INGENIERIA WEB II
FRAMEWORKS DE JAVASCRIPT: ANGULAR JS
ING. CARLOS SIMES

¿Qué es AngularJS?

AngularJS es un framework de aplicaciones web de código abierto que utilizan Javascript. Fue desarrollado originalmente por Misko Hevery y Adam Abrons.

Ahora es mantenido por Google. **Su última versión es la 1.8.2 y puede descargarse desde este enlace https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js**

Aclaración: Posterior a esta versión, Google redefinió todo el código creando Angular 2. Esta nueva versión es totalmente diferente conceptualmente a AngularJS, ya que está concebida para grandes proyectos web y su estudio requiere una asignatura completa dedicada a su aprendizaje.

Para proyectos pequeños y medianos, el uso de AngularJS es más que suficiente y en la actualidad uno de los frameworks más utilizados.

Por otro lado, Angular 2, actualmente va por la versión 2.10 que se denomina comercialmente Angular 10.

Definición de AngularJS como figura en su documentación oficial, es la siguiente:

AngularJS es un marco estructural para aplicaciones web dinámicas.

Permite usar **HTML** como idioma de plantilla y le permite **extender la sintaxis de HTML** para expresar los componentes de su aplicación de forma clara y sucinta.

El enlace de datos angular y la inyección de dependencias eliminan gran parte del código que actualmente tiene que escribir. Todo sucede dentro del navegador, por lo que es un socio ideal con cualquier tecnología de servidor.

Características

- AngularJS es un poderoso marco de desarrollo basado en JavaScript para crear RICH Internet Application (RIA).
- AngularJS proporciona a los desarrolladores opciones para escribir la aplicación del lado del cliente (usando JavaScript) usando el patrón MVC (Model View Controller).
- La aplicación escrita en AngularJS es compatible con varios navegadores.
- AngularJS gestiona automáticamente el código JavaScript adecuado para cada navegador.

- AngularJS es de código abierto, totalmente gratuito, y utilizado por millones de desarrolladores de todo el mundo. Está licenciado bajo la Licencia de Apache versión 2.0.

En general, AngularJS es un marco para crear aplicaciones web de pequeña y mediana escala y alto rendimiento, manteniéndolas muy fáciles de mantener.

Características principales

Las siguientes son las características principales más importantes de AngularJS:

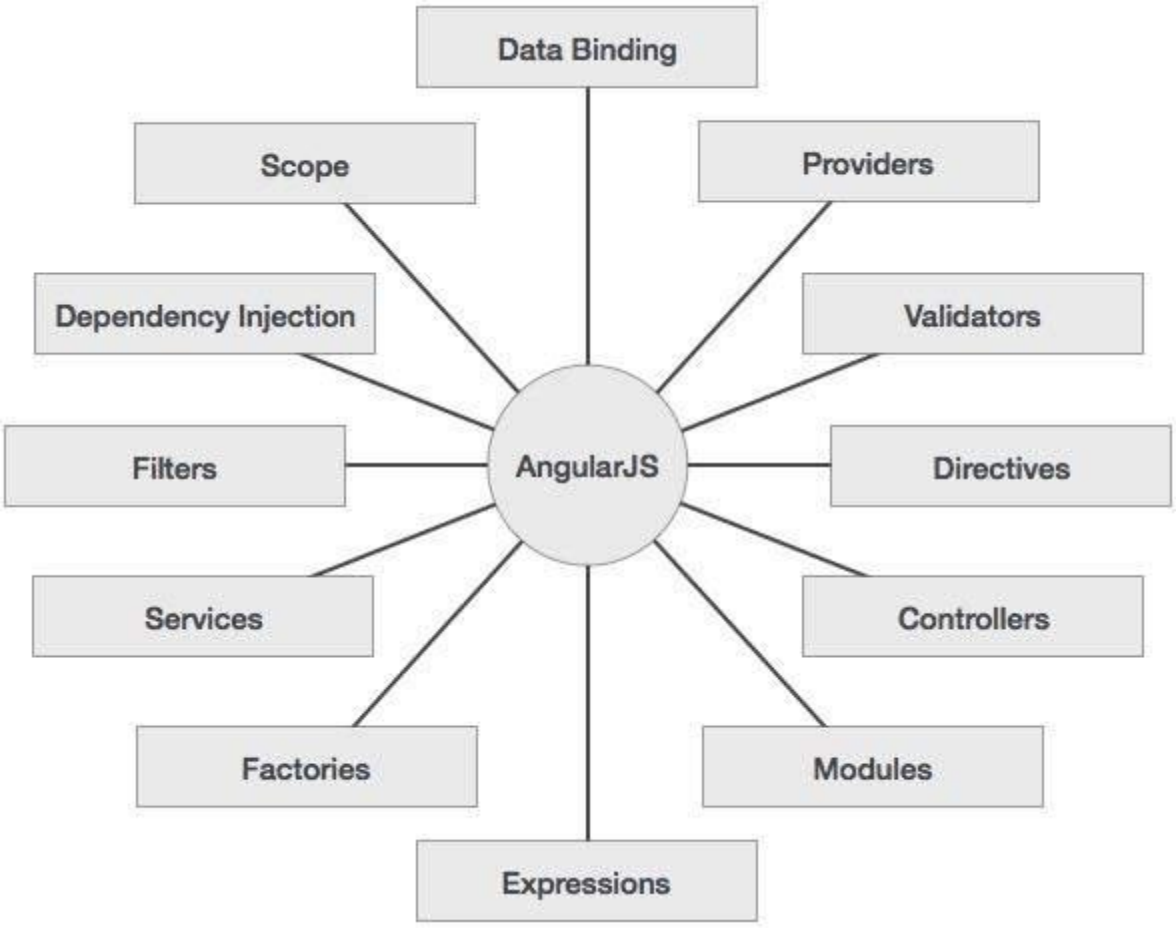
- **Enlace de datos** : es la sincronización automática de datos entre los componentes del modelo y de la vista.
- **Alcance** (**$scope**): Son objetos que se refieren al modelo, pensando este como los datos de la aplicación. Actúan como un pegamento entre el controlador y la vista.
- **Controlador** : son **funciones JavaScript** vinculadas a un ámbito determinado.
- **Servicios** - AngularJS viene con varios servicios integrados por ejemplo **$https** para hacer conexiones a un servidor web usando el protocolo HTTP/S. Este servicio lo emplearemos cuando querramos consumir datos desde un web server.
- **Filtros** : Permiten operar sobre los datos realizando ciertas funciones de uso habitual. Por ejemplo podemos aplicar un filtro a una matriz para que nos devuelva los datos de la misma ordenados de una determinada manera.
- **Directivas** - Las directivas son etiquetas especiales que podemos aplicar en elementos DOM (como elementos, atributos, css y más), para lograr ciertos comportamientos que HTML estandar no posee.
- Estas pueden usarse para crear etiquetas HTML personalizadas que sirvan como nuevos widgets personalizados (porciones de código más interfaces HTML con cierta funcionalidad específica y reusable). AngularJS tiene un conjunto de directivas integradas (ngBind, ngModel ...), a las que podemos agregar propias.
- **Plantillas** : son la vista renderizada con información del controlador y el modelo. Estos pueden ser un solo archivo (como index.html) o múltiples vistas en una página usando "partials".
- **Enrutamiento** - Es el concepto de cambiar de vista.
- **Modelo de vista Sea lo que sea** - MVC es un patrón de diseño para dividir una aplicación en diferentes partes (llamado Modelo, Vista y Controlador), cada uno con responsabilidades distintas. **AngularJS** no implementa MVC en el sentido tradicional, sino algo más cercano a MVVM (Model-View-ViewModel). El equipo de angular JS lo refiere humorísticamente como Model View Whatever.

- **Deep Linking** - Deep linking le permite codificar el estado de la aplicación en la URL para que pueda ser marcado. La aplicación puede restaurarse desde la URL al mismo estado.
- **Inyección de dependencia** - AngularJS tiene un subsistema de inyección de dependencias integrado que ayuda al desarrollador haciendo que la aplicación sea más fácil de desarrollar, comprender y probar.

No se preocupen si hasta aquí ciertos conceptos no le quedan muy claros aún.
A medida que avancemos iremos viendo en detalle cada uno de ellos y realizando ejemplos para que se entienda prácticamente el uso y la funcionalidad.

