El futuro digital es de todos

MinTIC

Universidad Industrial de Santander

MisiónTIC2022

INTRODUCCIÓN

1. Introducción

1.1. Metodología Scrum

1.1.1. ¿Qué es Scrum?

Scrum es un marco que permite el trabajo colaborativo entre equipos, que anima a aprender a través de las experiencias, a autoorganizarse mientras aborda un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente; además permite obtener el mejor resultado posible en la gestión de un proyecto.

1.1.2. Características

Scrum es una metodologías ágiles más populares, ya que aporta flexibilidad y una estructura organizada que aumenta las probabilidades de éxito en la gestión de proyectos. Las características o pilares ayudan a la administración de un proyecto aplicando este método; el cual se describe a continuación:

- El método apoya la colaboración y la autoorganización.
- Los equipos autogestionan sus actividades en tiempos específicos.
- Se realizan listas Priorizadas de las tareas llamadas Pila del producto o Product Backlog.
- Versiones funcionales del producto final continuamente en cada Ciclo (*sprints*).
- Los *sprints* se repiten hasta el final del proyecto.

- En Scrum, la adaptación a las condiciones del mercado tiene especial importancia.
- Scrum se enfoca en brindar respuestas rápidas y eficientes a los cambios.
- Ofrecer altos valores comerciales al cliente en poco tiempo.
- Reunión de seguimiento Diarias que no debe durar más de 15 minutos.
- Participación de todas las partes involucradas en el proyecto.
- El tamaño del equipo Scrum suele ser de seis a diez personas.

1.1.3. Factores claves

Los elementos que nos permiten alcanzar los objetivos que se han trazado en la metodología Scrum son los siguientes.

- Equipos autoorganizados que toman decisiones.
- Responsabilidad y autodisciplina
- Trabajo centrado en el compromiso de desarrollo
- Información, transparencia y visibilidad en el desarrollo del proyecto.
- Fases de desarrollo solapadas
- La incertidumbre como elemento consustancial y asumido en el entorno y cultura de la organización
- Difusión y transferencia del conocimiento
- Control sutil

1.1.4. El Sprint

Sprint es el nombre que va a recibir cada uno de los ciclos o iteraciones que vamos a tener dentro un proyecto Scrum, el cual posee las siguientes características.

- En Scrum, los proyectos avanzan por iteraciones Sprint de 2–4 semanas de duración (desarrollo de funcionalidad).
- Dentro de cada Sprint, se gestiona la evolución del proyecto mediante reuniones breves de seguimiento en las que se revisa el trabajo realizado desde el hito anterior y los planes para el hito siguiente.
- Las reuniones de seguimiento de cada Sprint deben ser diarias.

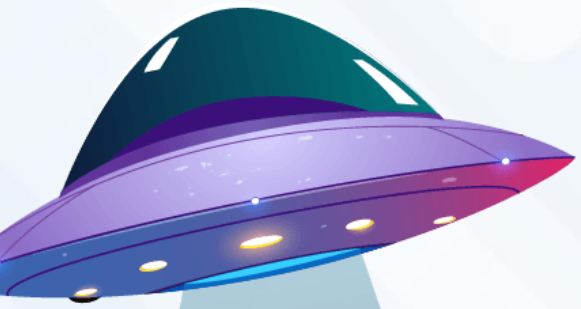
La duración del sprint hay que pensarla en función al tiempo que puede comprometerse a mantener los cambios fuera del sprint. y la únicas forma de abortar un Spring son las siguientes:

- Facilitador de proyecto (Scrum Master) aborte.
- La tecnología seleccionada no funciona.
- El Scrum Team ha tenido interferencias.
- Han cambiado las circunstancias del negocio.

1.1.5. Roles

En la metodología Scrum podemos identificar tres roles principales: Product Owner, Scrum Master y el Scrum Team.

