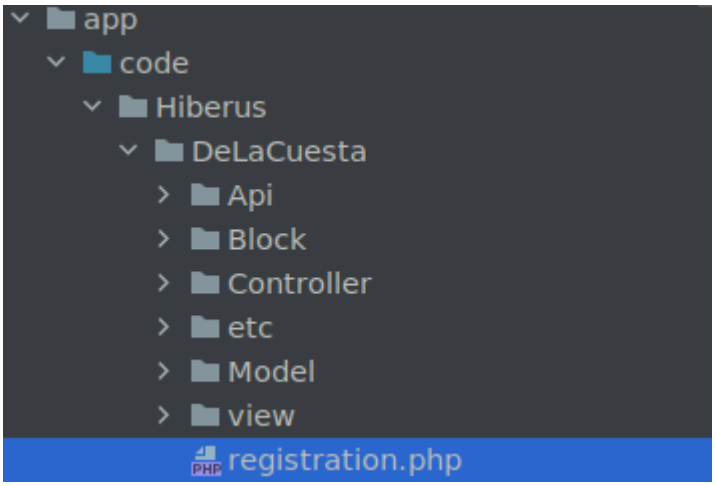
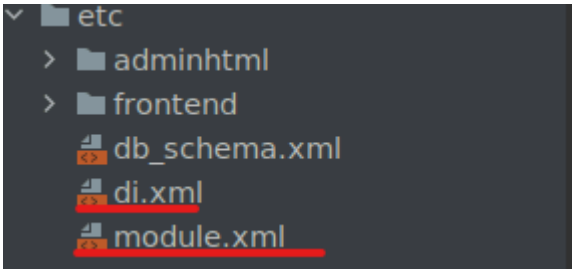
# ENLACE AL GITHUB

1.

Creo una carpeta llamada DeLaCuesta, la cual será el módulo, con su registration.php, el cual indicará a Magento como localizar nuestro módulo, en este caso “Hiberus_DeLaCuesta”.



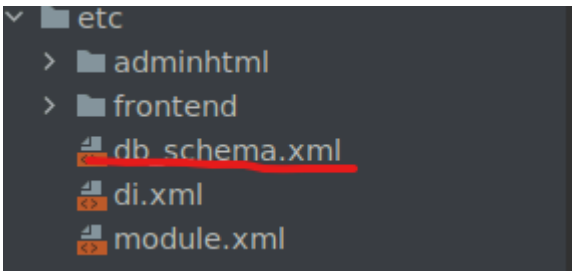

También, dentro de la carpeta etc, hago el archivo module.xml, para registrar primeramente el módulo. Dentro de ella, realizo un di.xml para futuros ejercicios, donde añadiré los plugins qué necesite.




Después de realizarlo, realizo el comando dockergento magento setup:upgrade

2.

Dentro de la carpeta etc, creo un archivo llamado db_schema.xml, donde haré mi tabla llamada “hiberus_exam”, con sus columnas. Para hacer el id autoincrementable, uso identity=”true”.

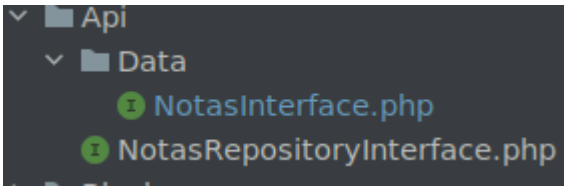



```xml
<?xml version="1.0"?>
<schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">
    <table name="hiberus_exam" resource="default" engine="innodb" comment="Tabla Examen Hiberus">
        <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>
```

Después, realizo el comando dockergento magento setup:upgrade y dockergento magento setup:di:compile para qué surjan efecto todos los cambios y compruebo la base de datos para verificarlo.

3.

