Jorge Fernandez, 3/12/2020

1. Crear el nuevo módulo

* En el directorio app/code/Hiberus, para crear el nuevo módulo, necesitamos primero crear la carpeta, en este caso la carpeta
se llamará Fernandez.
* Una vez creada la carpeta, para registrar nuestro módulo, necesitaremos crear en la raíz, el fichero registration.php en el cual
procederemos a registrar nuestro nuevo módulo Hiberus_Fernandez, una vez creado este fichero, necesitamos también crear, en el
subdirectorio etc, el fichero module.xml
* Una vez creados estos ficheros, procedemos a ejecutar setup:upgrade para que nos carge nuestro nuevo módulo y podemos ver como
aparece entre el listado de módulos que carga el comando. Una vez hecho esto, ya tendríamos nuestro modulo creado y registrado.

2. Crear una tabla en la base de datos

* Creamos bajo el directorio etc el ficher db_schema.xml en el cual introducimos las columnas de la tabla a crear. En este caso
hay que destacar la inclusión de una columna con valores decimales a la cual ha habido que indicarle los valores del parametro
precision y scale, referentes al número de digitos antes y despues del decimal.

* Una vez creado el fichero volvemos a ejecutar setup:upgrade

* Y una vez ejecutado podemos observar con ayuda de MySQL Workbench que la tabla se ha creado correctamente.

3,4. Generar las diversas clases que van a gestionar la tabla creada previamente

* Para comenzar, procedemos a definir las interfaces tanto de la tabla Exam que acabamos de crear, como de su

Repository, como de su SearchResult.

En cuanto a la interfaz de Exam:
** Creamos el directorio Api en la raiz de nuestro módulo y dentro de ese directorio, el subdirectorio Data.
** Creamos nuestro fichero ExamInterface.php y lo completamos.
** Una vez completado, creamos en el mismo subdirectorio Data, el fichero ExamSearchResultsInterface.php y lo completamos.
** Por último, necesitaremos crear la interfaz del Repository, para ello creamos el fichero ExamRepositoryInterface.php en el
directorio Api, pero fuera del subdirectorio Data y lo completamos. Esta primera parte se corresponde con la capa de Service Contracts.

Una vez completada la parte de interfaz en el directorio Api, comenzamos a crear las clases del Modelo, esto se hace en el directorio
Model.
Lo primero que encontraremos dentro del Model son las clases que implementan las interfaces que hemos definido previamente, situadas
en la raíz del directorio Model.
* Definimos estas clases en la raiz de Model.
* Posteriormente crearemos el Resource Model en este caso de la tabla hiberus_exam, que será la encargada de manipular los datos en la
bd.

Una vez completado todo este proceso, debemos crear el fichero di.xml para registrar las diversas dependencias y relaciones entre las
clases que hemos creado previamente, así como la relación con la tabla de la bd.

Con todo esto completado, setup:upgrade y al finalizar podremos observar como nuestra tabla hiberus_exam ya se encuentra poblada
con datos.

5. Crear controlador de frontend.

Para crear el controlador, el primer paso es crear el fichero routes.xml en el directorio etc/frontend. En el vamos a indicar la ruta
que queremos definir, su frontname así como el modulo que va a responder a esa ruta.

Una vez creado el fichero routes, procedemos a crear el fichero Controlador. Este fichero debe ser creado bajo la subcarpeta Controller
En el directorio Controller vamos a crear nuestro Controlador Exams, y dentro de el el fichero Index.php.
Una vez completados estos ficheros, al acceder a la url
http://curso.magento.local/fernandez/exams nos responderá con un "Hola".

6. Añadir layout, bloque y template a la página.

Para empezar, crearemos el layout y el template para que muestren inicialmente solo el título.
Para ello, crearemos el template que incluirá el titulo primero, el fichero se encontrará en view/frontend/templates/title.phtml.
Una vez creado, lo incluiremos en el fichero de layout, creado en view/frontend/layout/hiberusfernandez_exams_index.xml.
Antes de nada debemos modificar nuestro controlador para que nos renderize la página correspondiente.
Una vez hecho esto, al acceder a la página se nos mostrará nuestro titulo.

Una vez completado el titulo, debemos mostrar el listado de examenes.
Crearemos el bloque en Block/Exams/ExamList.php y posteriormente un template que utilize esos datos en view/frontend/templates/exam/exam_list.phtml
Posteriormente añadimos el bloque al fichero de layout creado en view/frontend/layout/hiberusfernandez_exams_index.xml.

Para añadir el total de examenes almacenados, debemos añadir una nueva función en el bloque, encargada de realizar la consulta a la bd
pero en este caso en vez de utilizar el metodo getElements(), utilizaremos el getTotalCount() que devuelve un entero correspondiente a la
cuenta.