- **Product Owner:** Es el encargado de tener la visión de lo que se desea construir y transmitirla al equipo y tiene las siguientes responsabilidades:
  - Responsable de la Pila de Producto y su correcta priorización.
  - Prioriza funcionalidades dependiendo del valor de mercado-
  - Cambiar la funcionalidad y prioridades para cada sprint.
  - Acepta o rechaza los resultados del sprint.
  - Responsabilidad del producto

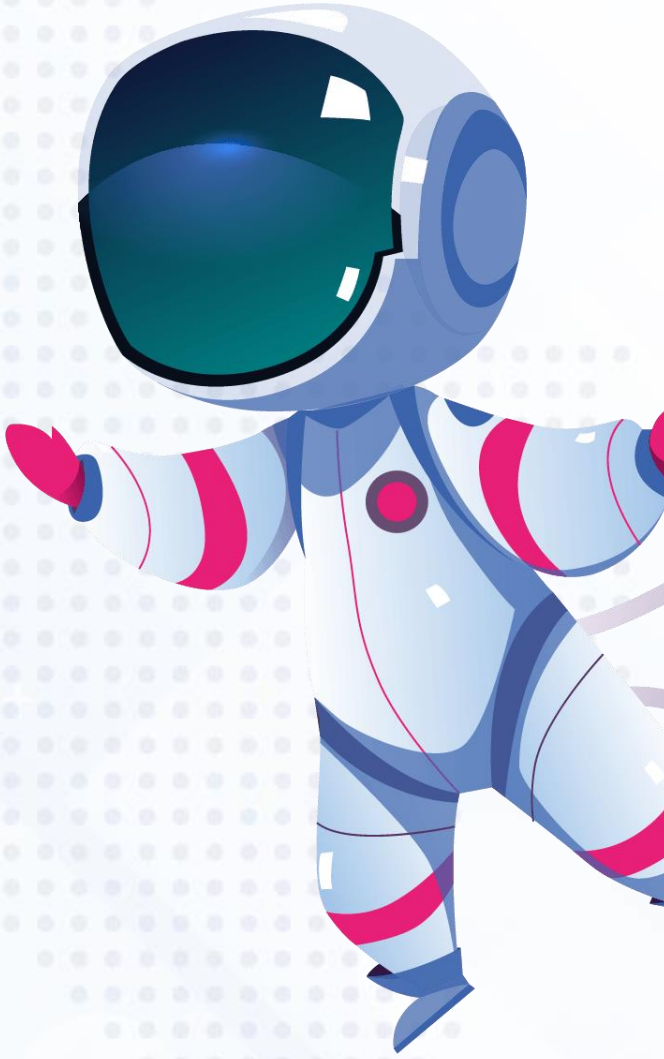


- **Scrum Master :** Responsable de la eficiencia del equipo Scrum y de que el equipo mejore de manera continua hacía su mejor versión, responsabilidades:

  - Asegura que el equipo es funcional y productivo.
  - Favorece la cooperación entre todos los roles y funciones.
  - Elimina barreras.
  - Aísla y defiende al equipo de interferencias externas
  - Asegura que el equipo siga Scrum.
  - Responsabilidad del funcionamiento.

- **Scrum Team:** grupo de Desarrolladores entre 3 a 9 profesionales que se encargan de desarrollar el producto, autoorganizándose y autogestionándose, tiene las siguientes responsabilidades:

  - Selecciona la meta del Sprint.
  - Equipo multidisciplinar con habilidades necesarias para poder cumplir la meta del Sprint.
  - Se autoorganiza así mismo y a su trabajo.
  - Hace sus problemas visibles.
  - Responsabilidad del desarrollo.



1.1.6. Reuniones

La metodología Scrum está basada en la gestión del proyecto de manera iterativa e incremental, siendo las reuniones diarias una de las claves para formar equipos Scrum altamente efectivos. Existen 4 tipos de reuniones en Scrum:

- **Planificación**. Se planifica el siguiente sprint, tratando temas como la duración de cada actividad, los recursos necesarios para su ejecución, las posibilidades de viabilidad de cada tarea, y otros asuntos que inciden en el futuro del proyecto.

- **Sprint diario**. La función principal de este tipo de reunión es la de comprobar que no existan obstáculos que impidan la ejecución de las actividades en progreso. El equipo de desarrollo ha de estar en sincronización, manteniéndose informado de cada cambio en el proyecto que pueda afectar a su trabajo.

