Diseñó	Asignatura	Clave	Versión
HGA	Tecnologías Web Multiplataforma	MTW204	1.0

Practica: CRUD con Bootstrap y Jquery

Objetivo

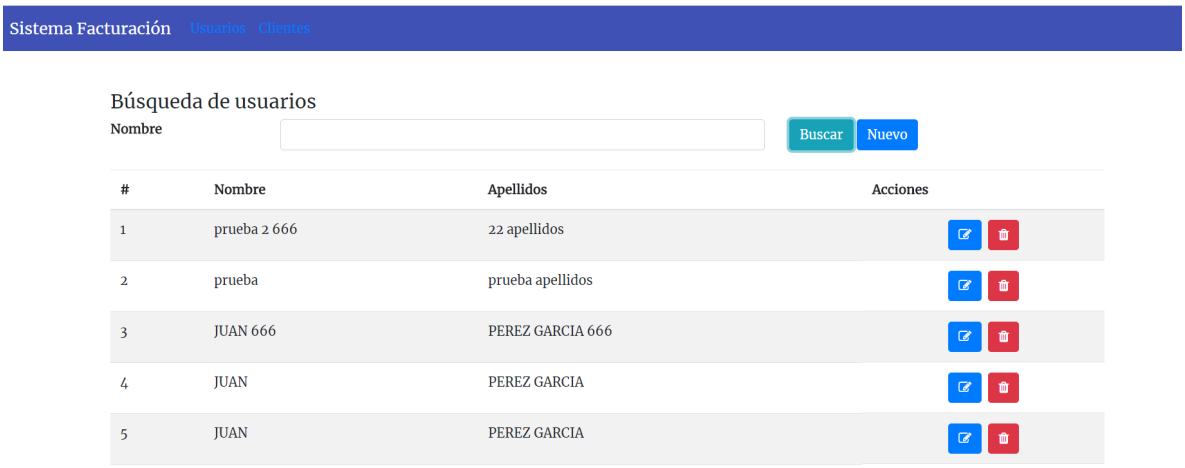
Desarrollar una aplicación web responsiva con Bootstrap y Jquery, la cual consumirá un servicio Rest para obtener, insertar, actualizar y eliminar la información.

Requisitos

- Conocimientos básicos de CodeIgniter
- Conocimientos básicos de consumo de servicios Rest
- Haber desarrollado la “Servicio Rest con CodeIgniter” o conocer la estructura del servicio .

Resultados esperados

En esta practica desarrollaras una app web responsiva con Bootstrap y Jquery que permita la manipulación de registros de usuarios. El resultado final será como el que se muestra en las siguientes pantallas.



Vista desde ancho de pantalla igual o superior al breakpoint md para listar usuarios

Diseñó	Asignatura	Clave	Versión
HGA	Tecnologías Web Multiplataforma	MTW204	1.0

Vista desde ancho de pantalla igual o superior al breakpoint md para edición de usuarios

Vista desde ancho pantalla menor al breakpoint md para listar usuarios

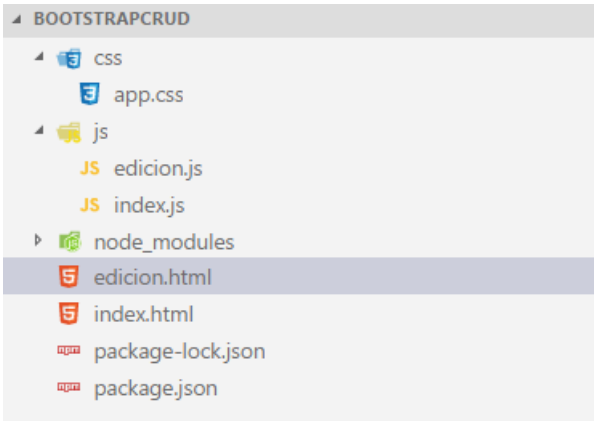
Diseñó	Asignatura	Clave	Versión
HGA	Tecnologías Web Multiplataforma	MTW204	1.0

Vista desde ancho de pantalla igual o superior al breakpoint md para edición de usuarios

Desarrollo de la practica

Creación del proyecto

Para esta práctica crea una carpeta con el nombre bootstrapcrud en donde se alojarán todos los archivos del proyecto. Utiliza el gestor npm para inicializar tu proyecto e instalar el servidor lite-server. Crea la siguiente estructura de archivos dentro del proyecto.



