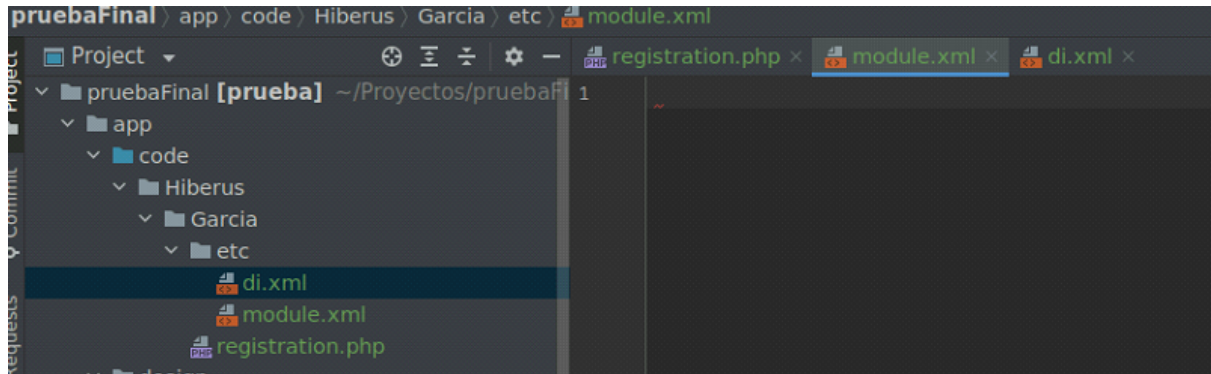


PRUEBA FINAL

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

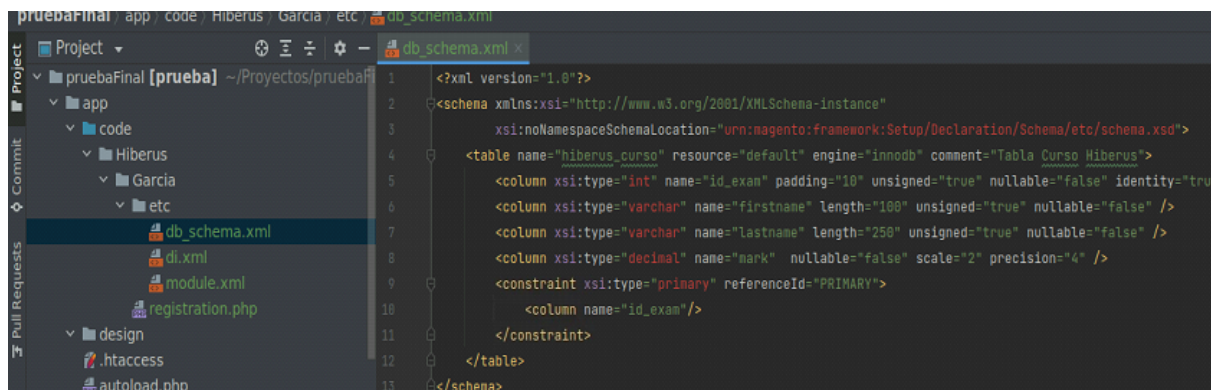


Me he creado un vendor llamado Hiberus, dentro mi modulo Garcia, dentro el fichero registration para que Magento sepa donde estará ubicado mi modulo. Luego nos creamos el fichero module.xml que este fichero es necesario para que exista nuestro modulo.

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

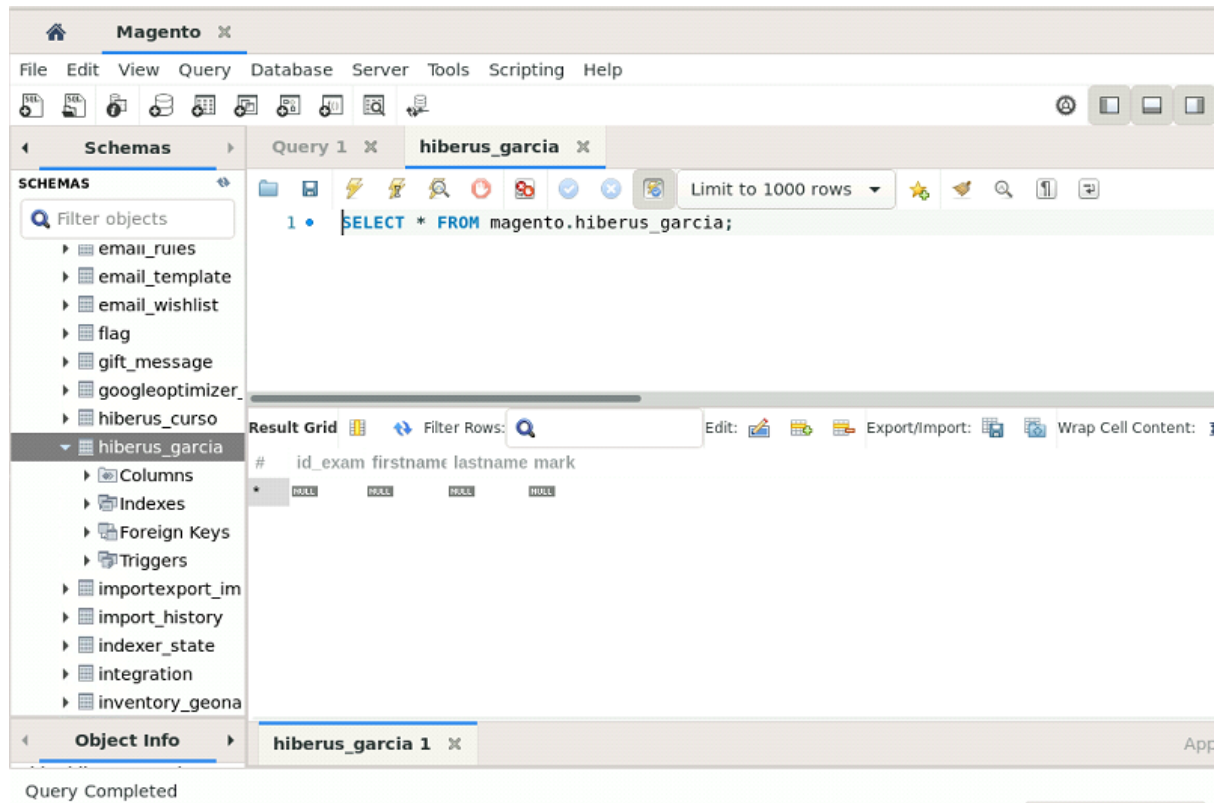
Field	Type	Null	Key	Default	Extra
id_exam	int(11)	NO	PRI	<null>	auto_increment
firstname	varchar(100)	NO		<null>	
lastname	varchar(250)	NO		<null>	
mark	decimal(4,2)	NO		<null>	