- **Demo del Sprint**. Se trata de una reunión de cierre en la que se expone el resultado final del proyecto a todo el equipo. El Project Owner será el responsable de validar estos resultados, mientras que el Scrum Master tomará nota de los posibles cambios a realizar.

- **Retrospectiva**. La metodología Scrum recomienda la celebración de este tipo de reuniones privadas entre el Scrum Master y el equipo de desarrollo, con el fin de revisar la manera en la que se ha gestionado el proyecto recién finalizado.

1.1.7. Artefactos

Son aquellos elementos físicos que se producen como resultado de la aplicación de Scrum.

- **Product Backlog**(pila de producto): es un listado de todas las tareas que se pretenden hacer durante el desarrollo de un proyecto.
    - Lista de funcionalidades y requisitos del producto.
    - Es un documento vivo
    - equipo de desarrollo pueden contribuir a él aportando ideas
    - El responsable de la Pila de producto y de su correcta priorización es el Product Owner.
    - Debe ser visible y fácilmente accesible por todo el mundo.
    - Proviene de un plan de negocio creado junto con el cliente.

- **Sprint Backlog(Pila del Sprint):** Subconjunto de objetivos/requisitos del Product Backlog seleccionado para la iteración actual y su plan de tareas de desarrollo. El equipo lo elabora en la reunión de planificación de la iteración (Sprint planning).
    - El equipo es el responsable de la Pila de Sprint
    - Contiene tareas de desarrollo requeridas para completar elementos de la Pila de Producto
    - Las tareas son estimadas y éstas actualizadas diariamente
    - Las tareas no suelen tener un responsable al principio del sprint y cualquiera puede cogerlas
    - La Pila de Sprint está cerrada durante el Sprint para cualquiera que no sea parte del equipo

¿Cómo se gestiona?

- Los individuos eligen las tareas
- El trabajo nunca es asignado.
- La estimación del trabajo restante es actualizada diariamente.
- Cualquier miembro del equipo puede añadir, borrar o cambiar el Sprint Backlog.
- Si el trabajo no está claro, definir un tema del Sprint Backlog con una mayor cantidad de tiempo y subdividirla luego
- Actualizar el trabajo restante a medida de que más se conoce.

- **Gráfico Burn-Down:** Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los objetivos/requisitos. Permite extrapolar si el equipo podrá completar el trabajo en el tiempo estimado.
    - Utilizado por el Scrum Team para el seguimiento del trabajo de cada Sprint.
    - Se actualiza diariamente
    - Muestra si la meta original del sprint será alcanzada o no.

1.2. ¿Qué es Desarrollo Web?

Desarrollo web es la construcción y mantenimiento de los sitios web; es el trabajo que permite que una web tenga una apariencia impecable, un funcionamiento rápido y un buen desempeño para permitir la mejor experiencia de usuario.

Los profesionales colaboran en desarrollo los podemos dividir en los siguientes perfiles:

- **Desarrollador Frontend:** Los desarrolladores front-end son los encargados de construir a partir de código las interfaces web que utilizamos a partir de ideas o diseños.

- **Desarrollador Backend:** Es el desarrollo del lado del servidor. Es el término utilizado para el desarrollo que ocurre detrás de escena, no visto por los usuarios. Que tienen como objetivo crear la regla de negocio, la lógica para hacer que un sistema web funcione correctamente y lo hacen mediante el uso de lenguajes de programación específicos del lado del servidor.

- **Desarrollador Full Stack:** Manejar cada uno de los aspectos relacionados con la creación y el mantenimiento de una aplicación web. Para ello es fundamental que el desarrollador Full Stack tenga conocimientos en desarrollo Front-End y Back-End.

- **Diseñador UX/UI:** Encargado de crear interfaces que satisfagan las necesidades de los usuarios y les proporcionen una experiencia intuitiva y diferenciadora.

1.3. ¿Qué es JavaScript?

Es un lenguaje de programación orientado a entornos web. Pertenece a la parte de "código cliente", que permite una gran interacción y dinamismo por parte del usuario con la página web.

