

## UD6 Desarrollo de aplicaciones web con ASP.NET Core.

### 6.4 Toques finales.

En esta última parte añadiremos algunos toques para mejorar la funcionalidad de nuestra aplicación, si bien se podría considerar que está completa ya que realiza todas las tareas que necesitamos e incluye autenticación y gestión de usuarios.

Entre las características que vamos a añadir se encuentran diversas mejoras en las páginas que muestran listados, además de añadir cierta personalización en el aspecto de la aplicación.

#### Búsqueda y ordenación de tablas.

Si bien es cierto que nuestra aplicación es capaz de mostrar toda la información que necesitamos, también lo es que los listados que muestra son simples tablas, sin paginado, ordenación ni capacidad de búsqueda. En tablas pequeñas como las de ubicaciones o departamentos esto podría ser aceptable, pero en un inventario limitaría mucho su capacidad, puesto que no disponer de estas características en una tabla de cientos de objetos haría que su uso fuera incómodo, o podría hacerlo incluso inmanejable.

Hay muchas formas de abordar estas tareas, y por supuesto, se podría programar todo ello en la vista y el controlador, pero en este caso vamos a recurrir a una de las muchas librerías disponibles para esta tarea, lo que nos facilitará enormemente el trabajo.

#### jQuery DataTables.

Se trata de un plug-in para jQuery, que permite añadir características avanzadas a cualquier tabla html, tales como paginación, búsqueda instantánea del lado cliente, ordenación multicolumna, etc..

Esta librería es open source y se puede obtener toda la información acerca de sus características y su utilización en su página web:

<https://datatables.net>

#### Añadir soporte para DataTables en nuestro proyecto.

Siguiendo las instrucciones de la web de DataTables, añadimos los enlaces necesarios en nuestra plantilla de vistas o Layout.

1. En la cabecera añadimos el enlace a la hoja de estilos.

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - CIFP A Carballeira</title>

  <environment include="Development">
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
  </environment>
  <environment exclude="Development">
```

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css"
asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
asp-fallback-test-class="sr-only"
asp-fallback-test-property="position"
asp-fallback-test-value="absolute" crossorigin="anonymous"
integrity="sha256-eSi1q2PG6J7g7ib17yAaWMcrr5GrtohYChqibrV7PBE=" />
</environment>
<link rel="stylesheet" href="~/css/site.css" />
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
integrity="sha384-fnmOCqbTlWIlj8LyTjo7m0UStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
crossorigin="anonymous">
<link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/v/bs4/dt-1.10.18/datatables.min.css" />
</head>
```

Esta es la hoja de estilos específica para Bootstrap 4, que es la librería que se utiliza por defecto en los proyectos ASP.NET Core desde la versión 2. El código JavaScript, en lugar de añadirlo a la plantilla, lo haremos en cada una de las vistas que haga uso de DataTables. Esto es así porque de este modo evitamos la carga de un fichero js en vistas que no lo necesitan. Con esto ya podemos comenzar a utilizar esta extensión en nuestras tablas.

## Lista de departamentos.

Comenzamos por la vista más sencilla. Aunque al ser la lista de departamentos en teoría una lista con pocos elementos y por lo tanto podría dejarse tal y como la tenemos, vamos a ver que usar DataTables es realmente sencillo.

Antes de convertir nuestra tabla en una DataTable, comenzaremos con mejoras en la presentación utilizando las clases de Bootstrap.

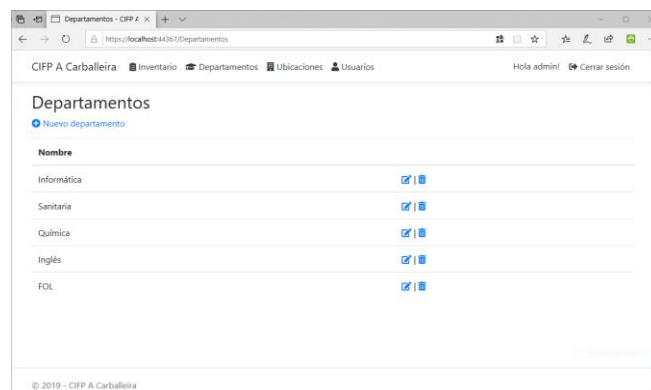
1. Comenzamos por cambiar la definición de la tabla:

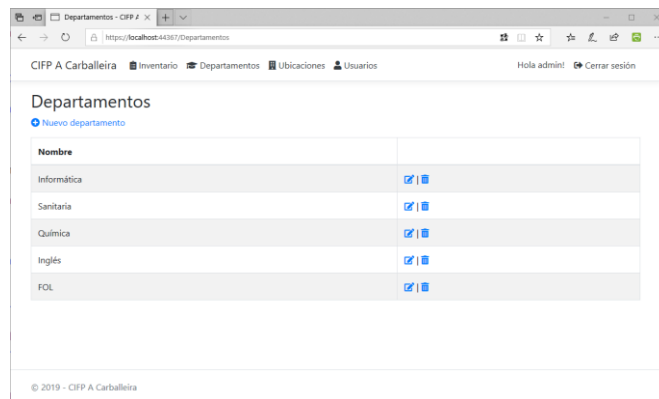
```
<table id="departments-table" class="table table-striped table-bordered table-hover">
```

Hemos identificado la tabla de modo que podamos luego utilizar el script de DataTables, y hemos aplicado estilos de tabla con borde, con colores de fondo alternativos en cada fila y que cambia de color al pasar el cursor por una fila.

