

PRUEBA TÉCNICA DE MAGENTO 2

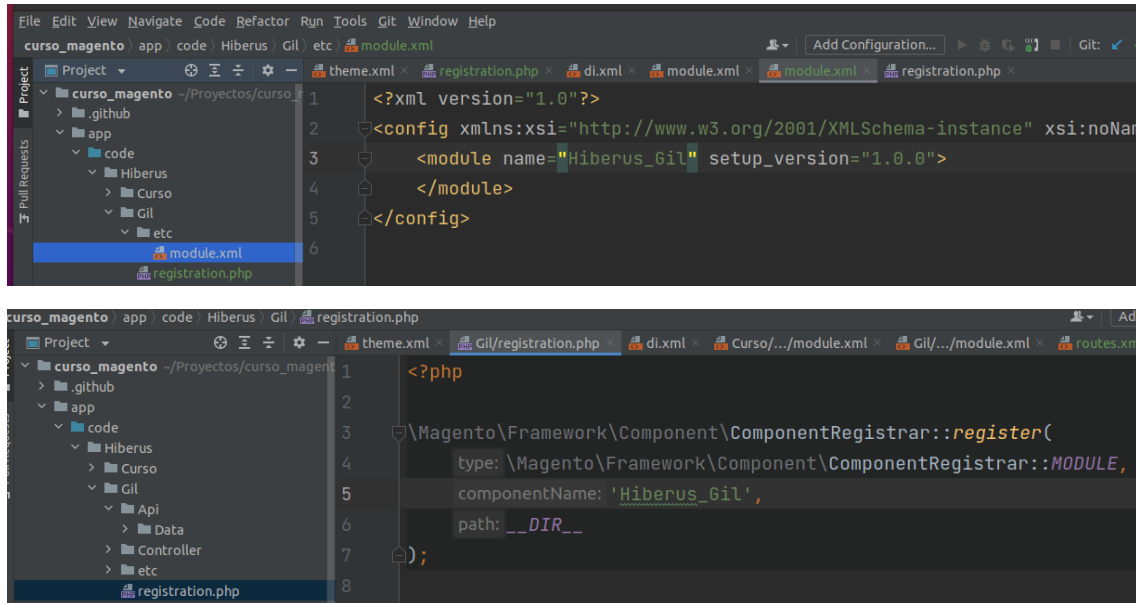
WILSON GIL LLANO

HIBERUS 2021

1 Crear un nuevo módulo

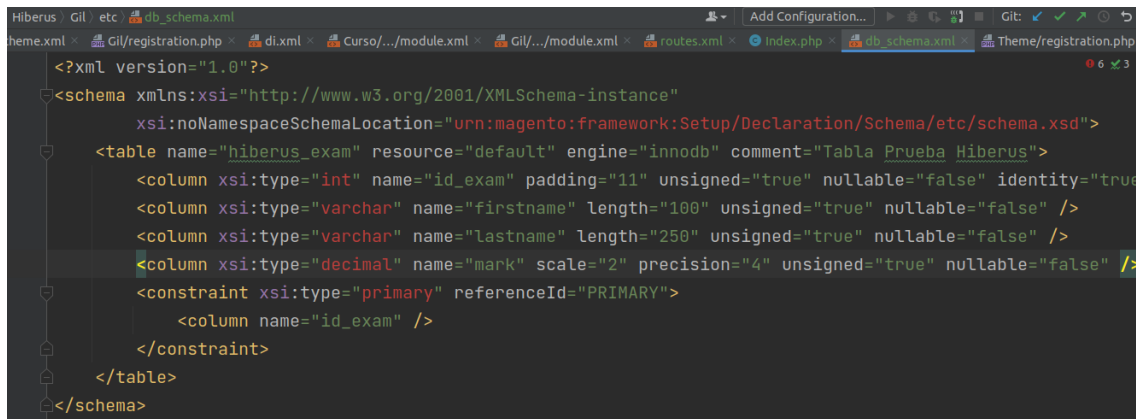
cuyo nombre sea tu apellido (sin tildes) y el vendor sea Hiberus, por ejemplo: Hiberus_Garcia.

