Contenido

[1. ANGULAR JS 1](#_Toc7075301)

[1.1. UN POCO DE HISTORIA 1](#_Toc7075302)

[1.2. EL RETO DE HOY 2](#_Toc7075303)

[1.3. JAVASCRIPT + HTML5 2](#_Toc7075304)

[1.4. FRAMEWORKS 3](#_Toc7075305)

[1.5. ANGULARJS 3](#_Toc7075306)

[1.6. CARACTERISTICAS DE ANGULARJS 4](#_Toc7075307)

[1.7. ELEMENTOS DE ANGULARJS 4](#_Toc7075308)

[1.7.1. asdasda 4](#_Toc7075309)

[1.8. ANGULARJS Y EL MODELO MVC 4](#_Toc7075310)

# ANGULAR JS

## UN POCO DE HISTORIA

Desde sus inicios la programación se ha visto determinada por la frase “El software sigue al hardware”, lo que simplemente quiere decir que los programadores crean aplicaciones para aquellas máquinas en las que los van a utilizar. Aunque parece obvio, esto indica que los programadores están limitados a las posibilidades que éstas.

Cuando empezó la informática, y aún hacia los años 60, las máquinas eran arcaicas y por lo tanto eran muchas las limitaciones para los programadores.

Conforme la informática fue avanzando, siguieron llegando otros ordenadores, sin embargo, aún no existía el internet y las redes locales por lo que el trabajo de los programadores era relativamente sencillo puesto que solamente debían preocuparse por lo que pasaba entro de los ordenadores y no tenían que preocuparse por otras cosas. Además las opciones para crear los programas no eran muy amplias, por lo que comúnmente los programadores se veían obligados a cierto lenguaje de programación para cada sistema operativo.

Conforme fueron avanzando los ordenadores apareció el internet y estos empezaron a conectare entre sí, aparecieron los servidores y las terminales dejaron de ser tontas. Esto permitió la aparición de lenguajes de programación como HTML, CSS, JavaScript y otros que podían realizar tareas sencillas.

## EL RETO DE HOY

Hoy en día el panorama ha cambiado en gran manera. Gracias a los bajos costos de las comunicaciones han aumentado los grandes centros de procesamiento de datos a los que inclusive las pequeñas empresas tienen acceso sin mayores dificultades.

Se ha terminado el “reinado de Windows”: hoy existen varios sistemas operativos u se usan para todo tipo de cosas. Los equipos Mac no solo los usan los diseñadores, los sistemas Linux no son terrenos únicamente para los desarrolladores, por el contrario, cualquier tipo de persona puede utilizar estos sistemas para todo tipo de propósitos. Por esta razón es que los desarrolladores no se pueden ver centrados en un solo sistema y limitar su servicio a un solo tipo de plataforma.

Otra cara del reto actual de los programadores tiene que ver con la larga lista de necesidades en el área de la informática y, por supuesto, la igual de larga lista de herramientas que se pueden utilizar para cubrir estas necesidades. Para los programadores puede ser un punto crítico el decidir entre programar para web o para escritorios, al igual que programar nativamente para un sistema o programar multiplataforma. La industria da soluciones y aporta todo tipo de alternativas como Java, .net, Python, RoR, Objective-C y otros más arcaicos como Cobol, pero es inevitable plantearse cuál de ellos es más adecuado para resolver los problemas.

## JAVASCRIPT + HTML5

En un entorno con tan amplia variedad de necesidades y de herramientas para cubrirlas, se hizo necesario encontrar una solución que pudiera ser común a todos los sistemas permitiendo así poder realizar un desarrollo que se adapte a todo tipo de sistemas y dispositivos que puedan llegar a aparecer.

Uno de los pocos lenguajes de programación que cumple con estas características es JavaScript. Para el desarrollo de aplicaciones para iPhone están disponibles Objective C, JavaScript, o Swift. Para el desarrollo de juegos o aplicaciones de escritorio que se ejecuten en Mac o Windows, se cuenta con más opciones incluyendo C, Java, o JavaScript. Para realizar juegos en linea interactivos o aplicaciones web, es necesario utilizar JavaScript. Por esto JavaScript es una de las prioridades para los que comienzan a estudiar programación. No solo por lo sencillo que resulta, sino también por lo útil que se muestra en el ámbito profesional. JavaScript es uno de los lenguajes estándar de Internet y una interesante alternativa para realizar aplicaciones móviles o de escritorio.

