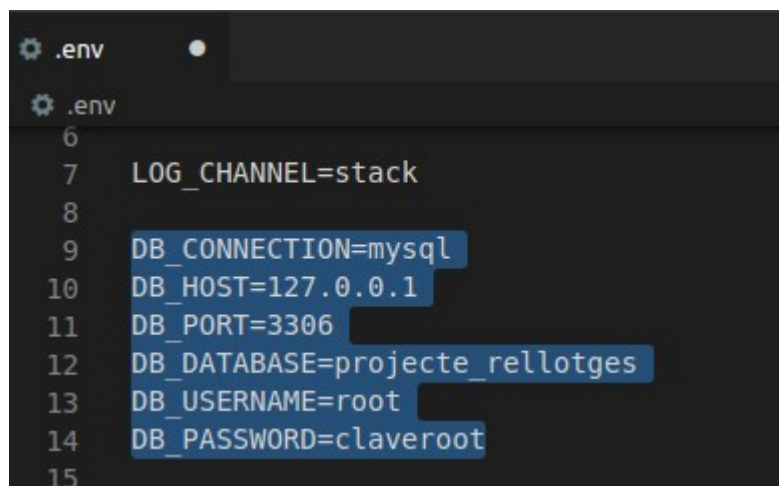


M07 - GUIA PROJECTE PRÀCTICA04

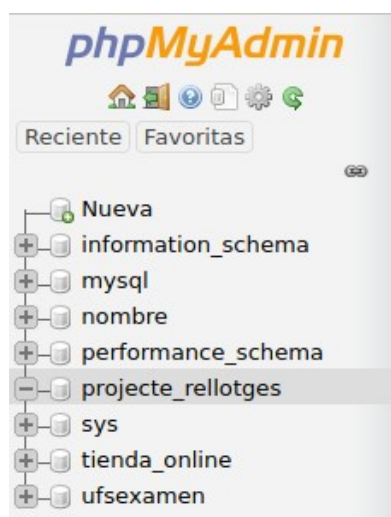
Exercici 1 - Configuració de la base de dades imigracions (1 punt)

En primer lloc configurarem correctament la base de dades. Així doncs, actualitzarem el fitxer `.env` per a indicar que farem servir una base de dades tipus MySQL anomenada "nom_del_vostre_projecte" juntament amb el nom d'usuari i contrasenya d'accés.



```
.env
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=projecte_rellotges
13 DB_USERNAME=root
14 DB_PASSWORD=claveroot
15
```

A continuació obrim PHPMyAdmin i crearem la nova base de dades de nom "nom_del_vostre_projecte". Per a comprovar que tot s'ha configurat correctament, obrim un terminal a la carpeta del nostre projecte i executem la comanda que crea la taula de migracions. Si tot va bé, refresquem dades des de PHPMyAdmin i comprovem que s'ha creat aquesta taula dins la nostra nova base de dades.



Però surt un error:

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/projecte_rellotges$ php artisan migrate:install

Illuminate\Database\QueryException : could not find driver (SQL: create table `migrations` (`id` int unsigned not null auto_increment primary key, `migration` varchar(255) not null, `batch` int not null) default character set utf8mb4 collate 'utf8mb4_unicode_ci')

at /home/adria/Escritorio/Servidor/miweb/projecte_rellotges/vendor/laravel/framework/src/Illuminate/Database/Connection.php:669
665|         // If an exception occurs when attempting to run a query, we'll format
the error
666|         // message to include the bindings with SQL, which will make this exception a
667|         // lot more helpful to the developer instead of just the database's errors.
668|         catch (Exception $e) {
> 669|             throw new QueryException(
670|                 $query, $this->prepareBindings($bindings), $e

```

Per solucionar-ho hem d'instalar:

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/projecte_rellotges$ sudo apt-get install php-mysql
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  gir1.2-geocodeglib-1.0 libllvm8 libllvm8:i386 ubuntu-web-launchers
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  php7.2-mysql
Se instalarán los siguientes paquetes NUEVOS:
  php-mysql php7.2-mysql
```

Anem a crear un nou usuari a phpMyAdmin:

Bases de datos

SQL

Estado actual

Cuentas de usuarios

Exportar

Importar

Configuración

Agregar cuenta de usuario

Información de la cuenta

Nombre de usuario:

Use el campo de texto:

projecte_rellotge

Nombre de Host:

Cualquier servidor

%

Contraseña:

Use el campo de texto:

.....