Por último, para la traducción, debemos primero crear el fichero csv de traducción al español es_ES.csv situado en el directorio
i18n/es_E.csv y en el incluimos nuestra traducción.

Una vez cargado, desde el administrador debemos modificar el idioma utilizado a español

7. Crear JS.

Primero, creamos el js en el directorio /view/frontend/web/js/ocultarnotas.js. EL js se encargará de una vez pulsado el boton,
buscar todos los campos referentes a la columna de notas y modificarlos para que no se muestren.

Una vez creado el JS, debemos modificar el template para incluir el boton así como el script y al acceder comprobar que funciona.

8. Maquetar con less.

Lo primero es crear el fichero _module.less en el directorio
view/frontend/web/css/source/_module.less.

Para modificar el color del título, incluimos tanto la linea que permite definir estilos genéricos(& when (@media-common = true) {)
como la linea que nos permite definir estilos responsive(.media-width(@extremum, @break) when (@extremum = 'min') and (@break = @screen__m) {).
En este caso vamos a utilizar la definición de estilos genéricos para definir el color por defecto del título, mientras que el otro se aplicará
solo cuando se cumplan las condiciones.

Para incluir el margin-left solo en los impares, podemos hacer uso de nth-child(odd) aunque en este caso, debido al li de cabeceras,
utilizaremos even en lugar de odd.

9. Alert con nota más alta

Para ello vamos a crear un nuevo fichero de javascript, notamasalta.js.
En el cual vamos a incluir la lógica del calculo de la nota más alta a partir de los datos mostrados
en la pagina web.

10. Nota media de la clase

Para sacar la nota media de la clase, vamos a hacer uso del bloque que definimos previamente para conseguir el listado
de Examenes y el número, de esta manera, creamos una nueva función que gracias al resultado de la llamada a las otras dos funciones
existentes en ese bloque, sea capaz de recorrer el array de datos extrayendo el campo que nos interesa, la nota, para tras conseguir
la suma completa, devolver ese número dividido por el numero de examenes obtenido inicialmente.
Para incluir esto en nuestro template phtml, simplemente debemos añadir una nueva linea en la cual insertaremos mediante php la llamada
a esta nueva función.

11. Plugin que ponga a todos un 4.9.

Para hacer este plugin, la idea mas lógica es tratar de ejecutarlo justo despues de la función del bloque que devuelve el listado final de los
examenes para que se muestren en la página web.

Primero, debemos crear la clase del plugin en el directorio Plugin/PluginNota.php. Una vez completado el código del Plugin, procedemos
a indicar en el fichero /etc/di.xml la definición del Plugin, indicando la clase que observa, la clase del plugin e indicando inicialmente que está habilitado.
Una cosa que ha debido utilizarse en el desarrollo del Plugin ha sido que en el mismo, al recorrer el array de Examenes buscando aquellos suspenidos,
la necesidad de introducir en el foreach los diversos valores por referencia, para poder hacer efectivas las modificaciones en los datos.

12. Distinguir color entre aprobados y suspensos.

Para este paso, podemos hacerlo directamente en el template phtml, en el cual para cada li que definimos uno por examen, podemos hacer
la comprobación de la nota y en función de ella asignarle un color u otro de background.

13. Destacar 3 mejores

Para este paso, podemos optar porque en la query en la cual recuperamos los examenes, realizar esa búsqueda ordenada por la nota, y de
esa manera poder mediante un contador saber cuales son el top 3 e indicarlo. Para ello, vamos a modificar el bloque que se encarga
de consultar la base de datos para que utilize este orden. Una vez añadido el campo que indica si el examen está en el top3, en el template
hacemos la comprobación del valor, y en caso afirmativo incluimos la indicación de ese Top3.

14. CLI Command

Primero, creamos el fichero del comando en Console/Command/ExamListCommand.php.

Una vez creado y definido el comportamiento del comando, creamos el directorio Options del comando, con el fichero Options/ExamList/ListOptions.php
Al igual que el fichero Input/ExamList/ListInputValidator.php

Una vez completados ambos, registramos el comando en el fichero di.xml del directorio etc.

15. Crear endpoints.

Para crear los endpoints, debemos crear el fichero /etc/webapi.xml y en el incluir nuestros nuevos endpoints con las funciones que utilizan.En
este caso se ha utilizado ref anonymous por facilidad en todos los endpoints.

17. Crear Logger.

Por último, para el logger vamos a hacer uso de la definición de virtual Types en el fichero etc/di.xml para redireccionar nuestro logger
al fichero que deseamos, posteriormente solamente debemos utilizar la clase Psr\Log\LoggerInterface para escribir en nuestro log.