1. Requisitos del sistema

Para este proyecto he creado varios CRUDs: Plataformas, Estados, Géneros que van a estar relacionados con el CRUD de Juegos. Los he creado mediante el crud-generator de laravel. Ésto nos ayuda a crear de forma rápida la estructura básica de los CRUDs.
Para esto he necesitado:

- **Xampp**: servidor local
- **Composer**: gestor de dependencias PHP
- **Node.js**: paquete de librerías en javascript

2. Creación del proyecto

Primero creé una base de datos llamada *coleccion* en el gestor PHPMyAdmin.
Después hay que crear el proyecto. Mediante la consola nos situamos en la carpeta htdocs de xampp y usamos el siguiente comando:

- `composer create-project laravel/laravel gameloggd`

Para comprobar que se ha creado correctamente lo ejecutamos mediante el siguiente comando:

- `php artisan serve`

Tenemos que modificar el nombre de la base de datos en el archivo .env del proyecto:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=coleccion
DB_USERNAME=root
DB_PASSWORD=
```

3. Creamos las migraciones

A la hora de crear las migraciones hay que tener en cuenta el orden de creación. Siempre se creará de última la tabla que contenga el campo relacional. En este caso la última tabla que tenemos que crear es *Juegos*.
Nos situamos en la carpeta del proyecto y usamos el comando `php artisan make:migration` por cada tabla que queramos crear:

- `php artisan make:migration plataformas`
- `php artisan make:migration generos`
- `php artisan make:migration estados`
- `php artisan make:migration juegos`

Creamos en las migraciones que acabamos de crear las columnas que van a tener las tablas. Estos archivos se encuentran en la carpeta *database/migrations* del proyecto.

- **Plataformas**:

```php
/**
 * Run the migrations.
 */
public function up()
{
    Schema::create('plataformas',function(Blueprint $table){
        $table->engine='InnoDB';
        $table->bigIncrements('id');
        $table->string('nombre');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::dropIfExists('plataformas');
}
```

- **Géneros**

```php
/**
 * Run the migrations.
 */
public function up()
{
    Schema::create('generos',function(Blueprint $table){
        $table->engine='InnoDB';
        $table->bigIncrements('id');
        $table->string('nombre');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::dropIfExists('generos');
}
```

- **Estados**

```php
/**
 * Run the migrations.
 */
public function up()
{
    Schema::create('estados',function(Blueprint $table){
        $table->engine='InnoDB';
        $table->bigIncrements('id');
        $table->string('nombre');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::dropIfExists('estados');
}
```

- **Juegos**

```php
/**
 * Run the migrations.
 */
public function up()
{
    Schema::create('juegos',function(Blueprint $table){
        $table->engine='InnoDB';
        $table->bigIncrements('id');
        $table->string('titulo');
        $table->integer('anho',false,true);
        $table->bigInteger('plataforma_id')->unsigned();
        $table->bigInteger('genero_id')->unsigned();
        $table->bigInteger('estado_id')->unsigned();
        $table->foreign('plataforma_id')->references('id')->on('plataformas')->onDelete('cascade');
        $table->foreign('genero_id')->references('id')->on('generos')->onDelete('cascade');
        $table->foreign('estado_id')->references('id')->on('estados')->onDelete('cascade');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 */
public function down()
{
    Schema::dropIfExists('juegos');
}
```

4. Autenticación

He utilizado UI con bootstrap. Hay que realizar en orden los siguientes comandos dentro del proyecto:

- `composer require laravel/ui`
- `php artisan ui bootstrap --auth`
- `npm install`
- `npm run dev` (Una vez usemos este comando hay que dejarlo ejecutándose junto a `php artisan serve`)

Cuando alguien se registre se guardarán los usuarios y contraseñas en la tabla users que se ha creado automáticamente al crear el proyecto.

5. Creamos los CRUDs.

El crud-generator nos va a crear todo automáticamente, los modelos, controladores y vistas. Ejecutamos estos comandos situados en la carpeta del proyecto:

- `composer require ibex/crud-generator --dev`
- `php artisan vendor:publish --tag=crud`

Y ahora, por cada tabla que hemos creado, ejecutamos el comando `php artisan make:crud nombreTabla`.

6. Acceso a los cruds

En el archivo *web.php* dentro de la carpeta *routes* tenemos que establecer una ruta para cada uno de nuestros cruds.

