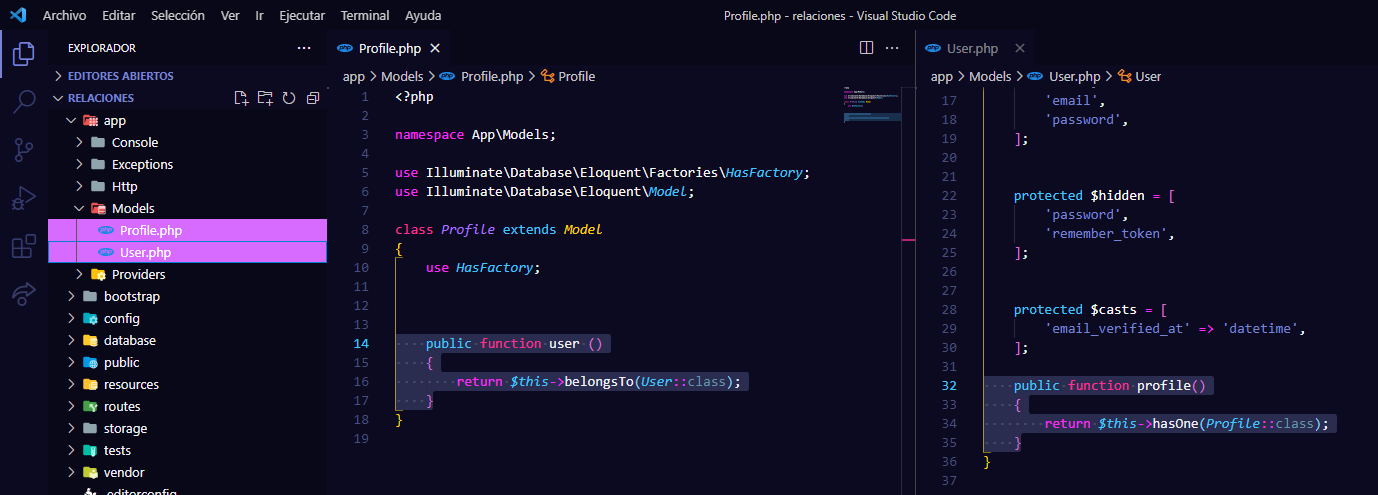
            $table->timestamps();

1. Ahora se debe hacer el procedimiento a nivel de **Eloquent de Laravel**, teniendo en cuenta que:

**Recordar que un usuario tiene un perfil y que el perfil le pertenece a un único usuario**

<https://laravel.com/docs/9.x/eloquent-relationships#one-to-one>

<https://laravel.com/docs/9.x/eloquent-relationships#one-to-one-defining-the-inverse-of-the-relationship>



     // RELACIÓN DE UNO A UNO

     public function user()

     {

         return $this->belongsTo(User::class);

     }

     // RELACIÓN DE UNO A UNO

    public function profile()

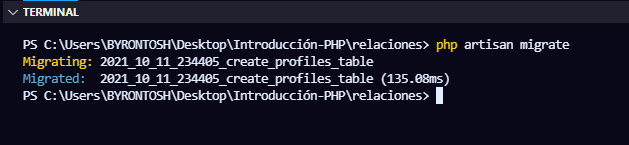
    {

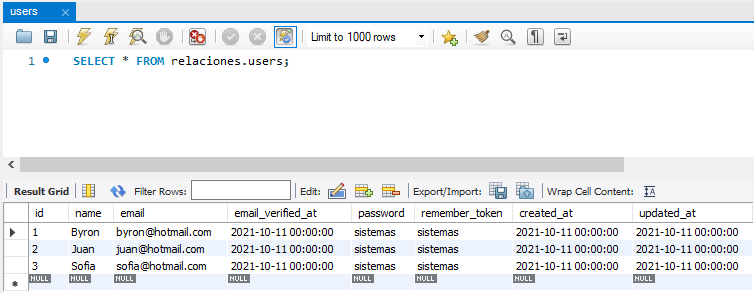
        return $this->hasOne(Profile::class);

    }

1. Ahora se procede a ejecutar la migración y verificar si realmente esta correcta la implementación de la relación uno a uno, ejecutando el siguiente comando.

**php artisan migrate**

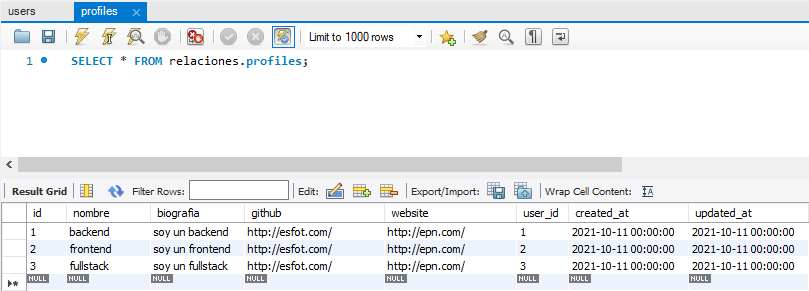




INSERT INTO `relaciones`.`users` (`id`, `name`, `email`, `email\_verified\_at`, `password`, `remember\_token`) VALUES ('1', 'Byron', 'byron@hotmail.com', '2021-11-20 13:19:25', 'sistemas', 'sistemas');

INSERT INTO `relaciones`.`users` (`id`, `name`, `email`, `email\_verified\_at`, `password`, `remember\_token`) VALUES ('2', 'Juan', 'juan@hotmail.com', '2021-11-20 13:19:25', 'sistemas', 'sistemas');

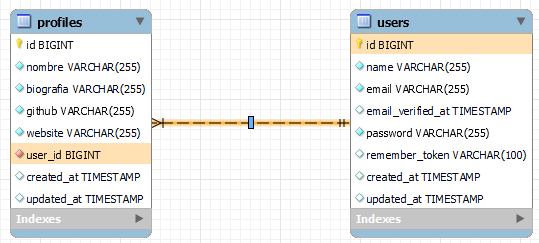
INSERT INTO `relaciones`.`users` (`id`, `name`, `email`, `email\_verified\_at`, `password`, `remember\_token`) VALUES ('3', 'Janneth', 'janneth@hotmail.com', '2021-11-20 13:19:25', 'sistemas', 'sistemas');



INSERT INTO `relaciones`.`profiles` (`id`, `nombre`, `biografia`, `github`, `website`, `user\_id`) VALUES ('1', 'backend', 'soy un backend', 'https://github.com/', 'https://esfot.epn.edu.ec/', '1');

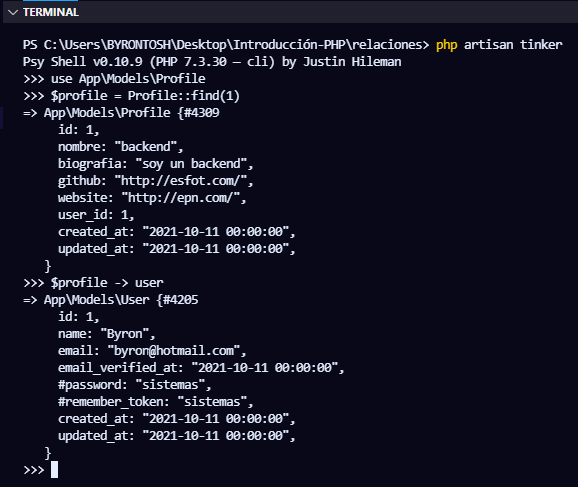
INSERT INTO `relaciones`.`profiles` (`id`, `nombre`, `biografia`, `github`, `website`, `user\_id`) VALUES ('2', 'frontend', 'soy un frontend', 'https://github.com/', 'https://esfot.epn.edu.ec/', '2');