El auge de JavaScript, fue más notorio con el surgimiento de HTML5, puesto que, aunque JavaScript ya existía antes de este, tomo mucha popularidad al permitir el acceso a APIs que extendieron las funcionalidades de la web, como el hacer Drag and Drop, tener un mecanismo de almacenar datos en el propio navegador del usuario, poseer un lienzo para poder dibujar en 2d e incluso en 3d, poder hacer conexiones bidireccionales con un servidor y muchas otras funcionalidades que aún han seguido aumentando.

Aunque es notoria su capacidad de desarrollo multiplafatorma, JavaScript + html5 sigue compitiendo con las soluciones nativas de cada sistema, ya que, por ejemplo, en términos de rendimiento es muy difícil que puedan llegar a equipararse. El lenguaje nativo, además, siempre tendrá mayor facilidad para acceder a las características y funcionalidades del sistema o dispositivo.

Además de esto, JavaScript + html5 también compiten con los lenguajes clásicos como C, PHP y Java en los cuales ya existe mucho trabajo realizado en forma de librerías disponibles para los programadores.

## FRAMEWORKS

JavaScript se ha convertido en uno de los lenguajes de programación más populares en internet. Como resultado de esto existen hoy en día un amplio conjunto de frameworks y librerías de ámbito general, que mejoran las prácticas de programación con JavaScript.

Por definición Un framework de software es una abstracción en la que el software que proporciona una funcionalidad genérica cambia de forma selectiva mediante un código adicional escrito por el usuario. El framework de JavaScript es un framework de aplicación escrito en JavaScript donde los programadores pueden manipular las funciones y usarlas para su comodidad.

La elección de un framework de JavaScript al momento de desarrollar un nuevo proyecto puede ser un momento crítico. Muchas son las opciones para elegir y cada año hay un nuevo grupo de ellos que empieza a ganar popularidad, sin que los ya existentes pierdan su fuerza. No obstante, la popularidad de un framework no determina el que sea el adecuado para un proyecto, y por esto el método de selección del framework a utilizar debe estar basado en los requisitos y exigencias del proyecto a desarrollar.

El objetivo de los frameworks no es tanto ofrecer al desarrollador una gran cantidad de funcionalidades ya lista, con implementaciones de piezas comunes en aplicaciones web, sino que están diseñados para ofrecer al equipo de trabajo un esquema de desarrollo sólido. Los frameworks obligan al desarrollador, en mayor o menor medida, a trabajar de un modo específico, que está probado facilita la creación de aplicaciones complejas, con un código fácilmente mantenible.

## ANGULARJS

AngularJs es probablemente uno de los frameworks de JavaScript, del lado del cliente, más populares disponibles en la actualidad. Se inició originalmente como un proyecto de google, pero que ha pasado a ser un framework de código abierto cuyo objetivo principal es la simplificación. Se destaca en la creación de aplicaciones web dinámicas de una sola página (SPA) y es compatible con la estructura de programación Modelo-Vista-Controlador (MVC).

AngularJs está completamente basado en HTML y JavaScript, por lo que no es necesario aplicar otra sintaxis o idioma. Este cambia el HTML estático a HTML dinámico. Amplía la capacidad del HTML al agregar atributos y componentes incorporados y también brinda la capacidad de crear atributos personalizados mediante JavaScript simple.

AngularJs aborda el problema de crear aplicaciones web dinámicas, lo que permite al desarrollador ampliar la funcionalidad de HTML al otorgarles la capacidad de crear nuevas construcciones con las conocidas directivas de AngularJs. En un nivel alto, las directivas son marcadores en un elemento DOM (como un atributo, nombre de elemento y comentario o clase CSS) que le dicen al compilador HTML de AngularJS que adjunte un comportamiento específico a ese elemento DOM. De esta manera, se abstrae la manipulación complicada del DOM, reduciéndola a elementos simples que se pueden incrustar directamente en una plantilla HTML. El ejemplo más famoso de esto es el enlace bidireccional de datos (doble-binding), una tarea pesada de código que ahora se relega a simplemente una expresión.