Genero las carpetas Api\Data, dentro de esta última una interfaz llamada NotasInterface genero todos los getters y setters, definiendo dos constantes, una llamada TABLE_NAME, donde se guardará el nombre de la tabla (en este caso hiberus_exam), y otra llamada COLUMN_ID, la cual hará referencia al id de la tabla (llamado id_exam).




```php
const TABLE_NAME = 'hiberus_exam';

const COLUMN_ID = 'id_exam';
```

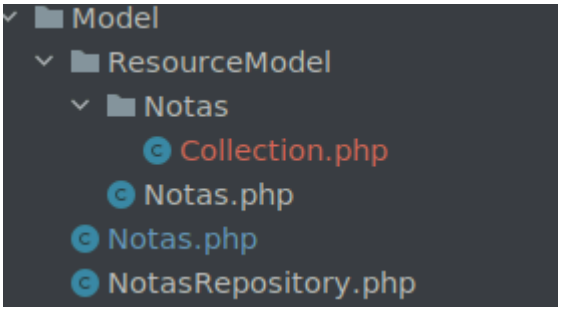
Después de generar los getters y setters con su PHDoc, en Api\, genero la interfaz NotasRepositoryInterface, para definir una interfaz al repositorio como indica su nombre.

```php
<?php

namespace Hiberus\DeLaCuesta\Api;
use \Hiberus\DeLaCuesta\Api\Data\NotasInterface;

interface NotasRepositoryInterface {

    /**
     * @param NotasInterface $notasInterface
     * @return NotasInterface
     */
    public function save(NotasInterface $notasInterface);

    /**
     * @param $idExam
     * @return NotasInterface
     */
    public function getById($idExam);
```

\Hiberus\DeLaCuesta\Api › NotasRepositoryInterface

Dentro, le genero 4 métodos: save, getById, delete y deleteById, trayéndome los datos de NotasInterface.

Por último, genero la carpeta Model\, dentro de ella, genero otra segunda carpeta llamada ResourceModel, y una clase llamada Notas.php donde generaré los métodos save, load, y delete con sus respectivos constructores.



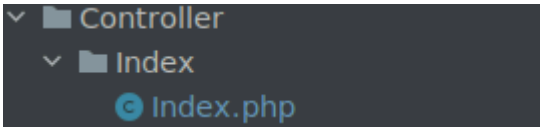

Tras ello, en la carpeta Model\, Me hago el modelo llamado Notas.php, donde usaré mi interfaz generada anteriormente para generar los getters y setters

```php
<?php

namespace Hiberus\DeLaCuesta\Model;

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


class Notas extends AbstractModel implements NotasInterface
{

    protected function _construct() {
        $this->_init( resourceModel: \Hiberus\DeLaCuesta\Model\ResourceModel\Notas::class);
    }
```

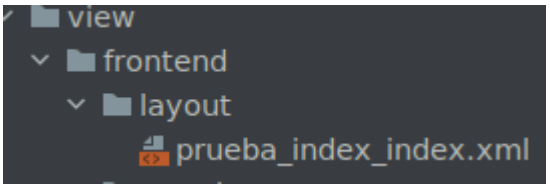
4.

Al no verlo en clase, Carlos nos dice qué usemos el Factory para que saque nombres y apellidos aleatoriamente desde un array, pero al no poder insertar en bases de datos los inserto manualmente y no los hago aleatorios.

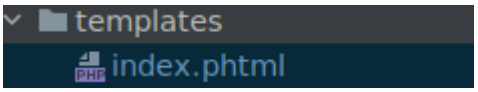
Para ello, creo el controlador Index.php dentro de Controller\Index.




Dentro de ella llamo a un constructor con las respectivas clases, y en un método execute() llamo al método create de pageFactory para generar la vista, la cual se encontrará en la carpeta view\frontend\layout, llamada en mi caso prueba_index_index.xml, ya qué en etc\frontend\routes.xml me hice un router con frontName y id prueba, del módulo Hiberus_DeLaCuesta.




Después, genero también dentro de view\frontend la carpeta templates, y dentro un archivo index.phtml, Donde declararé una variable con @var haciendo referencia al Block\Index el cual generaré ahora, y guardandolo en una variable $block, donde usaré el método getAlumno qué también crearé dentro de Block\Index.

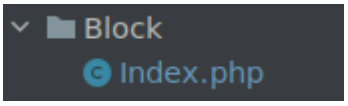



```php
<?php

/**
 * @var \Hiberus\DeLaCuesta\Block\Index $block
 */
$alumnos = $block->getAlumno();
$notas = $block->getAverageMarks()
```

Debajo, muestro los nombres, apellidos y notas respectivos con sus getters.

```php
<?php

    foreach($alumnos as $alumno) {

?>

<ul>
    <li>Nombre:<?= $alumno->getFirstName(); ?></li>
     <li>Apellido: <?= $alumno->getLastName(); ?></li>
    <li>Nota:<?= $alumno->getMark(); ?></li>
```

Genero la carpeta Block, y dentro un archivo llamado Index.php.