Dentro del archivo index.html colocaremos el listado de los usuarios. En el archivo edicion.html agregaremos el formulario para insertar o editar registros. Los archivos index.js y edicion.js es donde colocaremos el código Jquery para implementar la

funcionalidad de la aplicación. En el archivo app.css colocaremos los estilos generales del sitio, en especial los referentes a la barra de navegación.

Tanto en el archivo index.html como en el archivo edicion.html coloca el código del Starter Template que viene en la página de Bootstrap https://getbootstrap.com/docs/4.0/getting-started/introduction/ .

Una vez que lo tengas haremos un par de modificaciones para que quede de esta manera en los dos archivos :

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link href="https://fonts.googleapis.com/css?family=Merriweather" rel="stylesheet">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-G
    <link rel="stylesheet" href="css/app.css">
    <title>Sistema Facturación</title>
  </head>
  <body>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <!--<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qp
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QK
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6

  </body>
</html>
```

Crea la siguiente estructura de archivos dentro de tu proyecto.

Las modificaciones se encuentran resaltadas. Básicamente se hace referencia al archivo app.css como hoja de estilo y se ha cambiado el texto de la etiqueta title. Observa también que hemos cambiado la referencia a Jquery. El Starter Template de Bootstrap hace referencia a la versión “slim” de Jquery. Con este cambio hacemos referencia a la versión completa de Jquery con la finalidad de utilizar las funciones de ajax.

De igual forma en ambas páginas coloca el código necesario para contar con una barra de navegación responsiva :

```
<body>

  <header id="header-contenedor" >
        <!-- Fixed navbar -->
      <nav class="navbar navbar-expand-md">

        <span class="navbar-brand">Sistema Facturación</span>

        <button class="navbar-toggler" type="button" data-toggle="collapse"
        data-target="#navbarCollapse" aria-controls="navbarCollapse"
        aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon text-white">&#9776;</span>
        </button>

        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item ">
                <a class="nav-link" href="index.html">Usuarios</a>
                </li>
                <li class="nav-item">
                <a class="nav-link " href="#">Clientes</a>
                </li>
            </ul>
        </div>
      </nav>
  </header>
```

En el archivo app.css coloca los siguientes selectores .

```css
body{
    font-family: 'Merriweather', serif;

}

#header-contenedor{
    background-color: #3F51B5;
    color: white;
}

.btn.btn-programacionxyz{
    background-color: #FF4081;
    color: white;
    margin-left: 5px;
}

.btn.btn-programacionxyz:hover {
    background-color: #455a64;
}

.btn.btn-programacionxyz:active{
    background-color: #FF4081;
}

.nav-link:hover{
    color : white;
}

.activa{
    color: white;
}
```

Recuerda que esto se desarrollo en la practica “Barra de navegación Responsive con Bootstrap 4”. Las dos páginas deben lucir como se muestra en la siguiente imagen.

Creación del listado de usuarios

Procederemos a crear el listado de los usuarios. Abre el archivo index.html e inmediatamente después del header coloca un div con la clase container-fluid para que abarque todo el ancho de la pantalla.

```
<div class="container-fluid">

</div>
```

Dentro del div colocaremos todo el contenido de la pantalla. Coloca dentro del div container otro div con la clase row , en donde a su vez agregaremos un div para acomodar el titulo.