Mi fichero db_schema.xml:



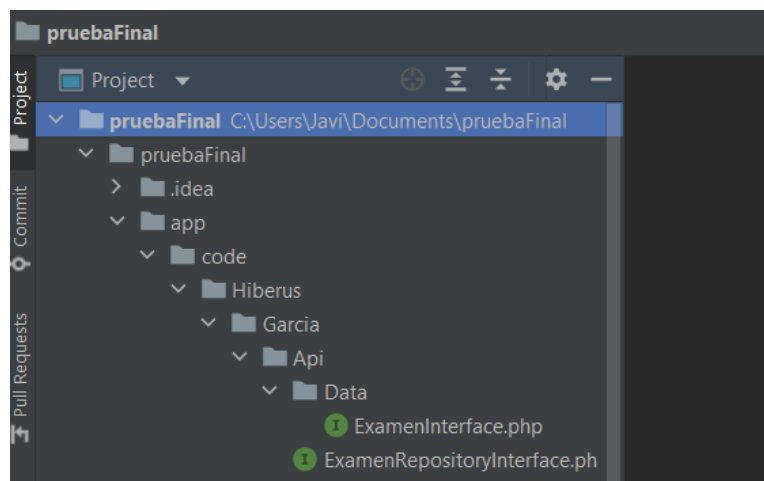
Mediante este fichero podremos crear la estructura de la base de datos de un módulo o de nuestro módulo mejor dicho.

Nos vamos al Workbench y comprobamos que nuestra tabla se haya creado correctamente:



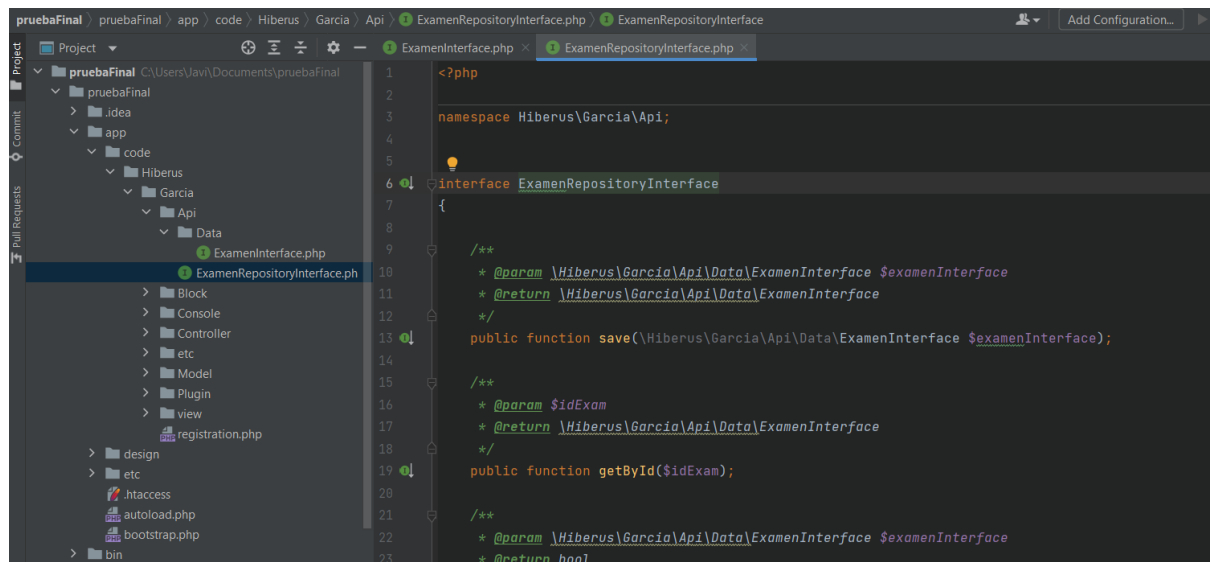
3 - Crear el Service Contracts y ORM que gestione esta entidad

Nos crearemos una estructura de carpetas dentro de nuestro módulo Garcia, dentro nos crearemos la siguiente estructura:



Nos crearemos una interfaz en donde crearemos los getters y setters, junto con dos constantes, en donde la TABLE_NAME identificará nuestra tabla de base de datos, **hiberus_garcia** y la COLUMN_ID será nuestro id primary, id_exam

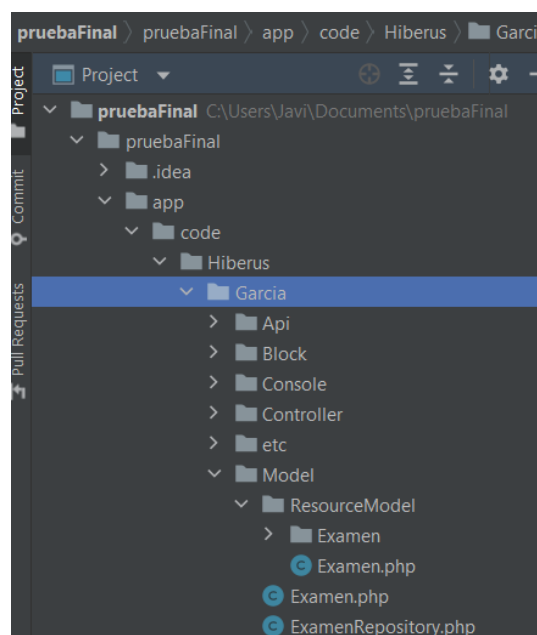
Una vez hecho esto generamos la interfaz **ExamenRepositoryInterface**:



```
<?php
1
2
3 namespace Hiberus\Garcia\Api;
4
5
6 interface ExamenRepositoryInterface
7 {
8
9     /**
10      * @param \Hiberus\Garcia\Api\Data\ExamenInterface $examenInterface
11      * @return \Hiberus\Garcia\Api\Data\ExamenInterface
12      */
13     public function save(\Hiberus\Garcia\Api\Data\ExamenInterface $examenInterface);
14
15     /**
16      * @param $idExam
17      * @return \Hiberus\Garcia\Api\Data\ExamenInterface
18      */
19     public function getById($idExam);
20
21     /**
22      * @param \Hiberus\Garcia\Api\Data\ExamenInterface $examenInterface
23      * @return bool
24     */
25 }
```

Mediante este repositorio podremos hacer consultas hacia la base de datos.

Una vez que hemos creado esto, crearemos otra estructura de carpetas. \Model\ResourceModel, dentro nos crearemos una clase llamada Examen.php:

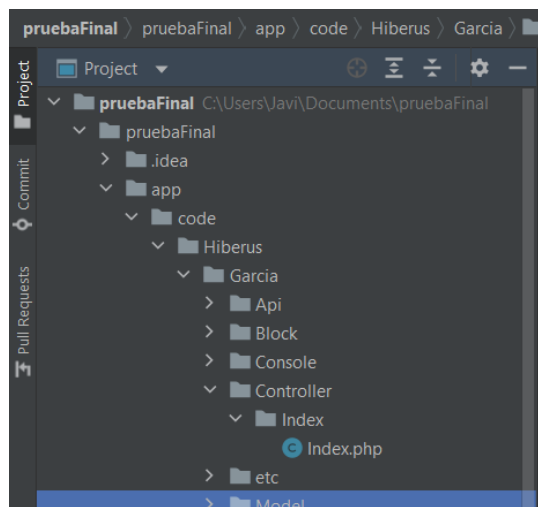


Esta clase contendrá constructores y métodos como el save, load y el delete.

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.

Tal y como nos pide en el ejercicio no podemos hacerlo ya que no lo hemos podido dar. Hemos hecho otra alternativa que nos dijo nuestro profesor Carlos, en donde mediante el Factory crearemos datos de los alumnos tanto nombres como apellidos y notas de manera aleatoria. Al no haber dado el insertar datos en base de datos directamente los he creado manualmente en base de datos.

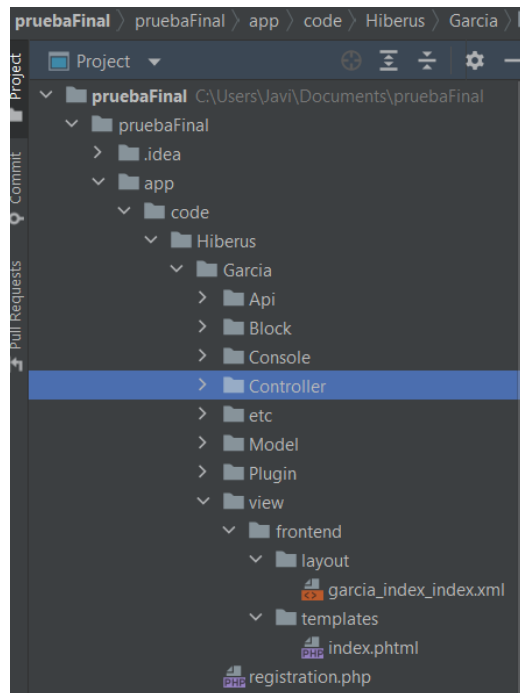
Para esto me he creado un controlador index:



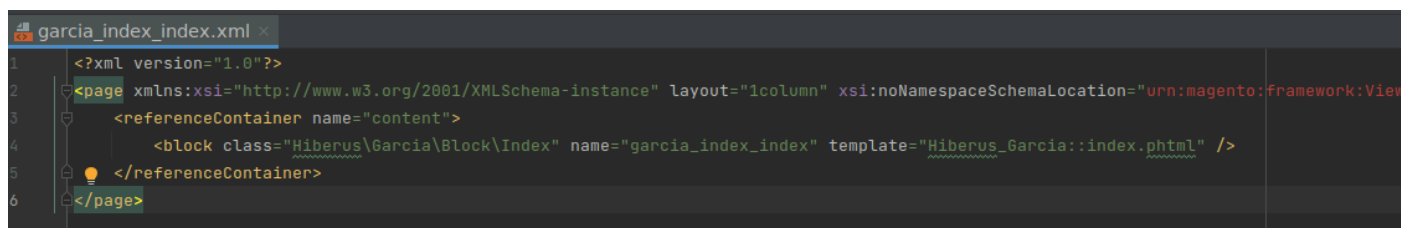
Dentro nos crearemos un constructor en donde usaremos diferentes clases.

```
1 <?php
2
3 namespace Hiberus\Garcia\Controller\Index;
4
5 use ...
6
12
13 class Index implements HttpGetActionInterface
14 {
15
16     protected \Magento\Framework\View\Result\PageFactory $pageFactory;
17     protected ExamenRepositoryInterface $examenRepository;
18     protected ExamenInterfaceFactory $examenInterfaceFactory;
19     protected Examen $examenResource;
20
21     public function __construct
22     (
23         Context $context,
24         PageFactory $pageFactory,
25         ExamenRepositoryInterface $examenRepository,
26         ExamenInterfaceFactory $examenInterfaceFactory,
27         Examen $examenResource)
28     {
29         $this->pageFactory = $pageFactory;
30         $this->examenRepository = $examenRepository;
31         $this->examenInterfaceFactory = $examenInterfaceFactory;
32         $this->examenResource = $examenResource;
33     }
34
35     public function execute()
36     {
37         return $this->pageFactory->create();
38     }
39 }
```

Además tendremos un método `execute()` en donde mediante el método `create` de `pageFactory` podremos crear nuestra vista que nos servirá para poder mostrar el resultado de todos los ejercicios. Esta vista lo crearemos en la siguiente ruta:

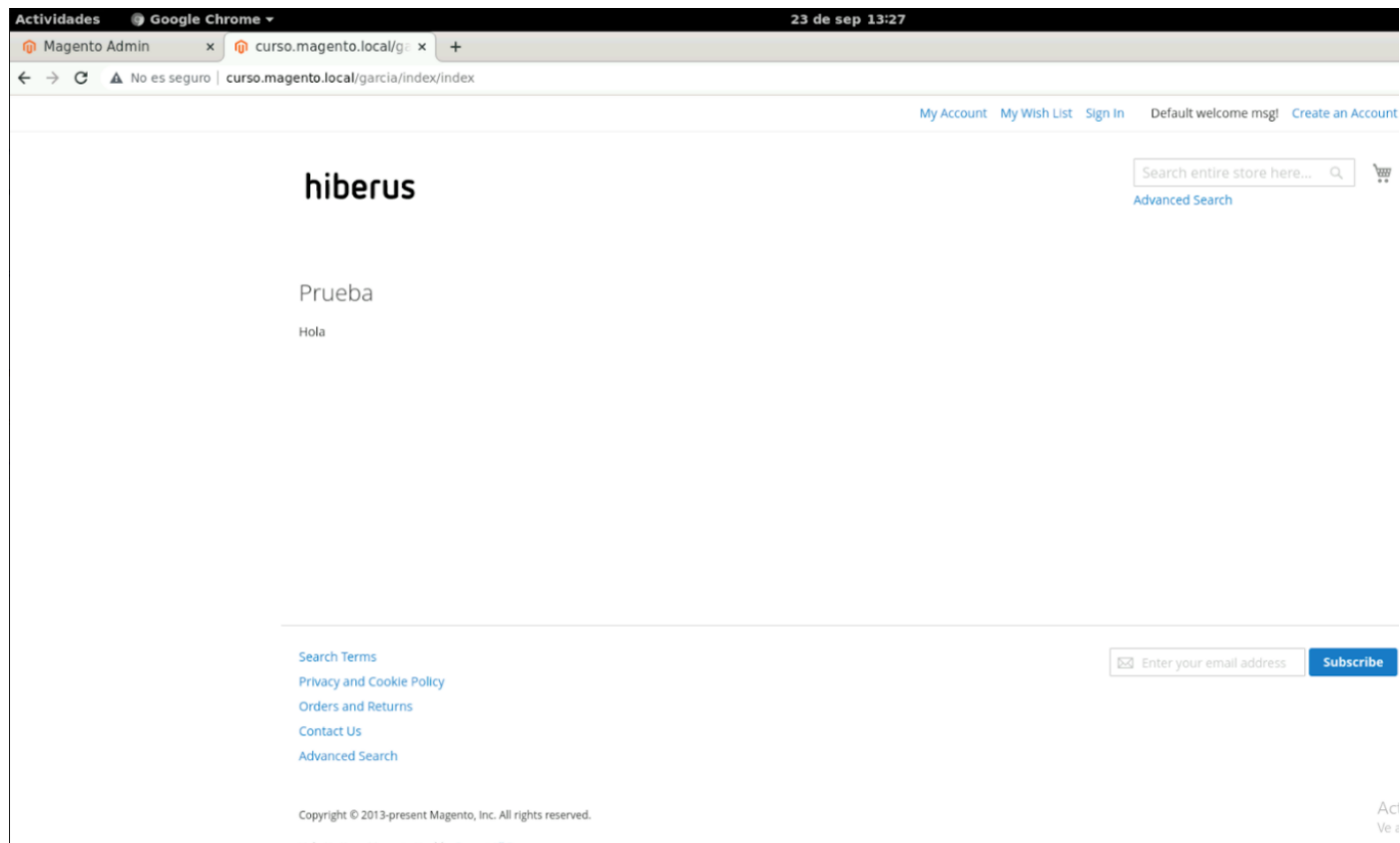


En el fichero `garcia_index_index.xml` nos permitira conectar nuestro block index con nuestra vista `index.phtml` para que podamos utilizar todos los métodos del block en vista:



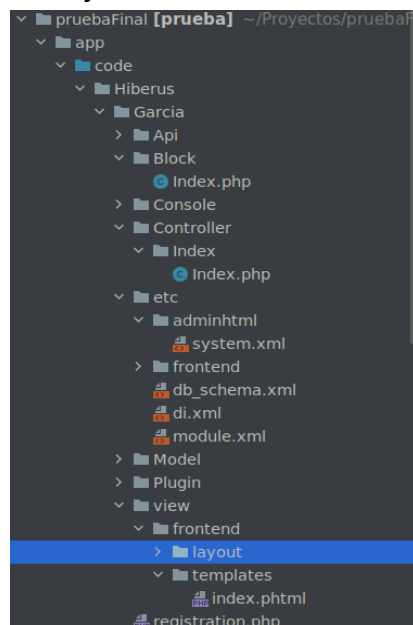
5 - Crear un nuevo controlador de frontend, el front name debe llamarse como tú apellido (ignorar tildes). Haz de momento que simplemente diga echo “Hola”, posteriormente lo modificaremos.

Para poder mostrar un hola en nuestra vista, hay dos maneras de poder hacerlo, o directamente desde vista mostrar con un `echo` un hola, o desde nuestro controlador en la función `execute`, con un `echo` 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.

Al igual que hemos hecho en el ejercicio 4, utilizaremos la misma estructura:



Realizamos todos los pasos, empezaremos mostrando un h2 con el título de nuestra vista, con una clase title :

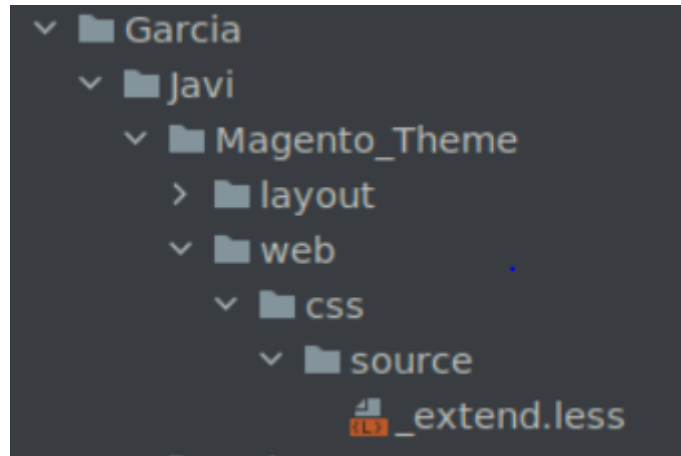
```
<h2 class="title">Prueba</h2>
```

Crearemos los respectivos li para que se puedan mostrar los datos de todos los alumnos y tambien pondremos el total de alumnos que tenemos en base de datos:

```
<?php
foreach ($alumnos as $alumno) {
    if ($i++ == $registros) break;
    ?>
    <ul>
        <li id="final">Nombre:<?= $alumno->getFirstName(); ?></li>
        <li id="final">Apellido: <?= $alumno->getLastName(); ?> </li>
        <?php
        if ($alumno->getMark() >= $notasMin) { ?>
            <li id="aprobado">Nota:<?= $alumno->getMark(); ?></li>
            <?php
        } else {
            ?>
            <li id="suspense">Nota:<?= $alumno->getMark(); ?></li>
            <?php } ?>
        </ul>
    <?php } ?>
    <p id="cantidad"> Total de alumnos: <?= (count($alumnos)) ?> </p>
```

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

Nos creamos la siguiente ruta en donde guardaremos nuestros estilos para aplicarlos en nuestra vista mediante el fichero `_extend.less`
`design\frontend\Hiberus\custom-theme\Magento_Theme\web\css\source,`



Una condición que pondrá el color del título de un color por defecto:

```
& when (@media-common = true) {  
  .title {  
    border: 1px solid black;  
    border-radius: 0.7rem;  
    color: #0a6c9f;  
    text-align: center;  
    background-color: #9b9b9b;  
    font-weight: bold;  
  }  
}
```

Y a partir de 768 píxeles ponerse de otro color:

```
.media-width(@extremum, @break) when (@extremum = 'min') and (@break = @screen__m){  
  .title {  
    border: 1px solid black;  
    border-radius: 0.7rem;  
    color: #7d2717;  
    text-align: center;  
    background-color: #9b9b9b;  
    font-weight: bold;  
  }  
}
```


Para los li que son impares le asigno un identificador en su correspondiente li, con este metodo podremos poner el margin-left a los li que son impares:

```
#final:nth-child(odd){  
    margin-left: 20px;  
}
```



Prueba		
•	Nombre: Javi	
•	Apellido: Garcia	
•	Nota: 9.00	
•	Nombre: Pablo	
•	Apellido: Lopez	
•	Nota: 8.00	
•	Nombre: Fran	
•	Apellido: Gonzalez	
•	Nota: 7.00	
•	Nombre: Angel	
•	Apellido: Victoria	
•	Nota: 6.00	
•	Nombre: Ruben	
•	Apellido: Sanchez	