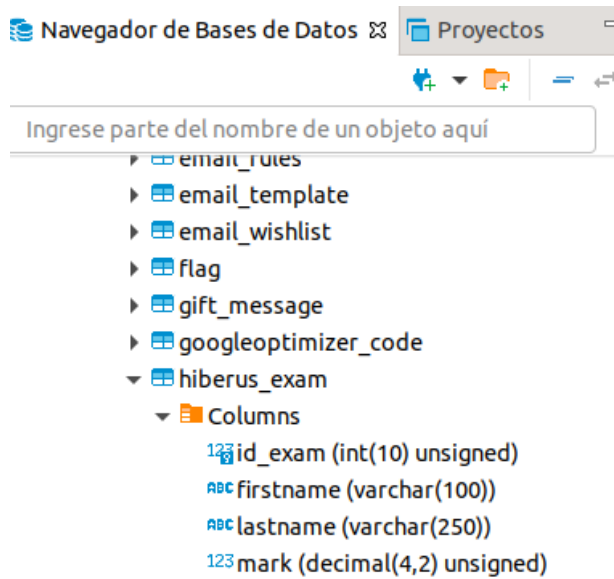
Se crea la carpeta el modulo Gil con el archivo .xml y el archivo registration.php



2 Crear una única tabla llamada hiberus_exam

Creación de la tabla





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

Se crea los dataEntitys son los que contiene la estructura de datos, contiene los gett y sett

Creación de la interface `GilInterface.php`

```
<?php

namespace Hiberus\Gil\Api\Data;

use Magento\Tests\NamingConvention\true\float;

interface GilInterface extends \Magento\Framework\Api\ExtensibleDataInterface
{

    public const TABLE_NAME = 'Hiberus_Gil';
    public const COLUMN_ID = 'id_exam';

    /**
     * @return int
     */
    public function getIdExam();

    /**
     * @param int $idExam
     * @return $this
     */
    public function setIdExam($idExam);

    /**
     * @return string
     */
}
```

```

public function getFirstName();

/**
 * @param string $name
 * @return $this
 */
public function setFirstName($name);

/**
 * @return string
 */
public function getLastName();

/**
 * @param string $lastname
 * @return $this
 */
public function setLastName($lastname);

/**
 * @return int
 */
public function getMark();

/**
 * @param float $mark
 * @return $this
 */
public function setMark($mark);

}

```

Creación de GilRepositoryInterface

```
<?php
```

```
namespace Hiberus\Gil\Api;
```

```
interface GilRepositoryInterface
```

```

{
    /**
     * @param Data\GilInterface $gilInterface
     * @return mixed
     */
    public function save(\Hiberus\Gil\Api\Data\GilInterface $gilInterface);

    /**
     * @param $idExam

```

```

* @return \Hiberus\Gil\Api\Data\GilInterface
*/
public function getById($idExam);

/**
* @param Data\GilInterface $gilInterface
* @return boolean
*/
public function delete(\Hiberus\Gil\Api\Data\GilInterface $gilInterface);

/**
* @param $idExam
* @return boolean
*/
public function deleteById($idExam);
}

```

Creación de la entidad Gil.php en el modelo

Los modelos contendrán lo que tiene que ver con la lógica del negocio, tendrá los métodos que realizará todas las acciones sobre la base de datos y cada modelo se conecta a la base de datos a través de un ResourceModel cuando necesitemos que persistan los datos

```

<?php

namespace Hiberus\Gil\Model;
use Hiberus\Gil\Api\Data\GilInterface;

class Gil extends \Magento\Framework\Model\AbstractModel implements
\Hiberus\Gil\Api\Data\GilInterface
{
    protected function _construct()
    {
        $this->_init(\Hiberus\Gil\Model\ResourceModel\Gil::class);
    }

    public function getIdExam()
    {
        return $this->getData('id_exam');
    }

    public function setIdExam($idExam)
    {
        return $this->setData('id_exam',$idExam);
    }

    public function getFirstName()
    {
        return $this->getData('first_name');
    }
}