```
Route::resource('juegos', App\Http\Controllers\JuegoController::class)->middleware('auth'); //middlewa
Route::resource('generos', App\Http\Controllers\GeneroController::class)->middleware('auth');
Route::resource('plataformas', App\Http\Controllers\PlataformaController::class)->middleware('auth');
Route::resource('estados', App\Http\Controllers\EstadoController::class)->middleware('auth');
```

7. Creación de los enlaces en el menú principal

Para poder acceder a las vistas que hemos creado tenemos que añadirle los enlaces. En *app\resources\views\layouts\app.blade.php* creamos los enlaces necesarios en el menú de navegación. Los haremos solo accesibles para quien haya iniciado sesión metiéndolos dentro de un `if(auth::check())`. También tenemos que añadir el middleware de autenticación a las rutas del archivo *web.php.*

```
Route::resource('juegos', App\Http\Controllers\JuegoController::class)->middleware('auth'); //middlewa
Route::resource('generos', App\Http\Controllers\GeneroController::class)->middleware('auth');
Route::resource('plataformas', App\Http\Controllers\PlataformaController::class)->middleware('auth');
Route::resource('estados', App\Http\Controllers\EstadoController::class)->middleware('auth');
```

```blade
<ul class="navbar-nav me-auto">
    <li class="nav-item">
        <a class="nav-link" href="{{ route('juegos.index') }}">{{ __('Juegos') }}</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{ route('generos.index') }}">{{ __('Generos') }}</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{ route('plataformas.index') }}">{{ __('Plataformas') }}</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{ route('estados.index') }}">{{ __('Estados') }}</a>
    </li>
</ul>
```

8. Acceso a los datos de las tablas relacionadas desde Juegos

Para poder acceder a las propiedades de las distintas tablas relacionadas hay que modificar los métodos `create()` y `edit()` en el controlador de Juegos. También hay que añadir los modelos al principio del controlador:

- **create():**

```php
/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public  function create(){
    $juego = new Juego();
    $plataformas =Plataforma::pluck('nombre', 'id');
    $generos =Genero::pluck('nombre', 'id');
    $estado =Estado::pluck('nombre', 'id');
    return view('juego.create', compact('juego','plataformas', 'generos', 'estado'));
}
```

- **edit():**

```php
/**
 * Show the form for editing the specified resource.
 *
 * @param  int $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
    $juego = Juego::find($id);
    $plataformas =Plataforma::pluck('nombre', 'id');
    $generos =Genero::pluck('nombre', 'id');
    $estado =Estado::pluck('nombre', 'id');
    return view('juego.edit', compact('juego','plataformas', 'generos', 'estado'));
}
```

- Dependencias:

```
use App\Models\Juego;
use Illuminate\Http\Request;
use App\Models\Estado;
use App\Models\Genero;
use App\Models\Plataforma;
```

9. Creación de un select para los elementos relacionados

En *form.blade.php* de Juegos cambiamos el tipo **text** por **select**.

```
<div class="form-group">
    {{ Form::label('plataforma_id') }}
    {{ Form::select('plataforma_id', $plataformas,$juego->plataforma_id, ['class' => 'form-control' . ($erro
    {!! $errors->first('plataforma_id', '<div class="invalid-feedback">:message</div>') !!}
</div>
<div class="form-group">
    {{ Form::label('genero_id') }}
    {{ Form::select('genero_id', $generos,$juego->genero_id, ['class' => 'form-control' . ($errors->has('gen
    {!! $errors->first('genero_id', '<div class="invalid-feedback">:message</div>') !!}
</div>
<div class="form-group">
    {{ Form::label('estado_id') }}
    {{ Form::select('estado_id', $estado,$juego->estado_id, ['class' => 'form-control' . ($errors->has('esta
    {!! $errors->first('estado_id', '<div class="invalid-feedback">:message</div>') !!}
</div>
```

10. Modificación del nombre del encabezado de los IDs en la tabla Juegos

Nos dirigimos a *app\resources\views\juego\index.blade.php* para ponerle el encabezado que queramos. También podríamos hacer esto en cualquier otra tabla.

```
<th>Titulo</th>
<th>Año</th>
<th>Plataforma</th>
<th>Genero</th>
<th>Estado</th>
```

11. Cambiar idioma

En los controladores podemos modificar el texto que nos muestra cuando editamos, mostramos o borramos algún elemento de las tablas. Por defecto viene en inglés, así que lo vamos a poner en español.
Tenemos que modificar el mensaje de las funciones store, update y destroy.

- **store**

```
return redirect()->route('juegos.index')
    ->with('success', 'El juego ha sido creado con éxito.');
```

- **update**

```
return redirect()->route('juegos.index')
    ->with('success', 'El juego ha sido editado con éxito');
```

- **destroy**

```
return redirect()->route('juegos.index')
    ->with('success', 'El juego se ha borrado con éxito');
```

Repetimos esto en todas las tablas.

12. Modificar apariencia

Para la vista de la página principal he añadido css en el archivo *welcome.blade.php.*
Para los CRUDs he modificado el archivo *resources/sass/_variables.scss.*

```
// Body
$body-bg: #fddaf2;

$primary: #c6e6fc;
$success: #e0fdda;
$table-accent-bg: #fcecf7;
$table-bg: rgb(224, 222, 222);
$table-color: rgb(131, 129, 129);
$table-active-color: rgb(255, 255, 203);
$table-hover-bg: #c6f8fc;
// Typography
$font-family-sans-serif: 'Nunito', sans-serif;
$font-size-base: 0.9rem;
$line-height-base: 1.6;
```