PymesAng paso a paso

**Objetivo** crear una Aplicación Web de múltiples capas integrando acceso a datos con EF, controladores de WebApi2, y vistas con Html5, Boostrap y código Javascript con AngularJs

**CREAR SOLUCION, PROYECTOS**

1. Crear Solucion/proyecto ‘PymesAng’ (asp.net vacio)
2. Agregar proyecto ‘Datos’ (tipo biblioteca de clases), en donde encapsulamos la funcionalidad de acceso a datos

En el proyecto Datos:

1. Agregar un entity data model, con nombre ‘Pymes’ (verificar que el DbContext quede como PymesEntities), mapear las tablas Artículos y ArticulosFamilias. (grabar y compilar la solución).
2. Agregar la clase /DatosAuxiliares/GestorArticulos.cs que tiene todos los métodos para el abm que usan EF. (proyecto compila con error, se soluciona en el próximo punto)
3. Observe que la ejecución del asistente entre otras cosas generó en el archivo app.config una cadena de conexión en la sección ‘<connectionStrings>’ y agregó referencia a las librerías de EF: EntityFramework.dll y EntityFrameworkSqlServer.dll

En el proyecto web PymesAng:

1. Agregar referencia proyecto Datos.
2. Agregar al archivo web.config la cadena de conexión usada en el proyecto Datos dentro del archivo app.config (ver punto 5), copiar toda la sección ‘<connectionStrings>’ como hija de la sección ‘<configuration>’

Para poder usar el proyecto ‘Datos’ que tiene EF necesitamos:

1. Agregar referencia a EntityFramework.dll (con la opción explorar de NuGet, antes ver la ruta de la misma referencia que ya existe en proyecto datos)
2. Agregar referencia a EntityFrameworkSqlServer.dll (con la opción explorar, antes ver la ruta de la misma referencia que ya existe en proyecto datos)

**WEBAPI**

1. Agregar nuevo elemento con scaffolding: Controlador de WebApi2 con acciones que usan EntityFramework llamarlo ‘ArticulosController’; relacionarlo con Datos.Articulos de Datos.PymesEntities.

(solución fin ETAPA1)

---------------------------------------------------------

1. Modificar en la Clase ArticulosController.cs recién generada, el metodo GetArticulos(int id) para que en lugar de entity framework use Datos.GestorArticulos.BuscarPorId(id).

Reemplazar: Articulos articulos = db.Articulos.Find(id);

Por: Articulos articulos = Datos.GestorArticulos.BuscarPorId(id);

1. Sobreescribir toda la clase ArticulosController por la misma clase provista en la carpeta DatosAuxiliares, la cual como se hizo en el punto anterior reemplaza todos los métodos para que use la clase Datos.GestorArticulos y no EF.
2. Agregar nuevo elemento con scaffolding WebApi2 con clases de EntityFramework llamado ArticulosFamiliasController que utilice el recurso de EF ArticulosFamilias (dejamos para otra etapa posterior hacer en Datos un GestorArticulosFamilias)

Ahora construiremos las Vista HTML:

1. Agregar carpeta content (archivos css) (está en DatosAuxiliares)
2. Agregar carpeta images (está en DatosAuxiliares)
3. Agregar carpeta font (está en DatosAuxiliares)
4. Agregar carpeta scripts (librerías de javascript) (está en DatosAuxiliares)
5. Agregar página html llamarla ‘inicio.html'
6. Desde el explorador de soluciones establecer como proyecto de inicio (botón derecho del mouse) al proyecto ‘PymesAng’
7. Desde el explorador de soluciones establecer como página de inicio a la página ‘Inicio.html’
8. Reemplazar el contenido de la página inicio según el provisto en el archivo:  
    /DatosAuxiliares/html\_inicio.txt
   1. Identificar referencias a carpetas agregadas en el punto anterior: content, images, scripts
   2. Identificar orden de las librerías js.
9. agregar codigo html para el menu bootstrap (segun ejemplos de https://getbootstrap.com/docs/3.3/examples/navbar-fixed-top/), copiar etiqueta <nav> con todo su contenido. (ejecutar el codigo para ver como quedo).
10. Modificar el menú para adaptarlo a nuestras necesidades, inicialmente poder navegar a páginas articulos.html y articulosfamilias.html (que próximamente vamos a agregar);
11. Reemplazar el codigo del menu agregado en el punto anterior por el provisto en el archivo:   
     /DatosAuxiliares/html\_menu\_inicio.txt
12. Agregar debajo del menu el código html-bootstrap (mensaje de bienvenida), según el provisto en el archivo:   
     /DatosAuxiliares/html\_bienvenido\_inicio.txt