```

```

    }

    public function setFirstName($name)
    {
        return $this->setData('first_name',$name);
    }

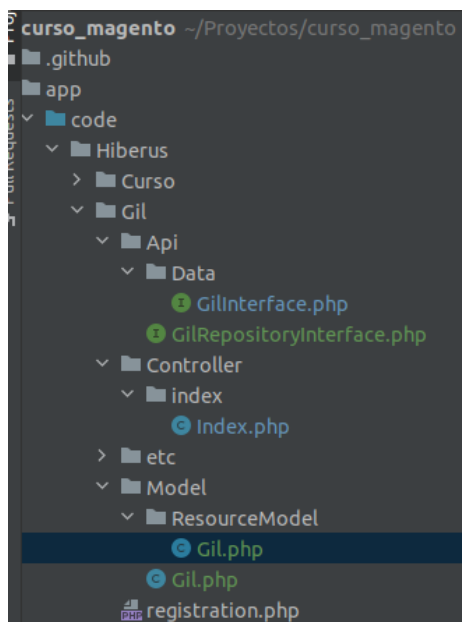
    public function getLastName()
    {
        return $this->getData('last_name');
    }

    public function setLastName($lastname)
    {
        return $this->setData('last_name',$lastname);
    }

    public function getMark()
    {
        return $this->getData('mark');
    }

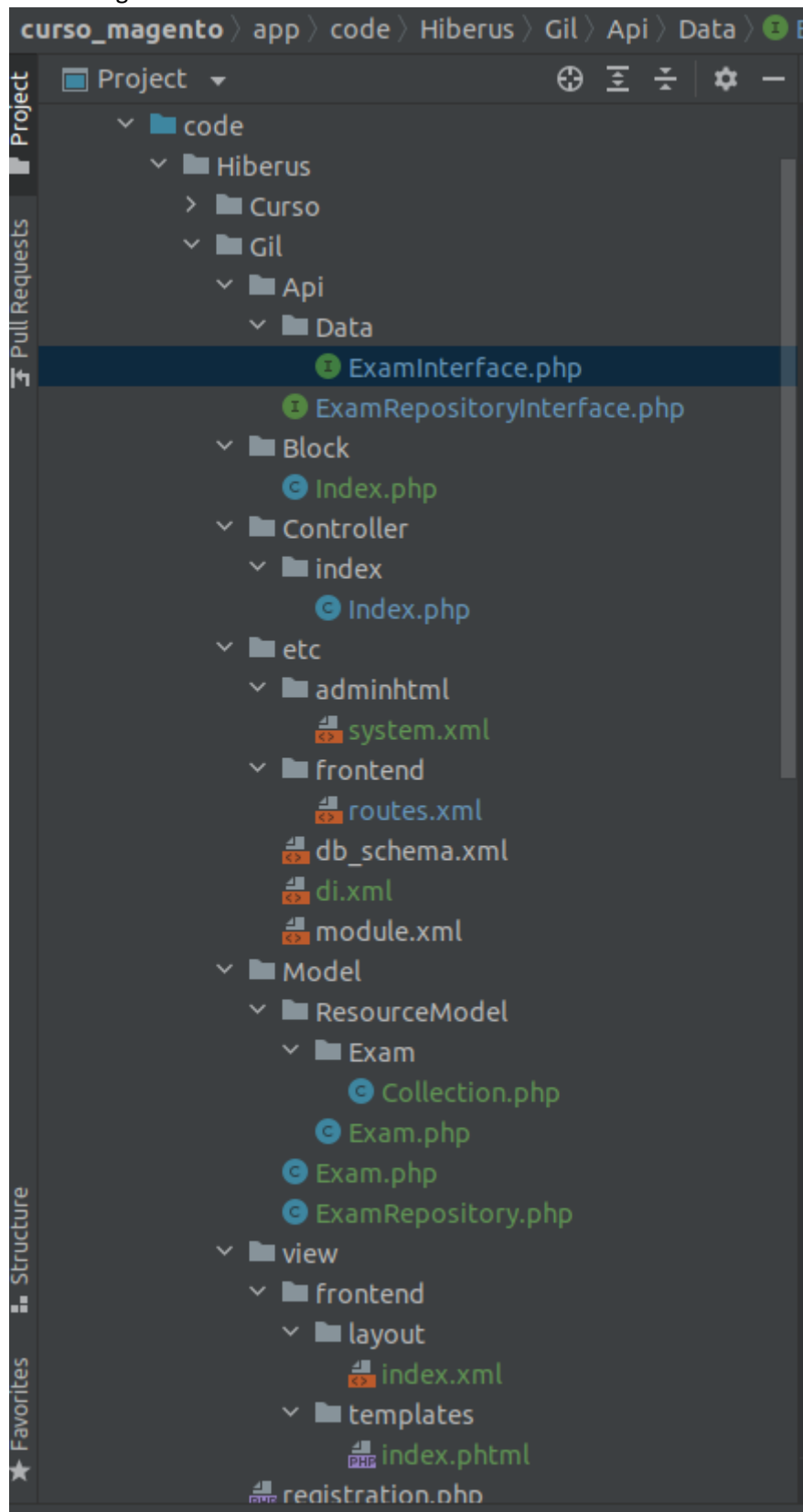
    public function setMark($mark)
    {
        return $this->setData('mark',$mark);
    }
}

