

Memoria de prueba técnica de Magento 2

Teresa Rodríguez García

Curso de PHP - CMS

hiberus

Septiembre 2021

Índice

[Notas aclaratorias](#)

[Introducción del enunciado](#)

[Ejercicio 1](#)

[Ejercicio 2](#)

[Ejercicio 3](#)

[Ejercicio 4](#)

[Ejercicio 5](#)

[Ejercicio 6](#)

[Ejercicio 7](#)

[Ejercicio 8](#)

[Ejercicio 9](#)

[Ejercicio 10](#)

[Ejercicio 11](#)

[Ejercicio 12](#)

[Ejercicio 13](#)

[Ejercicio 14](#)

[Ejercicio 15](#)

[Ejercicio 16](#)

[Ejercicio 17](#)

[Indicaciones para elaborar la presente memoria](#)

[Cómo entregar la prueba](#)

[Enlace a la prueba en GitHub](#)

Notas aclaratorias

En la prueba hay ejercicios sobre temas que por diversas cuestiones no se han abordado en la formación, por lo que no se han realizado. Circunstancia indicada en cada ejercicio.

Esto tiene dos excepciones: los ejercicios 7 y 9 sobre javaScript, no dado en clase, pero que gracias a la base previa, la documentación y la información que se ha podido obtener por internet se han podido realizado con éxito.

En mi caso he tenido una serie de problemas técnicos con el entorno y el proyecto restando días de ejecución del examen y recursos, de los que ya se ha informado y se amplió el plazo de entrega para compensar estos percances, por lo que no veo necesario entrar en más detalles en esta memoria a este respecto más que limitarme a reseñar que es la razón de que se entregue más tarde haciendo uso del tiempo ampliado.

Introducción del enunciado

Crear un nuevo módulo que va a usarse para gestionar las notas de los alumnos en un examen, para ello hay que crear una nueva entidad, un service contract que la administre, un data patch que permita añadir datos a la tabla y un controlador que muestre la información guardada para posteriormente añadirle js y darle estilos.

Ejercicio 1

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

En la carpeta de proyecto, dentro de la carpeta `app` creamos la siguiente estructura de directorios, donde `Hiberus` es el vendor y `Rodriguez` el módulo.

```
app/code/Hiberus/Rodriguez
```

Creamos la estructura de directorios y archivos mínimos necesarios, un módulo puede contener otros según necesidad.

Dentro del módulo `Rodriguez`:

- Creamos el archivo `registration.php`:

```
app/code/Hiberus/Rodriguez/registration.php
```

- Creamos la carpeta `etc` y dentro, el archivo `module.xml` y el `di.xml` necesario para futuros ejercicios ya que en él se registrarán futuros elementos como un plugin, o un comando personalizado entre otras cosas.

```
app/code/Hiberus/Rodriguez/etc/module.xml
```

```
app/code/Hiberus/Rodriguez/etc/di.xml
```

De momento el `di.xml` lo dejaremos vacío. El contenido para los archivos creados aparece en la documentación de Magento, pero también se puede obtener copiándolo de otro módulo. Para hacerlo hemos ido a `vendor/magento` donde están los módulos de Magento, y hemos entrado en uno de ellos, elegimos `module-catalog`.

Para nuestro archivo `registration.php` hemos copiado el contenido de:

```
vendor/magento/module-catalog/registration.php
```

y le hemos cambiado el nombre por `Hiberus_Rodriguez` (`nombreVendor_nombreMódulo`):

```
<?php

use Magento\Framework\Component\ComponentRegistrar;

ComponentRegistrar::register(ComponentRegistrar::MODULE,
'Hiberus_Rodriguez', __DIR__);
```

Para nuestro archivo module.xml hemos copiado el contenido de:

curso-magento/vendor/magento/module-catalog/etc/module.xml

y le hemos cambiado el nombre por `Hiberus_Rodriguez` (nombreVendor_nombreMódulo) y la versión `1.0.0` que irá aumentando con las futuras modificaciones.

```
<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../lib/internal/Magento/Framework/Module/etc/module.xsd">

<module name="Hiberus_Rodriguez" setup_version="1.0.0"></module>

</config>
```

En la terminal, para activar el módulo, ejecutamos en orden:

- `dockergento start` (para iniciar dockergento, si no lo estaba ya)
- `dockergento magento setup:upgrade`
- `dockergento magento setup:di:compile`
- `dockergento magento cache:flush`
- `dockergento magento module:enable Hiberus_Rodriguez`

Con esto el módulo queda habilitado.

Si queremos comprobarlo podemos hacerlo de varias formas, una es deshabilitarlo desde terminal con:

```
bin/magento module:disable Hiberus_Rodriguez
```

y volverlo a habilitar con:

```
bin/magento module:enable Hiberus_Rodriguez
```

observando el mensaje de la terminal que nos indica si está habilitado o no.

También podemos comprobarlo consultando el archivo `config.php` en la ruta:

```
app/etc/config.php
```

En `config.php` están todos los módulos de magento o propios, habilitados o no, indicando su estado con un 0 (deshabilitado) o un 1 (habilitado).

Hemos usado esta opción y vemos que nuestro módulo está correctamente habilitado ya que figura así en el `config.php`:

```
'Hiberus_Rodriguez' => 1,
```

Ejercicio 2

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

Field	Type	Null	Key	Default	Extra
id_exam	int(11)	NO	PRI	<null>	auto_increment
firstname	varchar(100)	NO		<null>	
lastname	varchar(250)	NO		<null>	
mark	decimal(4,2)	NO		<null>	

Dentro del módulo, en la carpeta etc, creamos un nuevo archivo llamado db_schema.xml en la ruta:

```
app/code/Hiberus/Rodriguez/etc/db_schema.xml
```