INSERT INTO `relaciones`.`profiles` (`id`, `nombre`, `biografia`, `github`, `website`, `user\_id`) VALUES ('3', 'fullstack', 'soy un fullstack', 'https://github.com/', 'https://esfot.epn.edu.ec/', '3');



1. Para probar se puede verificar ejecutando.

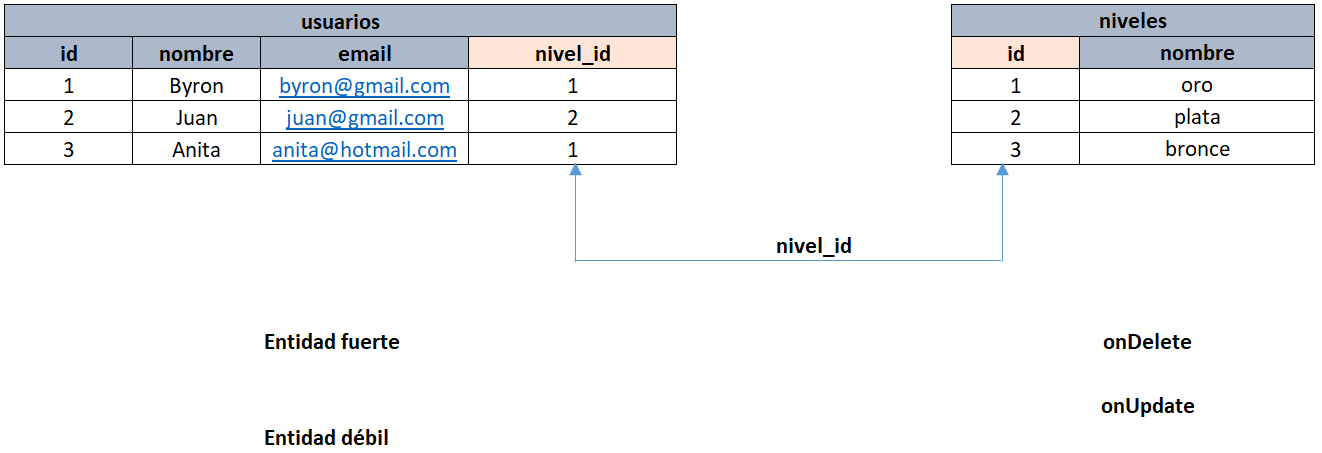
**php artisan tinker**

1. Ahora se procede a trabajar con las relaciones.

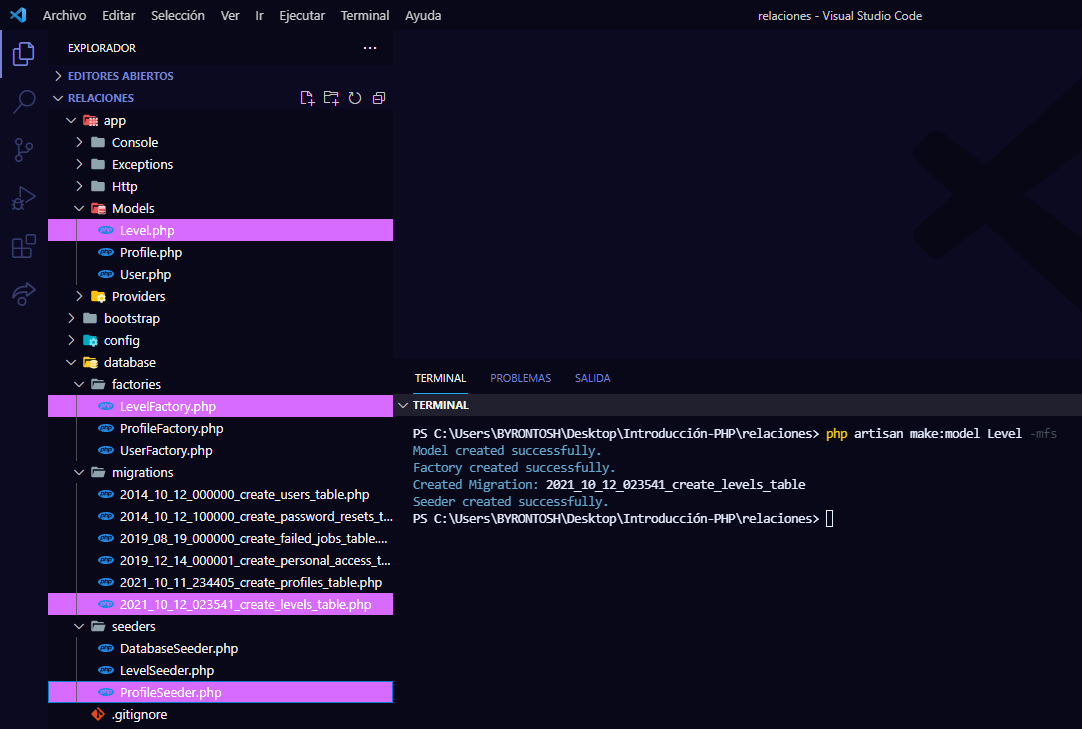
**RELACIÓN DE UNO A MUCHOS**

<https://laravel.com/docs/9.x/eloquent-relationships#one-to-many>



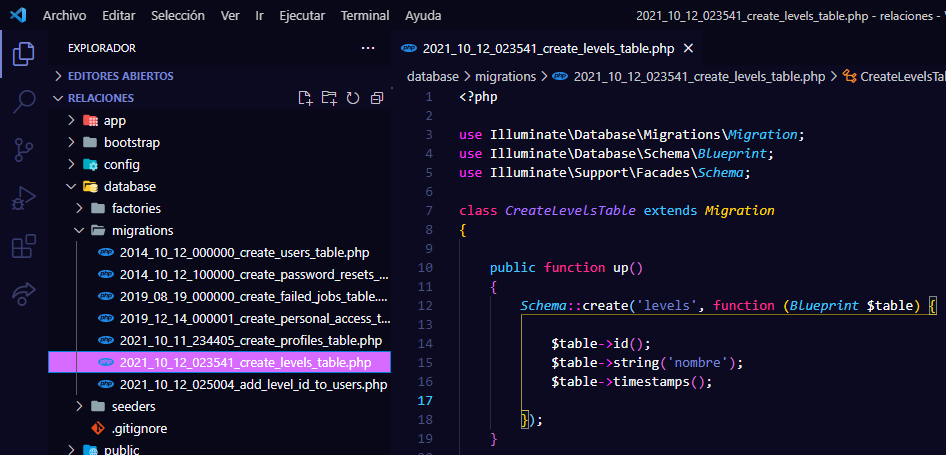
1. Se procede con la creación de un nuevo modelo llamado profile y adicional a ello se crea su migración, factory y sedder con el siguiente comando.