Conceptos

El siguiente diagrama muestra algunas partes importantes de AngularJS, las cuales veremos en detalle más adelante.



Ventajas de AngularJS

- AngularJS proporciona la capacidad de crear aplicaciones de **una sola página** de una manera muy limpia y fácil de mantener.

  Nota: Las aplicaciones de una sola página o SPA, son aplicaciones web en las que la navegación se sustituye por el reemplazo dinámico de componentes de manera tal de simular el funcionamiento y la dinámica de una aplicación de escritorio. Este comportamiento es fundamental a la hora de crear aplicaciones para dispositivos móviles. El mecanismo es similar a cuando hacíamos el reemplazo del contenido de un elemento por ejemplo un usando innerHTML, por otros, por ejemplo mostrar una tabla de resultados, y de esa forma evitar navegar de una página a otra. AngularJS implementa esta funcionalidad de una manera transparente que nos facilita enormemente el desarrollo de web apps.

- AngularJS proporciona capacidad de vinculación de datos a HTML dando así al usuario una experiencia rica y sensible
- El código AngularJS es comprobable (testeable) por unidad.
- AngularJS utiliza la inyección de dependencia y hace uso de la separación de responsabilidades.
- AngularJS proporciona **componentes reutilizables**.(widgets)
- Con AngularJS, el desarrollador escribe menos código y consigue más funcionalidad.
- En AngularJS, las vistas son páginas html puras, y los controladores escritos en Javascript hacen el procesamiento del negocio.

Además de todo, las aplicaciones de AngularJS pueden ejecutarse en todos los principales navegadores y teléfonos inteligentes, incluidos los teléfonos y tablets basados en **Android e iOS.**

Bueno hasta aquí, todo muy lindo .... ahora vamos a programar y ver que sale !!!

Arrancando con Angular ... Primera App

Lo primero que necesitamos para crear una aplicación con Angular es incluir sus librerías en la página HTML. Para ello podemos disponer de las mismas localmente, previa descarga, o directamente referenciarlas a una URL para que se descarguen el desplegar la página. Las librerías tienen solo 100k por lo que la descarga es instantánea.

La instrucción sería:

```
<head>
  <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
  </script>
</head>
```

Lo segundo es definir que porción de la página o toda va a ser parte de la aplicación Angular que estamos creando. Por ejemplo si queremos que toda la página sea parte de la aplicación, podríamos usar:

```
<html ng-app>
```

La directiva **ng-app** en la etiqueta html declara que toda la página será parte de la aplicación que estamos creando, por lo tanto la misma podrá acceder a todos los elementos contenidos en ella.

Si en cambio hubiésemos colocado un <div **ng-app** > nuestra aplicación solo podría acceder a los elementos dentro del div.

Esto nos permite trabajar de manera sectorizada y lograr acciones y resultados diversos dándonos la posibilidad de trabajar **"modularmente",** dado que podemos dividir la app en varias secciones y construir bloques o módulos con funcionalidades distintas si es necesario.

La página completa nos quedaría asi:

```
<!doctype html>
<html ng-app>

<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>
</head>

<body>
    <div>
        <label>Nombre:</label>
        <input type="text" ng-model="Nombre" placeholder="Ingresa tu Nombre">
        <hr />
        <h1>Hola {{Nombre}} !</h1>
    </div>
</body>
</html>
```

Si observamos la página, veremos que hemos declarado una directiva **ng-model,** básicamente aquí es simplemente una variable, pero podemos pensar el modelo como una **estructura de datos** más compleja, y luego hemos accedido a su valor.

En Angular para leer el valor de una variable se usa la notación **moustache {{** variable / modelo **}}**

También es posible usar la directiva **ng-bind** para enlazar datos. La página anterior nos quedaría así:

```
<html>
<head>
    <title>Primera aplicación Angular</title>
</head>
<body>
    <h1>Ejemplo 1</h1>

    <div ng-app="">
        <p>Ingresa tu Nombre: <input type="text" ng-model="Nombre"></p>
        <p>Hello <span ng-bind="Nombre"></span>!</p>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>

</body>
</html>
```

Aquí reemplazamos **{{ }}** por **ng-bind** y de esa manera vinculamos el modelo a una etiqueta.