**ANGULARJS**

1. Agregar en la carpeta scripts un archivo js, llamarlo ‘app.js’ en donde vamos a escribir el código javascript/angular de nuestra aplicación.
2. Escribir codigo básico de angular para iniciar la aplicación angular (‘myApp’) y su controlador (‘inicioCtrl’) en donde defina una variable del $scope: Titulo='Pymes Demo'.   
    Ver como quedaría este codigo en /DatosAuxiliares/js\_inicio.txt
3. Agregar a la página inicio.html la referencia al script app.js agregado en el punto anterior
4. Modificar el código html de la página inicio, para que funcione angular:

reemplazar: <body>

por <body ng-app='myApp' ng-controller='InicioCtrl'>

reemplazar: <h1>Pymes demo</h1>

por <h1>{{Titulo}}</h1>

1. Verificar el funcionamiento de la solución hasta este momento
2. Agregar la página ArticulosFamilias.html (tomar de base el mismo código que inicio.html)

reemplazar: <body ng-app='myApp' ng-controller='InicioCtrl'>

por <body ng-app='myApp' ng-controller='ArticulosFamiliasCtrl'>

Reemplazar el elemento <div class="jumbotron"> y su contenido por el provisto en el archivo:

/DatosAuxiliares/html\_ArticulosFamilias.txt

1. Agregar al archivo app.js el siguiente código que usará el controlador de angular de ArticulosFamilias.html

myApp.controller('ArticulosFamiliasCtrl',

function ($scope, $http) {

$scope.Titulo = ‘Gestionar Articulos Familias’;

$http.get('/api/ArticulosFamilias')

.then(function (response) {

$scope.Lista = response.data;

});

}

Al pasarle $scope y $http le paso el ambito del controlador con la informacion necesaria.

No le paso referencia a objeto que deba mantener si no que todo eso ocurre de forma automática

);

1. Verificar el funcionamiento de la solución hasta este momento

(Solucion fin ETAPA2)

---------------------------------------------------------

1. Agregar la pagina Articulos.html (tomar de base el mismo codigo que inicio.html)
2. Reemplazar <div class="jumbotron"> y su contenido por el provisto en el archivo:

/DatosAuxiliares/html\_Articulos.txt. Observe que los input ya se encuentran con la directiva ng-model que los enlazará a una variables del objeto $scope de angular.

1. Reemplazar: <body ng-app='myApp' ng-controller='InicioCtrl'>   
   por <body ng-app='myApp' ng-controller='ArticulosCtrl'>
2. Agregar al archivo app.js el siguiente codigo que usará el controlador de angular de Articulos.html

myApp.controller('ArticulosCtrl',

function ($scope, $http) {

$scope.Titulo = 'Gestionar Articulos'; // inicia mostrando el Listado

// articulo cargado inicialmente, como demo para probar la interface visual (luego comentar esta linea)

$scope.Lista = [{ IdArticulo: 1, Nombre: 'Articulo Demo', Precio: 110.25, Stock: 357, FechaAlta: '17/01/2017', Activo: true, CodigoDeBarra:'1234567890123', IdArticuloFamilia:1 }];

}

);