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

He declarado un nuevo metodo llamado `getNotaMedia()`:

```
//Metodo para obtener la nota media de todos los alumnos  
public function getNotaMedia()  
{  
    $notas = 0;  
    $alumnos = $this->getAlumno();  
    foreach ($alumnos as $alumno) {  
        $notas += $alumno->getMark();  
    }  
    return $notasTotal = $notas / count($alumnos);  
}
```

Obtengo todas las notas de los alumnos y hago la media, devolviendolo un return. Ahora lo muestro en vista creando en una variable un bloque llamando a este método:

```
$notasTotal = $block->getNotaMedia();
```

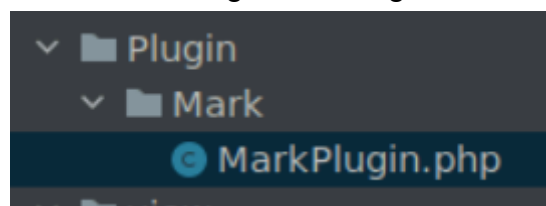
```
<p id="cantidad"> Nota Media: <?= round($notasTotal, precision: 2) ?> </p>
```

Vista de la nota media en mi vista:

•	Apellido: Dionis
•	Nota:5.00
•	Nombre:Pepe
•	Apellido: Iris
•	Nota:4.9
•	Nombre:Paco
•	Apellido: Lois
•	Nota:4.9
Total de alumnos: 8	
Nota Media: 6.35	

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

Me declaro un plugin llamado MarkPlugin, en la siguiente ruta:



```

<?php

namespace Hiberus\Garcia\Plugin\Mark;

use Hiberus\Garcia\Model\Examen;

class MarkPlugin
{

    public function afterGetMark(Examen $subject, $result){

        if($subject->getData( key: 'mark')<5){
            $subject->setMark( mark: 4.9);
            $result=$subject->getData( key: 'mark');
            //dd($result);die;
        }

        return $result;
    }

}

```

Este método recoge la nota que sea menor que 5 y le asigna el valor 4.9 despues de haber hecho el getMark() que devuelve la nota y mostrarlo en vista.

Una vez declarado el método lo registramos el plugin en nuestro fichero di.xml para que se pueda utilizar.

```

<!-- Plugin para los suspensos -->
<type name="Hiberus\Garcia\Model\Examen">
    <plugin name="examen_plugin" type="Hiberus\Garcia\Plugin\Mark\MarkPlugin" sortOrder="10" />
</type>

```

Resultado en vista:

•	Nombre:Pepe
•	Apellido: Iris
•	Nota:4.9
•	Nombre:Paco
•	Apellido: Lois
•	Nota:4.9

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)

```

<?php
if ($alumno->getMark() >= $notasMin) { ?>
    <li id="aprobado">Nota:<?= $alumno->getMark(); ?></li>
    <?php
} else {
    ?>
    <li id="suspense">Nota:<?= $alumno->getMark(); ?></li>
<?php } ?>

```

En vista mediante un condicional compruebo que las notas sean superior a la nota mínima que escojo yo, en el caso de que sea aprobado le asigno un identificador de aprobado con un color determinado y si es suspenso en otro color:

- Nombre:Alicia
- Apellido: Dionis
- Nota:5.00
- Nombre:Pepe
- Apellido: Iris
- Nota:4.9

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

Para poder mostrar las 3 mejores notas de nuestros alumnos me he creado un método en mi bloque:

```

//Metodo par obtener las tres mejores notas de todos los alumnos
public function getNotasMax()
{
    $totalAlumnos = $this->getAlumno();
    $notas = [];
    $notasMaximas = [];
    $notaMax = 0;
    foreach ($totalAlumnos as $item) {
        $notas[] = $item->getMark();
    }
    $max = max($notas);
    foreach ($notas as $nota) {
        $notaGeneral = $nota;
        if ($notaGeneral <= $max && count($notasMaximas) < 3) {
            $notaMax = $notaGeneral;
            $notasMaximas[] = $notaMax;
        }
    }
    return $notasMaximas;
}

```

Mediante este método lo que hago es me declaro una variable \$totalAlumnos en donde me guardo todos los alumnos. Me declaro un array en donde voy a guardar todas las notas de estos alumnos. Me declaro otra variable \$max en donde gracias a la función max() obtengo el la nota máxima de todos mis alumnos, recorro el array

de las notas, me declaro otro array en donde meteré todas las notas de los alumnos de ese array, voy comprobando que sea inferior a la nota maxima y que el array no supere un tamaño de 3, que sera los tres que queremos obtener. Luego simplemente lo voy añadiendo al array que hemos creado para guardar las 3 notas mejores y lo devuelvo.

Luego en vista simplemente me vuelvo a crear otra variable en donde guardo el método que hemos creado:

```
$notasAltas = $block->getNotasMax();
```

Y en vista simplemente recorro este array de 3 notas:

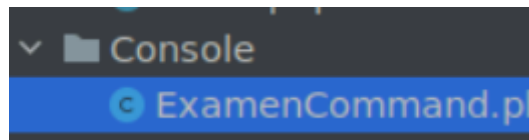
```
<h2 id="titulo">LOS MEJORES ALUMNOS</h2>
<ul>
  <?php
  foreach ($notasAltas as $notas) {
    ?>
    <li id="nota">Nota:<?= $notas ?></li>
    <?php
  }
  ?>
</ul>
```

Vista general:

LOS MEJORES ALUMNOS	
•	Nota:9.00
•	Nota:8.00
•	Nota:7.00

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. Este se debe llamar como tu apellido bajo el namespace Hiberus (hiberus:apellido).

