EXAMEN MAGENTO

1.Crear un módulo.

Para crear en magento un módulo lo que hace falta son 2 ficheros.

El primero de ellos va dentro de la carpeta etc llamado module.xml que incluye una cabecera y se define el tag module que de nombre tendrá el Vendor-name_module-name que en nuestro ejercicio es Hiberus_Lazaro. Al no depender de ningún otro modulo con eso sería suficiente, en caso de dependencia utilizaríamos la etiqueta sequence.

El segundo de ellos se llama registration.php (Que se encuentra dentro de nuestro modulo) este fichero solo registra nuestro modulo y lo incluye en el saco con los demás módulos. Incluye el mismo nombre (Hiberus_Lazaro), el tipo que es (en nuestro caso es MODULE) y además tiene la variable DIR que indica el directorio donde está este fichero.

2. 3. Crear Tabla ,service contracts y ORM

Dentro de la carpeta etc creamos el fichero db_schema.xml que sirve para la estructura de nuestro modelo de datos, para este ejercicio nos dan una serie de datos concretos que tiene que tener la tabla.

Definimos el modelo de datos y repositorio.

Dentro de la carpeta Api/Data creamos la interfaz del modelo AlumnoInterface.php, donde definimos las constantes que corresponden con las columnas de nuestra tabla y las definiciones de las funciones (get y set).

Dentro de la carpeta Api creamos la interfaz del repositorio AlumnoRepositoryInterface.php, donde definimos las operaciones contra la base de datos.

Dentro de la carpeta Model creamos nuestro modelo Alumno.php donde implementa cada una de las funciones de la interfaz (AlumnoInterface) y con el uso de _getData significa que si la clase ya tiene ese dato devuélvemelo.

En esta carpeta también se encuentra el modelo del repository que es AlumnoRepository.php donde se encuentran todas las funciones y como dato destacable en nuestra función getList() incluye el searchCriteriaInterface que son criterios de búsqueda que incluye magento.

Dentro de la carpeta Model/ResorceModel creamos Alumno.php donde se hace la comunicación con la base de datos y definimos la tabla a donde tiene que ir a buscar.

Dentro de la carpeta Model/ResorceModel/Alumno creamos Collection.php que sirve para enlazar todas las clases.

Por último, dentro de la carpeta etc creamos di.xml, con la etiqueta preference índica que clase concreta implementa esta interfaz, con la etiqueta virtual te crea clases ficticias y type te cambia los tipos.

4. Crear un Setup

Dentro de la carpeta Setup/Patch/Data creamos el fichero PopulateDataModel.php que sirve para rellenar nuestros valores en la tabla del modelo de datos.

En nuestro caso hicimos el extra, pero no lo pude comprobar ya que no me dejaba la opción de borrar el parche que tenía anteriormente. En el extra hay que crear el fichero import.csv dentro de Setup/Data que tendrá el nombre y apellido.

Tendrás que añadir su use, ponerlo en el constructor y a la hora de recoger los datos deberás añadirlo en getdata y ya los podrás recoger.

Una vez cogidos haces, los sets de las funciones.

5.Crear un nuevo controlador

Lo primero que hacemos es crear el fichero de rutas para que sea accesible desde el navegador. Dentro de la carpeta ect/frontend

Nuestra url será http:/curso.magento.local/<frontname>/<controller>/<action>

En el fichero de routes.xml declaramos el frontname que este caso será nuestro apellido lazaro y el module (Como en casos anteriores Hiberus_Lazaro)

Para completar la url creamos nuestro controlador dentro de la carpeta Controller/Lazaro que será la acción de nuestra url le hemos puesto de nombre index.php.

Por último, quedar el fichero asociado a la ruta(id_controller_action) por lo tanto será lazaro_lazaro_index.xml que se encuentra dentro de la carpeta view/frontend/layout.

Por tanto, nuestra url completa será:

Nuestra url será http:/curso.magento.local/lazaro/lazaro/index

Para crear un bloque:

Para ver que funciona printearemos un hola en el bloque, creamos un fichero llamado examenbloque.phtml dentro de la carpeta view/frontend/templates

Por lo tanto en el fichero lazaro_lazaro_index.xml asociaremos el template template="**Hiberus_Lazaro::examenbloque.phtm**

6.Listado de los alumnos

En este caso nos dicen de crear un bloque con datos de la base de datos por lo tanto tenemos que crear un fichero llamado BloqueExamen.php que se encuentra dentro de la carpeta Block que extiende de la clase Template y aquí cogemos nuestro AlumnoRepositoryInterface donde nos creamos una función llamada getListExam() donde nos muestra a través de la función getList() todos los datos de nuestra tabla.