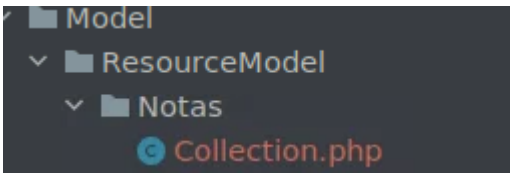



Dentro, llamaré a su respectivo constructor y después un getAlumno, donde llamaré a un objeto con un método llamado insertAlumno, donde insertaré los nombres, apellidos y notas de los alumnos, para después devolver el objeto qué almacenaré en una variable llamada $crearAlumno.

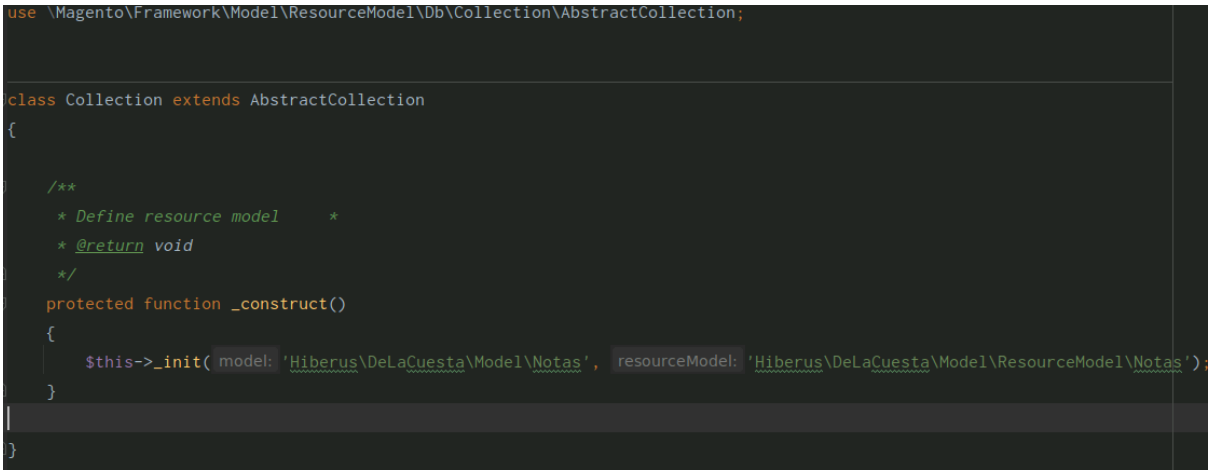
Debajo, creo el método insertAlumno al cual hago referencia en el anterior, pasándole primero el Factory->create() y generando los setters para los campos, guardandolos con save() y devolviendo el id de los alumnos para recogerlo en el getAlumno().

Al realizar todo esto e ir a la vista, me falla al decir qué firstname no puede ser null, cuando compruebo con Xdebugger qué este valor y los demás no sean null, corroborando qué tienen datos pero al momento de la inserción en la base de datos falla por algún motivo el cual no encuentro. He intentado todo lo posible para ello pero no deja de fallar, aún revisando la documentación y comprobando qué todo está correcto, buscando en stackOverflow, pero no logro dar con el problema aún comparando el código del ejemplo con el realizado.

