UNIVERSIDAD SERGIO ARBOLEDA

TODOS A LA U
Fórmate digital

MasterClass Unidad 5

Desarrollo de Software

Introducción

- Se propone durante esta sesión realizar la implementación guiada de un front end haciendo uso de bootstrap, lo cual tendrá como objetivo apoyar el desarrollo de la parte final del proyecto.

Archivo index.html

- Lo primero que haremos es cargar las librerías externas:

CDN via jsDelivr

Skip the download with jsDelivr to deliver cached version of Bootstrap's compiled CSS and JS to your project.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.
```

If you're using our compiled JavaScript and prefer to include Popper separately, add Popper before our JS, via a CDN preferably.

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js" in
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"
```

La última versión estable se puede encontrar en https://getbootstrap.com/docs/5.3/getting-started/download/

Archivo index.html

- Agregamos también JQuery

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Libraries!</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">
    <link rel="stylesheet" href="css/myStyle.css">
</head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<!-- JavaScript Bundle with Popper -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
<body>
</body>
</html>
```

La última versión estable de bootstrap se puede encontrar en https://getbootstrap.com/docs/5.3/getting-started/download/
Jquery puede ser invocado desde https://developers.google.com/speed/libraries?hl=es-419

Archivo index.html

- Crearemos unos campos de tipo input para recibir la información, lo que haremos será crear un contenedor de los elementos en los que pondremos los input.
- Es valioso a cada input establecerle un id (que debe ser único) con el fin de poder capturar la información posteriormente.

La última versión estable de bootstrap se puede encontrar en https://getbootstrap.com/docs/5.3/getting-started/download/
Jquery puede ser invocado desde https://developers.google.com/speed/libraries?hl=es-419

Archivo index.html

Bootstrap tiene una gran cantidad de clases que establecerán la manera en que se visualizan los elementos.

```
<div class="container  p-2">
    <div class="myForm">
        <input id="idLibrary" type="hidden">
        <label for="nameLibrary" class="form-label">Nombre</label>
        <input id="nameLibrary" type="text"><br>
        <label for="targetLibrary" class="form-label">Objetivo</label>
        <input id="targetLibrary" type="text"><br>
        <label for="capacityLibrary" class="form-label">Capacidad</label>
        <input id="capacityLibrary" type="number"><br>
        <label for="descriptionLibrary" class="form-label">Descripcion</label>
        <textarea id="descriptionLibrary"></textarea><br>
        <select id="categorySelect" class="form-select">

        </select><br>
    </div>
</div>
```

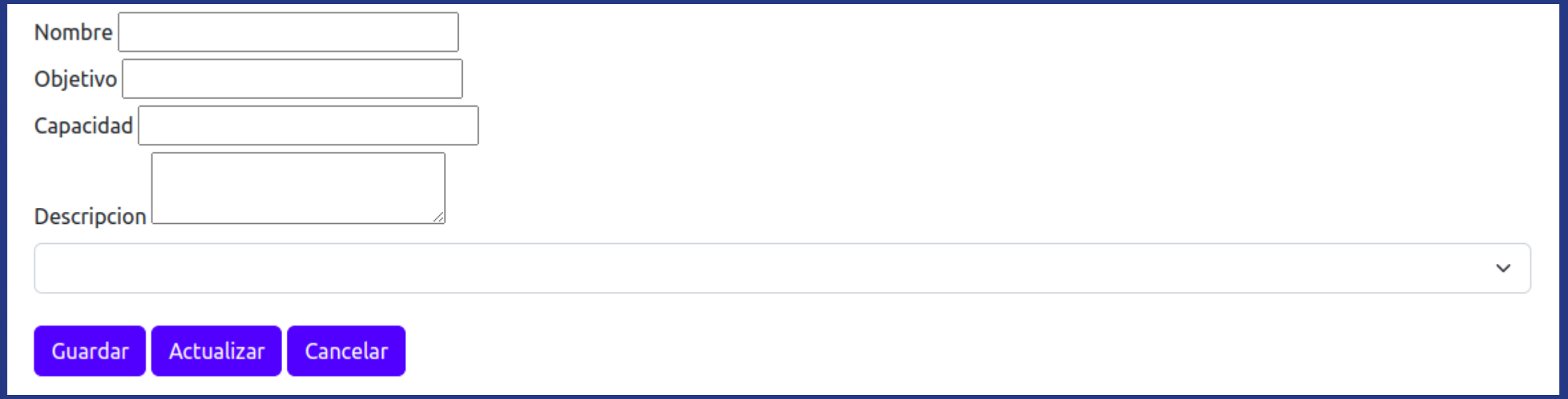
Archivo index.html

Agregamos los botones que servirán para realizar las acciones de guardar, actualizar y borrar.