**php artisan make:model Level -mfs**



1. Ahora se procede a definir los campos (nombre de las columnas) para la tabla de la base de datos, esto se lo realiza en la migración del modelo respectivo, recordando que:

**Recordar que un nivel tiene varios usuarios y que el usuario pertenece a un nivel**



            $table->id();

            $table->string('nombre');

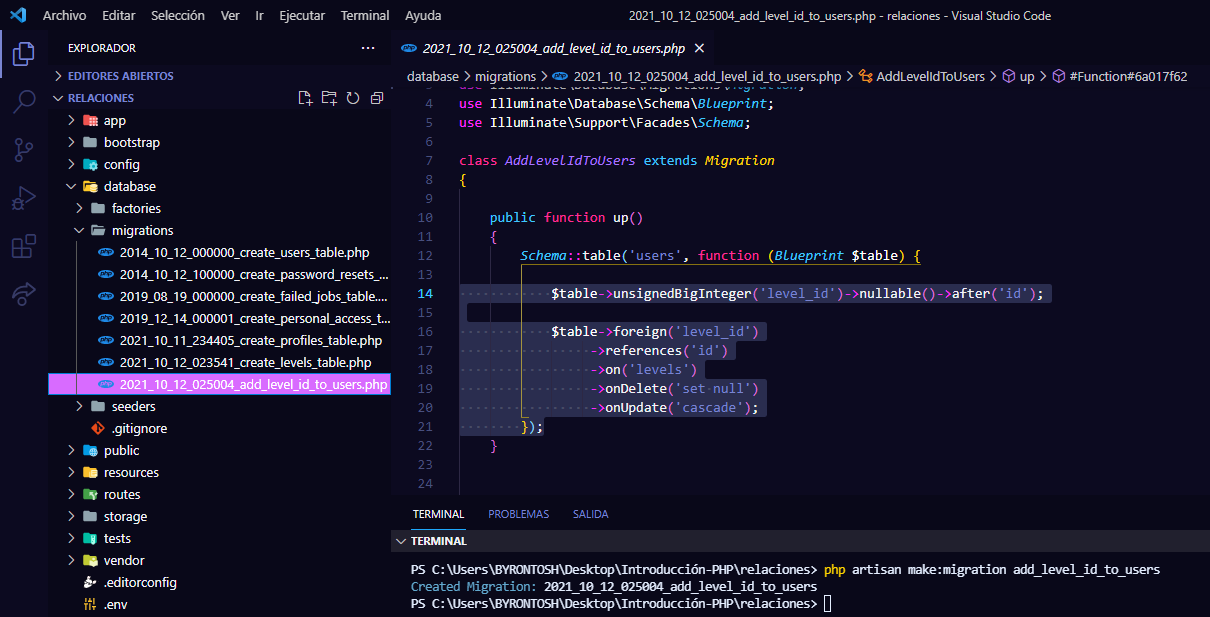
            $table->timestamps();

1. Con este procedimiento se realizó la relación a nivel de tablas de la **Base de Datos** únicamente, pero hay que tener en cuenta sobre las relaciones fuertes y débiles. Para lo cual se procede agregar una migración personalizada, ejecutando el siguiente comando.

<https://laravel.com/docs/7.x/migrations#column-modifiers>

<https://laravel.com/docs/9.x/migrations#column-modifiers>

**php artisan make:migration add\_level\_id\_to\_users**



            $table->unsignedBigInteger('level\_id')->nullable()->after('id');

            $table->foreign('level\_id')

                 ->references('id')

                 ->on('levels')

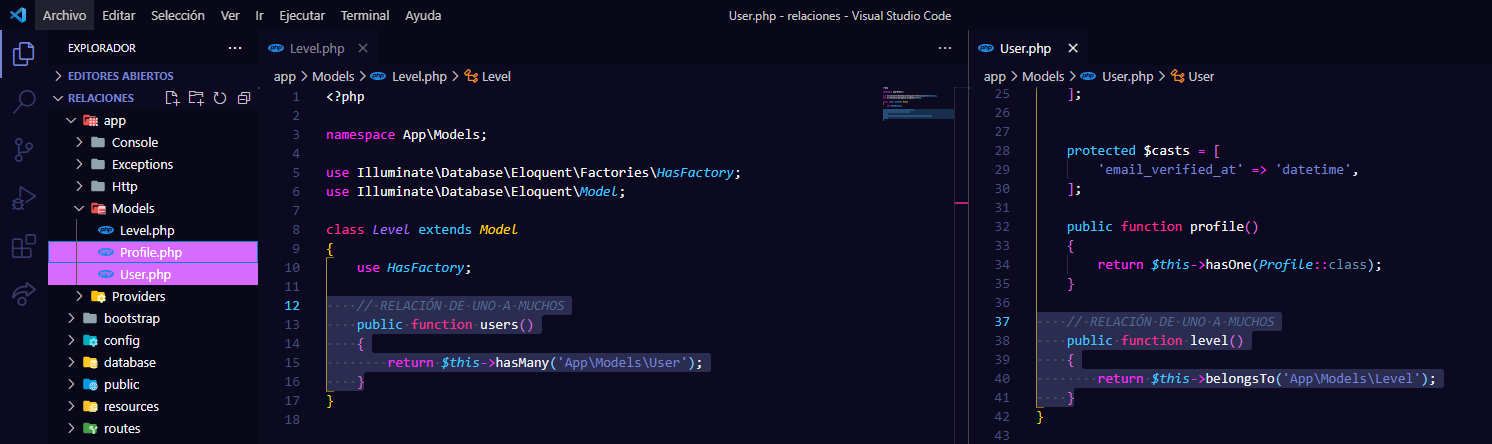
                 ->onDelete('set null')

                 ->onUpdate('cascade');

1. Ahora se debe hacer el procedimiento a nivel de **Eloquent de Laravel**, teniendo en cuenta que:

**Recordar que un nivel tiene varios usuarios y que el usuario pertenece a un nivel**

<https://laravel.com/docs/9.x/eloquent-relationships#one-to-many>



        // RELACIÓN DE UNO A MUCHOS

        public function users()

        {

            return $this->hasMany('App\Models\User');

        }

    // RELACIÓN DE UNO A MUCHOS

    public function level()

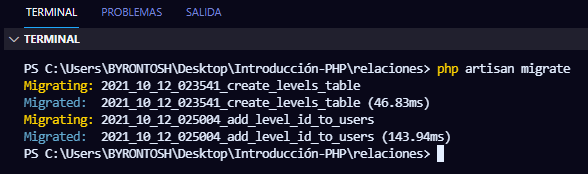
    {

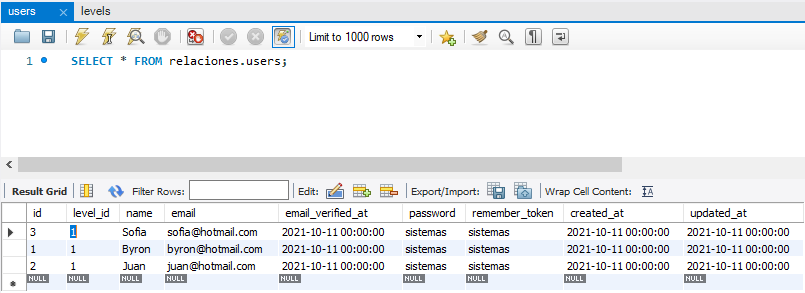
        return $this->belongsTo(Level::class);

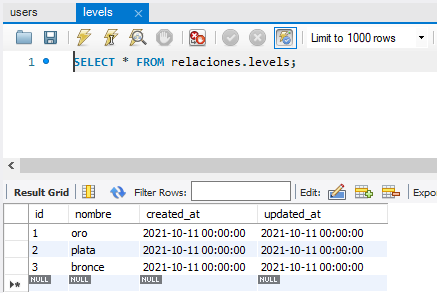
    }

1. Ahora se procede a ejecutar la migración y verificar si realmente esta correcta la implementación de la relación uno a uno, ejecutando el siguiente comando.

