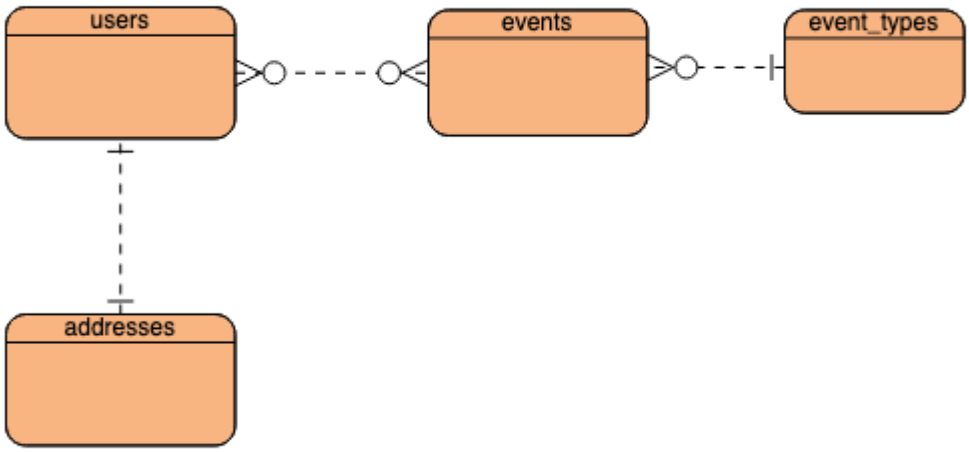
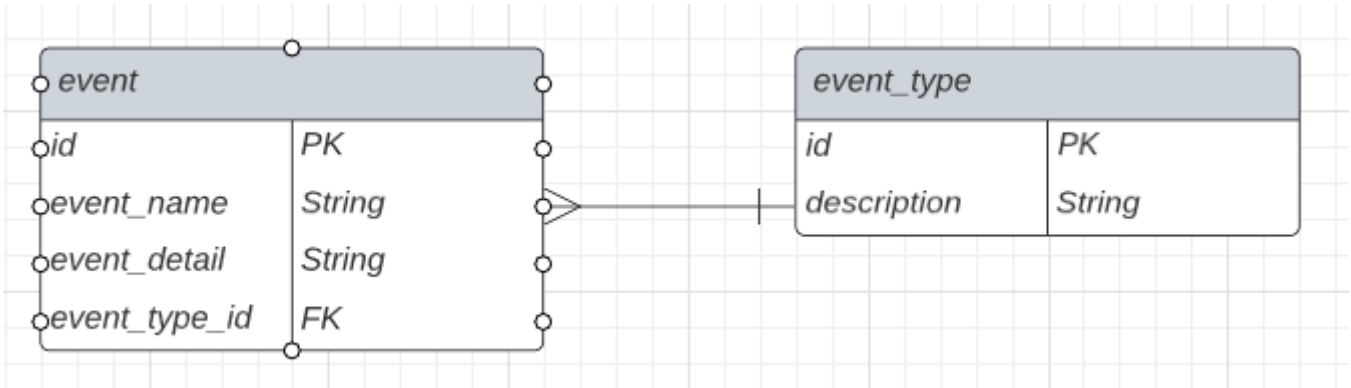
Inés Larrañaga Fernández de Pinedo	CSC Jesuitas - Logroño
	DWES

Table of Contents

One-to-many relationship



From the previous image, we can say that the last relationship one-to-many we need to implement is:



Seeing the picture we can say that an event belongs to a type of event and that a type of event has many events.

Database Migration

Now, we are going to create a new table with a the events type information. We are going to create the migration file first:

```
php artisan make:migration create_event_types_table
```

Inside of that script:

```
Schema::create('event_types', function (Blueprint $table) {
            $table->id();
            $table->string('description');
            $table->timestamps();
});
```

Execute the migration command NOW!

Now, we need to change our DB model including the corresponding FK inside `event` table. In order to do so, we create a migration file that changes the existing `event` table:

```
php artisan make:migration modify_events_table --table=events
```

And inside of the script:

```
Schema::table('events', function (Blueprint $table) {
            $table->bigInteger('event_type_id')->unsigned();
            $table->foreign('event_type_id')->references('id')-
>on('event_types');
        });
```

Execute again the migration command. Take into account that the migration will not work in case you have related data inside `events` table.

Eloquent ORM (Model)

We need to create a new model for our `event_type` entity:

```
php artisan make:model EventType
```

> Task1: On your own complete the EventType model class, so that you include the field `description` as fillable, and you will have to create a new `events()` method, in order to create the oneToMany relationship. Hint: use `hasMany` method.

On the other side, we need to add a new method called `eventType()` where it is going to return the related EventType:

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

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. In this case only one more:

```
//EventsController routing
Route::controller(EventTypeController::class)->group(function() {
    Route::get('type/{type}', 'listEvents');
});
```

Controllers

Let´s create the new controller file:

```
php artisan make:controller EventTypeController
```

New files should appear in app/Http/Controllers/. Let's modify the `EventTypeController` first in order to include the new method:

```
public function listEvents(EventType $type){
       $events = $type->events;
       return response()->json(['message'=>null,'data'=>$events],200);
}
```

Test the method properly using Postman.