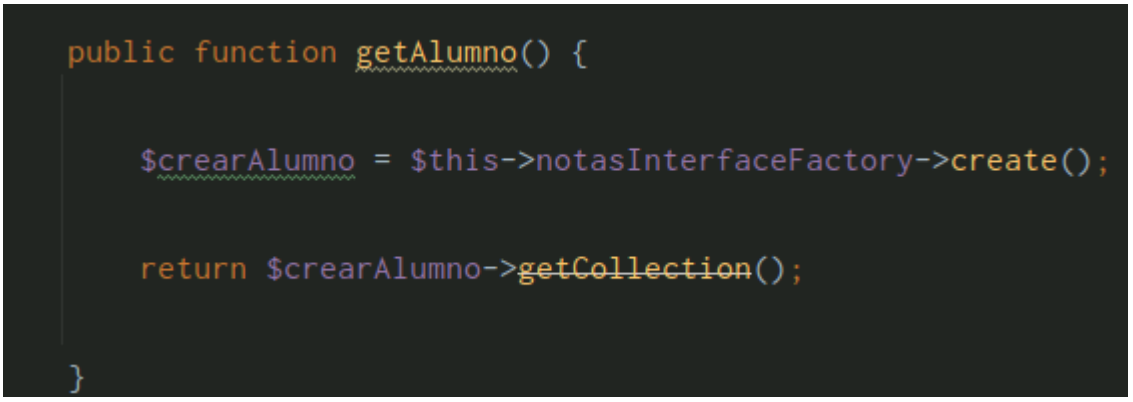
De nuevo, pruebo otra solución al crear dentro de Model\ResourceModel una carpeta llamada Notas, y dentro una clase llamada Collection.php



Dentro instancio un constructor en el qué le hago $this->_init() parametrizado con mi modelo Notas y mi resourceModel Notas.




Después, modifico el get de Block\Index para eliminar el set y llamar al Factory realizando un create, asignándolo a una variable y llamando a getCollection()




Al realizar todo esto, en la vista .phtml le coloco un foreach para sacar cada dato de la base de datos como he indicado en la captura anterior, lográndolo.

5.

Para hacer el echo, simplemente dentro del controlador le paso un pageFactory al metodo execute, y antes de qué ejecute create le agrego un echo de la siguiente forma:

```php
public function execute()
{
    echo "hola";
    return $this->pageFactory->create();
}
```

El resultado en vista es el siguiente:


6.

Realizo todos los pasos, añadiendo un h2 con la clase title

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

y un ul con sus respectivos li

```php
<?php

    foreach($alumnos as $alumno) {

?>

<ul>
    <li>Nombre:<?= $alumno->getFirstName(); ?></li>
     <li>Apellido: <?= $alumno->getLastName(); ?></li>
    <li>Nota:<?= $alumno->getMark(); ?></li>
</ul>
```

pero al momento de hacer la traducción no lo logro, ya qué no llegamos a dar eso en el curso de Magento, por lo qué tan solo hago un count de los alumnos y un párrafo con un literal donde coloco “Total de alumnos” y los muestro al lado.

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

7.

No llegamos a ver JavaScript durante el curso de Magento, pero aún así lo realizaré. (Todo esto está sacado de la documentación)

Dentro de la carpeta view\frontend, creamos un archivo llamado requirejs-config.js. Dentro de este, declaramos el módulo a utilizar y el archivo de Javascript qué usaremos:

```
var config = {
    map: {
        '*': {
            btndisplay: 'Hiberus_DeLaCuesta/js/btndisplay
        }
    }
};
```

Después, crearemos el archivo javascript en la siguiente ruta: view\frontend\web\js y se llamará btndisplay.js. Este archivo contendrá el script a realizar pero con variables para luego usarlo como un “método” en vista:

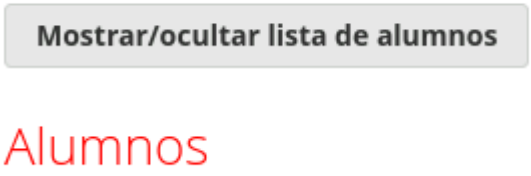
```
define(['jquery'], function($){
    "use strict";
    return function btndisplay(btn,div)
    {
        $(btn).click(function(){
            $(div).slideToggle();
        });
    }
});
```

Después en vista, declaramos el script y lo ejecutamos de la siguiente forma:

```
<script>
    require(['jquery', 'btndisplay'], function($, btndisplay) {
        btndisplay('#btnDisplay','#alumnos');
    });
</script>
```

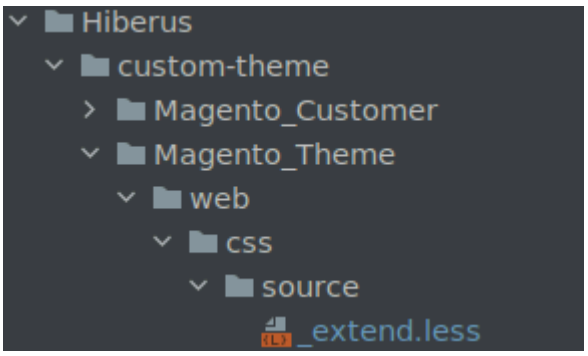
En mi caso, he hecho un botón con id btnDisplay y un div qué encierra toda la lista de alumnos con id alumnos. En vista se vería lo siguiente al mostrarlos:


Y al ocultarlos:




8.

