Memoria Escrita Examen Magento 2020 – Javier Soto Sierra

Ejercicio 1

Se ha creado una carpeta en la ruta app/code con el nombre del vendor (Hiberus) y dentro de esta carpeta el nombre del módulo, que en mi caso es Soto. También habrá que crear el archivo module.xml (dentro de la carpeta etc) y registration.php en la raíz del módulo. Esto archivos son obligatorios para crear un módulo nuevo.

Ejercicio 2

Para crear nuestra tabla "hiberus_exam" se ha utilizado el archivo dbschema.xml, que se ha creado en la carpeta etc del módulo.

En este archivo se crean todas las columnas de nuestra tabla con sus especificaciones.

Ejercicio 3

Para llevar a cabo este ejercicio se ha creado la carpeta "Api" , donde se ha creado nuestra interfaz del repositorio y las interfaces del objeto exam y examresults.

También se ha creado la carpeta "Model" donde se encuentra nuestro modelos que implementan las interfaces anteriores y la carpeta "ResourceModel" con las clases correspondientes, que se encargarán de estar en contacto con la base de datos, ya sea para guardar un elemento, obtenerlo de la base de datos o eliminarlo.

Ejercicio 4

Para realizar este ejercicio hay que crear la ruta Setup/Data/Patch dentro de nuestro modulo y dentro de esta ruta la clase "PopulateDataModel", que sirve para rellenar la base de datos. En este caso se ha puesto nombre y apellidos aleatorios, como también en la nota del examen.

Ejercicio 5

Se ha creado el archivo routes.xml dentro de nuestro directorio frontend en la carpeta etc. Aquí señalamos el nombre de nuestro frontname (soto). También se ha creado el directorio Controller/MiControlador donde encontramos nuestro archivo Index.php, el cual nos renderiza la página. La ruta de nuestra página en el front seria /soto/micontrolador/.

Ejercicio 6

Creamos nuestra carpeta web dentro del modulo y dentro la carpeta layout. Dentro creamos el archivo soto_micontrolador_index.xml (siguiendo el patrón <route_id>_<nombrecontrolador>_<action>).

Creamos la carpeta Block y dentro una carpeta llamada “Bloque” contendrá el archivo ExamList.php, el cual nos ayudará en este caso a pasarle los datos de los exámenes a nuestra plantilla.

Para crear nuestra plantilla, se crea la carpeta templates en la carpeta web y dentro se nos hemos creado la plantilla “examlist.phtml”. En esta plantilla se ha puesto un título y un listado con los exámenes, donde aparecen el nombre, apellido y nota del examen. Al final aparece un texto con el número total de exámenes mostrado. Para añadir la traducción de este texto final, hay que encerrar el texto con “__()” y crear el archivo “es_ES.csv” en una carpeta que tenemos que crear, que se llama “i18n”.

Ejercicio 7

Primero creamos nuestro archivo js dentro de la carpeta web, donde crearemos la carpeta js, en nuestro caso se llama examen.js . En nuestra plantilla creamos el pequeño script situado al final, donde le pasaremos el elemento “.examenes”, el cual contiene la lista de todos los exámenes.

Dentro de nuestro archivo js, capturamos el elemento y con la función on(‘click’,’boton’), haremos que cuando se pulse el botón de ocultar las notas se oculten los elementos con la clase nota. En nuestro caso se han puesto dos botones, uno para ocultar las notas y otras para mostrarlas de nuevo. Cuando se pulsa uno de estos el otro se oculta.

Ejercicio 8

Para ello se ha creado el archivo _module.less en la carpeta web/css/source . Se han creado dos bloques principales, el primero para tamaños estándares de pantalla y el segundo para los tamaños que sean superiores a 768px. En nuestro caso hemos aplicado la mayoría de estilos en la segunda opción ya que nuestra pantalla es mayor de 768px.

En el primer caso se ha puesto un título de color azul y en el segundo de color rojo.

Hemos puesto que los li dentro de la clase “exámenes” que sean impares tengan un margen izquierdo de 20px, pasándole el parámetro @margin-left-primary declarado al principio del archivo con el valor de 20px.

Ejercicio 9

Para realizar este ejercicio se le ha pasado por parámetro desde la plantilla la mejor nota con el nombre “config”. Para realizar la alerta se ha utilizado la función alert pasándole también la variable de la mejor nota, que está dentro de config (config.config).

Ejercicio 10

Para ello, se ha almacenado en variables php la suma de las notas y el número de elementos y se ha creado un párrafo el cual contiene la media de las notas.

Ejercicio 11

Se crea la carpeta plugin dentro de nuestro modulo con la clase PluginExample.php, donde tenemos nuestro método afterGetList el cual modifica todos las notas suspensas a un 4.9 sin modificar la base de datos. También creamos el archivo di.xml dentro de la carpeta frontend en etc, donde estableceremos que se use el plugin.

Ejercicio 12

Para este ejercicio, se le ha aplicado un fondo de color rojo a las notas que sean suspensas y verde a las aprobadas. Para comprobar esto se ha realizado un condicional que compruebe las notas de cada examen y si superan el 5 se pone en verde y si no se ponen en rojo.

Ejercicio 13

Primero mediante el sortOrderBuilder se han ordenado la lista de los exámenes (todo esto en el bloque) y se ha utilizado un contador establecido a 3 que irá decrementándose a medida que recorremos los exámenes. Así los tres primeros tendrán un elemento extra que será un párrafo con un texto que ponga “DESTACADO!!!”.

Ejercicio 14

Para ello se han creado todos los directorios que hay dentro de la carpeta Console. A destacar dentro del método configure establecemos el nombre del comando y el texto descriptivo del comando y el método process no hará la impresión de los exámenes.

También habrá que establecer el comando en nuestro di.xml .

Ejercicio 15

Se ha creado el archivo webapi.xml en nuestro directorio etc. Aquí estableceremos las rutas para cada endpoint donde utilizaremos los métodos de nuestra interfaz del repositorio getLIst(para obtener el listado), save(para guardar el examen) y deleteById(para eliminar algún examen).

Ejercicio 16

Para ello se ha creado la carpeta adminhtml en el directorio etc. Donde crearemos los archivos system.xml y menú.xml. Dentro de system.xml estableceremos el Tab de Hiberus y también estableceremos los parámetros elementos y nota de corte, los cuales serán los inputs de texto que utilizaremos en la vista de admin. Para asignarles un valor por defecto, lo hacemos en el archivo config.xml de la carpeta etc.

Se ha creado un helper para poder acceder a estos elementos y así utilizarlos en otros sitios, concretamente en nuestro bloque de nuestra plantilla, para así poder saber cuántos elementos mostrar y cuál es la nota que marca el aprobado.

Ejercicio 17

Nuestros virtualtypes los hemos definido en nuestro di.xml , donde hemos establecido el nombre del logger como hiberus_soto.log y lo hemos inyectado como dependencia en el archivo de nuestro bloque de nuestro front. Así cada vez que se utilice el bloque escribirá en nuestro logger la hora de acceso a esa página y la nota media de los alumnos.