Todas estas clases se pueden encontrar en la documentación referente a tablas de Bootstrap 4.

En las siguientes imágenes vemos como estos atributos cambian la presentación de nuestra tabla:





2. A continuación añadimos el script que añadirá las funcionalidades de DataTables a nuestra tabla:

```
@section scripts {
<script type="text/javascript"
src="https://cdn.datatables.net/v/bs4/dt-1.10.18/datatables.min.js">
</script>

<script type="text/javascript">
$(document).ready(function () {
$('#departments-table').DataTable({
language: {
"sProcessing": "Procesando...",
"sLengthMenu": "Mostrar _MENU_ registros",
"sZeroRecords": "No se encontraron resultados",
"sEmptyTable": "Ningún dato disponible en esta tabla",
"sInfo": "Mostrando registros del _START_ al _END_ de un total
de _TOTAL_ registros",
"sInfoEmpty": "Mostrando registros del 0 al 0 de un total de 0
registros",
"sInfoFiltered": "(filtrado de un total de _MAX_ registros)",
"sInfoPostFix": "",
"sSearch": "Buscar:",
"sUrl": "",
"sInfoThousands": ",",
"sLoadingRecords": "Cargando...",
"oPaginate": {
"sFirst": "Primero",
"sLast": "Último",
"sNext": "Siguiente",
"sPrevious": "Anterior"
},
"oAria": {
"sSortAscending": ": Activar para ordenar la columna de
manera ascendente",
"sSortDescending": ": Activar para ordenar la columna de
manera descendente"
}
}
});
</script>
}
```

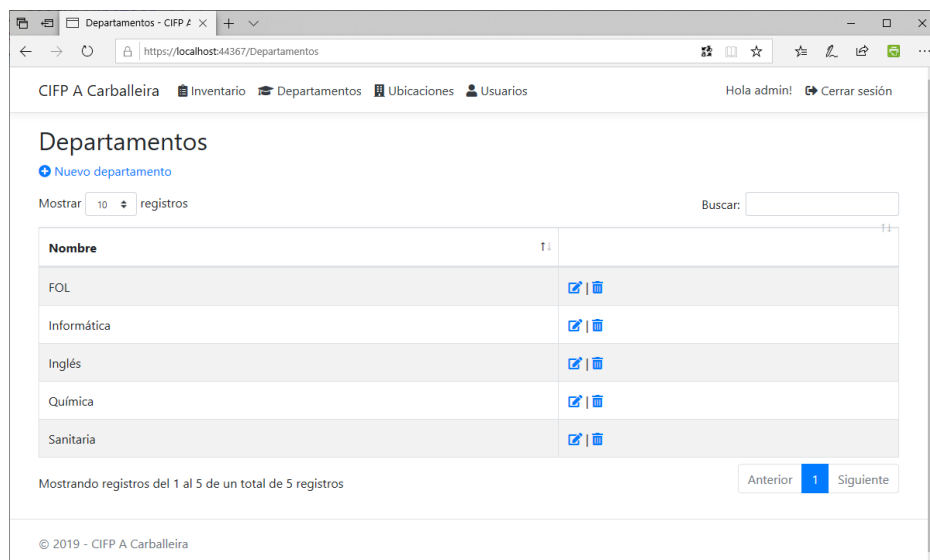
Lo que hacemos es, en primer lugar, añadir el fichero js de DataTables para Bootstrap. En segundo lugar llamamos al método DataTable sobre nuestra tabla, utilizando su identificador. De este modo añadimos toda la funcionalidad, simplemente llamando a un método.

En realidad, el código que hace todo esto es simplemente esta llamada:

```
<script type="text/javascript">
    $(document).ready(function () {
        $('#departments-table').DataTable();
    });
</script>
```

El resto de parámetros que hemos utilizado en nuestro código son parámetros de lenguaje, y sirven únicamente para traducir los mensajes de la tabla ya que los mensajes por defecto se encuentran en inglés. Todo esto se puede consultar en la documentación de jQuery DataTables.

- Si ejecutamos ahora nuestra aplicación y vamos a la vista de departamentos, veremos que sin demasiado esfuerzo hemos conseguido funcionalidad de paginado, ordenación y filtrado en nuestra tabla, mejorando enormemente la usabilidad de nuestra aplicación.