Directivas de Angular

Directiva ng-app

La directiva **ng-app** inicia una aplicación AngularJS.

Define el elemento raíz. Inicializa o arranca automáticamente la aplicación cuando se carga la página web que contiene la aplicación AngularJS.

También se utiliza para cargar varios módulos AngularJS en la aplicación AngularJS.

Directiva ng-init

La directiva **ng-init** inicializa los datos de la aplicación AngularJS. Se utiliza para poner valores a las variables que se utilizarán en la aplicación. En el siguiente ejemplo, inicializaremos una serie de países. Utilizamos la sintaxis de JSON para definir la matriz de países. Básicamente ng-init nos permite definir variables y su valor inicial.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'}, {locale:'en-GB',name:'United Kingdom'}, {locale:'en-FR',name:'France'}]">
  ...
</div>
```

Directiva ng-model

Esta directiva vincula los valores de los datos de la aplicación AngularJS a los controles de entrada HTML. En el ejemplo siguiente, hemos definido un modelo denominado "nombre".

```
<div ng-app = "">
  ...
  <p>Ingresa tu Nombre: <input type = "text" ng-model = "Nombre"></p>
</div>
```

Observemos aquí que en vez de crear una variable de tipo name="Nombre " y luego tener que capturar el valor ingresado usando

Nombre=document.getElementById("Nombre").value <=> ng-model = "Nombre"

simplemente colocando ng-model=Nombre, Angular interpreta que estamos queriendo crear una variable llamada Nombre a la que le asignaremos el valor que se ingrese en el input. Genial !!!

Expresiones en AngularJS

Las expresiones se utilizan para enlazar los datos de la aplicación a html. Las expresiones se escriben dentro de llaves dobles como **{{expresión}}.**

Las expresiones se comportan de la misma manera que las directivas **ng-bind**. Las expresiones de la aplicación AngularJS son expresiones javascript puras y emiten los datos donde se utilizan.

Uso de números

```
<p>Gastos en Libros $ : {{costo * cantidad}} </p>
```

Uso de cadenas

```
<p>Hola {{Estudiante.Nombre + " " + Estudiante.Apellido}}!</p>
```

Uso del objeto

```
<p>DNI No: {{Estudiante.DNI}}</p>
```

Uso de array

```
<p>Calificaciones: {{Calificaciones[3]}}</p>
```

Ejemplo

El siguiente ejemplo mostrará todas las expresiones mencionadas anteriormente.

Ejemplo02.html

```
<html>
<head>
    <title>Expresiones en Angular</title>
</head>
<body>
    <h1>Ejemplo 2</h1>
    <div ng-app=""
        ng-
init="cantidad = 12;costo = 30; Estudiante = {Nombre:'Gustavo',Apellido:'Paredes',DNI:101};
Calificaciones = [8,9,7.5,7.3,6]">
```

```
        <p>Hola {{Estudiante.Nombre + " " + Estudiante.Apellido}} !</p>
        <p>Gastos en Libros $: {{costo * cantidad}} </p>
        <p>DNI No: {{Estudiante.DNI}}</p>
        <p>Calificaciones: {{Calificaciones[3]}}</p>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>
</body>
</html>
```

AngularJS - Controladores

La aplicación AngularJS se basa principalmente en el uso de controladores para controlar el flujo de datos en la aplicación. Un controlador se define utilizando la directiva **ng-controller**.

Un controlador es un objeto JavaScript que contiene **atributos / propiedades y funciones.** Cada controlador acepta **$scope** como un parámetro que se refiere a la aplicación / módulo que el controlador debe controlar.

