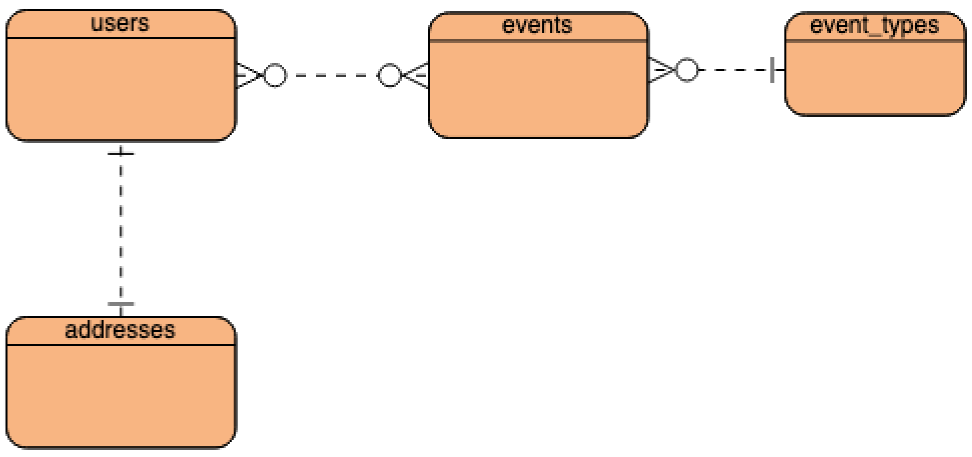
Inés Larrañaga Fernández de Pinedo	**CSC Jesuitas - Logroño**
	DWES

Table of Contents

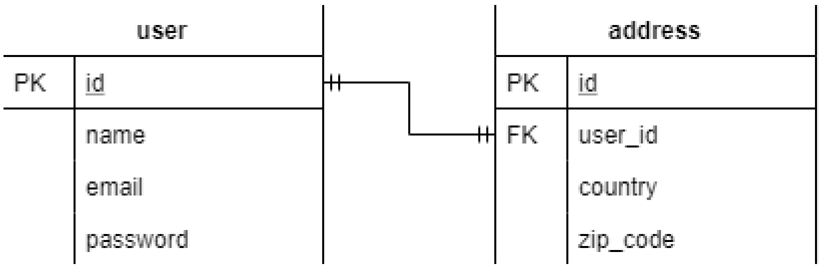
Eloquent ORM relationships

As backend developers, we need a DB to store data and we are already familiar with the 3 types of relationships:

- One-to-One
- One-to-Many
- Many-to-Many



One-to-one relationship

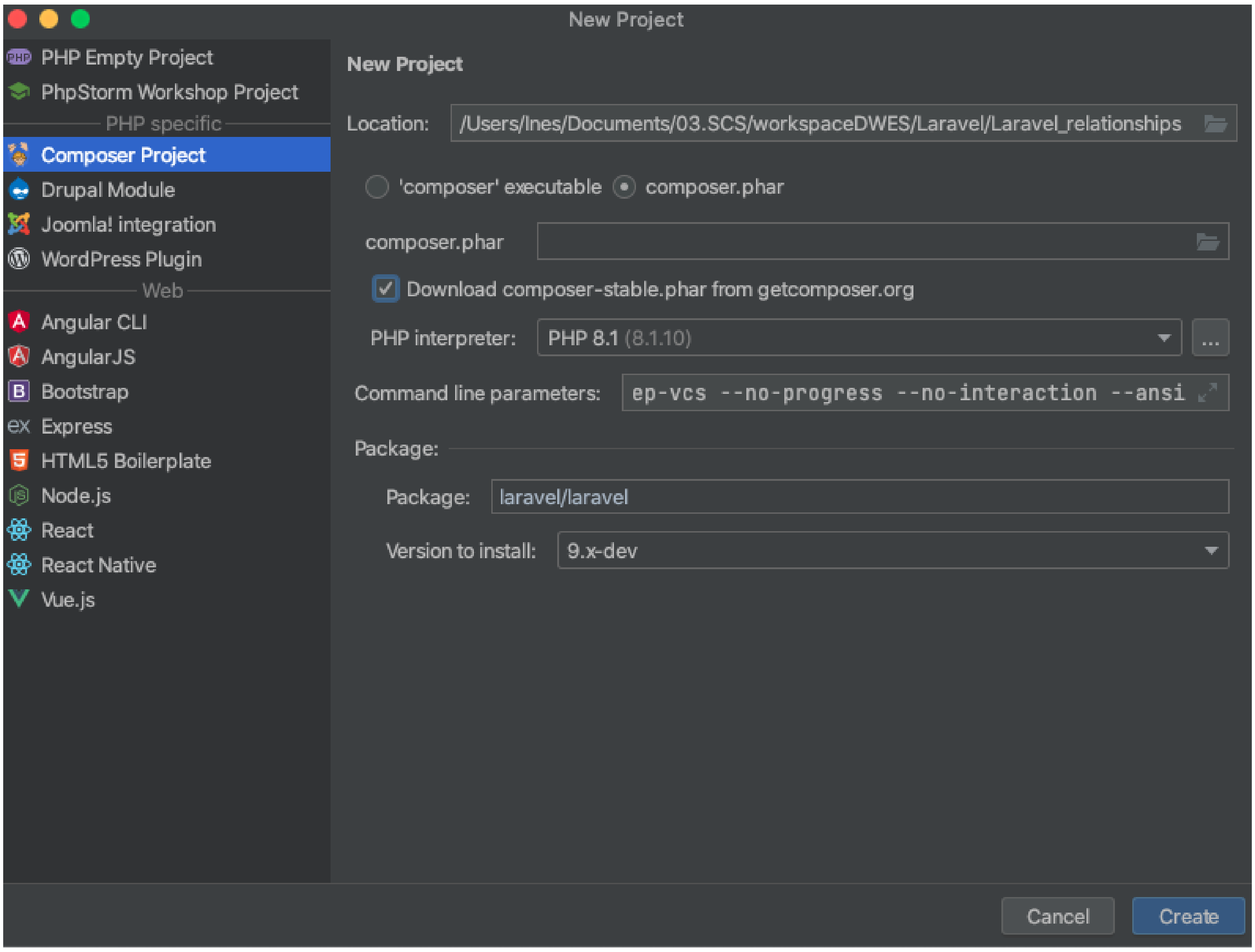


Seeing the picture, we can say that one user can only have one address, and one address must be owned by one user.

If we look at the address table, there is one column named user_id in the form of a Foreign Key. The user_id column must have the same value as one of the id in the user table.

Laravel Preparation

Using PHPStorm or the command line create a laravel project called `Laravel_relationships`.



New Project

Location: /Users/Ines/Documents/03.SCS/workspaceDWES/Laravel/Laravel_relationships

'composer' executable [✓ composer.phar]

composer.phar:

[✓] Download composer-stable.phar from getcomposer.org

PHP interpreter: PHP 8.1 (8.1.10)

Command line parameters: `ep-vcs --no-progress --no-interaction --ansi`

Package:

Package: laravel/laravel

Version to install: 9.x-dev

Database Migration

The first thing that needs to be done before storing any data into the database is, of course, creating the tables.

If you open app/database/migrations/, there is already a user_table migration. That will be enough for us.

However, we need to create the new `addresses` table. In order to do so:

```
php artisan make:migration create_addresses_table
```

```
public function up()
    {
        Schema::create('addresses', function (Blueprint $table) {
            $table->id();
            $table->bigInteger('user_id')->unsigned();
            $table->string('country');
            $table->string('zipcode');
            $table->timestamps();
            $table->foreign('user_id')
                ->references('id')
                ->on('users')
                ->onDelete('CASCADE');
        });
    }
```

At the end of the method, we create a foreign key for the `user_id` column, with having a reference id in the users table. `onDelete (‘cascade’)` means that when a user in the users table is deleted, the address related to that user will be deleted as well.

Now we are going to launch the migration command:

```
php artisan migrate
```

In case you have an error, check your ´.env´ file: "DB_HOST=127.0.0.1".

Check your new tables using DataGrip.

Eloquent ORM (Model)

Now, that we have create our tables, we need to create our model files:

```
php artisan make:model User
php artisan make:model Address
```

The new model files should appear under app/Models/ folder.

Let’s modify the User model first, adding the relationship. So, at the end of the User model add this method:

```
public function address(){
       return $this->hasOne('App\Models\Address');
}
```

And open Address model and include:

```
public function user(){
       return $this->belongsTo('App\Models\User');
}
```

Routing

These is the corresponding content por the `api.php` routing file with the services we want to implement and test in our controllers afterwards:

```
Route::controller(UserController::class)->group(function(){
    Route::post('register', 'register');
    Route::get('user/{user}', 'show');
    Route::get('user/{user}/address', 'show_address');
});

Route::controller(AddressController::class)->group(function() {
    Route::post('address', 'store');
    Route::get('address/{address}', 'show');
    Route::get('address/{address}/user', 'show_user');
});
```

Controllers

Let´s create two controller files:

```
php artisan make:controller UserController
php artisan make:controller AddressController
```

New files should appear in app/Http/Controllers/. Let’s modify the `UserController` first:

```
class UserController extends Controller
{
    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6',
            'c_password' => 'required|same:password',
        ]);

        if($validator->fails()){
            return response()->json($validator->messages(), 400);
        }
        $user = User::create([
            'name' => $request->get('name'),
            'email' => $request->get('email'),
            'password' => Hash::make($request->get('password')),
        ]);
        return response()->json(['message'=>'User Created','data'=>$user],200);
    }

    public function show(User $user)
    {
        return response()->json(['message'=>'','data'=>$user],200);
    }

    public function show_address(User $user)
    {
        return response()->json(['message'=>'','data'=>$user->address],200);
    }
}
```

> TASK1: Test the 3 methods using postman.

> TASK2: Complete the AddressController class, with next methods and the corresponding postman tests:
>
> ```php
>     public function store(Request $request)
>     {
>
>     }
>
>     public function show(Address $address)
>     {
>
>     }
>
>     public function show_user(Address $address)
>     {
>
>     }
> ```