Strength: Extremadamente débil

Debe volver a escribir:

.....

Complemento de autenticación

Autenticación de MySQL nativo

Generar contraseña:

Generar

Base de datos para la cuenta de usuario

☒ Crear base de datos con el mismo nombre y otorgar todos los privilegios.

☐ Otorgar todos los privilegios al nombre que contiene comodín (username_%).

Privilegios globales

☒ Seleccionar todo

Nota: Los nombres de los privilegios de MySQL están expresados en inglés.

☒ Datos

☒ SELECT

☒ INSERT

☒ UPDATE

☒ DELETE

☒ FILE

☒ Estructura

☒ CREATE

☒ ALTER

☒ INDEX

☒ DROP

☒ CREATE TEMPORARY TABLES

☒ SHOW VIEW

☒ CREATE ROUTINE

☒ ALTER ROUTINE

☒ EXECUTE

☒ CREATE VIEW

☒ EVENT

☒ TRIGGER

☒ Administración

☒ GRANT

☒ SUPER

☒ PROCESS

☒ RELOAD

☒ SHUTDOWN

☒ SHOW DATABASES

☒ LOCK TABLES

☒ REFERENCES

☒ REPLICATION CLIENT

☒ REPLICATION SLAVE

☒ CREATE USER

Límites de recursos

Nota: si cambia los parámetros de estas opciones a 0

MAX QUERIES PER HOUR

0

MAX UPDATES PER HOUR

0

MAX CONNECTIONS PER HOUR

0

MAX USER_CONNECTIONS

0

SSL

☒ REQUIRE NONE

☐ REQUIRE SSL

☐ REQUIRE X509

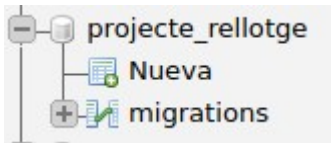
☐ SPECIFIED

Tornem a modificar el arxiu .env:

```
DB_CONNECTION=mysql
DB_HOST=172.18.0.1
DB_PORT=3306
DB_DATABASE=projecte_rellotge
DB_USERNAME=projecte_rellotge
DB_PASSWORD=projecte_rellotge
```

I ja està:

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/projecte_rellotges$ php artisan migrate:install
Migration table created successfully.
```



Ara anem a crear la taula que farem servir per emmagatzemar el catàleg d'ítems del vostre projecte(en aquest exemple, pel·lícules). Executem la comanda d' Artisan per a crear la migració, queportarà per nom "create_movies_table", per a la taula "movies". Un cop creat edita aquestfitxer per afegir tots els camps necessaris, que serien (recordeu que heu de personalitzar els camps al vostre projecte!):

camp	tipus	Valor per defecte
id	Autoincremental	
title	string	
year	String de longitud 8	
director	String de longitud 64	
pòster	string	
rented	booleà	false
synopsis	text	
timestamps	Timestamps de Eloquent	

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/projecte_rellotges$ php artisan make:migration create_rellotges_table --create=rellotges
Created Migration: 2020_02_17_072829_create_rellotges_table
```

```

public function up()
{
    Schema::create('rellotges', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('title');
        $table->string('year', 8);
        $table->string('color');
        $table->string('imagen');
        $table->boolean('rented')->default(false);
        $table->string('synopsis');
    });
}

```

Recordeu que en el mètode "down" de la migració has de desfer els canvis que has fet en el mètode "up", en aquest cas seria eliminar la taula.

Finalment executarem la comanda d'Artisan que afegeix les noves migracions i comprovarem aPHPMyAdmin que la taula s'ha creat correctament amb els camps que li hem indicat.

```

adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/proyecto_rellotges
$ php artisan migrate
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.22 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.13 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.05 seconds)
Migrating: 2020_02_17_072829_create_rellotges_table
Migrated: 2020_02_17_072829_create_rellotges_table (0.07 seconds)
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/proyecto_rellotges
$ 

```



The screenshot shows the PHPMyAdmin interface. On the left, a tree view displays the database structure under 'proyecto_rellotge', including tables like 'failed_jobs', 'migrations', 'password_resets', 'rellotges', and 'users'. The 'rellotges' table is selected. On the right, a table view shows the columns: 'id', 'title', 'year', 'color', 'imagen', 'rented', and 'synopsis'. Below the table view, there is a section titled 'Operaciones sobre los resultados de la consulta' with a button labeled 'Crear vista'.