```
<div ng-app = "" ng-controller = "studentController">
  ...
</div>
```

Aquí hemos declarado un controlador **studentController** usando la directiva ng-controller. Este controller, controlará todos los elementos contenidos dentro de la app que tiene el alcance del div donde lo hemos colocado. Aquí es importante comprender que el controller no podrá acceder a ningun elemento por fuera del div en que lo hemos colocado. Esta es la base para generar una estructura modular con diferentes controladores ubicados en diferentes secciones de nuestra app.

Como paso siguiente, definiremos el **studentController** de la siguiente manera:

En un script dentro de nuestra página en un archivo externo colocaremos

```
<script>
  function studentController($scope) {
    $scope.student = {
      firstName: "Carlos",
      lastName: "Torres",

      fullName: function() {
```

```
        var studentObject;
        studentObject = $scope.student;
        return studentObject.firstName + " " + studentObject.lastName;
      }
    };
  }
</script>
```

- **studentController** definido como una función JavaScript con **$scope** como argumento.
- **$ scope** se refiere al conjunto de datos que puede manejar la aplicación que usa la función **studentController.**
- $ scope.**student** es un objeto que incluimos dentro del cojunto global de datos $scope del controlador **studentController**.
- **firstName y lastName** son dos propiedades del objeto **$scope.student**.

  Hemos pasado los valores predeterminados a ellos.
- **fullName** es una función dentro del objeto **$scope.student** cuya tarea es devolver el nombre combinado.
- En la función **fullName** estamos recibiendo el objeto **estudiante** y luego devolviendo el nombre combinado.
- Como nota, también podemos definir el objeto controlador en un archivo JS separado y referirlo a la página html.

Ahora podemos usar la propiedad estudiante de studentController usando **ng-model** o usando expresiones de la siguiente manera.

```
Ingresa el Nombre: <input type = "text" ng-model = "student.firstName"><br>
Ingresa el Apellido: <input type = "text" ng-model = "student.lastName"><br>
<br>
Estas ingresando: {{student.fullName()}}
```

- Hemos limitado student.firstName y student.lastname a dos inputs de entrada.
- Hemos limitado student.fullName () a HTML.
- Ahora, siempre que escriba algo en los inputs de entrada de nombre y apellido, puede ver el nombre completo actualizándose automáticamente.

Ejemplo

El siguiente ejemplo mostrará el uso del controlador.

Ejemplo03.html

```
<html>

<head>
    <title>Controladores en Angular</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">
    </script>
</head>

<body>
    <h2>Ejemplo de Controlador</h2>
    <div ng-app="App" ng-controller="ControladorEstudiante">
        Ingrese el Nombre: <input type="text" ng-model="Estudiante.Nombre"><br><br>
        Ingrese el Apellido: <input type="text" ng-model="Estudiante.Apellido"><br>
        <br>

        Usted está Ingresando: {{Estudiante.NombreCompleto()}}
    </div>

    <script>
        var App = angular.module("App", []);
        App.controller('ControladorEstudiante', function ($scope) {
            $scope.Estudiante = {
                Nombre: "Carlos",
                Apellido: "Torres",

                NombreCompleto: function () {
                    var objetoEstudiante;
                    objetoEstudiante = $scope.Estudiante;
                    return objetoEstudiante.Nombre + " " + objetoEstudiante.Apellido;
                }
            };
        });
    </script>
</body>

</html>
```

Veamos ahora el ejemplo de la típica Calculadora hecha en Angular

CalculadoraAngular.html

```
<html>
<head>
    <title>Controladores en Angular</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>
</head>
<body>
    <h2>Calculadora AngularJS</h2>

    <div ng-app="CalculatorApp" ng-controller="CalculatorController">

        <p><input type="number" ng-model="Numbers.a"></p>
        <p><input type="number" ng-model="Numbers.b"></p>

        <input type="radio" value="+" name="op" ng-model="operator">Suma <br>
        <input type="radio" value="-" name="op" ng-model="operator">Resta <br>
        <input type="radio" value="*" name="op" ng- model="operator">Multiplicacion<br>
        <input type="radio" value="/" name="op" ng-model="operator">Division<br><br>

