



MEMORIA MAGENTO 2

Francisco Ruiz-Alejos

HIBERUS



1: Crear un nuevo módulo cuyo nombre sea tu apellido (sin tildes) y el vendor sea Hiberus, por ejemplo: Hiberus_Garcia.

Documentación seguida: <https://devdocs.magento.com/videos/fundamentals/create-a-new-module/>

Para crear el módulo lo primero que hice fue crear la carpeta Ruizalejos y luego el archivo `module.xml`, que nos da la definición inicial del módulo.

Después cree el archivo `registration.php`, que indica a magento como ubicar el módulo.

Para activar el módulo utilice el comando `dockergento magento setup:upgrade`.

La prueba de que el módulo se ha creado correctamente la hice a través del comando `dockergento magento module:status` y ver si aparece el módulo, como se aprecia en la siguiente imagen sí que aparece.

```
Hiberus_Ruizalejos
```

2: Crear una única tabla llamada hiberus_exam que responda exactamente a la siguiente estructura:

Para crear la tabla he creado el archivo `db_schema.xml` dentro de la carpeta `etc` del módulo Ruizalejos.






La siguiente imagen define toda la estructura de este archivo, y el resultado que aparece en mysql workbench después de introducir los inserts:

```
db_schema.xml
1 <?xml version="1.0"?>
2 <schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">
4     <table name="hiberus_exam" resource="default" engine="innodb" comment="Tabla Examen Hiberus">
5         <column xsi:type="int" name="id_exam" padding="11" unsigned="true" nullable="false" identity="true" />
6         <column xsi:type="varchar" name="firstname" length="100" unsigned="true" nullable="false" />
7         <column xsi:type="varchar" name="lastname" length="250" unsigned="true" nullable="false" />
8         <column xsi:type="decimal" name="mark" scale="2" precision="4" unsigned="true" nullable="false" />
9         <constraint xsi:type="primary" referenceId="PRIMARY">
10             <column name="id_exam" />
11         </constraint>
12     </table>
13 </schema>
```



```
1 • SELECT * FROM magento.hiberus_exam;
```

Limit to 1000 rows ▼

Result Grid  Filter Rows:  Edit:    Export

#	id_exam	firstname	lastname	mark
1	1	Francisco	Ruiz	9.00
2	2	Pepe	Perez	4.00
3	3	Raul	Perez	4.50
4	4	Jose	Martinez	3.50
5	5	Carlos	Dominguez	7.50
6	6	Juan	Lopez	8.00
7	7	Jesus	Ayora	8.50
*	NULL	NULL	NULL	NULL

3: Crear el Service Contracts y ORM que gestione esta entidad.

Documentación seguida: <https://devdocs.magento.com/guides/v2.4/extension-dev-guide/api-concepts.html>

Para el Service Contracts, creé el conjunto de interfaces PHP que se definen para el módulo. En este caso siguiendo la guía de magento la interfaz de datos `ExamenInterface` la he creado dentro de una carpeta `Api/Data`.

Esta interfaz contiene dos constantes, una el nombre de la tabla y otra su id; y los métodos getters y setters de esa misma tabla.

En la carpeta `Api` he creado otra interfaz, `ExamenRepositorioInterface.php`; la cual va a ser la interfaz del repositorio. Este contiene los métodos `save`, `getById`, `delete`, y `deleteById`.

Me he creado un archivo `di.xml` dentro de `etc` y he usado la etiqueta `<preference>` para declarar las implementaciones del `Api`.

Para crear todo lo relacionado con el ORM, me creó el modelo `Examen` que extiende de `AbstractModel`, el cual contiene el constructor y los getters y setters debido a que implementa a la interfaz `ExamenInterface`.

También he creado el repositorio `ExamenRepository` el cual implementa los respectivos métodos de `ExamenRepositoryInterface`.

Después creé el directorio `ResourceModel`, y dentro de la clase `Examen`, que extiende de `AbstractDB`, la cual sirve para conectar con la base de datos.

4: Crear un Setup (Db Schema y Data Patch) para introducir datos que introduzca en la tabla creada utilizando los service contracts. Por defecto podéis construir un array con la información a añadir.

Debido a que no habíamos dado nada relacionado con este ejercicio, he decidido hacerlo a través de `Factorys`. Para ello lo primero me creó en el `resourceModel` una clase `Collection`, que es lo que hace es llamar a lo que hay en BD y después en la clase `index de Block` (este se explicará en ejercicios posteriores), he creado principalmente el método `getAlumno()`, que lo que hace es crear un alumno a través del `ExamenInterfaceFactory`. (esta clase se crea automáticamente al seguir todos los pasos anteriores, y hacer un setup upgrade, compile y borrar la cache). Ese método lo que hace es crear alumnos y después devuelvo la colección de estos. Los datos de los campos los he introducido mediante consultas insert a través del `mysql workbench`.