**php artisan migrate**



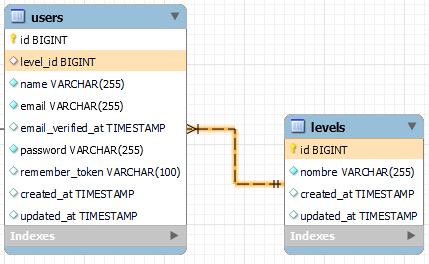




INSERT INTO `relaciones`.`levels` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('1', 'oro', '2021-11-20 13:19:25', '2021-11-20 13:19:25');

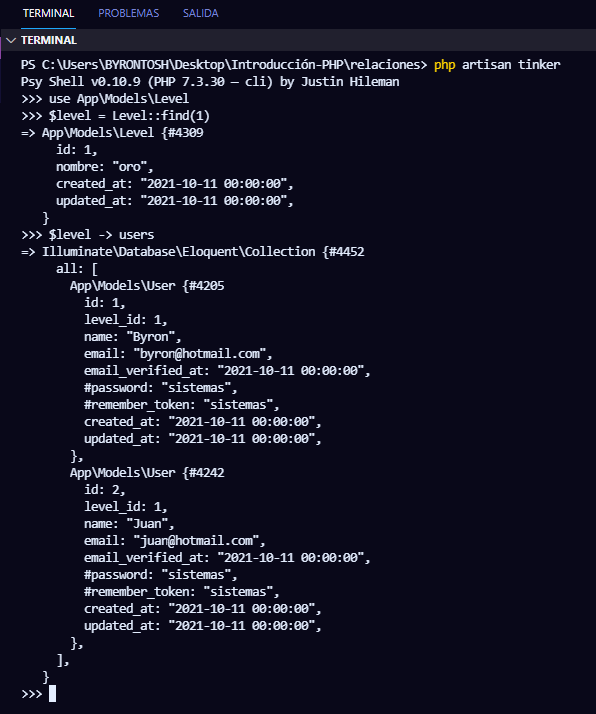
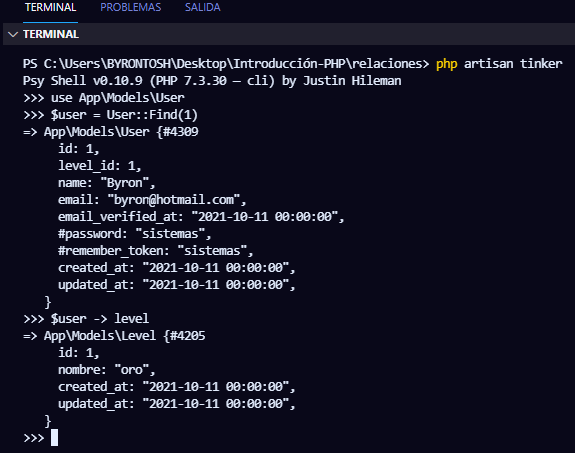
INSERT INTO `relaciones`.`levels` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('2', 'plata', '2021-11-20 13:19:25', '2021-11-20 13:19:25');

INSERT INTO `relaciones`.`levels` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('3', 'bronce', '2021-11-20 13:19:25', '2021-11-20 13:19:25');



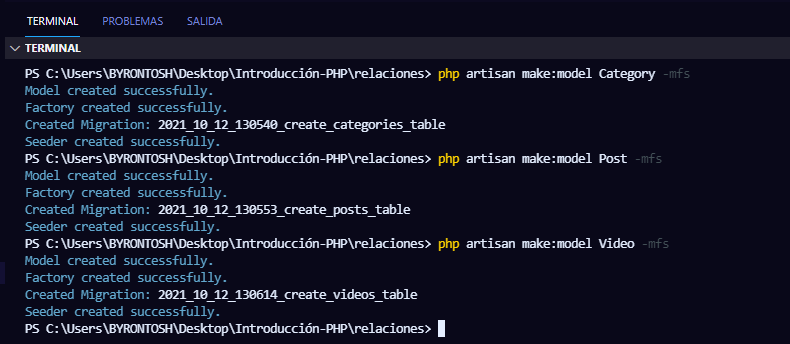
1. Para probar se puede verificar ejecutando

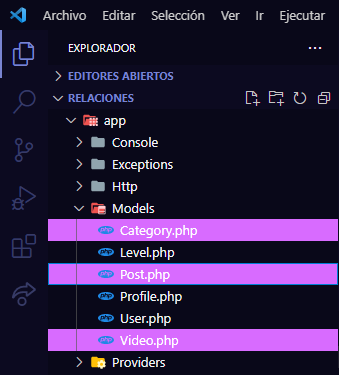
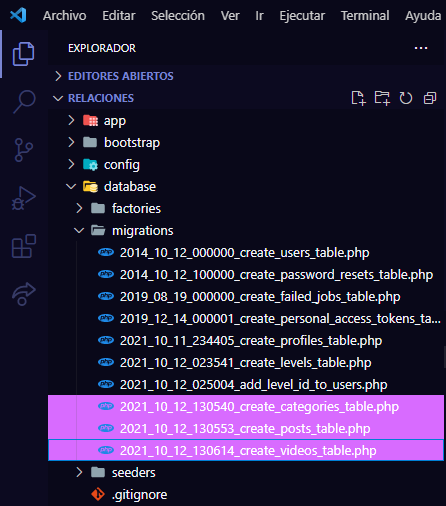
**php artisan tinker**



1. Ahora se procede a realizar las otras relaciones restantes

* **Recordar que un usuario puede hacer varios posts y que un post le pertenece a un usuario**
* **Recordar que un usuario puede tener varios videos y que un video le pertenece a un usuario**
* **Recordar que una categoría puede tener varios videos y que un video le pertenece a una categoría**
* **Recordar que una categoría puede tener varios posts y que un post le pertenece a una categoría**





            $table->id();

            $table->string('nombre');

            $table->timestamps();

            $table->id();

            $table->string('nombre');

            $table->unsignedBigInteger('category\_id');

            $table->unsignedBigInteger('user\_id');

            $table->foreign('category\_id')

                   ->references('id')

                   ->on('categories')

                   ->onDelete('cascade')

                   ->onUpdate('cascade');

            $table->foreign('user\_id')

                   ->references('id')

                   ->on('users')

                   ->onDelete('cascade')

                   ->onUpdate('cascade');

            $table->timestamps();

            $table->id();

            $table->string('nombre');

            $table->unsignedBigInteger('category\_id');

            $table->unsignedBigInteger('user\_id');

            $table->foreign('category\_id')

                  ->references('id')

                  ->on('categories')

                  ->onDelete('cascade')

                  ->onUpdate('cascade');

            $table->foreign('user\_id')

                  ->references('id')

                  ->on('users')

                  ->onDelete('cascade')

                  ->onUpdate('cascade');

            $table->timestamps();

    // RELACIÓN DE UNO A MUCHOS - U

    public function videos()

    {

        return $this->hasMany(Video::class);

    }

    // RELACIÓN DE UNO A MUCHOS

    public function posts()

    {

        return $this->hasMany(Post::class);