## CARACTERISTICAS DE ANGULARJS

Angular tiene las siguientes características claves que lo convierten en uno de los frameworks poderosos en el mercado.

1. **MVC** – Este framework se basa en el famoso concepto de MVC (Modelo-Vista-Controlador). Este patrón se basa en dividir la capa de lógica de negocios, la capa de datos y la capa de presentación en secciones separadas. La división en diferentes secciones se realiza para que cada una pueda ser manejada más fácilmente.
2. **Enlace de modelo de datos** – No es necesario escribir un código especial para vincular datos a los controles HTML. Esto se puede hacer mediante AngularJs simplemente agregando algunos fragmentos de código.
3. **Menos escritura de código** – Para lleva a cabo la manipulación del DOM, es necesaria una gran cantidad de código a la hora de diseñar cualquier aplicación. Pero AngularJs permite esta manipulación con mucha menos cantidad de código.
4. **Pruebas unitarias listas**: los diseñadores de Google no solo desarrollaron Angular sino que también desarrollaron un marco de prueba llamado "Karma" que ayuda a diseñar pruebas unitarias para aplicaciones AngularJS.

## ARTEFACTOS DE ANGULARJS

En AngularJs se destacan una serie de elementos y componentes que son las bases de toda aplicación construida con este framework. Elementos que se encontrarán a lo largo de los códigos Javascript y HTML del proyecto

## MÓDULOS

En su mayoría, las aplicaciones comúnmente poseen un método principal que se encarga de instanciar y conectar las diferentes partes de una aplicación.

Las aplicaciones AngularJs no tienen este método principal. En su lugar, los módulos especifican de forma declarativa cómo se debe arrancar una aplicación.

Un módulo de AngularJs se puede representar como el contenedor de las diferentes partes de la aplicación. Es una colección de proveedores, servicios, directivas, etc., y opcionalmente configuraciones que ejecuta los bloques que se aplican a la aplicación durante el proceso de arranque.

Hay varias ventajas en la utilización de módulos:

* El proceso declarativo es más fácil de entender.
* Se puede empaquetar el código como módulos reutilizables.
* Los módulos se pueden cargar en cualquier orden (o incluso en paralelo) porque los módulos demoran la ejecución.
* Las pruebas unitarias solo tienen que cargar los módulos relevantes, lo que las mantiene rápidas.

Para aprovechas las ventajas de una aplicación modular en AngularJs, la documentación oficial recomienda:

* Un módulo por cada funcionalidad
* Un módulo para cada componente reutilizable
* Un módulo a nivel de la aplicación que depende de los módulos anteriores y que contenga cualquier código de inicialización

## CONTROLADORES

Siempre contenidos en los módulos, los controladores en AngularJs controlan el flujo de datos de una aplicación. Son objetos de Javascript, creados por un constructor de objetos y en el HTML, mediante la directiva ng-controller, se define el alcance del controlador. De esta forma se tiene un ámbito acotado en el HTML en el que actuará dicho controlador, con las funcionalidades declaradas en él mediante el objeto Javascript.

Según las buenas prácticas sugeridas por la comunidad, la declaración de un controlador angular es la siguiente:

angular

.module('app')

.controller('DashboardController', DashboardController);

function DashboardController() {

var vm = this ;

vm.nombre = 'mi nombre';

vm.sendMessage = function () {};

}

Y su referencia en el HTML:

<div ng-controller = "DashboardController as ctrl" >

{{ctrl.nombre}}

</div>

Los controladores se pueden adjuntar al DOM de diferentes maneras. Para cada una de ellas, AngularJS creará una instancia de un nuevo objeto Controlador, utilizando la función de constructor del Controlador especificada:

* La directiva ng-controller. utilizando la sintaxis “controller as”, la instancia del controlador se asignará a una propiedad en el scope de la aplicación.
* Un controlador de ruta en una definición de $ruta.
* El controlador de una directiva regular, o una directiva de componentes.

En general, un controlador no debe tratar de hacer demasiado. Debe contener solo la lógica de negocios necesaria para una sola vista.

La forma más común de mantener a los Controladores delgados es encapsular el trabajo que no pertenece a los controladores en los servicios y luego usar estos servicios en los Controladores a través de la inyección de dependencia.