En él definimos la tabla de la base de datos de la siguiente forma:

```
<?xml version="1.0"?>

<schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">

    <table name="hiberus_exam" resource="default" engine="innodb"
comment="Tabla prueba tecnica Hiberus">

        <column xsi:type="int" name="id_exam" padding="10"
unsigned="true" nullable="false" identity="true"/>

        <column xsi:type="varchar" name="firstname" length="100"
unsigned="true" nullable="false"/>

        <column xsi:type="varchar" name="lastname" length="250"
unsigned="true" nullable="false"/>

        <column xsi:type="decimal" name="mark" nullable="false"
scale="2" precision="4"/>

        <constraint xsi:type="primary" referenceId="PRIMARY">

            <column name="id_exam" />

        </constraint>

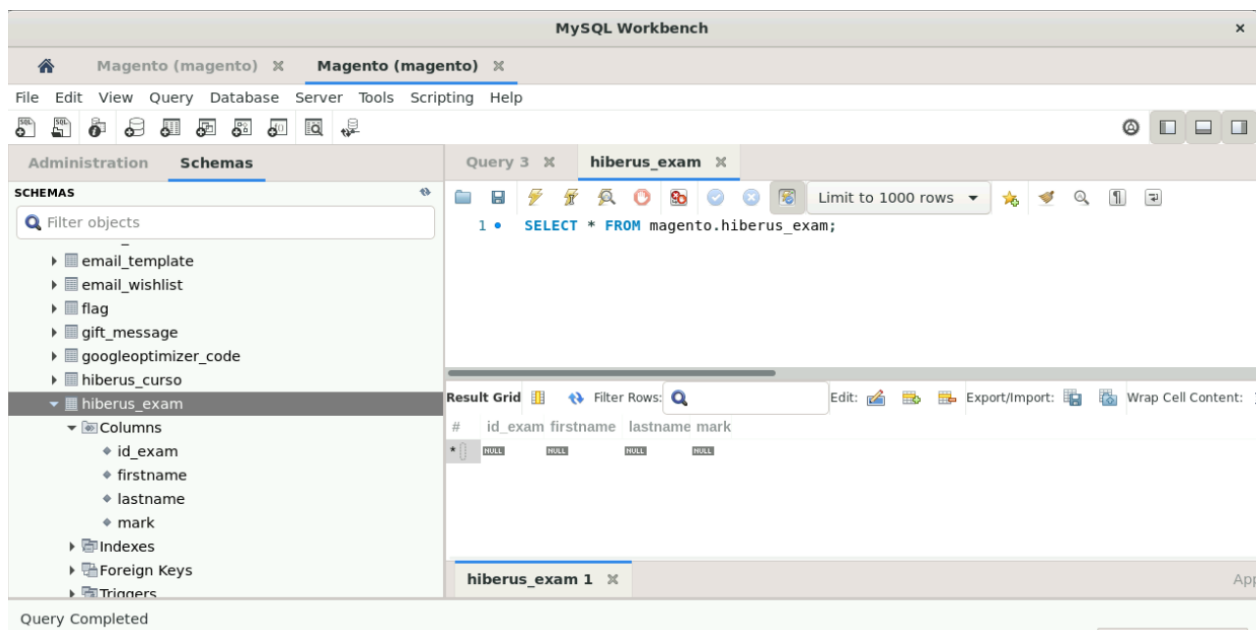
    </table>
```

```
</schema>
```

En la terminal, ejecutamos en orden:

- `dockergento magento setup:upgrade`
- `dockergento magento setup:di:compile`
- `dockergento magento cache:flush`

Tras lo que comprobamos en Workbench que se haya creado correctamente la nueva tabla con las columnas correspondientes y así es:



Ejercicio 3

Crear el Service Contracts y ORM que gestione esta entidad.

Creamos la carpeta `Api` en la ruta:

```
app/code/Hiberus/Rodriguez/Api
```

La carpeta `Api` contendrá las dos interfaces que necesitamos y que crearemos con la siguiente estructura de directorios a partir de la carpeta `Api`:

- **Api**
 - **NotasRepositoryInterface.php**
 - **Data**
 - **NotasInterface.php**

En la interfaz `NotasInterface.php` creamos dos constantes para guardar el nombre de la tabla y el id, declaramos los getters y setters correspondientes a las columnas de la tabla de la base de datos que hemos creado, auto-generamos la documentación con PHPDoc y repasamos cambiando lo necesario.

La interfaz `NotasRepositoryInterface.php` obtiene los datos de `NotasInterface.php` trayéndolos con el correspondiente

```
use \Hiberus\Rodriguez\Api\Data\NotasInterface;
```

En `NotasRepositoryInterface.php` declaramos los métodos de la interfaz del repositorio como son:

- `save` para guardar un registro en nuestra tabla de la base de datos, pasándole una instancia de `NotasInterface` como parámetro y retornando la instancia guardada.
- `delete` para borrar, igual que en el caso anterior, se le pasa una instancia de `NotasInterface` como parámetro y en este caso retorna un booleano según su éxito.
- `deleteById` para borrar un registro con un ID concreto de nuestra tabla de la base de datos, pasándole el ID del registro como parámetro y retornando un booleano según su éxito.

- `getById` para obtener un registro concreto por su ID de nuestra tabla de la base de datos , pasándole el ID del registro como parámetro y retornando la instancia de `NotasInterface` solicitada.

A continuación creamos la carpeta `Model` y dentro la clase `Notas.php` que extiende `AbstractModel` e implementa nuestra interfaz `NotasInterface.php` que acabamos . Esto lo hacemos en la ruta:

`app/code/Hiberus/Rodriguez/Model/Notas.php`

Dentro del archivo por tanto tenemos:

```
<?php
namespace Hiberus\Rodriguez\Model;

use Hiberus\Rodriguez\Api\Data\NotasInterface;
use Magento\Framework\Model\AbstractModel;

class Notas extends AbstractModel implements NotasInterface {
    protected function _construct() {

    }

    $this->_init(\Hiberus\Rodriguez\Model\ResourceModel\Notas::class);
}
... }
```

Ejercicio 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. * Como alternativa opcional, podéis traer esa información (nombre y apellido) desde un CSV, como pista, ese csv puede estar dentro de vuestro módulo en "Vendor\Apellido\Setup\data\import.csv" y leerlo haciendo uso de la clase de Magento 2: `Magento\Framework\File\Csv` y su función `"getData($file)"` a la cual le pasáis un path de fichero y os devuelve un array de lo que ha leído en el fichero. Únicamente el csv debe tener las columnas nombre y apellido. La nota deberá introducirse de manera aleatoria, lo ideal sería que generase notas del 0 al 10 con 2 decimales.

Este contenido no se ha visto en clase, pero como para poder continuar con los ejercicios necesitamos que la base de datos contenga registros, estos se insertan manualmente.

Ejercicio 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.

Creamos un controlador en la ruta:

```
/app/code/Hiberus/Rodriguez/Controller/Index/Index.php
```

Esta clase `Index.php` implementa el `HttpGetActionInterface` y en ella usamos el `PageFactory` y nuestro repositorio, creamos un constructor que inicia todo lo necesario y una función `execute` que devuelve, gracias al `pageFactory->create()` lo que va a la vista.

En la ruta siguiente creamos el archivo `routes.xml`

```
/app/code/Hiberus/Rodriguez/etc/frontend/routes.xml
```

que registra nuestro route `prueba` y el nombre de nuestro módulo con nuestro apellido `Hiberus_Rodriguez` (`nombreVendor_nombre_Modulo`):

```
<?xml version="1.0" ?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:App/etc/routes.x
sd">

    <router id="standard">

        <route frontName="prueba" id="prueba" >

            <module name="Hiberus_Rodriguez"/>

        </route>

    </router>

</config>
```

Para obtener los datos, dentro de la carpeta `Model`, creamos la carpeta `ResourceModel`, y dentro de ella la clase `Notas.php` que extiende `AbstractDb`

Nos queda en la ruta:

`app/code/Hiberus/Rodriguez/Model`

la siguiente estructura de directorios:

- `Model`
 - `Notas.php` // ejercicio 3
 - `NotasRepository.php` // ejercicio 3
 - **`ResourceModel`**
 - **`Notas.php`**
 - **`Notas`**