    }

    // RELACIÓN DE UNO A MUCHOS - V

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    // RELACIÓN DE UNO A MUCHOS

    public function category()

    {

        return $this->belongsTo(Category::class);

    }

    // RELACIÓN DE UNO A MUCHOS - P

    public function user()

    {

        return $this->belongsTo(User::class);

    }

    // RELACIÓN DE UNO A MUCHOS

    public function category()

    {

        return $this->belongsTo(Category::class);

    }

    // RELACIÓN DE UNO A MUCHOS - C

    public function posts()

    {

        return $this->hasMany(Post::class);

    }

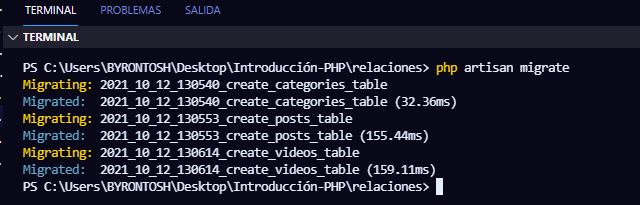
    // RELACIÓN DE UNO A MUCHOS

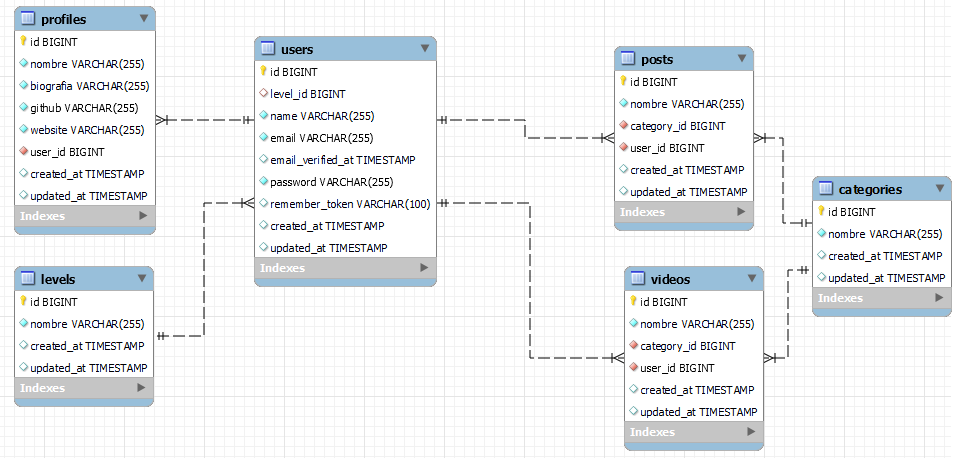
    public function videos()

    {

        return $this->hasMany(Video::class);

    }





INSERT INTO `relaciones`.`categories` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('1', 'backend', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`categories` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('2', 'frontend', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`categories` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('3', 'redes', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`categories` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('4', 'software', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`categories` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('5', 'hardware', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('1', 'el desarrollo de laravel', '1', '1', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('2', 'el desarrollo de nodejs', '1', '1', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('3', 'el desarrollo de React', '2', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('4', 'el desarrollo de Angular', '2', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('5', 'La arquitectura de una PC', '5', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`posts` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('6', 'La arquitectura de una Laptop', '5', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('1', 'el desarrollo de laravel - video', '1', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

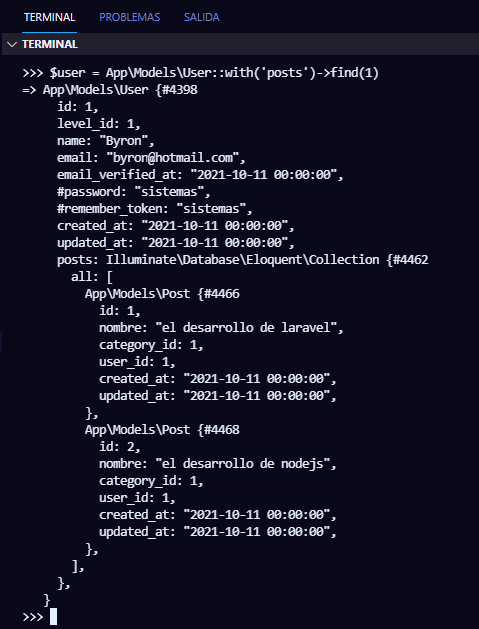
INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('2', 'el desarrollo de nodejs - video', '1', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

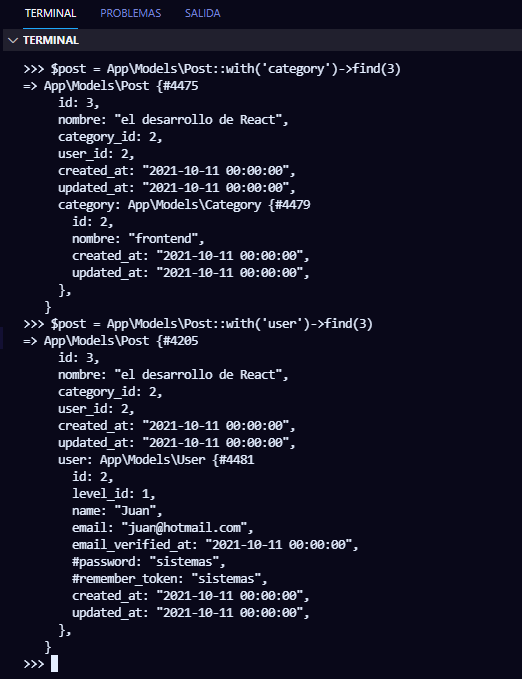
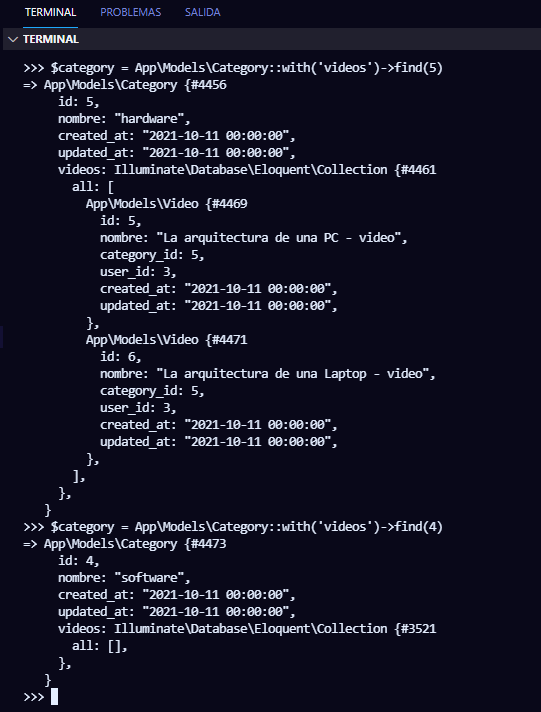
INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('3', 'el desarrollo de React - video', '2', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('4', 'el desarrollo de Angular - video', '2', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('5', 'La arquitectura de una PC - video', '5', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`videos` (`id`, `nombre`, `category\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('6', 'La arquitectura de una Laptop - video', '5', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