Para este ejercicio nos crearemos una carpeta llamada Console, dentro nos crearemos una clase llamada ExamenCommand.php:



Le pasamos un bloque y las notas al constructor:

```
public function __construct(
    \Hiberus\Garcia\Model\Examen $examens, Index $block
)
{
    $this->examens = $examens;
    $this->block = $block;
    parent::__construct(/*$name*/);
}
```

En el método configure() le asignamos el nombre al comando que vamos a crear y la descripción de lo que hará nuestro comando:

```
protected function configure()
{
    $this->setName( name: 'hiberus:garcia')
    ->setDescription( description: 'Mostrar Tabla de Exámenes')/*->addOption(
        self::NAME,
        null,
        InputOption::VALUE_OPTIONAL,
        'Name'
    )*/
    ;

    parent::configure();
}
```

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

Por último en el método execute, me declaro una variable alumnos en donde le asigno el método que creamos anteriormente , y muestro todos sus datos.

```
protected function execute(
    InputInterface $input,
    OutputInterface $output
)
{
    /* if ($input->getOption(self::NAME)) {
        $name = $input->getOption(self::NAME);
    } else {
        $name = $this->author->getAuthorName();
    } */
    $alumnos = $this->block->getAlumno();
    foreach ($alumnos as $alumno) {
        $this->examens->setIdExam($alumno->getIdExam());
        $this->examens->setFirstName($alumno->getFirstName());
        $this->examens->setLastName($alumno->getLastName());
        $this->examens->setMark($alumno->getMark());
        $output->writeln( messages: '<info> ID: ' . $this->examens->getIdExam() . ' | Nombre: ' . $this->examens->getFirstName() .
            ' | Nota: ' . $this->examens->getMark() . '</info>');
    }
}
```

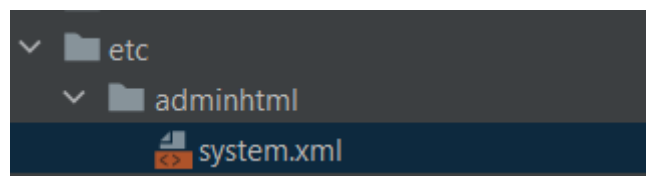
Metemos el comando que nos hemos creado en el terminal:

```
pue@pue-KVM:~/Proyectos/pruebaFinal$ dockergento magento hiberus:garcia
ID: 1 | Nombre: Javi | Apellidos: Garcia | Nota: 9.00
ID: 2 | Nombre: Pablo | Apellidos: Lopez | Nota: 8.00
ID: 3 | Nombre: Fran | Apellidos: Gonzalez | Nota: 7.00
ID: 4 | Nombre: Angel | Apellidos: Victoria | Nota: 6.00
ID: 5 | Nombre: Ruben | Apellidos: Sanchez | Nota: 6.00
ID: 6 | Nombre: Alicia | Apellidos: Dionis | Nota: 5.00
ID: 7 | Nombre: Pepe | Apellidos: Iris | Nota: 4.9
```

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)

Lo primero que vamos a hacer es crearnos la carpeta adminhtml y dentro nuestro fichero system.xml, en donde crearemos el panel para el administrador:



```

1 <?xml version="1.0"?>
2 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Config:etc/system_file.xsd">
4     <system>
5         <tab id="hiberus" sortOrder="999" translate="label">
6             <label>Hiberus</label>
7         </tab>
8         <section id="hiberus_nombre" showInDefault="1" showInStore="1" showInWebsite="1" sortOrder="10">
9             <label>Exámenes García</label>
10            <tab>hiberus</tab>
11            <resource>Hiberus_Garcia::config</resource>
12            <group id="general" showInDefault="1" sortOrder="10">
13                <label>General</label>
14                <field id="registros" type="text" sortOrder="10" showInDefault="1">
15                    <label>Numero de registros:</label>
16                    <validate>integer</validate>
17                </field>
18
19                <field id="notasMin" type="text" sortOrder="10" showInDefault="1">
20                    <label>La nota minima para probar es:</label>
21                    <validate>validate-number</validate>
22                </field>
23            </group>
24        </section>
25    </system>
26 </config>

```

El resultado lo vemos en el panel del administrador, en stores, configuration:

Scope: Default Config ?

GENERAL	▼	<p>General</p> <p>Numero de registros: <input type="text"/></p> <p>La nota minima para probar es: <input type="text"/></p>
CATALOG	▼	
SECURITY	▼	
CUSTOMERS	▼	
SALES	▼	
YOTPO	▼	
DOTDIGITAL	▼	
HIBERUS	▲	

Exámenes García

Una vez hecho, en nuestro bloque, nos creamos estos dos métodos mediante el uso del ScopeConfig, que nos permite traernos los datos desde el panel del administrador:


```

public function getRegistros()
{
    $data = $this->scopeConfig->getValue('hiberus_nombre/general/registros', ScopeInterface::SCOPE_STORE);
    $alumnos = $this->getAlumno();
    return $data ? count($alumnos);
}

public function getNotaMin()
{
    $data = $this->scopeConfig->getValue('hiberus_nombre/general/notaMin', ScopeInterface::SCOPE_STORE);
    return $data ? 5;
}

```

Una vez hecho esto, en nuestra vista, creamos nuestras variables que recojan estos dos métodos:

```

$registros= $block->getRegistros();
$notasMin= $block->getNotaMin();

```

Aprovechando el código del anterior ejercicio compruebo con la variable \$notasMin que el resultado a mostrar solo muestre aquellas notas que le hemos indicado:

```

<?php
if ($alumno->getMark() >= $notasMin) { ?>
    <li id="aprobado">Nota:<?= $alumno->getMark(); ?></li>
    <?php
} else {
    ?>
    <li id="suspense">Nota:<?= $alumno->getMark(); ?></li>
    <?php } ?>

```

Resultado en vista:

✓ You saved the configuration.