- **Collection.php** // se le llama desde Block/Index.php y contiene:

```
<?php

namespace Hiberus\Rodriguez\Model\ResourceModel\Notas;

use
\Magento\Framework\Model\ResourceModel\Db\Collection\AbstractCollection;

class Collection extends AbstractCollection {

    protected function _construct(){

        $this->_init('Hiberus\Rodriguez\Model\Notas',
        'Hiberus\Rodriguez\Model\ResourceModel\Notas');

    }

}
```

Creamos la siguiente ruta para la vista:

```
/app/code/Hiberus/Rodriguez/view/frontend/layout/prueba_index_index.xml
1
```

respetando el nombre del id del routes.xml en el archivo cuyo contenido es:

```
<?xml version="1.0"?>

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
layout="1column"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/
page_configuration.xsd">

    <referenceContainer name="content">

        <block class="Hiberus\Rodriguez\Block\Index"
name="prueba_index_index" template="Hiberus_Rodriguez::index.phtml" />

    </referenceContainer>
```

</page>

Y finalmente creamos en la siguiente ruta la plantilla para la vista `index.phtml`:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

que muestra los datos de la variable `Block` que viene de `Block/Index.php`

Tras comprobar que se puede hacer un hola mundo, nos traemos los datos que necesitaremos en los posteriores ejercicios. Para pintar los datos de la tabla en la base de datos en la plantilla ya podemos hacer:

```
$alumnos = $block->getAlumno();
```

y ya se pueden pintar los datos de `$alumnos` recorriéndolos con un bucle.

Ejercicio 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. Se deben seguir las siguientes especificaciones:

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."**

Parte de este ejercicio se hizo y explicó en el ejercicio anterior, nos quedarían las especificaciones, que se realizan en la plantilla en la ruta:

/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml

- Para la especificación 1:

```
<h2 class="title">Alumnos</h2>
```

- Para la especificación 2:

```
$alumnos = $block->getAlumno();
```

```
...
```

```
<?php foreach ($alumnos as $alumno) { ?>
```

```
    <ul>
```

```
        <li>Nombre:<?= $alumno->getFirstName(); ?></li>
```

```
        <li>Apellido: <?= $alumno->getLastName(); ?></li>
```

```
        <li class="<?= $classEvaluation ?>">Nota:<?= $alumno->getMark(); ?></li>
```

```
    </ul>
```

```
<?php } ?>
```

- Para la especificación 3:

```
<p> Total de alumnos: <?= (count($alumnos)) ?> </p>
```

- La especificación 4 de este ejercicio no ha podido realizarse al no haber llegado a ver cómo se realizan las traducciones en clase, por eso el apartado 3 se pinta en español.

El contenido renderizado (sin estilos) se muestra en la URL que especificamos en el ejercicio anterior:

`http://curso.magento.local/prueba/index/index`

Alumnos

- Nombre:Teresa
- Apellido: Rodriguez
- Nota:10.00
- Nombre:Jose
- Apellido: Romero
- Nota:9.00
- Nombre:Maria
- Apellido: Suarez
- Nota:8.00
- Nombre:Ignacio
- Apellido: Perez
- Nota:4.00
- Nombre:Sara
- Apellido: Alvarez
- Nota:3.00
- Nombre:Antonio
- Apellido: Garcia
- Nota:2.00

Media de notas: 6

Total de alumnos: 6

Ejercicio 7

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

JavaScript no ha dado tiempo a darlo en clase, aún así se intenta realizar el ejercicio consultando la documentación y posibles soluciones por internet. Los pasos realizados son los siguientes:

En esta ruta se crea el archivo js:

`app/code/Hiberus/Rodriguez/view/frontend/requirejs-config.js`

con el siguiente contenido que declara nuestro módulo y el archivo:

```
var config = {
    map: {
        '*': {
            toggle: 'Hiberus_Rodriguez/js/toggle',
        }
    }
};
```

Como hemos declarado en el archivo anterior, `toggle` será el nombre de nuestro archivo js que creamos en la ruta:

`app/code/Hiberus/Rodriguez/view/frontend/web/js/toggle.js`

y en el que creamos la función para que cuando se pulse el botón correspondiente, se active el toggle y se despliegue o recoja el listado que, en la plantilla phtml, contendremos dentro de un div.

```
define(['jquery'], function($) {
    "use strict";
    return function myscript(idBtn, idDiv)
```

```
{
    $(idBtn).click(function(){
        $(idDiv).slideToggle();
    });
}
});
```

En la plantilla, recordemos la ruta:

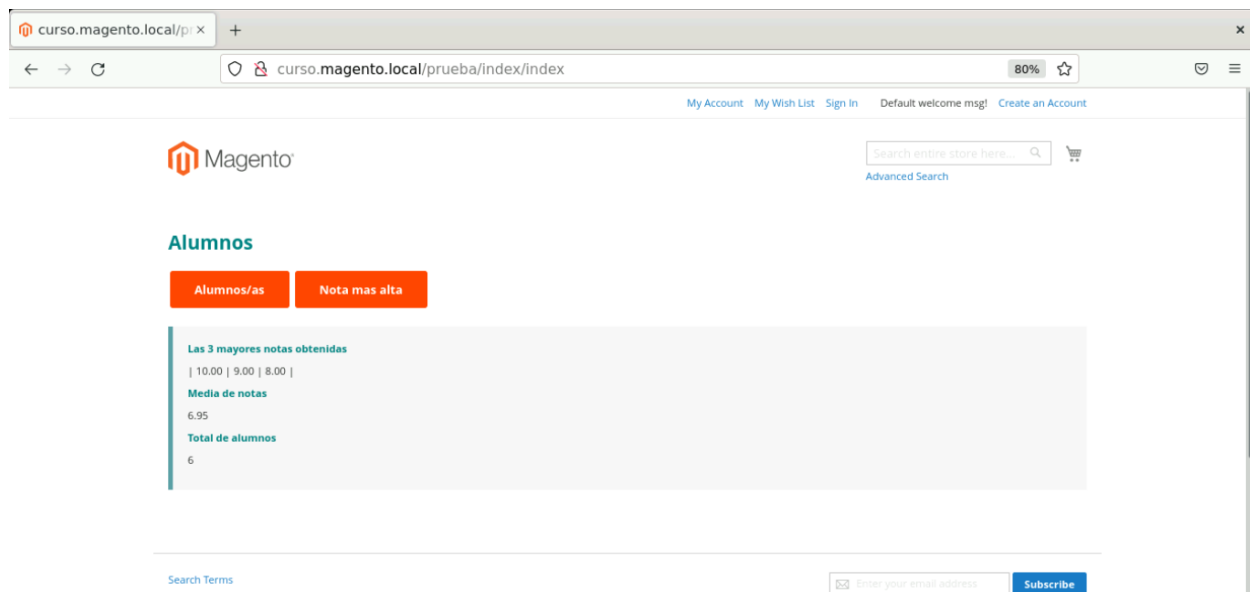
```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

Creamos el botón, metemos el listado dentro de un div y añadimos el script al final:

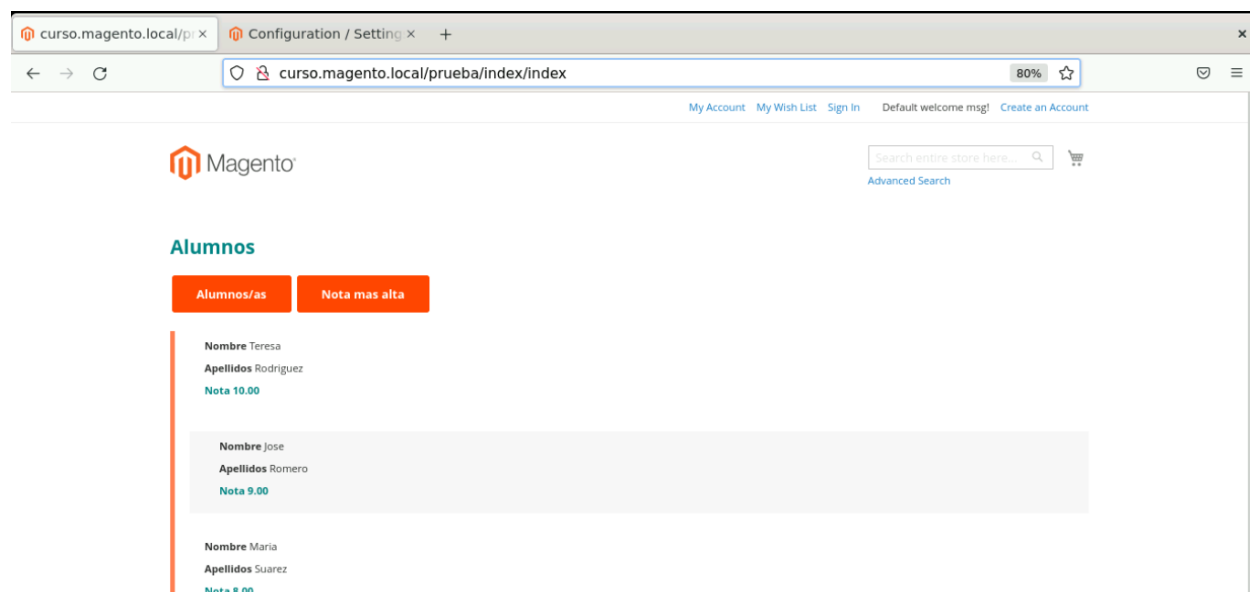
```
<button id="desplegar">Desplegar</button>
...
<div id="details">
    <!-- aquí se pinta el listado -->
</div>
...
<script>
    require(['jquery', 'toggle'], function ($, myscript) {
        myscript('#desplegar', '#details');
    });
</script>
```

Las capturas se han hecho al final con lo que se muestran también los estilos y otros ejercicios acabados, pero igualmente se ve la funcionalidad del botón Alumnos/as que despliega o recoge el listado de alumnos/as al ser pulsado.

- Botón Alumnos/as (listado plegado)



- Pulsamos el botón Alumnos/as (listado desplegado) si se vuelve a pulsar se pliega.



<p>Nombre Ignacio</p> <p>Apellidos Perez</p> <p>Nota 4.9</p>	
<p>Nombre Sara</p> <p>Apellidos Alvarez</p> <p>Nota 4.9</p>	
<p>Nombre Antonio</p> <p>Apellidos Garcia</p> <p>Nota 4.9</p>	
<p>Las 3 mayores notas obtenidas</p> <p> 10.00 9.00 8.00 </p> <p>Media de notas</p> <p>6.95</p> <p>Total de alumnos</p> <p>6</p>	

Ejercicio 8

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

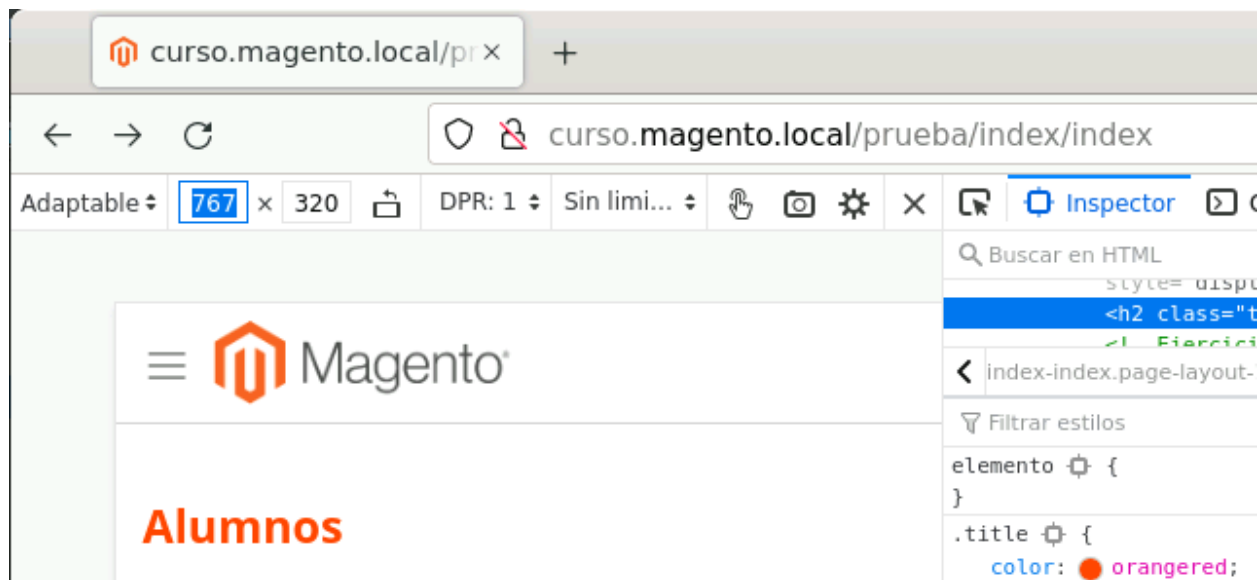
1. El título debe tener por defecto un color y a partir de 768 píxeles ponerse de otro.
2. Dejad el listado con la mejor apariencia que te parezca.
3. Haced por css que los impares tengan un margen izquierdo de 20px, este valor debe estar definido como variable al principio del less, por ejemplo @margin-left-primary.

Creamos las carpetas siguientes y el archivo `.less` para nuestros estilos en la ruta:

```
app/design/frontend/Hiberus/custom-theme/Magento_Theme/web/css/source/_extend.less
```

Siguiendo el principio “mobile first” aplicamos primero los estilos al **título cuando la anchura sea menor de 768px, se verá naranja y en negrita**:

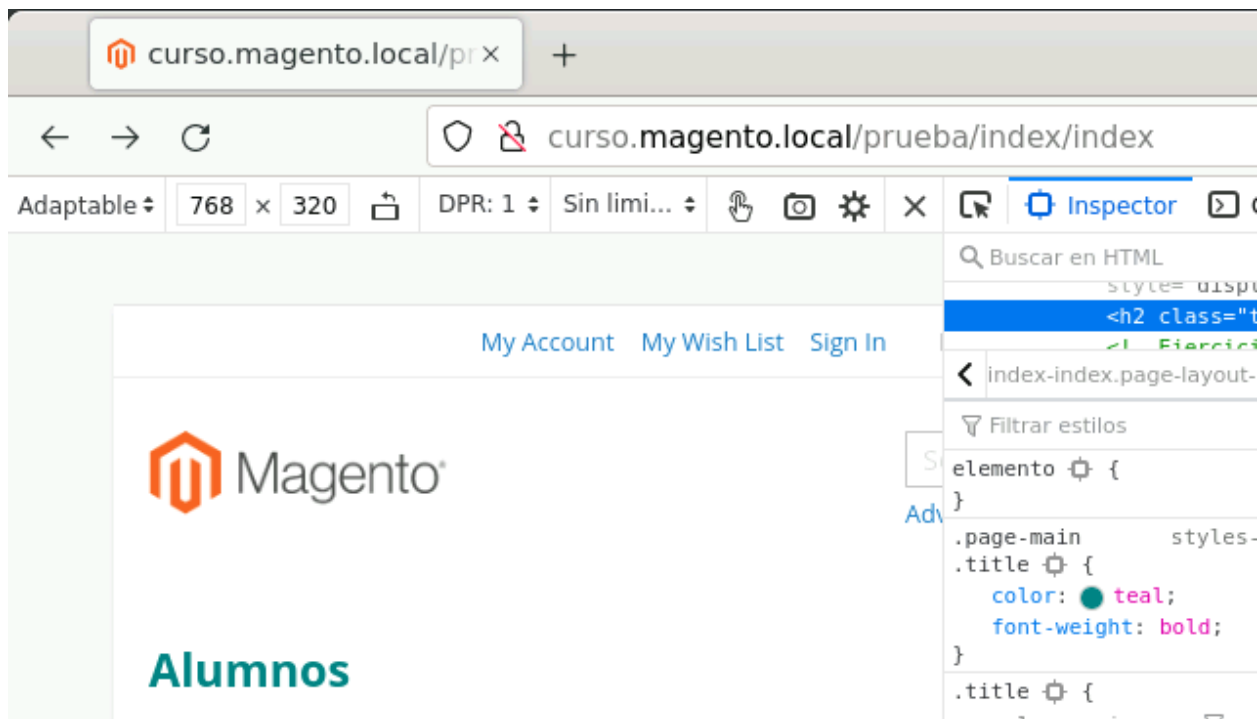
```
& when (@media-common = true) {
    .title{
        color:orangered;
        font-weight: bold;
    }
}
```



- Y luego, al ser **mayor o igual a 768px** de anchura, hacemos que el **título cambie su color a teal**:

```
.media-width(@extremum, @break) when (@extremum = 'min') and (@break
= @screen__m) {
    .page-main {
        .title{
            color:teal;
            font-weight: bold;
        }
    }
}
```

}



- Para aplicar el margen de 20 px a los impares del listado:

Al principio del archivo `_extend.less` declaramos la variable `@margin-left-primary` con los 20px requeridos en el enunciado para el margen izquierdo:

```
@margin-left-primary: 20px;
```

Haciendo uso de la variable y selectores, aplicamos el margen a los impares como se nos solicita en el enunciado:

```

ul:nth-child(even) {
  .notas-mixin(@margin-left-primary);
}

```

El margen quedaría así antes de estilos, como vemos aumenta 20px en los impares:

- Nombre:Teresa
- Apellido: Rodriguez
- Nota:10.00
- Nombre:Jose
- Apellido: Romero
- Nota:9.00
- Nombre:Maria
- Apellido: Suarez
- Nota:8.00
- Nombre:Ignacio
- Apellido: Perez
- Nota:4.9
- Nombre:Sara
- Apellido: Alvarez
- Nota:4.9

Por último damos un mejor aspecto al listado.

NOTA: Este apartado se ha realizado al final, tras haber completado otros ejercicios que también pintan elementos en la página, como por ejemplo los ejercicios 7 y 9 en los que se añaden botones y funcionalidades con js que también aparecen en las capturas, ya que se le han aplicado los estilos correspondientes para unificar el aspecto de nuestra página.

- Sin pulsar botones (sin desplegar el listado de alumnos) y con una anchura de 768 px o más:

The screenshot shows a web browser at the URL `curso.magento.local/prueba/index/index`. The page features the Magento logo and a search bar. Below the header, there is a section titled "Alumnos" with two orange buttons: "Alumnos/as" and "Nota mas alta". A light gray panel is expanded, displaying the following statistics:

- Las 3 mayores notas obtenidas**: | 10.00 | 9.00 | 8.00 |
- Media de notas**: 6.95
- Total de alumnos**: 6

- Pulsando el botón Alumnos/as, se despliega la información gracias al toogle de Js con los estilos personalizados que se pueden ver, realizados modificando la plantilla de nuestro tema (`index.phtml`) y el archivo less correspondiente:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

```
app/design/frontend/Hiberus/custom-theme/Magento_Theme/web/css/source/_extend.less
```

En el archivo:

```
app/design/frontend/Hiberus/custom-theme/web/css/source/_extend.less
```

tenemos la variable `@margin-left-primary` para el margen de 20px agregada a la clase y se importa desde `_mixin.less`

```
app/design/frontend/Hiberus/custom-theme/web/css/source/_mixin.less
```


para aplicar el mixin al archivo less del tema:

```
app/design/frontend/Hiberus/custom-theme/Magento_Theme/web/css/source/_extend.less
```

Como en anteriores ejercicios se ha explicado ya y mostrado la captura de el listado desplegado en versión desktop, para no repetir vamos a mostrarlo en esta captura en la versión para pantallas de menos de 768 px (el título cambia a naranja) ya con los estilos personalizados:



The screenshot shows a mobile browser view of a Magento website. The address bar displays 'curso.magento.local/prueba'. The page header features the Magento logo and a search icon. The main content area is titled 'Alumnos' in orange. Below the title are two orange buttons: 'Alumnos/as' and 'Nota mas alta'. A vertical orange bar is on the left. The student list is as follows:

Alumnos/as	Nota mas alta
Nombre Teresa Apellidos Rodriguez Nota 10.00	
Nombre Jose Apellidos Romero Nota 9.00	
Nombre Maria Apellidos Suarez Nota 8.00	

Nombre Ignacio

Apellidos Perez

Nota 4.9

Nombre Sara

Apellidos Alvarez

Nota 4.9

Nombre Antonio

Apellidos Garcia

Nota 4.9

Las 3 mayores notas obtenidas

| 10.00 | 9.00 | 8.00 |

Media de notas

6.95

Total de alumnos

6

Para aplicar los cambios y verlos en el navegador (solo ha funcionado así) ejecutamos en el terminal:

- `dockergento magento setup:upgrade`
- `dockergento magento setup:di:compile`
- `dockergento magento cache:flush`

Error frecuente: Nos aseguramos que esté asignado nuestro tema en el admin para que se reflejen los cambios aplicados y podemos ver el resultado final en la URL:

`http://curso.magento.local/prueba/index/index`

Ejercicio 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. (EXTRA: Utiliza el jQuery widget "alert" para mostrar esta nota)

Como ya indicamos en el ejercicio 7, JavaScript no se ha dado en clase, pero se intenta realizar con ayuda de internet, documentación y conocimientos previos. Estos son los pasos:

Creamos un método para obtener la nota más alta de clase en:

`app/code/Hiberus/Rodriguez/Block/Index`

Ahora vamos al archivo js:

`app/code/Hiberus/Rodriguez/view/frontend/requirejs-config.js`

Aquí ya teníamos el toggle para el botón del ejercicio 7, ahora le añadimos la línea del alert:

```
var config = {
  map: {
    '*': {
      toggle: 'Hiberus_Rodriguez/js/toggle',
      alert: 'Hiberus_Rodriguez/js/alert'
    }
  }
}
```

```
};
```

En la plantilla, en la ruta:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

creamos el botón y añadimos el script, y al script del final le incluimos el alert:

```
<button id="notaMasAlta">nota más alta</button>
```

```
...
```

```
<script>
```

```
    var maxNote = <?= $block->getMaxMark() ?>
```

```
</script>
```

```
<script>
```

```
    require(['jquery', 'toggle'], function ($, myscript) {
```

```
        myscript('#desplegar', '#details');
```

```
    });
```

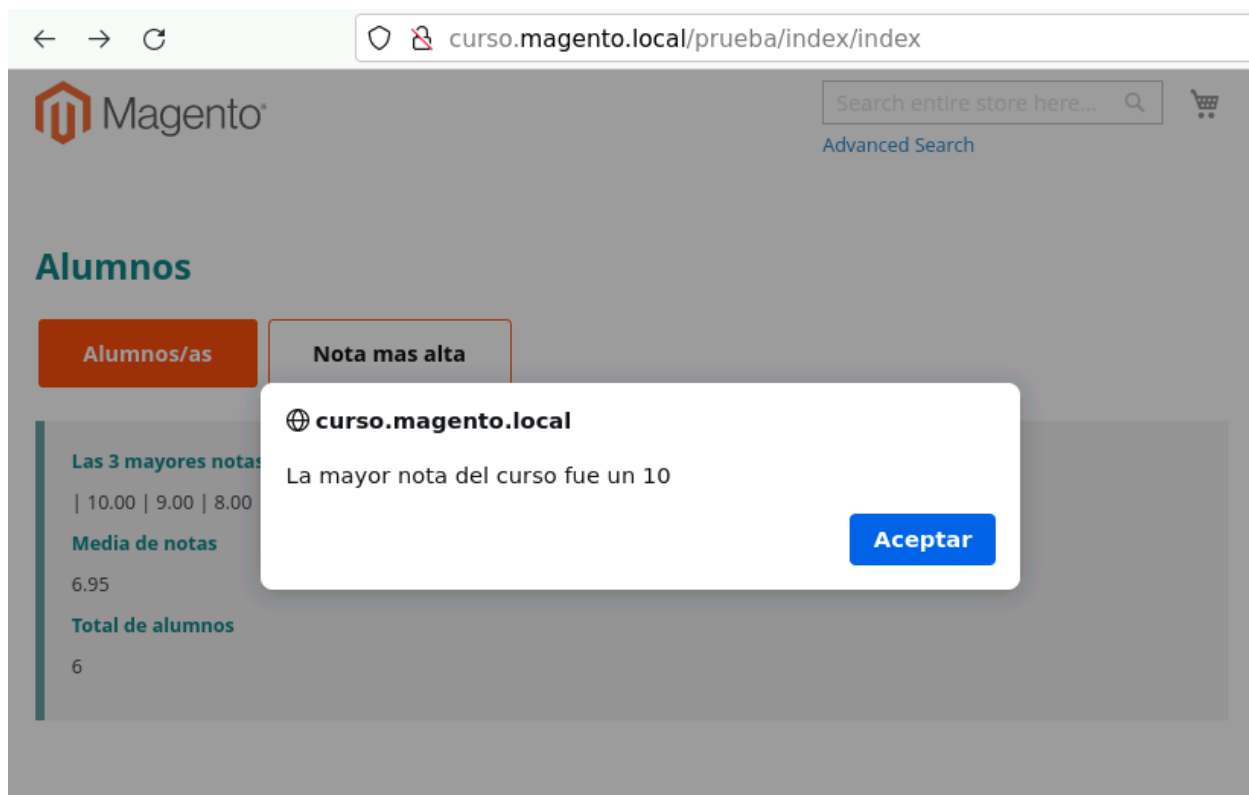
```
    require(['jquery', 'alert'], function ($, alertNotaAlta) {
```

```
        alertNotaAlta('#notaMasAlta');
```

```
    });
```

```
</script>
```

El resultado al pulsar el botón de la nota más alta es el siguiente:



Ejercicio 10

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

Creemos un método que realice el cálculo en la ruta:

`/app/code/Hiberus/Rodriguez/Block/index.php`

el método `getAverageMarks` que tras obtener los datos de la tabla de nuestra base de datos, recorre los registros para extraer de cada uno las notas de cada alumno que guarda en un array, del que se hace la media sumándolas y dividiendo la suma por el total de notas.

Esa media de las notas es el dato que retorna el método y el que necesitamos llevarnos a la vista.

Para ello en nuestra plantilla, en la ruta:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

Llamamos al método creado en Block/Index.php y asignamos el resultado a una variable,

```
$notas = $block->getAverageMarks();
```

que posteriormente pintamos, asignándole las etiquetas y clases adecuadas a nuestro diseño:

```
<p class="textBold textColor"> Media de notas </p>
```

```
<p><?= $notas ?></p>
```

También pintamos el total de alumnos:

```
<p class="textBold textColor"> Total de alumnos </p>
```

```
<p><?= (count($alumnos)) ?></p>
```

```
</div>
```

El resultado estará en la parte inferior de la página y el siguiente:

Media de notas
6.95
Total de alumnos
6

Ejercicio 11

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

Creamos el plugin en la siguiente ruta:

```
app/code/Hiberus/Rodriguez/Plugins/Marks/MarksPlugin.php
```

En la clase `MarksPlugin` creamos un método que cargue las notas y con una lógica condicional, asignamos la nota de 4.9 a todos los suspensos.