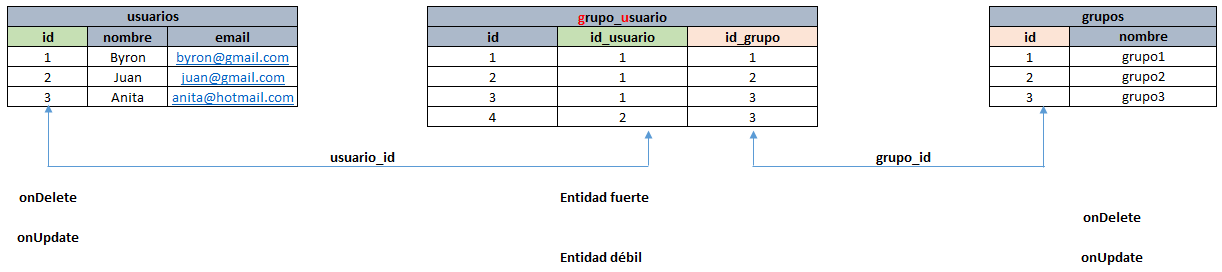




1. Ahora se procede a trabajar con las relaciones

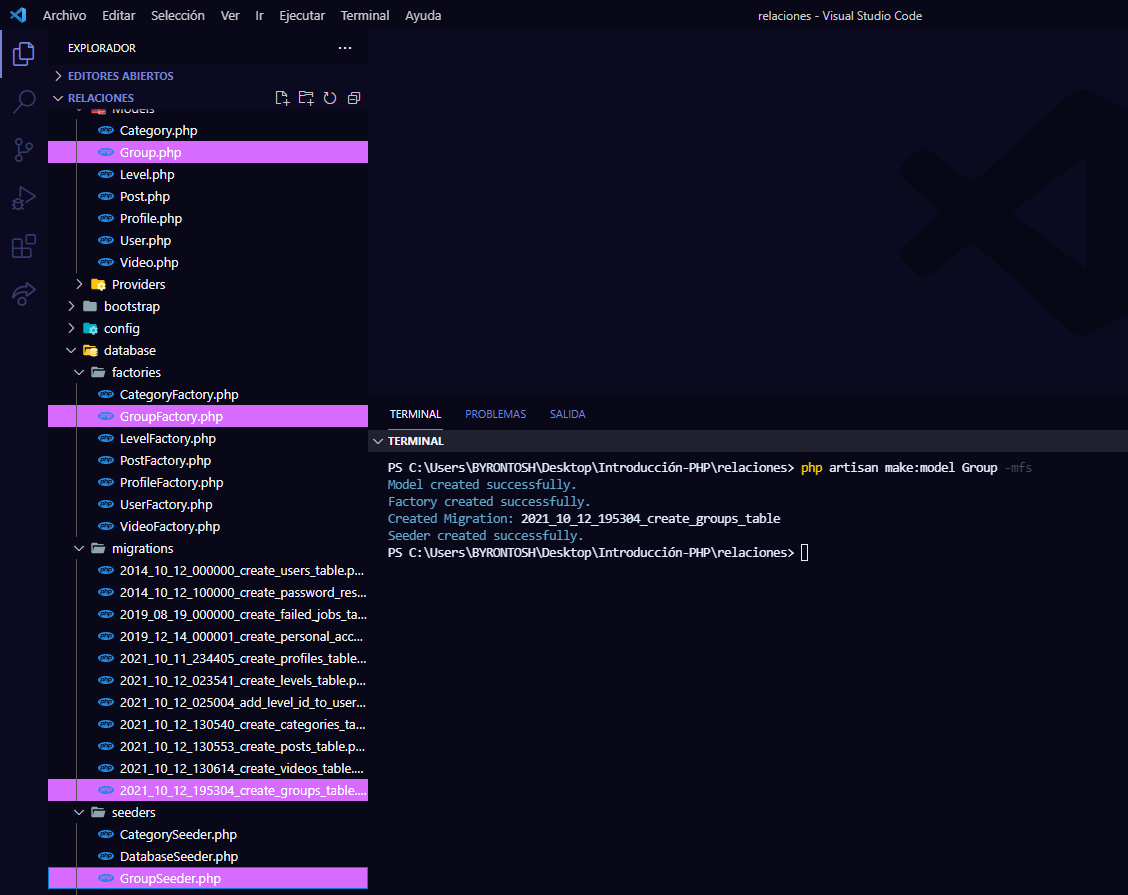
**RELACIÓN DE MUCHOS A MUCHOS**

<https://laravel.com/docs/9.x/eloquent-relationships#many-to-many>



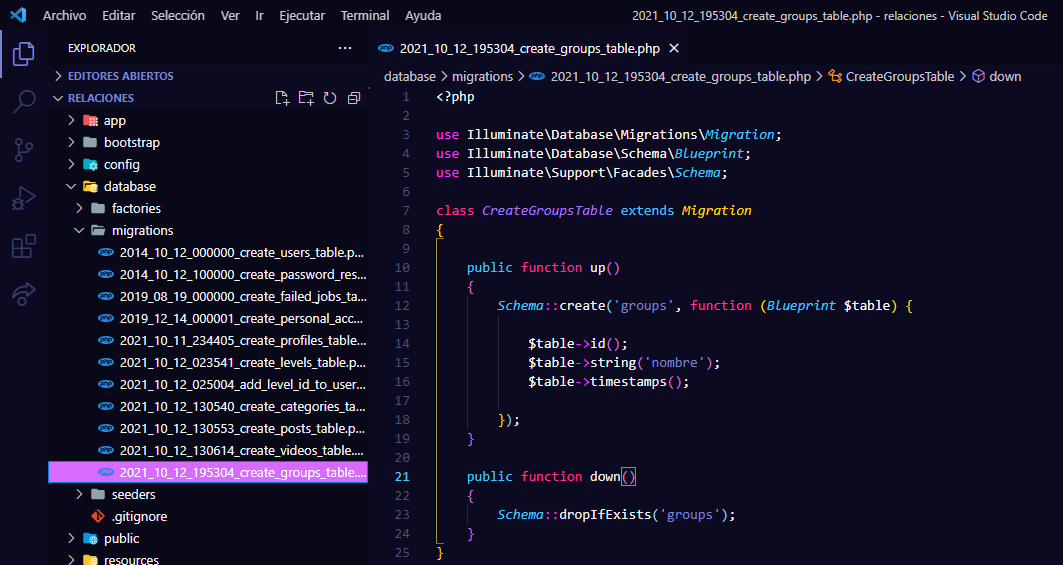
1. Se procede con la creación de un nuevo modelo llamado Group y adicional a ello se crea su migración, factory y sedder con el siguiente comando.