```
<br>
<br>
<div class="row">
   <div class="col-12 col-md-10 offset-md-1">
      <h4>Búsqueda de usuarios</h4>
   </div>
</div>
```

Observa que al div que contiene el titulo le hemos implementado las clases de col. De tal manera que cuando se detecte que el ancho de la pantalla esta en breakpoints igual o superiores a md, el div abarque 10 columnas (col-md-10) y deje un espacio proporcional a su izquierda de 1 (offset-md-1), con ello el div se “centrará” en dichos breakpoints. Cuando se detecten breakpoints menores a md el div abarcara todo el ancho de la pantalla.

Enseguida colocaremos una fila que contenga el código con la implementación del formulario de búsqueda.

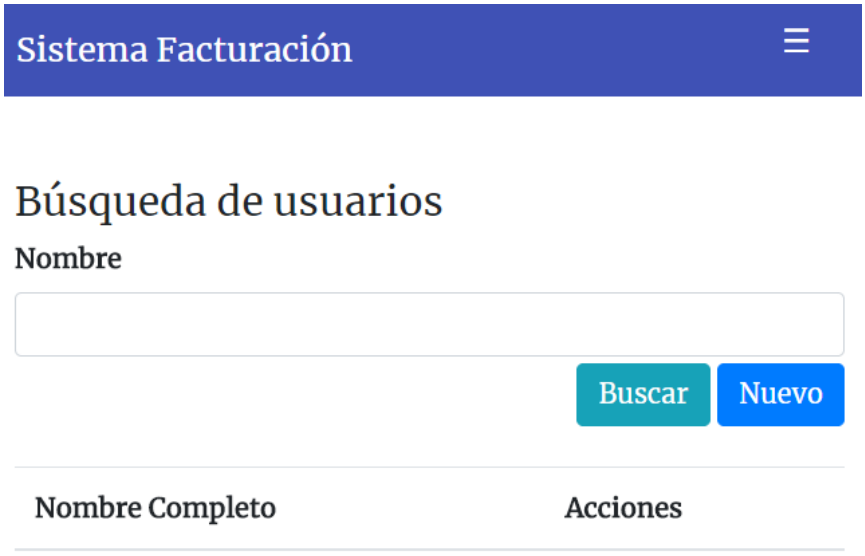
```
<div class="row">
    <!-- md o superior 10 columnas con offset de 1. Extra small o menores a md 12 columnas -->
    <div class="col-12 col-md-10 offset-md-1">
        <div class="row">
            <!-- extra small 12 columnas, md o superiores solo tomará 2 columnas -->
            <div class="col-12 col-md-2">
                <label><strong>Nombre</strong></label>
            </div>
            <!-- extra small 12 columnas, md o superiores tomará 6 columnas -->
            <div class="col-12 col-md-6">
                <input id="cliente" name="cliente" class="form-control" onpaste="return false" value="" maxlength="100">
            </div>

            <!--  extra small 12 columnas y contenido se coloca al final, md o superior 4 columnas y contenido al inicio -->
            <div class="col-12 col-md-4 d-flex justify-content-end justify-content-md-start">
                <button id="btnBuscar" name="btnBuscar" class="btn btn-info mt-1 mt-md-0">Buscar</button>
                <button id="btnNuevo" name="btnNuevo" class="btn btn-primary ml-1 mt-1 mt-md-0">Nuevo</button>
            </div>
        </div>

    </div>

</div>
```

Observa que se ha implementado las clases de columnas de tal manera el formulario tenga un comportamiento u otro según el breakpoint en el que se encuentre la pantalla. Por ejemplo para el breakpoint md igual o superior el div de columna principal abarcará 10 especias y en menores a 10 abarcará 12 columnas. El formulario deberá verse de esta forma en extra small :

Lo siguiente es agregar el código para la tabla.