```



4 Crear un Setup (Db Schema y Data Patch)

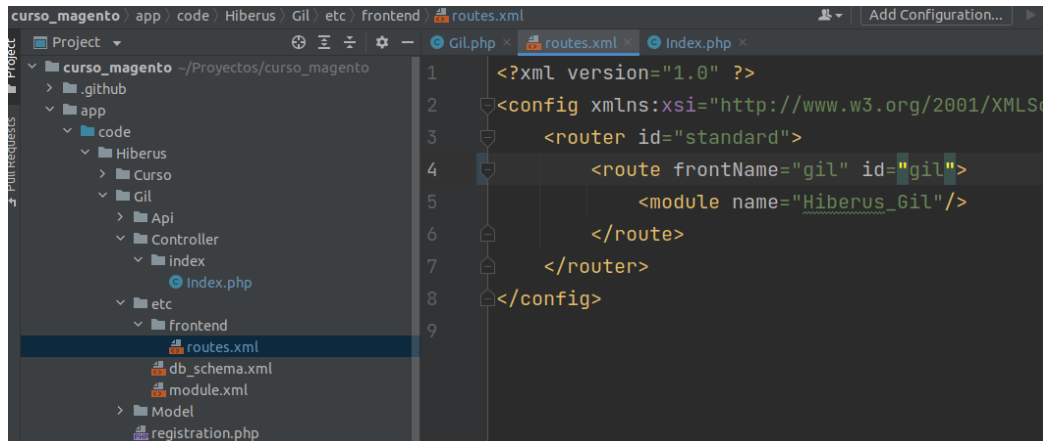
para introducir datos que introduzca en la tabla creada utilizando los service contracts. Se crea la siguiente estructura de carpetas y ficheros para interactuar con la base de datos desde magento.



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.