**php artisan make:model Group -mfs**



1. Ahora se procede a definir los campos (nombre de las columnas) para la tabla de la base de datos, esto se lo realiza en la migración del modelo respectivo, recordando que:

**Un usuario puede estar en varios grupos y que un grupo tiene varios usuarios**



            $table->id();

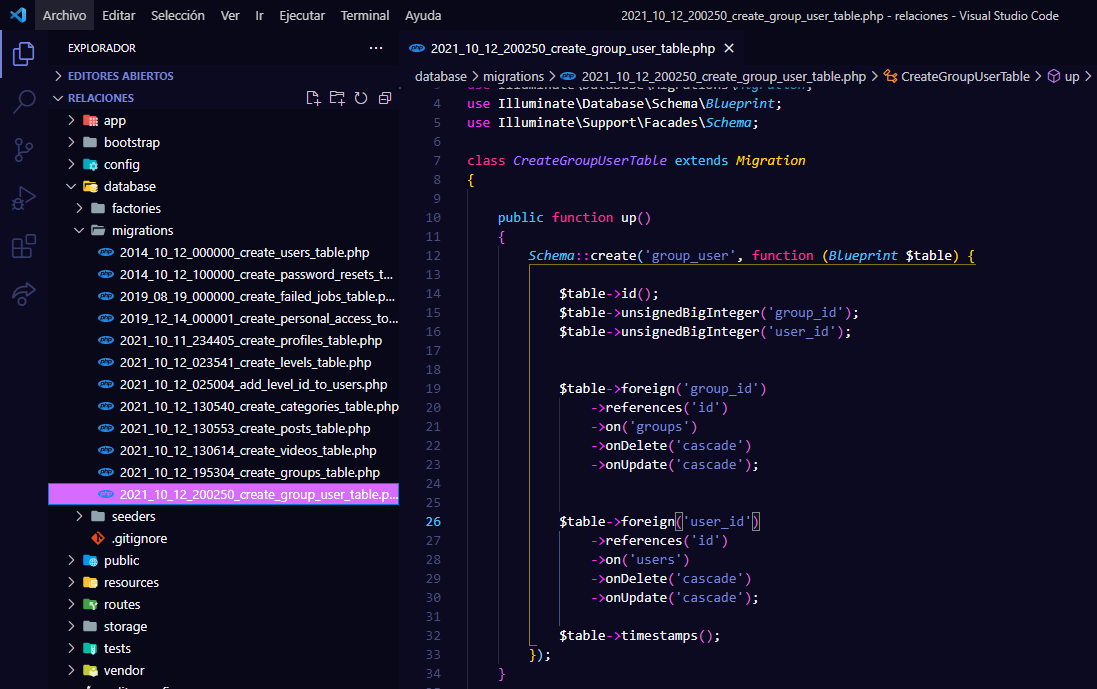
            $table->string('nombre');

            $table->timestamps();

1. No obstante, hay que tener en cuenta que si se trata de una relación de muchos a muchos se debe trabajar con una tabla intermedia o tabla pivote que haga referencia a las dos tablas que son usuarios y grupos. Para lo cual se debe crear una nueva migración, ejecutando el siguiente comando:

**php artisan make:migration create\_group\_user\_table**

<https://laravel.com/docs/9.x/eloquent-relationships#many-to-many-table-structure>



            $table->id();

            $table->unsignedBigInteger('group\_id');

            $table->unsignedBigInteger('user\_id');

            $table->foreign('group\_id')

                ->references('id')

                ->on('groups')

                ->onDelete('cascade')

                ->onUpdate('cascade');

            $table->foreign('user\_id')

                ->references('id')

                ->on('users')

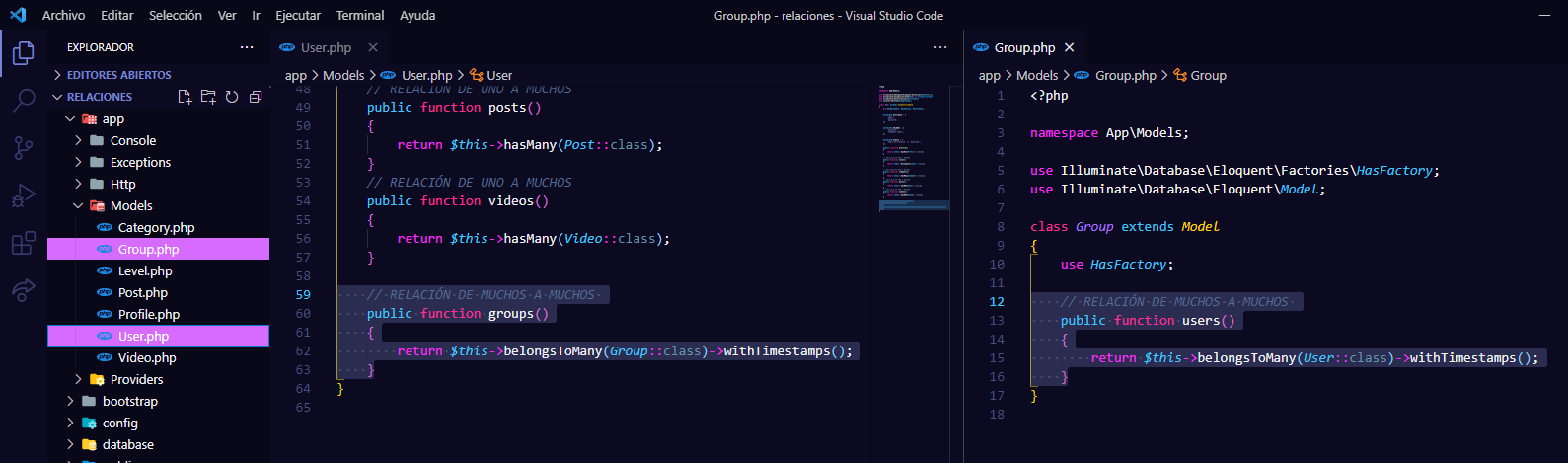
                ->onDelete('cascade')

                ->onUpdate('cascade');

            $table->timestamps();

1. Ahora se debe hacer el procedimiento a nivel de **Eloquent de Laravel**, teniendo en cuenta que:

**Un usuario puede estar en varios grupos y que un grupo tiene varios usuarios**



    // RELACIÓN DE MUCHOS A MUCHOS

    public function groups()

    {

        return $this->belongsToMany(Group::class)->withTimestamps();

    }

    // RELACIÓN DE MUCHOS A MUCHOS

    public function users()

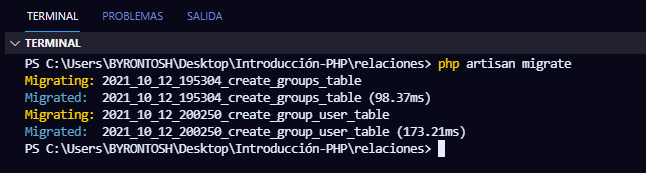
    {

        return $this->belongsToMany(User::class)->withTimestamps();

    }

1. Ahora se procede a ejecutar la migración y verificar si realmente esta correcta la implementación de la relación muchos a muchos, ejecutando el siguiente comando.

**php artisan migrate**



INSERT INTO `relaciones`.`groups` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('1', 'grupo-1', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`groups` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('2', 'grupo-2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`groups` (`id`, `nombre`, `created\_at`, `updated\_at`) VALUES ('3', 'grupo-3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('1', '1', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

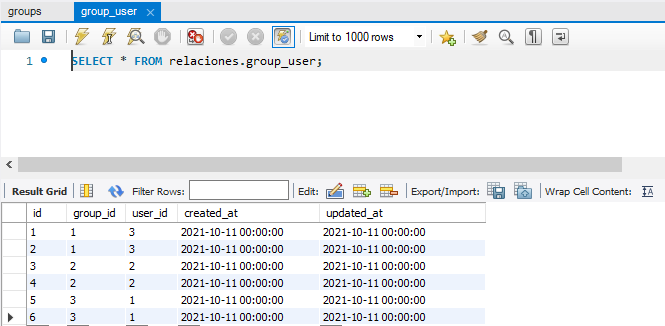
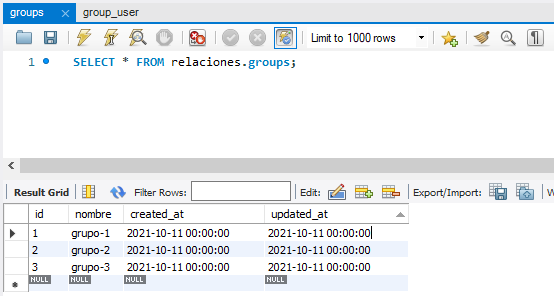
INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('2', '1', '3', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