```
<?php
namespace Hiberus\Rodriguez\Plugin\Marks;
use \Hiberus\Rodriguez\Model\Notas;
class MarksPlugin{
    public function afterGetMark(Notas $subject, $result){
        if ($subject->getData('mark') < 5.0) {
            $subject->setMark(4.9);
            $result = $subject->getData('mark');
        }
        return $result;
    }
}
```

Y en la ruta de nuestro archivo `di.xml`:

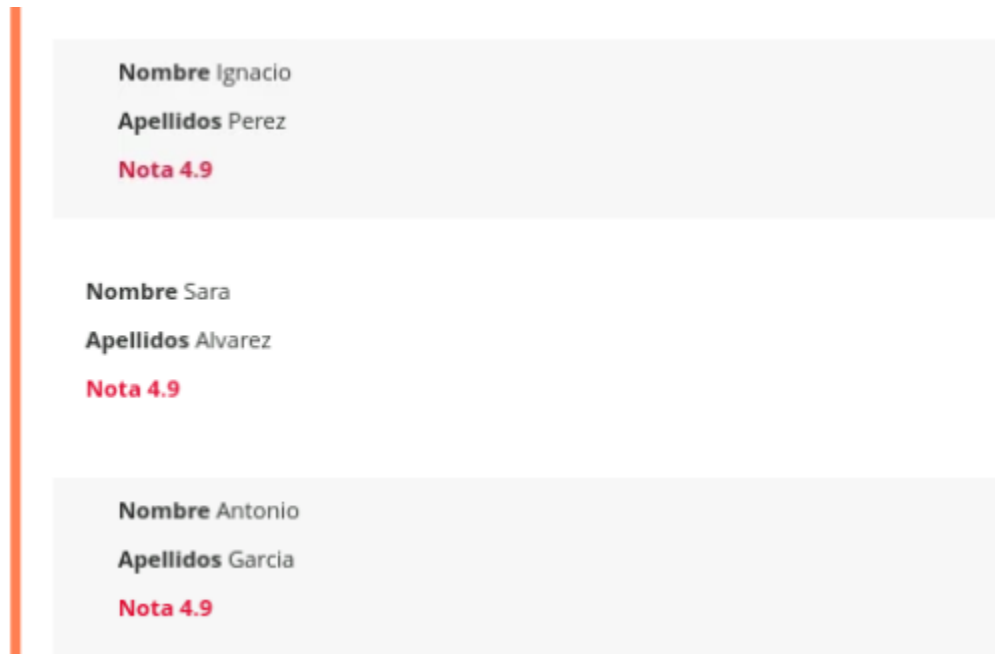
```
app/code/Hiberus/Rodriguez/et/di.xml
```

Declaramos el plugin que acabamos de crear incluyendo lo siguiente:

```
<type name="Hiberus\Rodriguez\Model\Notas">
```

```
<plugin name="notas_plugin"  
type="Hiberus\Rodriguez\Plugin\Marks\MarksPlugin" sortOrder="10" />  
</type>
```

El plugin cambia los suspensos a 4.9 como puede verse en la captura:



Ejercicio 12

Que los alumnos aprobados aparezcan en un color y los suspensos en otro. (EXTRA: Define estos estilos mediante un LESS MIXIN, de modo que el color a aplicar sea un parámetro de entrada del mixin).

En nuestra plantilla index.phtml, en la ruta:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```


Aplicamos la lógica necesaria para que si un alumno tiene una nota de 5 o más, se le aplique una clase que pinte verde la nota, y si es menos, se aplique otra clase que pinte roja la nota usando un simple condicional del bucle que recorre y pinta los alumnos.

```
if ($alumno->getMark() >= 5) {  
    $classEvaluation = "approved";  
} else {  
    $classEvaluation = "suspended";  
}
```

Los colores de las clases se definen en el archivo `_mixin.less` y en el `_extend.less` se hace uso de la variable `color` para asignar el color rojo para los suspendidos o verde para los aprobados.

En el archivo `_extend.less` para que se apliquen los estilos los ponemos dentro del `common`:

```
& when (@media-common = true) {  
...  
    .approved{  
        .notas-color(teal);  
        font-weight: bold;  
    }  
    .suspended{  
        .notas-color(crimson);  
        font-weight: bold;  
    }  
...  
}
```

En la captura se ve una muestra del listado en la que se aprecia cómo quedan en un tono rojizo las notas los suspensos y de color “Teal” los aprobados.

Nombre Jose Apellidos Romero Nota 9.00
Nombre Maria Apellidos Suarez Nota 8.00
Nombre Ignacio Apellidos Perez Nota 4.9
Nombre Sara Apellidos Alvarez Nota 4.9
Nombre Antonio Apellidos Garcia Nota 4.9

Ejercicio 13

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

Vamos a mostrar en el div inferior de nuestra plantilla, junto a la información referida a la clase, las tres mejores calificaciones obtenidas por los alumnos de la clase siguiendo el estilo que hemos aplicado para esta zona.

Para ellos creamos el método `getMaxMarks` en:

```
/app/code/Hiberus/Rodriguez/Block/index.php
```

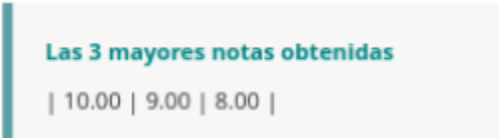
este método obtiene las notas de la base de datos y nos devuelve un array con las 3 calificaciones más altas obtenidas, esto luego es lo que pintamos en nuestra plantilla en la ruta:

```
/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml
```

haciendo uso de un bucle para pintar las tres calificaciones con los estilos pertinentes y separados por barras.

```
<div class="calculate">
<p class="textBold textColor">Las 3 mayores notas obtenidas</p>
|
<?php foreach($maxMarks as $mark) { ?>
    <?= $mark ?> |
<?php } ?>
...
</div>
```

Se ve en el bloque inferior de la siguiente forma:



```
Las 3 mayores notas obtenidas  
| 10.00 | 9.00 | 8.00 |
```

Ejercicio 14

Crear un CLI command nuevo que permita ver todos los datos de la tabla de exámenes, se valorará que NO se haga uso del object manager. Este se debe llamar como tu apellido bajo el namespace Hiberus (hiberus:apellido).

Creamos la clase `NotasCommand` en la ruta:

```
app/code/Hiberus/Rodriguez/Console/NotasCommand.php
```

Creamos un `construct` para tener `block` y `notas`.

Creamos el método `configure` con el nombre del comando `'Hiberus_Rodriguez'` y su descripción.

Creamos el método `execute` donde nos traemos todos los alumnos con `getAlumnos()`, con un bucle los recorremos y modificamos haciendo uso de los `getters` y `setters` y con el `output` sacamos por consola.

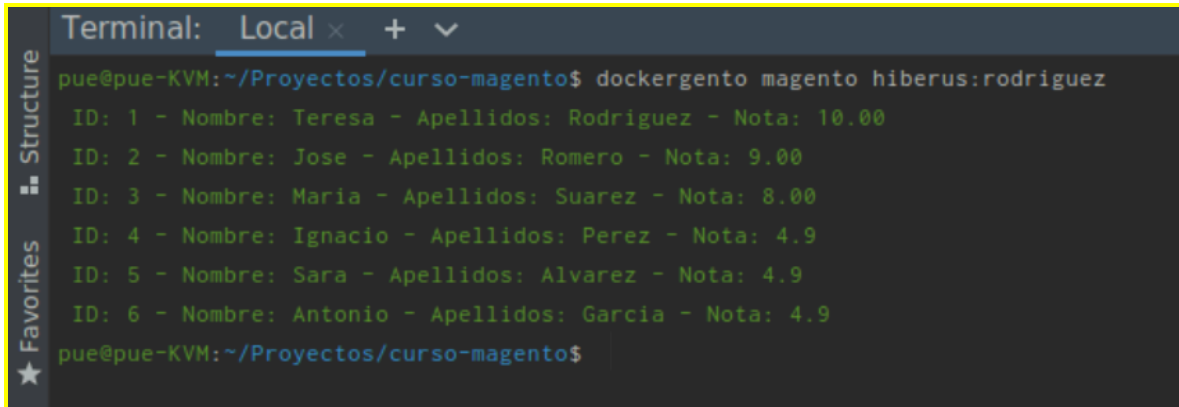
En la terminal ejecutamos los comandos:

- `dockergento magento setup:upgrade`
- `dockergento magento setup:di:compile`
- `dockergento magento cache:flush`

Y finalmente probamos nuestro comando poniendo en la terminal:

- `dockergento magento hiberus:rodriguez`

Obteniendo la salida esperada por consola:



```
Terminal: Local x + v
pue@pue-KVM:~/Proyectos/curso-magento$ dockergento magento hiberus:rodriguez
ID: 1 - Nombre: Teresa - Apellidos: Rodriguez - Nota: 10.00
ID: 2 - Nombre: Jose - Apellidos: Romero - Nota: 9.00
ID: 3 - Nombre: Maria - Apellidos: Suarez - Nota: 8.00
ID: 4 - Nombre: Ignacio - Apellidos: Perez - Nota: 4.9
ID: 5 - Nombre: Sara - Apellidos: Alvarez - Nota: 4.9
ID: 6 - Nombre: Antonio - Apellidos: Garcia - Nota: 4.9
pue@pue-KVM:~/Proyectos/curso-magento$
```

Ejercicio 15

Crear 3 endpoint nuevo de Api Rest con Swagger:

- Permitir ver todos los datos de la tabla de exámenes.
- Crear otro endpoint que permita borrar alumnos por id.
- Crear otro que permita guardar un nuevo alumno y su nota.

No se ha dado en clase.

Ejercicio 16

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

- Poder configurar cuantos elementos mostraremos en el listado de exámenes que hemos creado en el frontend, en la nueva página.
- Poder configurar cual es la nota que marca el aprobado (por defecto 5.0)

Creamos el archivo `system.xml` en la ruta

```
app/code/Hiberus/Rodriguez/etc/adminhtml/system.xml
```

en el que creamos la nueva sección de configuración para la administración, con la tab Hiberus.

```
<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Config/etc/
system_file.xsd">

    <system>

        <tab id="hiberus" sortOrder="999" translate="label">

            <label>Hiberus</label>

        </tab>

        <section id="hiberus_elementos" showInDefault="1"
sortOrder="10">

            <label>Notas</label>

            <tab>hiberus</tab>

            <resource>Hiberus_Rodriguez::config</resource>

            <group id="general" showInDefault="1" sortOrder="10">

                <label>General</label>

                <field id="elementos" type="text" sortOrder="10"
showInDefault="1">

                    <label>Número de elementos a mostrar</label>

                    <validate>integer</validate>
```

```

</field>

    <field id="aprobados" type="text" sortOrder="10"
showInDefault="1">

        <label>Nota mínima para el aprobado</label>

        <validate>validate-number</validate>

    </field>

</group>

</section>

</system>

</config>

```

Y su reflejo en la administración lo vemos en Stores/ Configuración está el tab Hiberus / Notas

The screenshot shows the Magento Configuration interface. On the left is a sidebar with navigation icons for Dashboard, Sales, Catalog, Customers, Marketing, Content, Reports, Stores, System, and Find Partners & Extensions. The main area is titled 'Configuration' and shows the 'Default Config' scope. A 'Save Config' button is in the top right. The left sidebar of the configuration area lists categories: GENERAL, CATALOG, SECURITY, CUSTOMERS, SALES, YOTPO, DOTDIGITAL, HIBERUS (expanded), and Notas. The 'General' section is active, showing two fields: 'Numero de elementos a mostrar [global]' with a value of 20, and 'Nota mínima para el aprobado [global]' with a value of 5.

Posteriormente en:

`app/code/Hiberus/Rodriguez/Block/Index.php`

Se declaran los métodos necesarios. Para poder obtener los datos desde el tab Hiberus que hemos creado, tenemos dos métodos, `getElementos` que nos trae la cantidad de registros a mostrar y `getNota` que nos trae la nota mínima que se considera como aprobado.

...

```
public function getElementos() {
    $elementos = $this->scopeConfig->getValue(
        'hiberus_elementos/general/elementos', ScopeInterface::SCOPE_STORE);
    return $elementos;
}
```

...

```
public function getNota() {
    $nota = $this->scopeConfig->getValue(
        'hiberus_elementos/general/aprobados', ScopeInterface::SCOPE_STORE);
    return $nota;
}
```

...

En nuestra plantilla, en la ruta:

`/app/code/Hiberus/Rodriguez/view/frontend/templates/index.phtml`

Cambiamos el valor numérico de 5 que teníamos como aprobado, por la variable que guarda la nota mínima que se considera aprobada.

```
<div id="details"> <!-- div toggle btn -->
```



```
<?php
foreach ($alumnos as $alumno) {
    // Para que salga rojo suspensos y verdes aprobados
    if ($i++ == $elementos) break; // corta el listado
    if ($alumno->getMark() > $notaMinima) { // antes >= 5
        $classEvaluation = "approved";
    } else {
        $classEvaluation = "suspended";
    }
}
?>
...
</div>
```

Ahora el aprobado se mostrará en rojo o verde, según se ha configurado el valor de aprobado en el panel de administración en nuestro tab Hiberus.

Alumnos

Alumnos/as	Nota mas alta
Nombre Teresa Apellidos Rodriguez Nota 10.00	
Nombre Jose Apellidos Romero Nota 9.00	
Nombre Maria Apellidos Suarez Nota 8.00	
Nombre Ignacio Apellidos Perez Nota 4.9	

Ejercicio 17

Crear un custom Logger que registre cada vez que se accede a la página nueva del listado de exámenes y nos indique cuántos alumnos se van a mostrar y cuál es la nota media, para tener un control de qué se le está mostrando al cliente cuando accede a esta página.

- El fichero de log nuevo, deberá llamarse con tu apellido bajo el namespace hiberus: `hiberus_garcia.log`
- No se deberá crear ninguna clase nueva, se deberán utilizar los virtual types para solucionar este problema.

No se ha dado en clase.

Indicaciones para elaborar la presente memoria

Por último, se pide que durante la realización de la prueba vayáis anotando las cosas que vais haciendo para redactar una memoria donde se explique lo que hacéis y las tomas de decisiones que realicéis en cada uno de los ejercicios.

Con esto se redactará un documento que se exportará a PDF y se adjuntará con el módulo desarrollado. A la hora de escribir la memoria, imaginad que os preguntan cara a cara qué habéis hecho en este módulo.

Cómo entregar la prueba

Se creará un repositorio de Github público donde o bien vais incorporando progresivamente todo lo que vais haciendo durante los días o bien lo subís todo de golpe al final, de esta manera luego revisarlo es mucho más accesible y por otro lado os sirve para desenvolveros con este control de versiones que vais a necesitar usar en cualquier trabajo. En este repositorio deberán estar tanto el módulo desarrollado, como el PDF de la memoria escrita. Finalmente lo que nos tendréis que mandar es tan solo el enlace a vuestro repositorio al correo que os indiquemos

Enlace a la prueba en GitHub

<https://github.com/grafismoweb/examen-magento>