5: Crear un nuevo controlador de frontend, el front name debe llamarse como tú apellido (ignora tildes). Haz de momento que simplemente diga echo "Hola", posteriormente lo modificaremos.

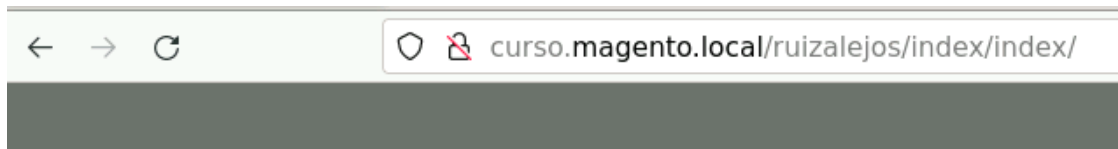
Documentación seguida: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/routing.html?itm_source=devdocs&itm_medium=search_page&itm_campaign=federated_search&itm_term=routes.xml

Para crear un controlador, he creado en el módulo/etc una carpeta `frontend` y dentro un fichero `routes.xml`. En este fichero he definido la ruta que iba a llevar mi controlador, en este caso `/Ruizalejos/index/index`.

Después he creado la carpeta `Controller` en la que dentro creaba una carpeta `Index` que contenía la clase `Index.php`.

El principal método de esta clase es el `execute()`, en el cual se indica el código que queremos ejecutar, en este caso el echo "hello".

Como se aprecia en la siguiente imagen aparece la palabra `Hola` que se ha indicado, en la ruta mencionada anteriormente.



hola

6: Asociar un layout, bloque y template a nuestro controlador de acción para poder devolver un listado de los exámenes de los alumnos en el frontend.

Para crear el `layout`, he creado una carpeta `view` en el módulo, y dentro las carpetas `frontend`, `layout` y dentro un fichero `ruizalejos_index_index.xml`

Para el `bloque`, dentro de mi modulo, he creado la carpeta `Block` con una clase `index.php` que extienda del `template`.

Por último, para crear el `template` en `view->frontend` he creado la carpeta `templates`, y dentro el fichero `index.phtml`, que es donde vamos a definir nuestra vista.

En la siguiente imagen se muestran las siguientes especificaciones que debía tener este `index.phtml`, y el resultado obtenido:

1. Añadir un título `h2` a la página con el `class="title"`.
2. En el listado (`ul`) debe mostrarse el nombre, apellido y la nota.
3. Debajo del listado, añadir un texto "Total number of students: XX." donde XX debe corresponder al total de alumnos almacenados.
4. Añadir una traducción al español a nivel de módulo para el literal anterior, de modo que el texto mostrado sea: "Total: XX alumnos."

Lista de alumnos

- Name: Francisco
- Surname: Ruiz
- Mark 9.00
- Name: Pepe
- Surname: Perez
- Mark 4.00
- Name: Raul
- Surname: Perez
- Mark 4.50
- Name: Jose
- Surname: Martinez
- Mark 3.50
- Name: Carlos
- Surname: Dominguez
- Mark 7.50

Total number of students: 5

7: Asociar un js por require a la página para que desde un botón se pueda ocultar y desocultar las notas de los alumnos.

Pese a que no la habíamos dado he estado investigando un poco en la documentación de Magento como añadir un módulo js, cosa que he hecho mas tarde para el ejercicio 9.

<https://devdocs.magento.com/videos/fundamentals/add-a-javascript-module/>

8. Maquetar el listado usando less en el módulo siguiendo las siguientes especificaciones:

1. El título debe tener por defecto un color y a partir de 768 píxeles ponerse de otro.
2. Dejad el listado con la mejor apariencia que te parezca.
3. Haced por css que los impares tengan un margen izquierdo de 20px, este valor debe estar definido como variable al principio del less, por ejemplo @margin-left-primary.

Antes de maquetar nada lo primero que he hecho es crearme mi tema `custom-Theme` con sus archivos `theme.xml` y `registration.php`. En el fichero `theme.xml` le he asignado el título del tema y el padre del que hereda (`Magento/Blank`).

Para maquetar usando `less` lo primero que hay que hacer es en la carpeta `design` crear toda la estructura necesaria, para ello creé las carpetas `Magento_Theme/web/css/source`. Dentro de esta última he creado el archivo para extender el `less`, el `_extend.less`.