1. Verificar el funcionamiento de la solución hasta este momento.
2. en el archivo app.js reemplazar solo el código que define el controlador ArticulosCtrl por el provisto en el archivo: /DatosAuxiliares/js\_Articulos.txt
3. Observe en el código javascript agregado los siguiente elemento definidos en el $scope::
   1. el siguiente objeto que indica las acciones que pueden hacerse en el abmc:  
      $Scope.TituloAccionABMC = { A: '(Agregar)', B: '(Eliminar)', M: '(Modificar)', C: '(Consultar)', L: null}  
      Se usa en el código html para mostrar al lado del título de la página la acción actual del ABM
   2. una propiedad que indica la acción actual que se está haciendo del abm:  
      $scope.AccionABMC = ‘L’; // inicialmente inicia el el listado (buscar con parámetros)  
      Se usará en el código html para poner visible / invisible los diferentes elementos de la interface.
   3. un objeto con los mensaje para informar al usuario:  
      $cope.Mensajes = {SD: ' No se encontraron registros...', RD: ' Revisar los datos ingresados...' }
   4. Los métodos: Agregar(), Buscar(), Consultar(), Modificar(), Grabar(), ActivarDesactivar(), Cancelar(), ImprimirListado() que son invocados desde la vista html por las directiva ng-click y ejecutar las acciones propias del ABM
4. A continuación modificaremos la vista para que sólo esté visibles/habilitados los elementos según correspondan a la acción actual del abmc, definida por la variable $scope.AccionABMC
   1. Reemplazar : <form name="FormFiltro">  
      Por: <form name="FormFiltro" ng-show="AccionABMC == 'L'">
   2. Reemplazar: <div id="TablayPaginador">  
      Por:<div id="TablayPaginador" ng-show="AccionABMC == 'L' && Lista.length">
   3. Reemplazar: <div id="msjSD" class="alert alert-info">  
      Por: <div id="msjSD" ng-show="AccionABMC == 'L' && Lista && Lista.length== 0" class="alert alert-info">
   4. Reemplazar: <form name="FormReg">  
      Por: <form name="FormReg" ng-show="AccionABMC != 'L'">
   5. En cada input del <form name=’FormReg’> agregar la siguiente condición para que esté deshabilitado en las consultas  
      ng-disabled="AccionABMC === 'C'"
5. Para que el botón Grabar se muestre según corresponda agregarle la siguiente directiva:  
   ng-show="AccionABMC == 'A' || AccionABMC == 'M' "
6. Para que el botón que tiene el texto: ‘Volver/Cancelar’; muestre el texto “Cancelar” cuando quiera salir de una edición o alta sin grabar ó muestre el texto ‘Volver’ cuando esté en una consulta; reemplazar el texto ‘Volver/Cancelar’ por la siguiente expresión:  
   {{AccionABMC == 'A' || AccionABMC == 'M' ? 'Cancelar' : 'Volver'}}
7. Modificar el div que muestra el mensaje ‘Revisar Datos’, que debe activarse cuando no son válidos los datos, por ahora lo ponemos invisible agregándole la siguiente directiva: ng-show="false”  
     
   A continuación implementaremos alguna funcionalidad en el código javascript/angular:
8. En el archivo app.js en el Controlador ArticulosCtrl vamos a eliminar el artículo de demo que se usa para cargar la tabla, porque vamos a traer los datos del servidor. Elimine la siguiente línea:  
   $scope.Lista = [{ IdArticulo: 1, Nombre: 'Articulo Demo', Precio: 110.25, Stock: 357, FechaAlta: '17/01/2017', Activo: true, CodigoDeBarra:'1234567890123', IdArticuloFamilia:1 }];
9. En el archivo app.js, agregar la siguiente funcionalidad al método Buscar()  
     
   $scope.Buscar = function () {

// las propiedades del params tienen que coincidir con el nombre de los parámetros de c# (case sensitive)

params = { Nombre: $scope.DtoFiltro.Nombre, Activo: $scope.DtoFiltro.Activo, numeroPagina: $scope.PaginaActual };

$http.get('/api/Articulos', { params: params })

.then(function (response) {

$scope.Lista = response.data.Lista; // variable para luego imprimir

$scope.RegistrosTotal = response.data.RegistrosTotal; // var para mostrar en interface

});

};

1. En articulos.html vamos a permitir que pulsando enter sobre el nombre del artículo a buscar se ejecute la búsqueda asociada al botón buscar. Al input del dtofiltro.nombre agregarle la siguiente directiva:

ng-keypress="$event.which == 13 && Buscar()"

(Solucion fin ETAPA3)

-------------------------------------------------

1. Observe la técnica usada en la tabla de artículos en la columna Activo para que aparezca según el valor de la propiedad dto.Activo (true/false) el texto SI/NO  
   <td>{{dto.Activo ? 'SI' : 'NO'}}</td>
2. Observe la técnica usada en la tabla de artículos en el columna acciones para que aparezca según el valor de la propiedad dto.Activo (true/false) el icono de glyficons-remove o glyficons-ok en color rojo o verde.

<button class="btn btn-{{ dto.Activo ? 'danger':'success'}}"

title="{{dto.Activo ? 'Desactivar':'Activar'}}"

ng-click="ActivarDesactivar(dto)">

<span class="glyphicon glyphicon-{{dto.Activo ? 'remove':'ok'}}"></span>

</button>