## DIRECTIVAS

Las directivas son marcadores en los elementos del DOM (como un atributo, nombre de elemento, comentario o clase CSS) que le dicen al compilador HTML de AngularJs ($compile) que adjunte un comportamiento específico a ese elemento DOM (por ejemplo, a través de detectores de eventos), o Incluso para transformar el elemento DOM y sus hijos.

Por lo tanto, las directivas son el primer recurso de AngularJs para la manipulación e interacción con el HTML, y siempre que sea posible se deben hacer usos de ellas para gestionar los cambios.

Para crear una directiva personalizada, la estructura declarativa sigue el formato básico de los controladores:

angular

.module('app')

directive('myCustomer', function() {

var directive {

restrict: 'E',

scope: {

customerInfo: '=info'

},

templateUrl: 'my-customer-iso.html',

link:link

};

function link(scope, element, attrs) {

/\* \*/

}

});

## SERVICIOS

El término “servicios” es muy utilizado en programación, principalmente cuando se trata de un desarrollo web. En AngularJs los servicios son un conjunto de funciones relacionadas a una tarea en particular, controladas por el framework y que están a disposición del programador para ser usadas en la aplicación.

Los Servicios son objetos sustituibles conectados entre sí y utilizados en los controladores a través de la inyección de dependencias. Se utilizan para organizar y distribuir código reutilizable a lo largo de la aplicación.

Cabe mencionar que los servicios de AngularJs solamente son instanciados cuando cuando un componente de la aplicación depende de él (lo que en programación se conoce como “Lazy Instance”). Además de esto, todos los componentes que dependan de un servicio mismo son referenciados a una misma instancia de este.

Para usar un servicio en AngularJs, Se lo agrega como una dependencia para el componente (controlador, servicio, filtro o directiva) donde se quiera utilizar el servicio. El subsistema de inyección de dependencias de AngularJs se encarga del resto. Adicional, un servicio puede contener sus propias dependencias de otros servicios.

El core de AngularJs provee una variedad de servicios ya disponibles dentro de su framework los cuales están disponibles para el programador y siempre van precedidos por el carácter “$” en su nombre. Algunos de estos servicios son:

* **$http:** es una función que toma un solo argumento, un objeto de configuración, que se utiliza para generar una solicitud HTTP y devuelve una promesa que se resuelve (solicitud exitosa) o se rechaza (solicitud de falla) con un objeto de respuesta.
* **$filtro:**

## ANGULARJS Y EL MODELO MVC

AngularJs utiliza la estructura del diagrama Modelo-Vista-Controlador de la siguiente manera:

* **Controlador:** Representa la capa del sistema en donde se encuentra lógica de la aplicación y sobre todo las llamadas "Factorías" y "Servicios" para mover datos contra los servidores o la memoria local en HTML5. El controlador responde a las acciones del usuario y realiza interacciones con los objetos del modelo de datos. El controlador recibe la entrada, la valida y luego realiza las operaciones del negocio que modifican el estado del modelo de datos.
* **Vistas:** Se utilizan para representar la capa de presentación o interfaz que se proporciona a los usuarios finales, es decir, todo aquello que el usuario puede ver en la pantalla e interactuar.
* **Modelo:** El modelo representa la estructura de datos de la aplicación y es el responsable de la gestión de estos datos. El modelo responde a la solicitud del usuario a través de las vista y a las instrucciones del controlador para actualizarse.

<https://blog.faztweb.com/2016/10/javascript-el-proximo-lenguaje-universal.html>

<https://www.arsys.es/blog/programacion/por-que-angularjs/>

<https://www.arsys.es/blog/programacion/aprender-javascript/>

<https://desarrolloweb.com/manuales/manual-angularjs.html>

<https://desarrolloweb.com/articulos/por-que-angularjs.html>

<https://hackr.io/blog/10-best-javascript-frameworks-2019>

<https://raygun.com/blog/popular-javascript-frameworks/>

<https://www.javatpoint.com/what-is-angularjs>

<https://www.upwork.com/hiring/development/angularjs-basics/>

<https://www.guru99.com/angularjs-introduction.html>

<https://www.tutorialsteacher.com/angularjs/what-is-angularjs>

<https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm>

<https://www.guidacode.com/2017/angularjs/explicacion-del-patron-mvc-en-angularjs/>

<https://www.guidacode.com/category/angularjs/>