Para que el título tenga un color y a partir de 768 píxeles otro lo he hecho de la siguiente forma:

Para poner que los estilos sean genéricos para todas las resoluciones lo he hecho a través de la siguiente línea.

```
& when (@media-common = true) { ... }
```

Para estilos `responsive`, en este caso para cuando la vista tenga más de 768 px es a través de un `media-width`, que, si no se cumple la condición, mostrará el `media-common`:

```
.media-width(@extremum,@break) when (@extremum='min') and  
(@break=@screen__m){ .. }
```

Para que los impares tengan un margen izquierdo de 20 px he utilizado el siguiente código, el cual destaca el `nth-child(even)` para que sean las filas impares.

```
ul:nth-child(even) {  
    margin-left: @margin-left-primary;  
}
```

9: Añadir un nuevo botón que muestre la nota más alta de todos los alumnos en un alert, busca la manera más eficiente para el servidor.

[Alert widget | Adobe Commerce Developer Guide \(magento.com\)](#)

Aunque no lo habíamos dado he estado viendo la documentación de `magento` para poder realizar este ejercicio.

Lo primero que he hecho es crearme el botón en la vista.

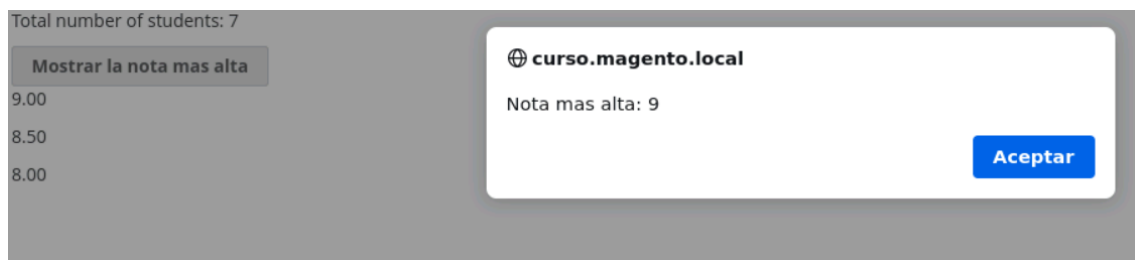
```
<button id="notaAlta">Mostrar la nota mas alta</button>
```

Después he creado un archivo de `module` `require-config.js` del cual he sacado el código de la url que sale en el ejercicio 7, en este se especifica la ruta en la que va a estar el archivo `js`.

Creo este archivo en mi caso `notaAlta.js` en el cual invoco al evento `.click` y dentro la función por defecto en la que muestro un `alert` con la variable de máxima nota.

Por último, asigno esta variable en vista y con un `require.js` invoco a la función creada en el paso anterior.

El resultado que aparece es el adecuado



10: Sacar en una nueva fila la media de notas que ha sacado la clase.

Para sacar la media lo he hecho directamente en la vista, aunque lo suyo hubiera sido hacerlo en una función en el controlador.

Dentro del for en el que recorro los alumnos con `$suma+=$al->getMark();` lo que hago es suma a la variable suma las notas de cada alumno, y para sacar la media, fuera ya del for divido esa variable entre el total de alumnos `count($alumnos)`.

```
<?php $suma+=$al->getMark();  
}  
?>  
<p>Average of Marks <?= $suma/(count($alumnos)) ?></p>
```

11: Crear un plugin que ponga un 4.9 a todos los alumnos que hayan suspendido (no se tiene que guardar en db).

Documentación seguida: <https://devdocs.magento.com/guides/v2.4/extension-dev-guide/plugins.html>

Lo primero para crear el plugin es crear la carpeta `Plugin` en la que va a estar alojado. Después he creado la carpeta del plugin `Examen` y dentro la clase `NotaSuspensa`, que es donde se va a poner un 4.9 a los alumnos suspensos.

En el hago la función `afterGetMark()`, para que se obtenga las notas de todos los alumnos, y a aquellos que el resultado sea `<5` les seteo, en la variable `result` que es donde está la nota a un 4.9

12: Que los alumnos aprobados aparezcan en un color y los suspensos en otro.

Lo primero que he hecho es un `_mixin` el cual paso el color como parámetro de entrada del `mixin`. Para ello me he creado el fichero `_mixin.less` dentro del tema las carpetas `web/css/source`. Este lo importo en otro fichero `_extend.less` que me he creado en la misma carpeta, con la intención de que se pueda usar el `mixin` en el `extend.less` de `Magento_Theme`.

El código usado en el `mixin` para pasar el color es el siguiente:


```
.examenColor(@color) {
    color: @color;
}
```

Para distinguir entre aprobados y suspendidos en la vista creo dos clases que varían. Una clase aprobado si la nota ≥ 5 y otra suspendido para nota < 5 , de la siguiente forma:

```
<ul class="<?php if($al->getMark()<5){ ?> suspendido <?php
} else { ?> aprobado <?php }?>">
```

Para usarlo en el extend.less del Magento_Theme lo he pasado de la siguiente forma, nombrando con un . la clase del mixin.

```
.aprobado {
    .examenColor(green);
}
.suspendido {
    .examenColor(red);
}
```

La siguiente imagen muestra el resultado del ejercicio:

Students' List

Name: Francisco
Surname: Ruiz
Mark 9.00
Name: Pepe
Surname: Perez
Mark 4.9
Name: Raul
Surname: Perez
Mark 4.9
Name: Jose
Surname: Martinez
Mark 4.9
Name: Carlos
Surname: Dominguez
Mark 7.50
Average of Marks 6.24
Total number of students: 5

13: Que además los 3 mejores aparezcan destacados de otra forma aún más destacada, podéis utilizar cualquier forma que se os ocurra, js, php.