```
<br>
<div class="row">
  <!-- En extra small 12 columnas, en md en adelate 10 columnas  -->
  <div class="col-12 col-md-10 offset-md-1">

    <!-- Clases table y table-striped para mejorar apariencia de la tabla  -->
    <table class="table table-striped">
        <thead>

          <tr>
            <!-- Se oculta en extra small, en md en adelante se muestra -->
            <th class="d-none d-md-table-cell">#</th>
            <th class="d-none d-md-table-cell">Nombre</th>
            <th class="d-none d-md-table-cell">Apellidos</th>

            <!-- Se muestra en extrasmall y se oculta en md en adelante -->
            <th class="d-table-cell d-md-none">Nombre Completo</th>
            <th>Acciones</th>
          </tr>

        </thead>

        <!-- Cuerpo de la tabla -->
        <tbody id="usuarios-table-body">

        </tbody>

    </table>

  </div>
</div>
```

Observa que a la tabla se le han implementado la clase table para mejorar su apariencia y la clase table-striped para hacer una alternación visual entre filas. Los títulos de las columnas se mostrarán u ocultarán según se vayan requiriendo acorde al breakpoint en el que se encuentre la pantalla.



Para ocultar o mostrar las columnas según el breakpoint de la pantalla utilizamos las clases d-none (para ocultar) y d-table-cell (para mostrar celda). El contenido de la tabla lo llenaremos por medio de Jquery, es por ello que le hemos colocado el id usuarios-table-body para hacer referencia a el desde código. Ahora procederemos a implementar la funcionalidad. En el archivo index.html coloca la referencia al archivo index.js .

```
<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<!--<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.j
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/pop
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.mi

<script src="js/index.js"></script>
```

Abre el archivo index.js para editarlo. Lo primero que haremos sera colocar los cuerpos de las funciones que utilizaremos .

```
const URL_API = 'http://programacion.xyz/mtw/204/crud/index.php/api/';

function editar(id)
{

}

function tblUsuarios(result)
{

}

$(document).ready(function(){

    $('#btnBuscar').click(function(){

    });

    $('#btnNuevo').click(function(){

    });

});
```

Hemos colocado la constante URL_API para hacer referencia a donde se encuentra el servicio web implementado con CodeIgniter. En este caso puedes utilizar tanto el que desarrollaste tu como el publicado en programacion.xyz.

La función editar(id) será llamada cuando deseemos editar un registro en especifico, básicamente nos redireccionará a la pantalla de edición proporcionandole como parámetro por la url el identificador del registro.

En la función tblUsuarios(result) se enviara la respuesta obtenida del servicio web con la lista de usuarios para armar la cadena html que se incoporarán a la tabla. En esta parte tendremos varias consideraciones también acorde al breakpoint en el que se encuentre la pantalla.

Dentro de document.ready, que se coloca una vez el código de la página se encuentra listo para ser usado, se definen las acciones a ejecutar cuando se ha lanzado el evento click tanto para el boton btnBuscar como para btnNuevo. Observa que con Jquery estamos haciendo referencia a los botones por medio de su id.

Dentro de la función para el evento click del botón btnNuevo colocaremos el código para que nos redireccione a la pagina edicion.html

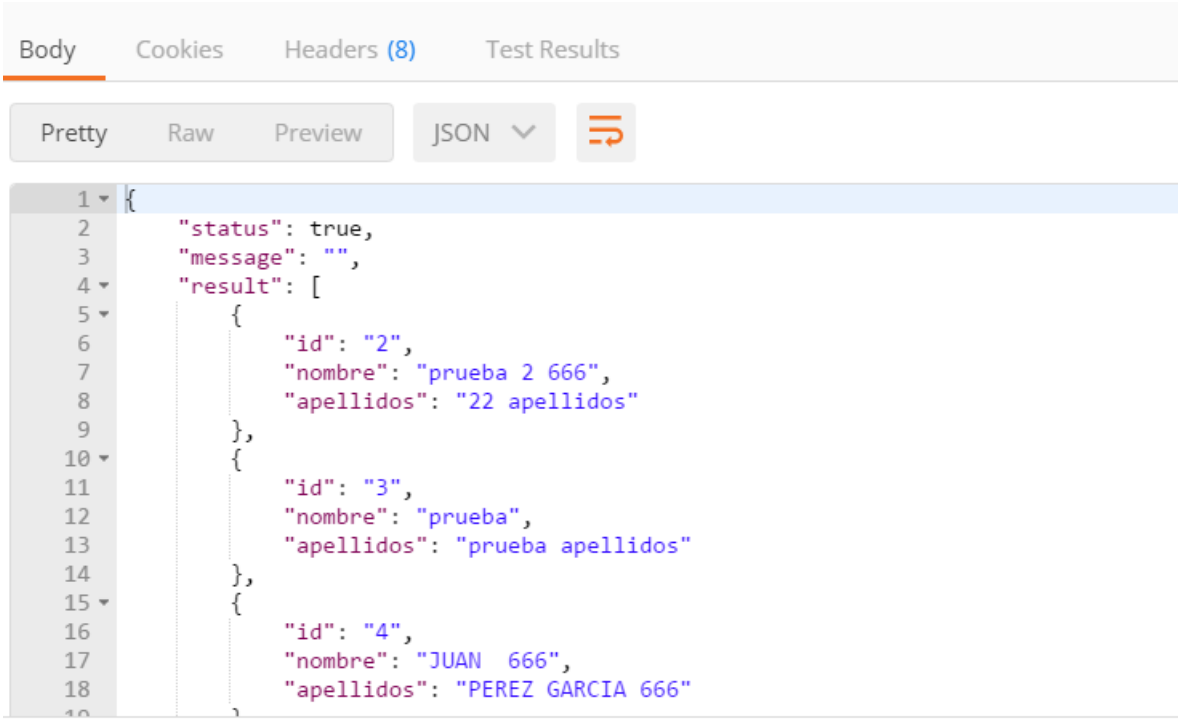
```
$('#btnNuevo').click(function(){
   window.location.href = 'edicion.html';
});
```

En el evento para el botón buscar colocaremos la llamada al servicio Rest para obtener toda la lista de usuarios.