## Lista de ubicaciones.

- Repetiremos la misma operación con nuestra vista de ubicaciones, comenzando por identificar nuestra tabla y cambiar los estilos:

```
<table id="locations-table" class="table table-striped table-bordered table-hover">
```

- A continuación utilizamos el mismo código que hemos usado para nuestra tabla de departamentos, asegurándonos de cambiar el nombre del identificador de la tabla:

```
<script type="text/javascript" src="https://cdn.datatables.net/v/bs4/dt-1.10.18/datatables.min.js"></script>

<script type="text/javascript">
    $(document).ready(function () {
        $('#locations-table').DataTable({
```

## Lista de usuarios.

1. Los pasos son los mismos para nuestra tabla de usuarios, siendo lo primero identificar nuestra tabla y cambiar los estilos:

```
<table id="users-table" class="table table-striped table-bordered table-hover">
```

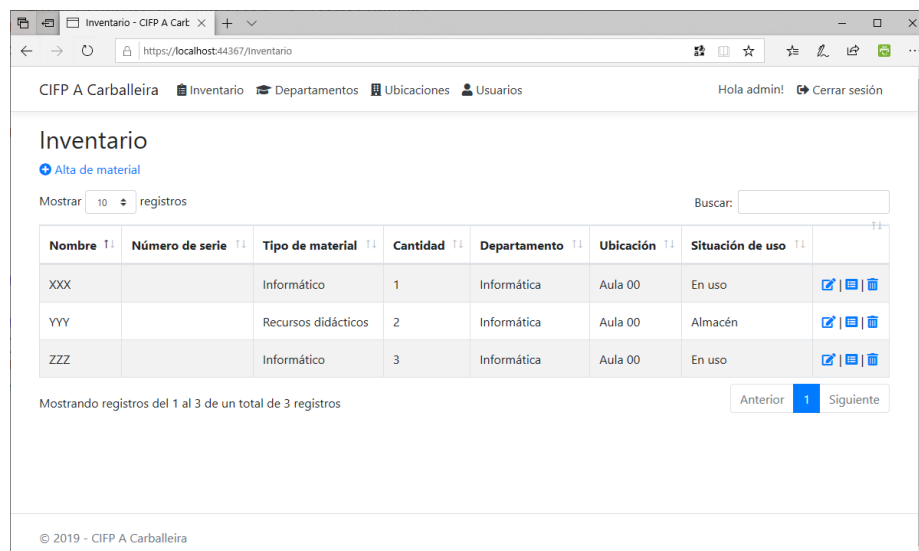
2. De nuevo utilizamos el mismo código, esta vez sobre la tabla #users-table.

## Inventario.

Y llegamos ya a la última tabla. Los pasos, como es de esperar serán los mismos, siendo el proceso de convertir una tabla sencilla en una tabla con funcionalidades extendidas muy fácil utilizando la librería jQuery DataTables. En una tabla como esta, que previsiblemente contendrá un gran número de objetos, es donde realmente esta funcionalidad extra supondrá una gran ventaja y una mejora importante en la usabilidad de nuestra aplicación.

1. Como hemos hecho hasta ahora, cambiamos la definición de la tabla para identificarla y luego utilizamos el código que llama a la función DataTable para nuestra tabla de inventario:

```
<table id="materials-table" class="table table-striped table-bordered table-hover">
```



The screenshot shows a web application interface for an inventory system. The header includes the application name 'CIFP A Carballeira' and navigation links for 'Inventario', 'Departamentos', 'Ubicaciones', and 'Usuarios'. The main content area is titled 'Inventario' and includes a link for 'Alta de material'. Below this, there is a search bar and a table with 3 records. The table has columns for 'Nombre', 'Número de serie', 'Tipo de material', 'Cantidad', 'Departamento', 'Ubicación', and 'Situación de uso'. Each row has a set of icons for editing and deleting records. The footer shows the copyright information '© 2019 - CIFP A Carballeira'.

Nombre	Número de serie	Tipo de material	Cantidad	Departamento	Ubicación	Situación de uso
XXX		Informático	1	Informática	Aula 00	En uso
YYY		Recursos didácticos	2	Informática	Aula 00	Almacén
ZZZ		Informático	3	Informática	Aula 00	En uso

## Personalización de estilos.

Una vez tenemos una aplicación completa, otro aspecto en el que se podría trabajar es en la presentación. Aunque este es un tema que queda fuera de los contenidos de este módulo, cabe indicar que una de las formas más simples para personalizar la presentación de nuestra aplicación es mediante la creación de un tema Bootstrap personalizado. Existen multitud de temas listos para descargar, o se pueden encontrar sitios que permiten editar nuestro propio tema desde cero, o bien partiendo de una plantilla. Algunos de estos sitios son, por ejemplo:

<https://bootstrap.build>

<https://pikock.github.io/bootstrap-magic>