```
<button type="button" class="btn btn-primary saveButtonJL" onclick="saveLib()">Guardar</button>
<button type="button" class="btn btn-primary updateButtonJL" onclick="updateLib()">Actualizar</button>
<button type="button" class="btn btn-primary updateButtonJL" onclick="cancelUpdateLib()">Cancelar</button>
```

Archivo index.html

En este momento se visualiza de la siguiente manera.



Funcionamiento con JS

Ahora crearemos el espacio en el que serán visualizados los elementos. Para ello preparamos un contenedor en el archivo index.html y haremos un llamado AJAX, desde allí pintaremos cada elemento del resultado en forma de tarjeta y lo agregaremos al contenedor.

```
 <hr>
<div class="row row-cols-4 row-cols-md-4 g-4"
id="results">


</div>
```

Index.html

```
function getLibs(){
    $.ajax({
        url : "api/Lib/all",
        type : 'GET',
        dataType : 'json',
        success : function(p) {
            console.log(p);
            $("#results").empty();
            let l="";
            for (let i=0;i<p.length;i++){
                    //UBICAREMOS CADA RESULTADO COMO UN CARD DENTRO DE RESULTS
            }
            $("#results").append(l);
        },
        error : function(xhr, status) {
            alert('ha sucedido un problema');
        },
        complete : function(xhr, status) {
            //  alert('Petición realizada');
        }
    });

}
```

MyScript.js

El archivo MyScript.js debe estar relacionado en el index.html para poder acceder a él

Funcionamiento con JS

En el archivo JS podemos mediante el carácter ` insertar código html como si se tratara de una cadena de texto. Este carácter que es la tilde que marca el acento grave, se utiliza para abrir y cerrar la cadena de texto. Es útil porque nos permite introducir saltos de línea sin tener que cerrar la cadena. Solo se cierra cuando se termine la cadena. Si se quiere utilizar variables, se debe utilizar la secuencia ${VARIABLE} donde la palabra variable es el nombre de la variable y esta al ejecutarse, será evaluada con su valor actual.

Veamos la manera como se inyectaría el código html desde el javascript:

MyScript.js

Funcionamiento con JS

```
let l="";
for (let i=0;i<p.length;i++){
   l+=`<div class="col">
         <div class="card"><div class="card-header">
                  <h5 class="card-title">${p[i].name}</h5>
               </div>
               <div class="card-body">
                  <p class="card-text">${p[i].description}</p>
                  <p class="card-text">Capacidad: ${p[i].capacity}</p>
                  <p class="card-text">Objetivo: ${p[i].target}</p>
                  <p class="card-text">Categoria: ${p[i].category.name}</p>
               </div>
               <div class="card-footer">
                    <div class="btn-group" role="group">
                     <button type="button" class="btn btn-outline-primary" onclick='getLibById(${p[i].id})'>Actualizar</button>
                     <button type="button" class="btn btn-outline-primary" onclick='deleteLibById(${p[i].id})'>Borrar!</button>
                   </div>