```
$('#btnBuscar').click(function(){

    var url = URL_API+'usuarios/obtener';

    $.ajax({
        type: 'get',
        url: url,
        data: '',
        contentType: 'application/json;charset=utf-8',
        traditional: true,
        success: tblUsuarios,
        error : function(result){
            alert('Ocurrio un error al llamar el servicio')
        }
    });
});
```

Observa los parámetros que se le proporcionan, como type se le define get , como url proporcionamos la dirección con el controlador (usuarios) y el método para obtener los registros de usuarios (obtener). Cuando el retorno de la información es exitoso, es decir en success, hemos especificado que el resultado sea enviado a función tblUsuarios y en caso de error hemos implementado una función anónima para mandar mensaje al usuario.

Recuerda probar que tu servicio se encuentre funcionando correctamente, cerciorate que al llamarlo te retorne correctamente la información. La estructura retornada debe ser como se muestra en la siguiente imagen.



Ahora en la función tblUsuarios implementaremos el código para verificar que no hay error en la obtención de la información desde el servicio. En caso de error enviaremos un mensaje al usuario y en caso de éxito armaremos el contenido que colocaremos en la tabla. En ese contenido debemos de considerar las columnas que se mostrarán u ocultaran dependiendo del breakpoint en el que se encuentre la pantalla.

```
function tblUsuarios(result)
{
    if(result.status){

        var tbl = '';

        $.each(result.result, function(i, usuario){

            tbl+= '<tr>';
            tbl+= '<td class="d-none d-md-table-cell">'+(i+1)+'</td>';
            tbl+= '<td class="d-none d-md-table-cell">'+usuario.nombre+'</td>';
            tbl+= '<td class="d-none d-md-table-cell"">'+usuario.apellidos+'</td>';
            tbl+= '<td class="d-table-cell d-md-none">'+usuario.nombre+' '+usuario.apellidos+'<td>';
            tbl+= '<div class="d-flex justify-content-center">';
            tbl+= '<button class="btn btn-primary" onclick="editar('+usuario.id+')">Editar</button>';
            tbl+= '<button class="btn btn-danger ml-2">Eliminar</button>';
            tbl+= ' </div>';
            tbl+= '</td>';
            tbl+= '</tr>';
        });