GENERAL	▼	General
CATALOG	▼	
SECURITY	▼	
CUSTOMERS	▼	

Numero de registros:

[global]

2

La nota minima para probar es:

[global]

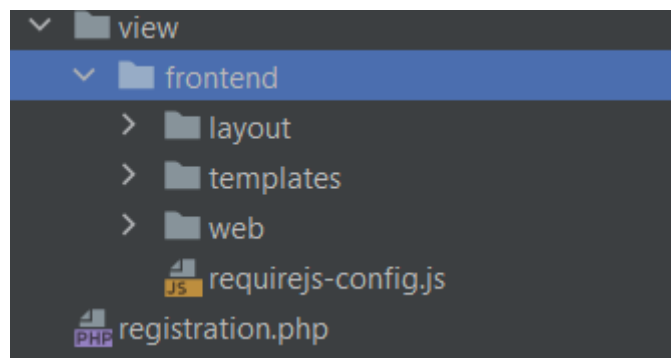
6

Prueba	
•	Nombre: Javi
•	Apellido: Garcia
•	Nota: 9.00
•	Nombre: Pablo
•	Apellido: Lopez
•	Nota: 8.00
Total de alumnos: 8	
Nota Media: 6.35	
LOS MEJORES ALUMNOS	
•	Nota: 9.00
•	Nota: 8.00
•	Nota: 7.00

Ejercicios que he podido hacer a pesar de no haberlo dado, 7 y 9:

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

Para este ejercicio nos crearemos la siguiente ruta:



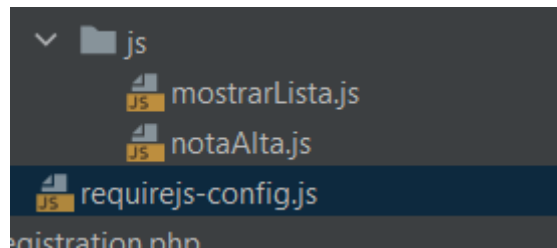
Nos crearemos un fichero requirejs-config.js:

```

requirejs-config.js
1  var config = {
2    map: {
3      '*': {
4        mostrarLista: 'Hiberus_Garcia/js/mostrarLista',
5        notaAlta: 'Hiberus_Garcia/js/notaAlta',
6      }
7    }
8  };
  
```

En este fichero incluimos los ficheros javascript que vamos a utilizar.

Posteriormente nos creamos ambos ficheros de javascript:



```
1 define(['jquery'], function($) {
2     "use strict";
3     return function mostrarLista(btn, div)
4     {
5         $(btn).click(function() {
6             $(div).slideToggle();
7         });
8     }
9 });
```

En el fichero mostrarLista, mostraremos los alumnos que tenemos y a su vez lo ocultaremos.

```
1 define(['jquery'], function($) {
2     "use strict";
3     return function alertNotaAlta(btn)
4     {
5         $(btn).click(function() {
6             alert('La nota mayor de la clase es: ' + maxNote);
7         });
8     }
9 });
```

Y en este fichero crearemos un alert() en donde mostraremos la nota más alta de todos nuestros alumnos.

Nos creamos en nuestro bloque index un metodo en donde obtendremos la nota mas alta de todos nuestros alumnos:

```

public function getNotaMasAlta() {
    $alumnos = $this->examenInterfaceFactory->create()->getCollection()->getData();
    $notaMaxima = 0;
    foreach ($alumnos as $alumno) {
        if ($alumno['mark'] > $notaMaxima) {
            $notaMaxima = $alumno['mark'];
        }
    }
    return $notaMaxima;
}

```

Una vez creado nuestro metodo, nos vamos a nuestra vista:

```

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

<script>
    require(['jquery', 'mostrarLista'], function($, mostrarLista) {
        mostrarLista('#mostrar', '#lista');
    });

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

```

Los botones:

```

<button id="mostrar">Mostrar lista</button>
<button id="notaAlta">Nota mas alta</button>

```

Resultado en vista:

Prueba

Mostrar lista Nota mas alta

- Nombre:Javi
- Apellido: Garcia
- Nota:9.00
- Nombre:Pablo
- Apellido: Lopez
- Nota:8.00

Total de alumnos: 8

Nota Media: 6.35

LOS MEJORES ALUMNOS

- Nota:9.00

Prueba

Mostrar lista Nota mas alta

Total de alumnos: 8

Nota Media: 6.35

LOS MEJORES ALUMNOS

- Nota:9.00
- Nota:8.00
- Nota:7.00

NOTAS

curso.magento.local dice
La nota mayor de la clase es: 9

Aceptar

[Advanced Search](#)

Mostrar lista Nota mas alta

Total de alumnos: 8

Nota Media: 6.35