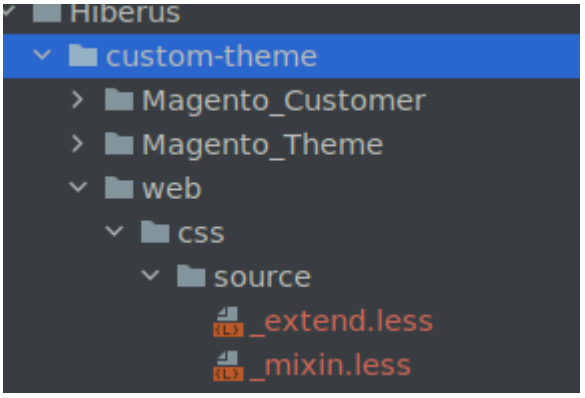
Primero, dentro de la siguiente ruta design\frontend\Hiberus\custom-theme\Magento_Theme\web\css\source añado un fichero llamado _extend.less




donde le paso un @media-common = true para qué aplique el color del título, en este caso verde. También le agrego qué a los impares les agregue un margin-left de 20px a través de una variable llamada @margin-left-primary declarada al principio del archivo.

```
@margin-left-primary: 20px;
& when (@media-common = true) {
    .page-main {
        .title{
            color:green;
        }
    }

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

Esta variable la tengo en otra ruta distinta la cual está en custom-theme\web\css\source_mixin.less donde le agrego a .notas-mixin la variable antes dicha, poniendo dentro el margin-left qué se le indique con esta variable.




```
.notas-mixin(@margin-left-primary){
    margin-left: @margin-left-primary;
}
```

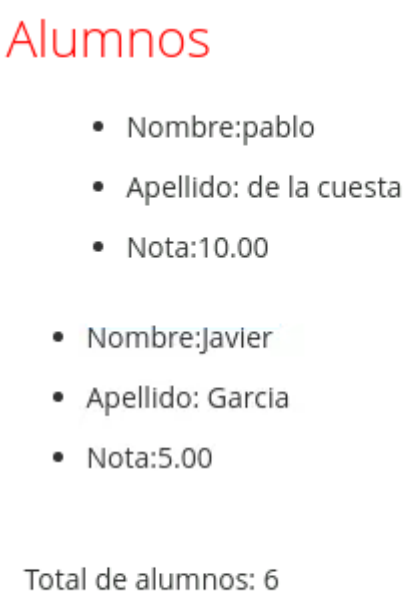
Luego, esta variable la importo desde otro archivo qué se encuentra en la misma carpeta llamado _extend.less, donde hago @import '_mixin.less' para qué se pueda aplicar este mixin a mi archivo _extend.less del Magento_Theme.

```
@import '_mixin.less';
```

Luego, hago un media-width para hacerlo responsive y qué a partir de los 768 píxeles el color cambie al qué tengo en el media-common, y de mientras qué no cambie y sea rojo.

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

Los resultados son los siguientes:




9.

No se ha visto en clase, pero aún así se intenta realizar el ejercicio. (Todo visto en la documentación)

Lo primero, hacemos un método en block para poder traernos la nota más alta de clase. El método realizado es el siguiente:

```php
public function getMaxMark() {
    $alumnos = $this->notasInterfaceFactory->create()->getCollection()->getData();
    $maxMark = 0;
    foreach ($alumnos as $alumno) {
        if ($alumno['mark'] > $maxMark) {
            $maxMark = $alumno['mark'];
        }
    }
    return $maxMark;
}
```

Después, creamos un archivo nuevo en la misma ruta qué el anterior JavaScript, esta vez llamado alertNotaAlta.js con un function para luego llamarlo en vista, donde le paso el id del botón para el alert

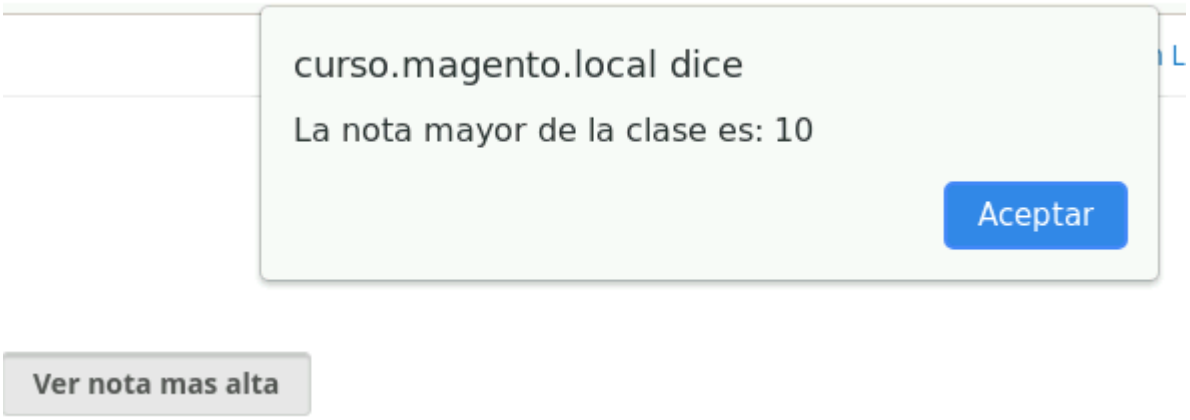
```php
public function getMaxMark() {
    $alumnos = $this->notasInterfaceFactory->create()->getCollection()->getData();
    $maxMark = 0;
    foreach ($alumnos as $alumno) {
        if ($alumno['mark'] > $maxMark) {
            $maxMark = $alumno['mark'];
        }
    }
    return $maxMark;
}
```

Por último, para la vista, le asignamos la variable a través del block directamente a una variable de JavaScript, y ejecutamos el script con un botón qué he creado en vista con el id notaMasAlta:

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

```
require(['jquery', 'alertNotaAlta'], function($, alertNotaAlta) {
    alertNotaAlta('#notaMasAlta');
});
</script>
```

El resultado en vista al darle al botón es el siguiente:




10.
Hago el método getAverageMarks(), donde le asigno a total los datos de la base de datos, y me creo un array vacio de notas. Luego, recorro total y voy sacando la nota de cada alumno y guardandola en el array notas. Una vez tengo todas ellas, saco la media de las notas de su sumatorio dividido la cantidad de notas qué hay, retornando la media a la vista.

```
public function getAverageMarks(){
    $resultPage = $this->notasInterfaceFactory->create();
    $total = $resultPage->getCollection();
    $notas = [];
    foreach ($total as $item){
        $notas[] = $item->getMark();
    }
    $mediaNotas = array_sum($notas)/count($notas);
    return $mediaNotas;
}
```

Después, la muestro en vista de la siguiente manera:

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

```
<p> Media de notas: <?=$notas?> </p>
```

El resultado de todo esto en vista es el siguiente:

- Nombre:Fran
- Apellido: Ruiz
- Nota:7.00

Media de notas: 5.5

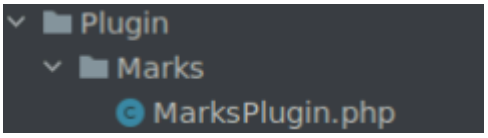
Total de alumnos: 6

.11.

Dentro del di.xml, declaramos el plugin el cual vamos a utilizar

```
<!-- Plugin de notas suspensas -->

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

Qué se encontrará dentro de una nueva carpeta qué haremos llamada Plugins/Marks/MarksPlugin.php




Dentro de ella, creo un método donde después de cargar las notas, las recojo y le asigno con un if a los valores qué sean menores de 5.0 un 4.9, es decir, la gente qué ha suspendido.

```
class MarksPlugin{

    public function afterGetMark(Notas $subject, $result)
    {
        if ($subject->getData( key: 'mark') < 5.0) {
            $subject->setMark( mark: 4.9);
            $result = $subject->getData( key: 'mark');
        }
        return $result;

    }

}
```

El resultado en vista es el siguiente:

# Alumnos

- Nombre:pablo
- Apellido: de la cuesta
- Nota:10.00

- Nombre:Javier
- Apellido: Garcia
- Nota:5.00

- Nombre:Javier
- Apellido: Garcia
- Nota:5.00

- Nombre:Jose
- Apellido: Perez
- Nota:4.9

- Nombre:Luis
- Apellido: Fernandez
- Nota:4.9

- Nombre:Fran
- Apellido: Ruiz
- Nota:7.00

12.