        <p>{{ result() }}</p>
    </div>

    <script>
        angular.module('CalculatorApp', [])
            .controller('CalculatorController', function ($scope) {
                $scope.result = function () {
                    if ($scope.operator == '+') {
                        return $scope.Numbers.a + $scope.Numbers.b;
                    }
                    if ($scope.operator == '-') {
                        return $scope.Numbers.a - $scope.Numbers.b;
                    }
                    if ($scope.operator == '*') {
                        return $scope.Numbers.a * $scope.Numbers.b;
                    }
                    if ($scope.operator == '/') {
                        return $scope.Numbers.a / $scope.Numbers.b;
                    }
                };
            });
    </script>
</body>
</html>
```

Como vemos en el ejemplo, Angular gestiona los eventos en forma transparente y actualiza las vistas ante la interacción del usuario.

Filtros en AngularJS

Los filtros se utilizan para modificar los datos y pueden ser incluidos en una expresión o las directivas con el carácter pipa |.

A continuación se muestra la lista de filtros utilizados.

uppercase convierte un texto en mayúsculas.

lowercase convierte un texto en minúsculas.

currency formatos de texto en un formato de moneda.

filter filtrar la matriz a un subconjunto de ella basada en criterios proporcionados.

orderby ordena una matriz basado en criterios proporcionados.

Ejemplo

El siguiente ejemplo mostrará todos los filtros mencionados anteriormente.

Filtros.html

```
<html>
<head>
    <title>Filtros en Angular</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>
</head>
<body>
    <h2>Filtros en Angular</h2>
    <div ng-app="mainApp" ng-controller="ControladorEstudiante">
        <table border="0">
            <tr>
                <td>Ingresa el Nombre:</td>
                <td><input type="text" ng-model="Estudiante.Nombre"></td>
            </tr>
            <tr>
                <td>Ingresa el Apellido: </td>
                <td><input type="text" ng-model="Estudiante.Apellido"></td>
            </tr>
            <tr>
                <td>Ingresa el Costo de la Matrícula: </td>
                <td><input type="text" ng-model="Estudiante.CostoMatricula"></td>
            </tr>
            <tr>
                <td>Ingresa la Materia: </td>
                <td><input type="text" ng-model="Materias"></td>
            </tr>
        </table>
```

```
        <br />
        <table border="0">
            <tr>
                <td>Nombre Completo en Mayúsculas: </td>
                <td>{{Estudiante.NombreCompleto() | uppercase}}</td>
            </tr>
            <tr>
                <td>Nombre Completo en Minúsculas: </td>
                <td>{{Estudiante.NombreCompleto() | lowercase}}</td>
            </tr>
            <tr>
                <td>Costo Matricula : </td>
                <td>{{Estudiante.CostoMatricula | currency}}
                </td>
            </tr>
            <tr>
                <td>Materias:</td>
                <td>
                    <ul>
                        <li ng-
repeat="materia in Estudiante.Materias | filter: Materias |orderBy:'Nota'">
                            {{ materia.Nombre + ', Nota:' + materia.Nota }}
                        </li>
                    </ul>
                </td>
            </tr>
        </table>
    </div>
    <script>
        var mainApp = angular.module("mainApp", []);
        mainApp.controller('ControladorEstudiante', function ($scope) {
            $scope.Estudiante = {
                Nombre: "Carlos",
                Apellido: "Simes",
                CostoMatricula: 1350,
                Materias: [
                    { Nombre: 'Fisica', Nota: 7 },
                    { Nombre: 'Quimica', Nota: 8 },
                    { Nombre: 'Matemática', Nota: 6.5 }
                ],
                NombreCompleto: function () {
                    var ObjetoEstudiante;
                    ObjetoEstudiante = $scope.Estudiante;
                    return ObjetoEstudiante.Nombre + " " +
                        ObjetoEstudiante.Apellido;
                }
            };
        });
    </script>
</body>
</html>
```

Ahora vemos un ejemplo de Tablas … como las tablas tienen una estructura repetitiva se presta para usar la directiva **ng-repeat** para generarla.

Tablas.html

```
<html>
<head>
    <title>Angular JS Table</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>

    <style>
        table,
        th,
        td {
            border: 1px solid grey;
            border-collapse: collapse;
            padding: 5px;
        }

        table tr:nth-child(odd) {
            background-color: #f2f2f2;
        }

        table tr:nth-child(even) {
            background-color: #ffffff;
        }
    </style>

</head>
<body>
    <h2>Ejemplo de Tablas en Angular</h2>
    <div ng-app="mainApp" ng-controller="ControladorEstudiante">

        <table border="0">
            <tr>
                <td>Ingresa el Nombre:</td>
                <td><input type="text" ng-model="Estudiante.Nombre"></td>
            </tr>

            <tr>
                <td>Ingresa el Apellido: </td>
                <td>
                    <input type="text" ng-model="Estudiante.Apellido">
                </td>
            </tr>