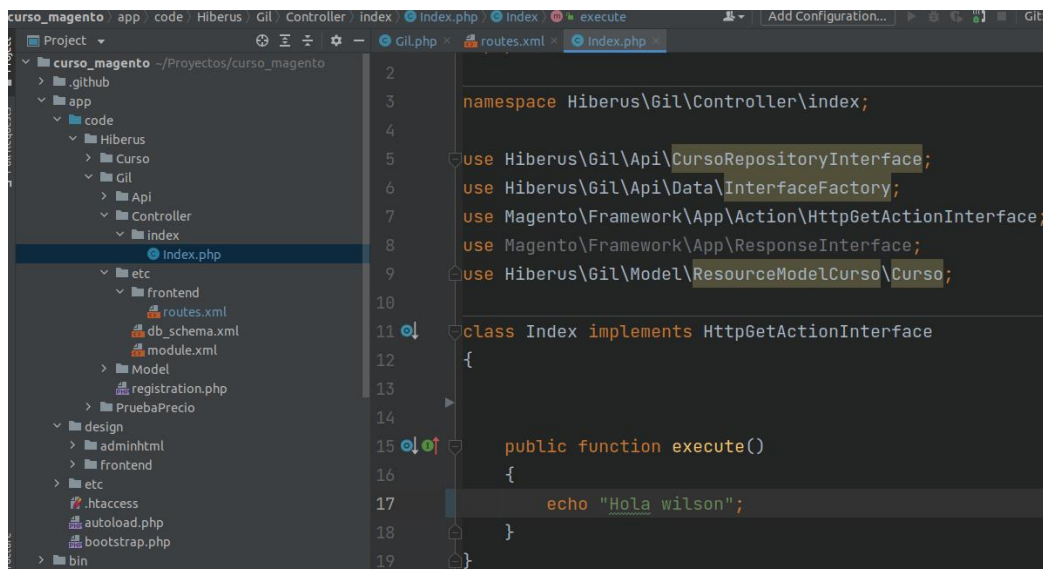
Se configura el archivo routes.xml



The screenshot shows an IDE with the project structure on the left and the routes.xml file open in the editor. The project structure includes folders like .github, app, code, Hiberus, Curso, Gil, Api, Controller, index, etc, frontend, db_schema.xml, module.xml, and Model. The routes.xml file contains the following XML code:

```
<?xml version="1.0" ?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="config.xsd">
    <router id="standard">
        <route frontName="gil" id="gil">
            <module name="Hiberus_Gil"/>
        </route>
    </router>
</config>
```

Configuración del archivo controlador index.php

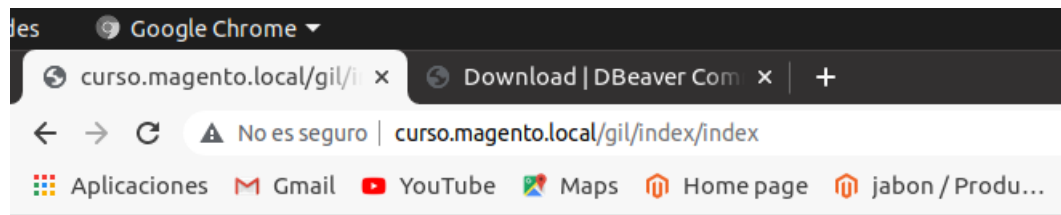


The screenshot shows the same IDE with the project structure on the left and the index.php file open in the editor. The index.php file contains the following PHP code:

```
namespace Hiberus\Gil\Controller\index;

use Hiberus\Gil\Api\CursoRepositoryInterface;
use Hiberus\Gil\Api\Data\InterfaceFactory;
use Magento\Framework\App\Action\HttpGetActionInterface;
use Magento\Framework\App\ResponseInterface;
use Hiberus\Gil\Model\ResourceModelCurso\Curso;

class Index implements HttpGetActionInterface
{
    public function execute()
    {
        echo "Hola wilson";
    }
}
```

```
Hola wilson! exception(s):
Exception #0 (InvalidArgumentException): Invalid return type

Exception #0 (InvalidArgumentException): Invalid return type
<pre>#1 Magento\Framework\App\Http\Interceptor->launch() called at [vendor/magento/framework/Controller/Action/Interceptor.php:40]
#2 Magento\Framework\App\Bootstrap->run() called at [pub/index.php:40]
</pre>
```

https://github.com/wilson347/PruebaMagento2_Wilson_Gil/tree/master