        $('#usuarios-table-body').html(tbl);

    }else{
        alert('Ocurrio un error');
    }
}
```

Observa las partes resaltadas, básicamente estamos armando con la variable tbl el código html del contenido de la tabla iterando cada uno de los elementos que se encuentran en la propiedad result de la respuesta del servicio. Al igual que en el código html para los nombres de las columnas, estamos considerando mostrar o no columnas con las clases d-none y d-table-cell con sus correspondientes breakpoints. También por cada registro estamos armando en la cadena la llamada a la función editar a la cual se le proporcionará el id del registro. Al final el código html generado se le establece como html al tbody usuarios-table-body.

Por último coloca dentro de la función editar el redireccionamiento a la página edicion.html especificando el identificador del registro a modificar como parámetro en la url.

```
function editar(id)
{
    window.location.href = 'edicion.html?id='+id;
}
```

Con esto ya podrás listar los registros de usuario y ejecutar el redireccionamiento para

Creación del formulario de edición

Procederemos a crear la interfaz para la edición y creación de nuevos registros de usuarios. Dentro del archivo edicion.html coloca el código que se muestra a continuación.

```
<div class="container">

    <br>
    <br>
    <div class="row">
        <div class="col-12 col-md-6 offset-md-3">
            <div class="table-header"><h4 id="titulo">Registrar usuario</h4></div>
        </div>
    </div>

    <div class="row">
        <div class="col-12 col-md-6 offset-md-3">

            <form>
                <div class="form-group">
                    <label class="d-none d-md-block">Nombre</label>
                    <input type="text" class="form-control" id="nombre" placeholder="Nombre completo">
                </div>
                <div class="form-group">
                    <label class="d-none d-md-block">Apellidos</label>
                    <input type="text" class="form-control" id="apellidos" placeholder="Apellidos">
                </div>

                <div class="d-flex justify-content-end ">
                    <button type="button" id="btnGuardar" class="btn btn-primary">Guardar</button>
                    <button type="button" id="btnVolver" class="btn btn-default ml-2">Volver</button>
                </div>

            </form>

        </div>
    </div>

</div>
```

Verifica que es lo que sucede en los diferentes breakpoints que pueda adoptar el navegador sobre el que estés probando y observa su comportamiento. Después coloca la referencia al archivo edicion.js .

```
<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstra
<!--<script src="https://code.jquery.com/jquery
<script src="https://ajax.googleapis.com/ajax/l
<script src="https://cdnjs.cloudflare.com/ajax/
<script src="https://maxcdn.bootstrapcdn.com/bo

<script src="js/edicion.js"></script>
```

Abre el archivo edicion.js para colocar la funcionalidad de inserción de registros. Primero crearemos el cuerpo de las funciones que vamos a necesitar .

```
$(document).ready(function(){

    const URL_API = 'http://programacion.xyz/mtw/204/crud/index.php/api/';
    const url = document.URL;

    var id = 0;
    var parametros = [];

    var usuario = {
        id : 0,
        nombre : '',
        apellidos : ''
    };

    function cargaPagina()
    {


    }

    $('#btnGuardar').click(()=>{

    });

    $('#btnVolver').click(function(){

    });

    cargaPagina();

});
```

Hemos colocado dentro de document.ready y de nueva cuenta constante con la URL base del servicio Rest. Ademas hemos definido diferentes variables, por ejemplo el objeto usuario con las propiedades que enviaremos al servicio. Recuerda que acorte a la practica de Servicio Rest con CodeIgniter el controlador será el segmento de la url “usuario” y el método el segmento “insertar” a través del verbo HTTP POST.

Dentro de la función cargaPagina() coloca el siguiente código.