Ahora nos vamos al fichero lazaro_lazaro_index.xml y aquí debemos meter la clase del bloque: class="Hiberus\Lazaro\Block\BloqueExamen"

Y en nuestro fichero examenbloque.phtml como nos piden pintar nombre, apellido, nota deberemos usar las funciones correctas guardadas en nuestro repository.

Añadir la traducción:

En el fichero examenbloque.phtml creamos al final una línea.

Magento permite almacenar traducciones en ficheros CSV con el nombre <idioma>_<PAIS>

Para poder traducirlo dentro de la carpeta i18n/es_ES.csv cogemos "Linea sin traducir","linea traducida".

7. Asociar un js con un botón dónde desaparezca las notas

Aquí vamos a trabajar con 2 ficheros, nuestro template que se llama examenbloque.html y nuestro js llamado notas.js que lo creamos dentro de la carpeta view/frontend/js donde hemos elegido hacerlo con notación declarativa.

El primer paso es crear en nuestro template un botón con una clase llamada inputbutton.

Como queremos que desaparezcan las notas tenemos que crear una clase donde cogemos las notas que será en nuestro getMark().

Con esta parte del código se creará una instancia para el botón.

```
<script type="text/x-magento-init">{
 ".inputbutton": {
 "Hiberus_Lazaro/js/nota": { } } }
</script>
```

En nuestro fichero nota.js lo que nos piden es que al hacer click desaparezcan y aparezcan las notas por lo tanto con el evento toggle() nos hace ese efecto y como

solo queremos que afecte a las notas, llamamos a la clase que hemos nombrado en nuestro template que es “notas”.

$(element).click(function(){

$(".notas").toggle();});

8.Maquetar el listado

Lo primero que nos piden es que el titulo tenga un color y a partir de 768 pixeles cambie a otro.

Esto es posible gracias a un fichero llamado _module.less que se encuentra dentro de la carpeta view/frontend/web/css/source.

Aquí hay que saber que etiquetas tienen nuestro template. En el enunciado nos decían que nuestro título tenía la etiqueta h2 con una clase llamada title por la tanto en nuestro _module.less para acceder a ese título bastará con poner h2.title.

Para que a partir de 768 pixeles cambie utilizamos esta expresión:

.media-width(@extremum, @break) when (@extremum = 'min') and (@break = @screen__m) {

Donde @screen__m es a partir de 768 pixeles y para el resto & when (@media-common = true)

Para el fondo es suficiente con poner background-color:(color que quieras) dentro de h2.title.

Otro punto nos decía que los impares tengan un margen izquierdo de 20px, aquí hay diferenciar entra pares e impares.

Con la función getId() cogemos todos los id, entonces tenemos que hacer una lógica para que diferencie **las clases** entre pares e impares y así nombrarlas en nuestro _module.less y poder meterle el margen.

```
<?php $value = ($lista->getId()%2==0) ? "par" : "impar";?>

<li class="<?= $value?>">
```

En este pequeño código dentro de nuestro template lo que hacemos es guardarnos en la variable value la clase par o impar según el id que toque.

Ahora en nuestro _module.less como queremos que modifique los impares, solo habría que poner li.impar porque nuestra clase está dentro de un li y dentro poner

margin-left: @margin-left-primary; @margin-left-primary es una variable que hemos declarado arriba como 20px ya que nos decía así el enunciado.

9.Botón de alerta con la nota más alta

En este apartado lo primero que hacemos es descubrir quien tiene la nota más alta.

Para ello cogemos el primero de la lista (con la función getId()==1) y nos la guardamos en una variable llamada alumno que la igualamos a la lista, por tanto este alumno tendrá que ser comparado con el resto de la lista, en cuanto uno de la lista supere al del alumno se guardará en la variable alumno.

Para crear un botón con alert:

Primero en el template creamos un botón con la clase “bottonAlert”

<input type="button" class="buttonAlert" value="Mostrar nota mas alta" />

Lo instanciamos a la clase del botón: Cuando pretemos el botón nos saldrá los datos del alumno con mejor nota.

```
<script type="text/x-magento-init">{
    ".buttonAlert": {
        "Hiberus_Lazaro/js/notaMasAlta": {
          "nombre": "<?= $alumno->getFirstName() ?>",
          "apellido": "<?= $alumno->getLastName() ?>",
          "nota": "<?= $alumno->getMark() ?>"}}}
</script>
```

El fichero js llamado notaMasAlta.js creamos el alert para que nos pille bien el nombre correcto.

alert(config.nombre + ' ' + config.apellido + ' con la nota: '+ config.nota);

10. Sacar la media en una nueva fila

Aquí solo trabajamos con el fichero de template (examenbloque.html), lo primero que hacemos es declarar una variable suma que vale 0.