En este ejercicio he creado un script php dentro de la vista en el que lo que hago es ir comparando el alumno con la mejor nota, la segunda y la tercera, y si es mayor la cambio por la nota que esta asignada en ese momento. Esta ultima en caso de que sea la mejor nota o la segunda mejor la paso a la segunda y tercera mejor nota correspondiente. Código creado:

```
foreach($alumnos as $al) {  
    //en este if calculo las 3 maximas notas  
    if ($al->getMark() > $max1) {  
        $max2 = $max1;  
        $max1 = $al->getMark();  
    } else {  
        if ($al->getMark() > $max2) {  
            $max3 = $max2;  
            $max2 = $al->getMark();  
        } else {  
            if ($al->getMark() > $max3) {  
                $max3 = $al->getMark();  
            }  
        }  
    }  
}
```

A estas notas para diferenciarse de otras lo que he hecho es aumentarles el tamaño de letra.

Students' List

Name: Francisco

Surname: Ruiz

Mark 9.00

Name: Pepe

Surname: Perez

Mark 4.9

Name: Raul

Surname: Perez

Mark 4.9

14: Crear un CLI command nuevo que permita ver los todos los datos de la tabla de exámenes, se valorará que NO se haga uso del object manager.

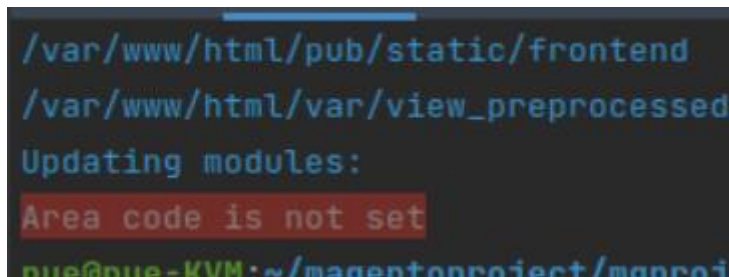
Lo primero que he hecho es crear la carpeta `Console`, para después crear la clase en la que crearemos el comando, a esa clase le llamo `RuizalejosCommand` y la extiendo de `Command`.

En el método `configure()` configuro el nombre que va a tener el comando y su descripción.

En el método `execute()`, lo que hago es recorrer la lista de la colección de alumnos y a cada campo de la clase del modelo examen les seteo todos los campos del alumno.

Por último, devuelvo mediante un `$output->writeln` la info que queremos que salga al ejecutar el comando y en el fichero `di.xml` añado la información relacionada con el comando.

NOTA: He dejado comentado el código debido a que yo creo que lo tengo bien y me da el error que se aprecia en la imagen cuando intento borrar cache o hacer un `setup:upgrade`.



```
/var/www/html/pub/static/frontend
/var/www/html/var/view_preprocessed
Updating modules:
Area code is not set
nue@nue-KVM:~/magento-project/magento2
```

15: Crear 3 endpoint nuevo de Api Rest con Swagger:

No le podido hacer debido que no habíamos dado este contenido.

16: Crear una nueva sección de configuración para vuestro módulo (con su tab asociada de Hiberus) que permita añadir los siguientes campos configurables:

He creado la carpeta `adminhtml` dentro de la `etc` de mi modulo, y dentro un archivo `system.xml`.

Lo primero que hago es montar mi propio `tab`, para después añadirle `sections`. En cada sección añado un elemento hijo, que va a ser el `group`. Dentro de cada grupo creo los campos (`fields`)

En el block he creado dos métodos, uno en el que obtengo el número de elementos y el otro el que obtengo la nota de corte.

He utilizado también el `scopeConfig` que me coge los campos que le meta en la configuración del admin, con la ruta pasada el `section/group/field`. En esa clase

Luego ya en la vista lo que hago es cambiar donde tenía el 5 por la nota introducida, y para los elementos que cuando llegue al numero introducido se pare de buscar más alumnos.

GENERAL	▼
CATALOG	▼
SECURITY	▼
CUSTOMERS	▼
SALES	▼
YOTPO	▼
DOTDIGITAL	▼
RUIZALEJOS	▲
Insertar Configuración	

Campos

Numero de elementos <small>[global]</small>	<input type="text" value="5"/>
Nota aprobado <small>[global]</small>	<input type="text" value="5"/>