               </div>
         </div>
      </div>
      `;
}
```

MyScript.js

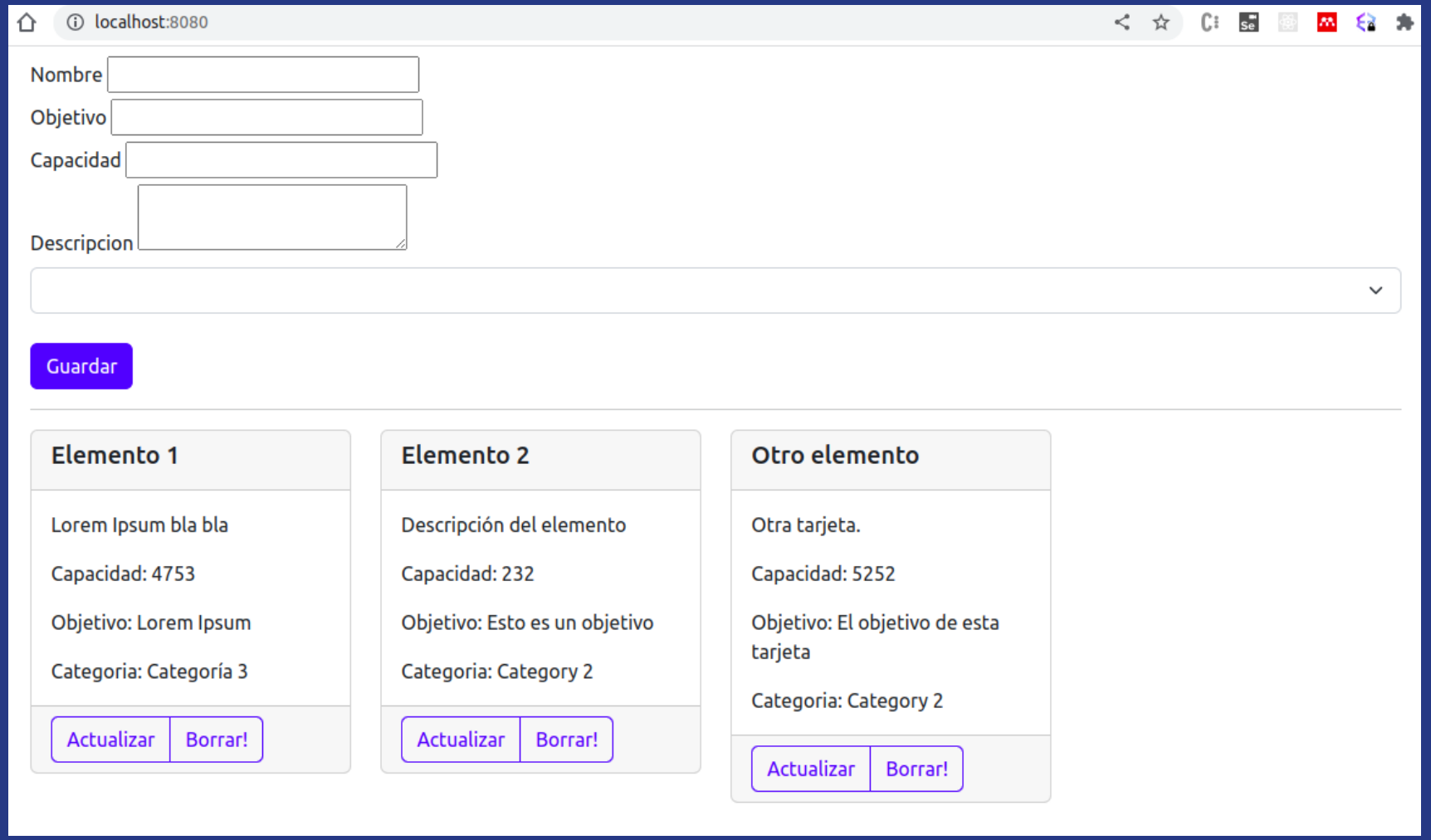
Funcionamiento con JS

El ciclo anterior, escrito dentro de el método que hace get, ubica las tarjetas con la información guardada, también escribe algunos botones, los cuales invocarán métodos que se utilizarán para la actualización y borrado de la información.

Veamos cómo luce la aplicación al ejecutarla:

MyScript.js

Funcionamiento con JS



Funcionamiento con JS

La captura de datos para el envío de Información lo realizaremos de la siguiente manera:

1) Creamos un método que capture la información de los campos visuales:

```
function getFrontLibData(){
    let k={
        id:$("#idLibrary").val(),
        name:$("#nameLibrary").val(),
        capacity:$("#capacityLibrary").val(),
        target:$("#targetLibrary").val(),
        description:$("#descriptionLibrary").val(),
        category:{
            id:$("#categorySelect").val()
        }
    }
    return k;
}
```

MyScript.js

Funcionamiento con JS

2) Creamos un método que haga un llamado AJAX tipo POST e invoque el método que captura la información.

```
function saveLib(){
    let data=getFrontLibData();
    data.id=null;
    let dataToSend=JSON.stringify(data);
    $.ajax({
        url : "api/Lib/save",
        type : 'POST',
        dataType : 'json',
        contentType:'application/json',
        data:dataToSend,
        success : function(p) {
            console.log(p);
            cleanFields();
            getLibs();

        },
        error : function(xhr, status) {
            alert('ha sucedido un problema');
        },
        complete : function(xhr, status) {
            //  alert('Petición realizada');
        }
    });
}
```

Es importante que la información sea convertida a formato JSON

MyScript.js

Funcionamiento con JS

3) Limpiamos los campos escritos en caso de haber sido exitosa la petición. El método es invocado desde el llamado ajax.

```
function cleanFields(){
    $("#idLibrary").val("");
    $("#nameLibrary").val("");
    $("#capacityLibrary").val("");
    $("#targetLibrary").val("");
    $("#descriptionLibrary").val("");
    $("#categorySelect").val("").change();
}
```

MyScript.js

Trabajo Propuesto

Desarrollar las funcionalidades de borrado y actualización.

TIP: Se pueden utilizar los campos del formulario para editar la información, para eso, al presionar el botón actualizar de cada tarjeta se puede enviar la información contenida a los elementos visuales del formulario; una vez precargados, se modifican y el botón actualizar deberá invocar un llamado ajax tipo PUT.

MyScript.js