            <tr>
                <td>Name: </td>
                <td>{{Estudiante.NombreCompleto()}}</td>
            </tr>
```

```
            <tr>
                <td>Materia:</td>

                <td>
                    <table>
                        <tr>
                            <th>Nombre</th>.
                            <th>Notas</th>
                        </tr>

                        <tr ng-repeat="materia in Estudiante.Materias">
                            <td>{{ materia.Nombre }}</td>
                            <td>{{ materia.Nota }}</td>
                        </tr>

                    </table>
                </td>
            </tr>
        </table>
    </div>
    <script>
        var mainApp = angular.module("mainApp", []);
        mainApp.controller('ControladorEstudiante', function ($scope) {
            $scope.Estudiante = {
                Nombre: "Carlos",
                Apellido: "Simes",
                CostoMatricula: 1350,

                Materias: [

                    { Nombre: 'Fisica', Nota: 7 },
                    { Nombre: 'Quimica', Nota: 8 },
                    { Nombre: 'Matemática', Nota: 6.5 },
                    { Nombre: 'Informática', Nota: 9.5 },
                    { Nombre: 'Bases de datos', Nota: 8 }
                ],

                NombreCompleto: function () {
                    var ObjetoEstudiante;
                    ObjetoEstudiante = $scope.Estudiante;
                    return ObjetoEstudiante.Nombre + " " + ObjetoEstudiante.Apellido;
                }
            };
        });
    </script>
</body>
</html>
```

Manipulación del DOM

Angular posee una serie de directivas que permiten manipular el DOM

Las principales son:

ng-disabled

Agregue el atributo **ng-disabled** a un botón HTML y pásele un modelo. Vincule el modelo a una casilla de verificación y vea la variación.

<input type = "checkbox" ng-model = "enableDisableButton">Disable Button

<button ng-disabled = "enableDisableButton">Click Me!</button>

ng-show

Agregue el atributo ng-show a un botón HTML y pásele un modelo. Vincule el modelo a una casilla de verificación y vea la variación.

<input type = "checkbox" ng-model = "showHide1">Show Button

<button ng-show = "showHide1">Click Me!</button>

ng-hide

Agregue el atributo ng-hide a un botón HTML y pase un modelo. Vincule el modelo a una casilla de verificación y vea la variación.

<input type = "checkbox" ng-model = "showHide2">Hide Button

<button ng-hide = "showHide2">Click Me!</button>

ng-click

Agregue el atributo ng-click a un botón HTML y actualice un modelo. Enlazar el modelo a html y ver la variación.

Total click: {{ clickCounter }}

<button ng-click = "clickCounter = clickCounter + 1">Click Me!</button>

Aquí un ejemplo de todas ellas:

Dom.html

```
<html>
<head>
    <title>Manipulacion del DOM</title>
</head>
<body>
    <h2>Manipulación del DOM</h2>
    <div ng-app="">
        <table border="0">
            <tr>
                <td>
                    <input type="checkbox" ng-model="enableDisableButton">
                    Deshabilitar Botón
                </td>
                <td>
                    <button ng-disabled="enableDisableButton" ng-
click="clickCounter = clickCounter + 1">
                        Click !
                    </button>
                </td>
            </tr>
            <tr>
                <td>
                    <input type="checkbox" ng-model="showHide1">Show Button</td>
                <td><button ng-show="showHide1" ng-
click="clickCounter = clickCounter + 1">Click Me!</button></td>
            </tr>
            <tr>
                <td><input type="checkbox" ng-model="showHide2">Hide Button</td>
                <td><button ng-hide="showHide2" ng-
click="clickCounter =            clickCounter + 1">Click !</button></td>
            </tr>
            <tr>
                <td>
                    <p>Total de clicks: {{ clickCounter }}</p>
                </td>
                <td><button ng-click="clickCounter = clickCounter + 1">
                        Click !</button></td>
            </tr>
        </table>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></sc
ript>
</body>
</html>
```

Esto es todo como una pequeña introducción de lo que se puede hacer con Angular.

El próximo Lab vemos como consumir servicios REST desde un WEB SERVER, que es lo que necesitarán para el final de la asignatura.