```
function cargaPagina()
{

    var urlApi = URL_API+'usuarios/obtener/';

    usuario.id = url.split('?').length > 1 ? url.split('?')[1].split('=')[1] : 0;

    $('#titulo').text('Registrar usuario');

    if(usuario.id > 0){

        $('#titulo').text('Editar usuario');


        urlApi += usuario.id;

        $.ajax({
            type: 'get',
            url: urlApi,
            data: '',
            contentType: 'application/json; charset=utf-8',
            success: function(data){
                console.log(data);
                $('#nombre').val(data.result[0].nombre);
                $('#apellidos').val(data.result[0].apellidos);

            },
            error : function(result){
                alert('Ocurrio un error al llamar el servicio')
            }
        });
    }

}
```

Cuando se le da clic al botón editar de un registro en la página index, nos redirecciona a la página edicion.html y le pasa como parámetro por la URL el id del registro.

En la función cargaPagina() estamos verificando si el id viene por la URL y en caso de ser así le proporcionamos el valor a la propiedad id del objeto usuario .

```
usuario.id = url.split('?').length > 1 ? url.split('?')[1].split('=')[1] : 0;
```

Por lo tanto en el código siguiente se verifica que si el id es mayor a 0 hace la solicitud al servicio para obtener la información del registro y mostrarlo en los controles de entrada.

```
urlApi += usuario.id;

$.ajax({
    type: 'get',
    url: urlApi,
    data: '',
    contentType: 'application/json; charset=utf-8',
    success: function(data){
        console.log(data);
        $('#nombre').val(data.result[0].nombre);
        $('#apellidos').val(data.result[0].apellidos);

    },
    error : function(result){
        alert('Ocurrio un error al llamar el servicio')
    }
});
```

La url es la misma para obtener todos los registros, pero le estamos concatenando el id del usuario para indicar al servicio que solo deseamos obtener un registro determinado. Si la llamada se realiza con éxito le asignamos a los campos de entrada nombre y apellidos los valores obtenidos. En ese caso como el servicio retorna una lista, retornara una lista con un solo elemento , es por eso que accedemos al indice 0.

Si todos es correcto, en la lista cuando des clic en editar a un registro te deberá cargar su información.

En el evento click para el botón btnGuardar coloca el siguiente código.

```
$('#btnGuardar').click(()=>{

    var method = 'post';
    var urlApi = URL_API+'usuarios/insertar';

    usuario.nombre =  $('#nombre').val();
    usuario.apellidos = $('#apellidos').val();

    if(usuario.id > 0){
        method = 'put';
        urlApi = URL_API+'usuarios/actualizar';
    }

    $.ajax({
        type: method,
        url: urlApi,
        data: JSON.stringify(usuario),
        contentType: 'application/json; charset=utf-8',
        success: function(result){
            window.location.href = 'index.html';
        },
        error : function(result){
            alert('Ocurrio un error al llamar el servicio')
        }
    });

});
```

Observa que por defecto el método es POST a menos de que se detecte que se esta realizando una modificación de registro. Lo mismo pasa con el método que se llamará al controlador del servicio Rest. Algo que también es importante considerar es el envió de la información al servicio. Observa en la linea data : JSON.stringify , en la cual se convierte el objeto que estamos proporcionando a un string json. Con esto, si todo ha sido ejecutado correctamente se redireccionara al index a través de la función anonima implementada para ello.

Por último coloca el código para el evento click del boton Volver .

```
$('#btnVolver').click(function(){
    window.location.href = 'index.html';
});
```

Eliminación de registros de usuario

Con lo que ya hemos visto deberás de implementar la funcionalidad para eliminar el usuario. Recuerda que el controlador del servicio es usuario/eliminar y debe usarse el verbo HTTP DELETE. Considera lo siguiente :

- Verifica la implementación del servicio. ¿Pasa el id por la url o por el cuerpo del mensaje?
- ¿ Como implementarás la eliminación , a través de un dialogo modal o una nueva pantalla?