Módulo del proyecto: app/code/Hiberus/HiberusRebollar

1- He creado el módulo dentro del Vendor "**Hiberus**"

2- Dentro del módulo he creado sus primeros ficheros necesarios, **registration**.**php** y **module**.**xml** dentro de la carpeta "**etc**".

3- Una vez creados los ficheros necesarios para el módulo he creado el fichero **db_chema**.**xml** dentro de "**etc**" para definir la tabla de la base de datos. Una vez definido he creado el modelo dentro de la carpeta "model" y la interface que importa el modelo dentro de "**Api**/**Data**".

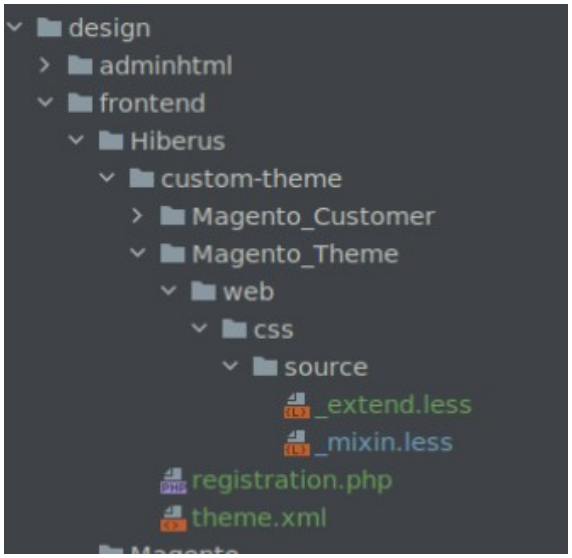
Dentro de la interface creada en el directorio "**Data**" definimos los métodos que utilizaremos en el modelo para poder pedir e insertar los datos. En el directorio "**Api**" definimos la interface con los métodos para guardar,borrar y consultar datos, esta interface se implementará en el modelo **ExamenRepository**.php
En el directorio "**etc**" he añadido el fichero **di**.**xml** con la estructura para referenciar las interfaces y modelos anteriores y así poder utilizar los datos de la tabla creada anteriormente.

4 - Los datos los he introducido mediante inserts directamente en la tabla de la base de datos, las interfaces y modelos definidos anteriormente las utilizo para consultar y mostrar los datos.

5 y 6- Dentro del directorio "**view**" he creado la carpeta "**frontend**" que contiene los directorios "**layout**" "**templates**". Dentro de "templates" defino el fichero index.phtml el cual importa la clase index que se encuentra dentro del directorio "**Block**", esta clase contiene un método que nos devuelve un collection con la informacion de los alumnos. Para obtener este collection he creado una carpeta "**Examen**" dentro de **ResourceModel** la cual contiene una clase Collection con un constructor en el que definimos el modelo en el _init. Esta colección la recorremos con un foreach para recoger los alumnos y poder llamar a sus métodos y así pintar el nombre, apellido y nota de cada uno.
Dentro del directorio "layout" se crea el fichero xml que define el contenedor en el que vamos mostramos los datos, en este caso es el fichero **index** que tenemos en "**templates**".

7- Para ocultar las notas he creado un botón en el index con un id para referenciarlo en la función de javascript. He creado 2 **ficheros js**, uno dentro de frontend llamado **requirejs-config**.**js** el cual contiene la ruta de la función que estará alojada dentro de **web/js/notas.js** encargada de la funcionalidad para ocultar la etiqueta del div que contiene las notas. En la etiqueta <**script**> del index hago un require de la función y la llamo dentro de un onclick que recoge el id del boton que se pulsa.

8-Los estilos los he definido dentro de los ficheros **_extend.less** y **_mixin**.**less** en la siguiente ruta:



Con el tema personalizado asignado en el adminstrador el cual extiende del tema **blank** de Magento. El título tiene una clase llamada "**title**" y para poder cambiar el color en función de la resolución le he dado un color por defecto dentro del fichero**_extend** y después le indico que se cambie con la resolución:

```
.media-width(@extremum, @break) when (@extremum = 'min') and (@break = @screen__m) {
    .title{
        color: #eb5202;

    }
}
```

Los alumnos impares tienen un margen definido utilizando **nth-child(even)** donde even hace referencia a los impares. El margen se coge desde la variable declarada en la parte superior.

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

9- El alert lo he creado con *jQuery widget "alert ". Defino la función en el script dentro la vista index y creo un botón que llamará esta f unción. También he creado una etiqueta con la clase modal que se abrirá al pulsar el botón.*

10- Para sacar la media de las notas he creado un acumulador dentro del propio **index**.**phtml** que muestra los datos de los alumnos para que vaya sumando todas las notas. Una vez finalizado el foreach se dividen los datos de esa variable por un una funcion count que tiene como parámetro el collection de alumnos.

11. Para definir el plugin y darle funcionalidad se crea dentro del módulo una carpeta "**Plugin**" la cual contiene otra carpeta que he llamado "**Mark**" que a su vez contiene el fichero php que va ejecutar la función para cambiar la nota. En ese fichero he definido una función llamada afterGetMark que hace referencia al método **GetMark** y recibe como parámetro la clase Examen y el resultado de ese método. Aquí dentro le doy la funcionalidad cambiando la nota a los que tengan menos de un 5 utilizando el $result.
Por último dentro del fichero **di.xml** le añado con la etiqueta <type> con la referencia al modelo donde se encuentra el método voy a modificar y dentro la etiqueta <plugin> la referencia al plugin creado.

12- Los alumnos aprobados o suspensos tienen una clase "suspenso" o "aprobado" en la vista dependiendo de si su nota es menor o mayor a 5. Una vez definida esta funcionalidad indico en el fichero **_mixin** el color que debe tener cada uno de ellos. Para utilizar este fichero en el **_exend.less** importo el fichero **_mixin** en la parte superior: **@import '_mixin.less';**

```
.aprobado{
    color: green;

}

.suspenso{
    color: red;

}
```

13- Para poder agrupar las 3 mejores notas he creado una función dentro de **Block/Index.php** llamada **getMejoresNotas**() la cual devuelve un array con las notas de los alumnos teniendo las mismas ordenadas de mayor a menor. Este array lo recibo en la vista y recorro los 3 primeros valores para comparar cada uno con la nota de los alumnos que estoy pintando, si la nota.

14- Para poder listar las notas de los alumnos mediante command he creado una clase **ExamenCommand**.**php** dentro de la carpeta **Console** en el módulo. Desde el constructor se define la clase que contiene los métodos de los alumnos y el bloque donde está el método que devuelve la colección, en este caso "**Examen**.**php**".Se define el nombre del comando en el config y se recorren los alumnos llamando al método del block **getNotas().** Le pasamos a cada alumno los datos recogidos de **getNotas()** y recogemos su valor para mostrarlo en el writeln del comando.

16- Para poder enviar datos desde el administrador he creado el directorio adminhtml dentro de etc el cual contiene el fichero **system**.**xml** con las etiquetas que definen los campos que se muestran en el admin. Lo siguiente ha sido añadir al **di**.**xml** el bloque que referencia al tema sobre el que quiero aplicar la modificación. Para poder procesar los datos enviados desde el admin de magento se crean 2 métodos dentro del **Index**.**php** alojado en "**Block**" uno para recibir el valor del primer campo y otro para el valor del segundo. Dentro de la vista donde se pintan los alumnos y sus notas llamo a estos 2 métodos y guardo el valor que tienen configurado para definir mediante condiciones cuantos alumnos mostrar y la nota necesaria para aprobar.