Exercici 2 - Model de dades (0.5 punts)

En aquest exercici crearem el model de dades associat amb la taula "movies" . Per això farem servir la comanda apropiada de Artisan per a crear el model anomenat Movie.

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/proyecto_rellotges
$ php artisan make:model rellotges
Model created successfully.
```

Un cop creat aquest fitxer l'obrirem i comprovarem que el nom de la classe sigui el correcte i que hereti de la classe Model. I ja està, no cal fer res més, el cos de la classe pot estar buit ({}), tota la resta es fa automàticament!

```
rellotges.php  x
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class rellotges extends Model
{
    //
}
```

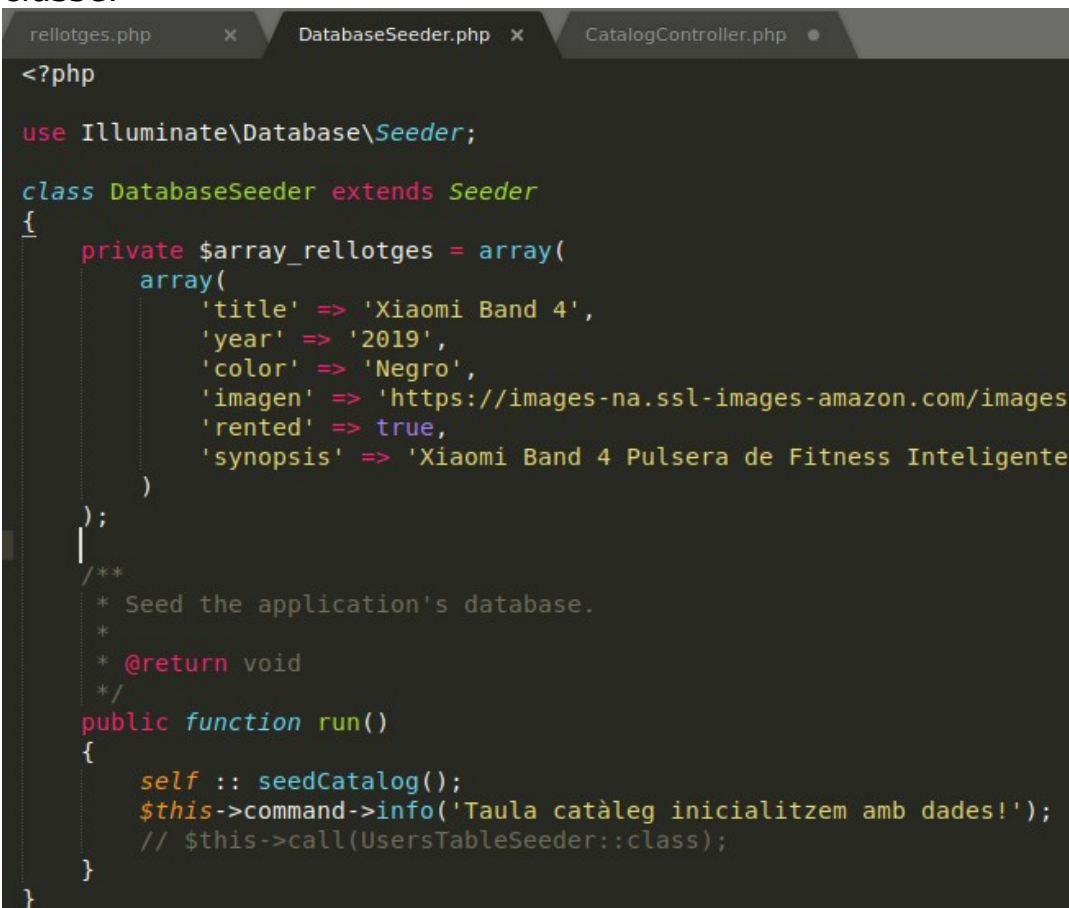

Exercici 3 - Llavors (1 punt)

Ara procedirem a omplir la taula de la base de dades amb les dades inicials. Per a fer-ho, caldrà editar el fitxer de llavors situat a "database/seeds/DatabaseSeeder.php" i seguirem els següents passos:

- Crearem un mètode privat (dins de la mateixa classe) anomenat seedCatalog() que s'haurà de cridar des del mètode run de la següent forma:

```
public function run()
{
    self :: seedCatalog();
    $this->command->info('Taula catàleg inicialitzem amb dades!');
    // $this->call(UsersTableSeeder::class);
}
```

- Movem l'array de "pel·lícules" (en realitat no serà "pel·lícules" sinó el nom que hagueu posat a l'array d'ítems del vostre projecte) que havíem copiat hardcoded dins del controlador CatalogController a la coneguda com a classe de llavors, DatabaseSeeder.php, crearem el model de dades guardant-ho de la mateixa forma, com a variable privada de la classe.



```
<?php
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    private $array_rellotges = array(
        array(
            'title' => 'Xiaomi Band 4',
            'year' => '2019',
            'color' => 'Negro',
            'imagen' => 'https://images-na.ssl-images-amazon.com/images',
            'rented' => true,
            'synopsis' => 'Xiaomi Band 4 Pulsera de Fitness Inteligente'
        )
    );

    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        self :: seedCatalog();
        $this->command->info('Taula catàleg inicialitzem amb dades!');
        // $this->call(UsersTableSeeder::class);
    }
}
```




- Dins del nou mètode `seedCatalog()` realitzem les següents accions:
 - En primer lloc esborrem el contingut de la taula `movies` amb `DB::table('movies')->delete();`
 - I a continuació afegim el següent codi:

```
public function seedCatalog(){
    DB::table('rellotges')->delete();
    foreach ($this->array_rellotges as $rellotge) {
        # code...
        $r=new Rellotge;
        $r->title=$rellotge['title'];
        $r->year=$rellotge['year'];
        $r->color=$rellotge['color'];
        $r->imagen=$rellotge['imagen'];
        $r->rented=$rellotge['rented'];
        $r->synopsis=$rellotge['synopsis'];
        $r->save();
    }
}
```

Finalment haurem d'executar la comanda d'Artisan que processa les llavors i un cop realitzat obrirem PHPMyAdmin per a comprovar que s'ha omplert la taula `movies` amb el llistat de "pel·lícules".

Si t'apareix l'error " Fatal error: Class 'Movie' not found " revisa si has indicat l'espai de noms de el model que utilitzaràs (use `App\Movie;`).

```
adria@adria-HP-ProDesk-600-G1-SFF:~/Escritorio/Servidor/miweb/proyecto_rellotges$ php artisan db:seed
Taula catàleg inicialitzem amb dades!
Database seeding completed successfully.
```

	id	created_at	updated_at	title	year	color	imagen	rented	synopsis
<input type="checkbox"/>  Editar  Copiar  Borrar	2	2020-02-17 08:43:04	2020-02-17 08:43:04	Xiaomi Band 4	2019	Negro	https://images-na.ssl-images-amazon.com/images/I/5...	1	Xiaomi Band 4 Pulsera de Fitness Inteligente Monit...

Exercici 4 - Ús de la base de dades (1 punt)

En aquest últim exercici actualitzarem els mètodes del controlador CatalogController perquè obtinguin les dades des de la base de dades. Seguirem els següents passos:

- Modificar el mètode getIndex perquè obtingui tota la llista de pel·lícules des de la base de dades usant el model Movie i que se la passi a la vista.

```
public function getIndex () {  
    $rellotges = rellotges::all();  
    return view('catalog.index', ['la_meva_llista_de_rellotges' => $rellotges->  
        toArray()]);  
}
```

Rellotgeria



Xiaomi Band

4

- Modificar el mètode `getShow` perquè obtingui la pel·lícula passada per paràmetre usant el mètode `findOrFail` i se la passi a la vista.