Después nombramos con otra variable llamada nota la calificación de cada uno, que la cogemos de la función getMark(), después haríamos la expresión: suma=suma+nota donde nos va calculando la nota de cada alumno, por último printeamos la media que es la suma/total de las notas.

11. Crear Plugin que ponga 4.9 a los suspendidos

Dentro de la carpeta plugin creamos nuestro fichero PluginExample.php

Dentro de este fichero llamamos a nuestro repositorio para que nos haga un recorrido de todos los valores de nuestra tabla que es con la función getItems()

Recorremos este array y hacemos la lógica.

Si la calificación es menor a 5.00 (Lo hacemos con el getMark())

Con un setMarck(4.90) (Modificamos el valor de la nota y siempre valdrá 4.9 en caso de ser suspendido)

Para que funcione tenemos que ir a nuestro di.xml y hacer la inyección del Plugin.

```
<!-- Plugin -->

<type name="Hiberus\Lazaro\Api\AlumnoRepositoryInterface"><plugin type="Hiberus\Lazaro\Plugin\PluginExamen" name="hiberus-lazaro-plugin"/></type></config>
```

12. Alumnos aprobados en un color y suspendidos en otro

En este apartado lo hemos pensado igual que el de pares e impares.

En el fichero de templates creamos la lógica para diferenciar en clases si estas aprobado o suspendido.

```
<?php $cal = ($lista->getMark()>5.00) ? "aprobado" : "suspenso";?>

<div class="<?= $cal?>">
```

Una vez diferenciado vamos a nuestro _module.less y llamamos a cada una de las clases div.aprobado o div.suspenso y dentro ponemos el color.

En el apartado de extra te pide realizarlo con MIXIN.

Dentro del fichero _module.less creamos una clase llamada calificación que tienen como parámetro una variable (en nuestro caso @fondo) y dentro de esa clase elegimos que se quiere hacer (en nuestro caso cambiar el color)

Por tanto, igualamos: background: @fondo dentro de la clase calificación.

Para que nuestras clases de aprobado y suspenso lo lean correctamente dentro de ellas solo hay que llamar a la clase .calificacion(color que quieras).

13.Las 3 mejores notas destaquen

En este apartado añadiremos una función llamada get3notas() en nuestro fichero de bloque BloqueExamen.php que nos pille las 3 mejores notas, para ello usaremos 2 criterios de búsqueda (Ordene de manera ascendente y que corte en 3)

```
->addSortOrder($this->_sortOrderBuilder->setField('mark')->setDirection('DESC')->create())

->setPageSize(3)
```

Ahora como en ejercicio anteriores tenemos que diferenciar en clases los 3mejores y el resto.

En el fichero template creamos una variable donde nos recoja los3mejores gracias a la función creada en el bloque.

Después en el fichero de templates creamos la lógica para diferenciar en clases si eres de los 3 mejores o no.

```
<?php $mejor = (in_array($lista,$mejores)) ? "mejores" : "peores";?>

<div class="<?= $mejor?>">
```

Por último en el fichero _modules llamamos a div.mejores y he aumentado el tamaño y color respecto al resto.

14. Crear CLI Command

En este apartado visualizaremos por la línea de comandos nuestros valores de la tabla.

Para ello crearemos dentro la carpeta Console/Command el fichero ShowAlumnosCommand.php

Donde este fichero tiene 3 funciones principales configure donde le damos el nombre que en este caso tiene que ser hiberus:lazaro

El segundo es el execute que es el tiempo de ejecución y por último el process que son los valores que queremos que se muestren por linea de comandos.

Para que funcione debemos ir al di.xml y meterle los type ,argument e ítem necesarios.

Por último ponemos por línea de comandos php bin/magento donde dentro de hiberus tenemos el nombre hiberus:lazaro y para visualizarlo pondremos php bin/magento hiberus:lazaro

15. Api Rest

Dentro de la carpeta etc crearemos el fichero webapi.xml, en este fichero crearemos 3 rutas, una para visualizar los elementos de la tabla que en este caso el método será GET y será la única ruta que podremos acceder con URL que será: http://curso.magento.local/rest/V1/lazaro/alumnos?searchCriteria[pageSize]=10

La siguiente será para crear un elemento de la tabla en este caso para hacerlo tendrás que irte a Postman poner el token, accedes a la petición en este caso de crear alumnos y es de método POST (recibe los parámetros en el Body y no Request)

La ultima URL sirve para eliminar un elemento de la tabla, hemos escogido deleteById y esta URL tiene de característica que metes el idAlumno:

http://curso.magento.local/rest/V1/lazaro/alumnos/10