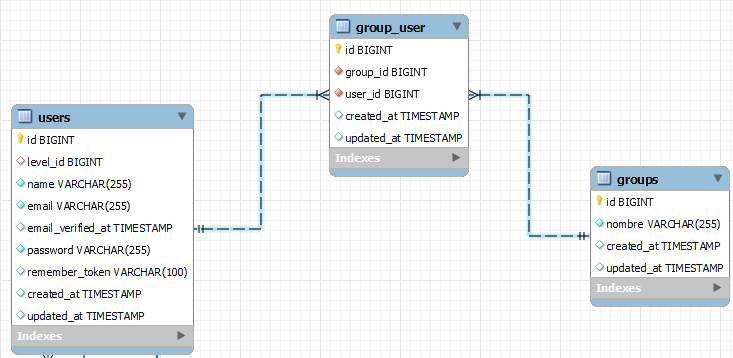
INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('3', '2', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('4', '2', '2', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('5', '3', '1', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

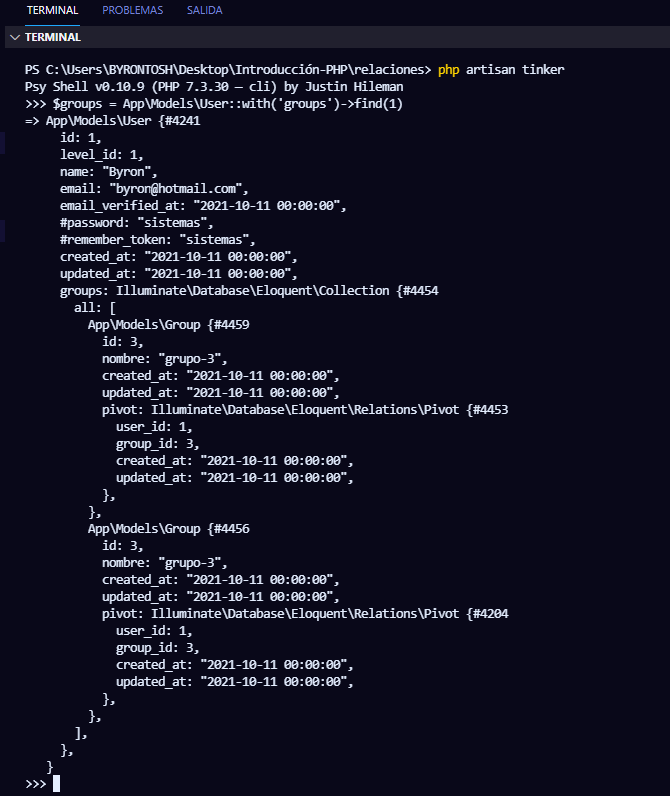
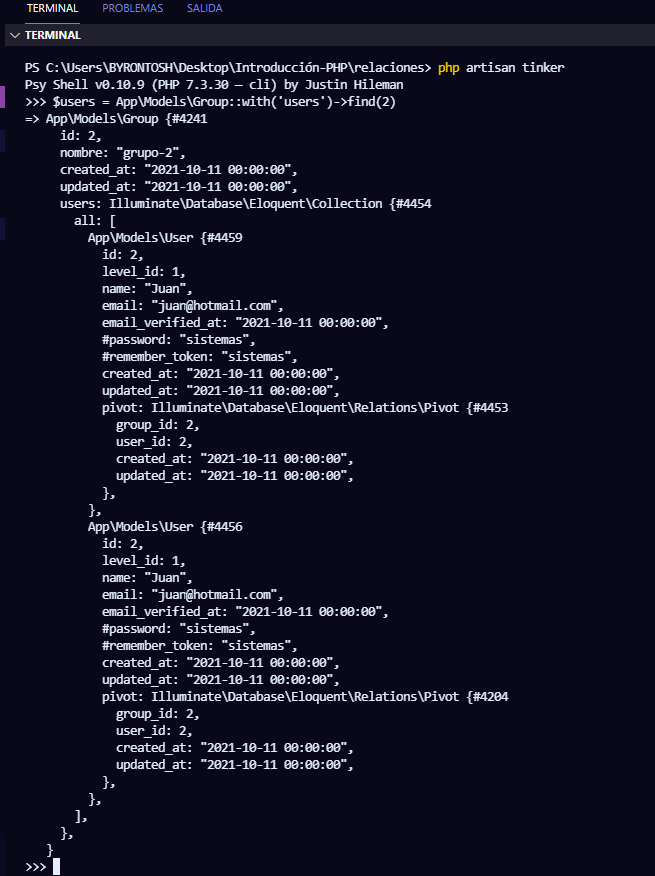
INSERT INTO `relaciones`.`group\_user` (`id`, `group\_id`, `user\_id`, `created\_at`, `updated\_at`) VALUES ('6', '3', '1', '2021-10-11 00:00:00', '2021-10-11 00:00:00');

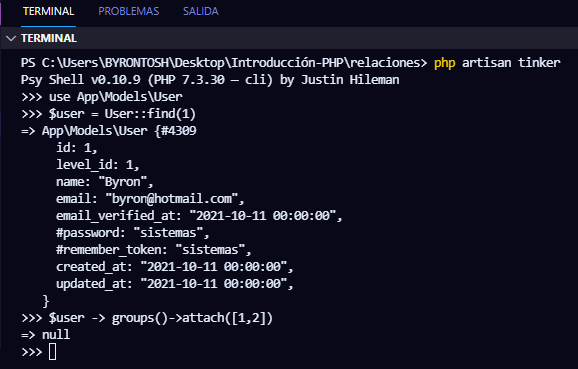
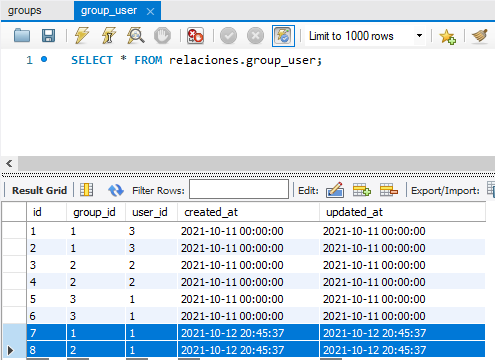




1. Para probar se puede verificar ejecutando

**php artisan tinker**



<https://github.com/Byrontosh/04_Relaciones>

***TAREA DESAFÍO***



1. Realizar lo siguiente:

Crear 4 relaciones adicionales a las vistas en clase, las culaes están divididas de la siguiente manera:

* 1 relación de uno a uno
* 2 relaciones de uno a muchos
* 1 relación de muchos a muchos

Recueda llenar con datos cada una de las tablas en la Base de Datos y hacer consultas con tinker.

Referencias bibliográficas

* <https://geekflare.com/es/laravel-eloquent-model-relationship/>
* <https://laravel.com/docs/9.x/eloquent-relationships>
* [https://www.oulub.com/docs/laravel/es-es/eloquent#retrieving-models](https://www.oulub.com/docs/laravel/es-es/eloquent" \l "retrieving-models)