```
// Vista detall rellotge
public function getShow ($id){

    $rellotges = rellotges::findOrFail($id);

    return view('catalog.show', ['id'=>$rellotges->toArray()]);
}
```

localhost:8000/catalog/show/2

Relotgeria



Xiaomi Band 4 ~ 2019

Color: Negro

Descripció:

Xiaomi Band 4 Pulsera de Fitness Inteligente Monitor de Ritmo cardíaco 135 mAh Pantalla Color Bluetooth 5.0 más Reciente 2019 (Negro).

Estat: Rellotge disponible!

Llogar rellotge Editar Rellotge Tornar al llistat

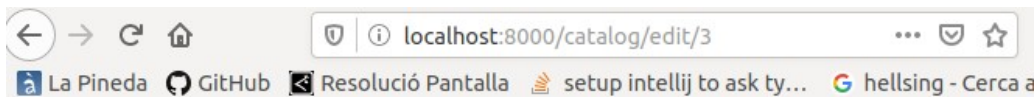
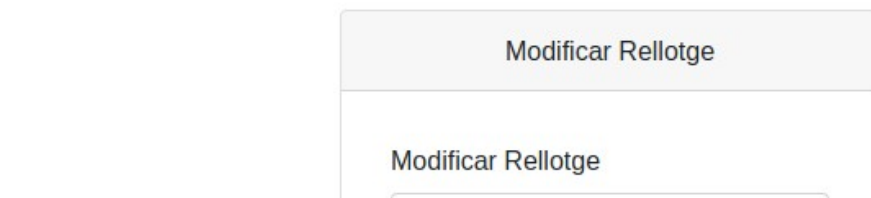
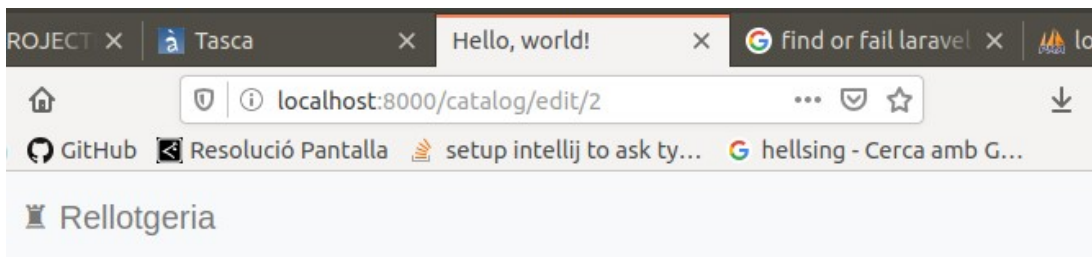
404 | Not Found

- Modificar el mètode `getEdit` perquè obtingui la pel·lícula passada per paràmetre usant el mètode `findOrFail` i se la passi a la vista.

```
// Modificar rellotge
public function getEdit ($id){

    $rellotges = rellotges::findOrFail($id);

    return view('catalog.edit', ['id'=>$rellotges->toArray()]);
}
```



Si al provar-ho t'apareix l'error " Class 'App \ Http \ Controllers \ Movie' not found "revisa si has indicat l'espai de noms de el model que utilitzaràs (use App\Movie;).

Ja no necessitem més l'array de pel·lícules hardcoded (\$arrayPelículas) que havíem posat al controlador, així que definitivament ja podem comentar o eliminar aquesta array!

Ara haurem d'actualitzar les vistes perquè en lloc d'accedir a les dades de l'array les obtingui del·lobjecte amb la "pel·lícula". Canviarem a tot arreu on hàgim posat \$película['campo'] per \$película->campo.

A més, a la vista catalog/index.blade.php, en lloc d'utilitzar l'índex de la matriu (\$key) com identificador per crear l'enllaç a catalog/show/{id}, haurem d'utilitzar el camp id de la pel·lícula (\$película->id). Farem el mateix a la vista catalog/show.blade.php, per agenerar l'enllaç d'editar pel·lícula haurem d'afegir l'identificador de la pel·lícula a la ruta catalog/edit.

```
index.blade.php  x DatabaseSeeder.php  x show.blade.php  x
@extends('layouts.master')
@section('content')

<div class="row">
    @foreach( $la_meva_llista_de_rellotges as $key => $rellotge )
        <div class="col-xs-6 col-sm-4 col-md-3 text-center">
            <a href="{{ url('/catalog/show/' . $rellotge['id'] ) }}">
                
                <h4 style="min-height:45px;margin:5px 0 10px 0">
                    {{ $rellotge['title'] }}
                </h4>
            </a>
        </div>
    @endforeach
</div>

@stop
```