Dentro del index.phtml, hago el siguiente código PHP para aplicarle las clases según esten aprobados o suspendidos:

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

        if($alumno->getMark() > 4.9){

            $classEvaluation = "approved";
        }else{
            $classEvaluation = "suspensed";
        }

?>

<ul>
    <li>Nombre:<?= $alumno->getFirstName(); ?></li>
     <li>Apellido: <?= $alumno->getLastName(); ?></li>
    <li class="<?=$classEvaluation?>">Nota:<?= $alumno->getMark(); ?></li>
</ul>
```

Con el if compruebo qué si están aprobados les asigne la clase approved, y sino suspensed. La definición de los colores los hago a través de un mixin en el archivo _mixin.less:

```less
.notas-mixinMark(@color){
    color: @color;
}
```

Después, en el _extend.less, a través de la variable @color le coloco según quiera el color qué necesite:

```less
    .approved{
        .notas-mixinMark(green);
    }

    .suspensed{
        .notas-mixinMark(red);
    }
```

La vista quedaría de la siguiente forma:

# Alumnos

- Nombre:pablo
- Apellido: de la cuesta
- Nota:10.00

- Nombre:Javier
- Apellido: Garcia
- Nota:5.00

- Nombre:Javier
- Apellido: Garcia
- Nota:5.00

- Nombre:Jose
- Apellido: Perez
- Nota:4.9

- Nombre:Luis
- Apellido: Fernandez
- Nota:4.9

- Nombre:Fran
- Apellido: Ruiz
- Nota:7.00

13.

Para los 3 mejores alumnos, he decidido mostrar la nota de estos mismos. Para ello, dentro del block me he creado el siguiente método en el qué recojo las 3 mejores notas para luego mostrarlas en vista:

```php
public function getMaxMarks(){
    $total = $this->getAlumno();
    $marks = [];
    $maxMarks = [];
    foreach ($total as $item){
        $marks[] = $item->getMark();
    }
    $max = max($marks);
    foreach ($marks as $mark){
        $nota = $mark;
        if($nota <= $max && count($maxMarks) < 3){
            $notaMax = $nota;
            $maxMarks[] = $notaMax;
        }
    }
    return $maxMarks;
}
```

Después, en la vista hago lo siguiente al declararla en el block:

```php
Notas mas altas:
<?php foreach($maxMarks as $mark) { ?>
    <p><?= $mark ?></p>
<?php } ?>
```

El resultado en vista es el siguiente:

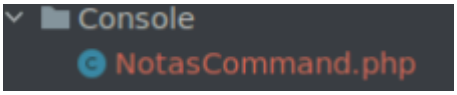
Notas mas altas:
10.00

5.00

5.00

14.

Para realizar el comando, nos creamos una nueva carpeta llamada Console, dentro de ella hago una clase llamada NotasCommand.php:




Dentro de NotasCommand, hago los constructores respectivos pasando block y notas:

```php
public function __construct(Index $block, Notas $notas)
{
    $this->block = $block;
    $this->notas = $notas;
    parent::__construct();
}
```

Después, en el configure, le asigno el nombre al comando y la descripción de lo qué hará:

```php
protected function configure()
{
    $this->setName('hiberus:delacuesta')
        ->setDescription('Mostrar examenes')
    parent::configure();
}
```

En el método execute, en $items le asigno el metodo qué creamos con anterioridad para coger todos los alumnos respectivos, getAlumno(). Después con un foreach recorro el array items y voy seteando con los getters qué recibo del array, para después en el output escribirlos por consola:

```php
protected function execute(InputInterface $input, OutputInterface $output)
{
    $items = $this->block->getAlumno();
    foreach ($items as $item) {
        $this->notas->setIdExam($item->getIdExam());
        $this->notas->setFirstName($item->getFirstName());
        $this->notas->setLastName($item->getLastName());
        $this->notas->setMark($item->getMark());
        $output->writeln( messages: '<info> ID: ' . $this->notas->getIdExam() . ' | Nombre: ' . $this->notas->getFirstName() . ' | Apellidos: ' . $this-
            ' | Nota: ' . $this->notas->getMark() . '</info>');
    }
}
```

Por último, al hacer los comandos setup, di:compile y cache:flush. El resultado del comando es el siguiente:

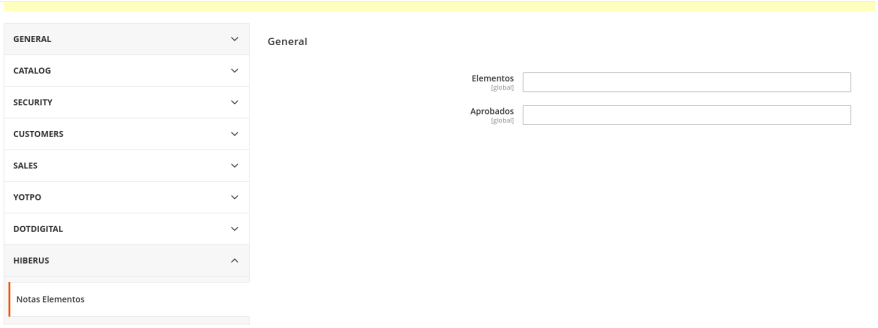
```
pue@pue-KVM:~/proyectos/curso-magento$ dockergento magento hiberus:delacuesta
ID: 1 | Nombre: pablo | Apellidos: de la cuesta | Nota: 10.00
ID: 2 | Nombre: Javier | Apellidos: Garcia | Nota: 5.00
ID: 3 | Nombre: Javier | Apellidos: Garcia | Nota: 5.00
ID: 4 | Nombre: Jose | Apellidos: Perez | Nota: 4.9
ID: 5 | Nombre: Luis | Apellidos: Fernandez | Nota: 4.9
ID: 6 | Nombre: Fran | Apellidos: Ruiz | Nota: 7.00
```

16.

Primero, creo el menú en el panel de administrador. Para ello, dentro de la carpeta etc\adminhtml creo un archivo llamado system.xml. Después, declaro la siguiente estructura para crear el menú:

```
<tab id="hiberus" sortOrder="999" translate="label">
    <label>Hiberus</label>
</tab>
<section id="hiberus_elementos" showInDefault="1" sortOrder="10">
    <label>Notas Elementos</label>
    <tab>hiberus</tab>
    <resource>Hiberus_DeLaCuesta::config</resource>
    <group id="general" showInDefault="1" sortOrder="10">
        <label>General</label>

        <field id="elementos" type="text" sortOrder="10" showInDefault="1">
            <label>Elementos</label>
            <validate>integer</validate>
        </field>

        <field id="aprobados" type="text" sortOrder="10" showInDefault="1">
            <label>Aprobados</label>
            <validate>validate-number</validate>
```

El resultado de todo esto desde el panel de administración es el siguiente:



Después desde el bloque, declaro los siguientes métodos a través del ScopeConfig, lo cual me trae los datos desde el panel del administrador:

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

```php
public function getNota() {
    $nota = $this->scopeConfig->getValue( 'hiberus_elementos/general/aprobados', ScopeInterface::SCOPE_STORE);
    if(is_null($nota)){
        $nota = 5;
    }
    return $nota;
}
```

En ellos recogeré la cantidad de elementos y el mínimo para aprobar respectivamente para enviarlos a vistas y recogerlos a través de un block. Al recogerlos, aprovecho un ejercicio anterior para mostrar como verde los aprobados y rojo los suspensos, poniendole la variable asignada a la nota mínima:

```php
if($alumno->getMark() > $notaMinima){

    $classEvaluation = "approved";
}else{
    $classEvaluation = "suspensed";
}
```

Para la cantidad de elementos, declaro una variable $i = 0 y dentro del foreach hago una comprobación con un if aumentando i por cada iteración, para cuando llegue a la cantidad de elementos directamente haga un break y no muestre más:

```php
<?php
    foreach($alumnos as $alumno) {
        if ($i++ == $elementos) break;
```