1.3.1. ¿Cómo nace?

Nace en 1995 con Brendan Eich desarrolla LiveScript para incluirlo en el Netscape Navigator 2.0. Poco tiempo después, la Sun Microsystems pasa a llamarlo JavaScript. En 1996, Microsoft crea JScript para competir con JavaScript y evitar problemas de licenciamiento y en 1997, se envía la especificación del JavaScript 1.1 a la ECMA y se estandariza el ECMAScript, hoy es lenguaje que aparece en el 90% de la páginas web creadas.

1.3.2. Características

Las principales características de JavaScript son las siguientes:

- **Lenguaje del lado del cliente:** se refiere a que se ejecuta en la máquina del propio cliente a través de un navegador.
- **Lenguaje Orientado a Objetos:** utiliza clases y objetos como estructuras que permiten organizarse de forma simple y son reutilizables durante todo el desarrollo.
- **Tipado Nivel o no Tipado:** no es necesario especificar el tipo de dato al declarar una variable. Esta característica supone una gran ventaja a la hora de ganar rapidez programando, pero puede provocar que cometamos más errores.
- **De Alto Nivel:** su sintaxis es fácilmente comprensible por su similitud al lenguaje de las personas. Se le llama de "alto nivel" porque su sintaxis se encuentra alejada del nivel máquina.
- **Lenguaje Interpretado:** utiliza un intérprete que permite convertir las líneas de código en el lenguaje de la máquina.

1.3.3. Estructura de una página web

Una página web cuenta de tres partes las cuales interactúan entre sí, como muestra la siguiente figura:

1.3.4. Inclusión de Javascript en las páginas

Para hacer que un documento HTML incluye instrucciones en JavaScript se debe hacer uso de la etiqueta <SCRIPT>

```
<script language="JavaScript">    código JavaScript </script>
```

o mediante un archivo externo separado con la extensión js.

```
<script language="Javascript" src="archivo.js">
```

1.3.5. Normas de Escritura

- Los comentarios deben empezar con el símbolo // si son de una sola línea o iniciarse con los símbolos /* y finalizar con */ si son de varias líneas.
- Las líneas de código terminan con el signo de punto y coma (;)
- JavaScript distingue entre mayúsculas y minúsculas
- Las llaves ({ y }) permiten agrupar código.

1.3.6. Variables y Operadores

Una variable es un elemento que permite almacenar valores, en JavaScript debe empezar con una letra la cual puede ir seguida de números, el signo "_" o más letras, puede almacenar cadenas de texto, valores numéricos o booleanos y se declara como se muestra a continuación.

var variable **=** valor; o simplemente variable **=** valor;

1.3.7. Conversión de datos

JavaScript realiza conversiones implícitas. Ejemplo:

var x=”50” //x es una variable de texto

var y=30 //y es una variable numérica

z1=x+y //z1 es variable de texto y vale “5010”

z2=y+x //z2 es numérica y vale 60

1.3.8. Operadores

Operadores aritméticos

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	Dividir
%	Resto de la división
++	Incremento
--	Decremento

Operadores Comparación

Operador	Significado
==	Igual
!=	Distinto
>=	Mayor o igual
<=	Menor o igual
>	Mayor
<	Menor

Operadores lógicos

Operador	Significado
&&	AND (Y lógico)
||	OR (O lógico)
!	NOT (NO lógico)

1.3.8.1. Estructuras

Las estructuras de este tema son sentencias que permiten tomar decisiones o repetir instrucciones dentro del código.

- **Instrucción if**

```
if(condición) {

 ..código que se ejecuta si la condición es cierta

 }else{

... ..código que se ejecuta si la condición es falsa

}
```

- **Instrucción switch**

switch (objetodeanálisis**) {**

case valor1: ..instrucciones

case valor2:...instrucciones

....

default: instrucciones **}**

- **Bucle while**

while(condición**){**

... sentencias que se ejecutan mientras la condición se cumpla

}

- **Bucle for**

  for(valor_inicial; condición; actualización**){**

  ..sentencias que se ejecutan mientras la condición se cumpla

  }

- **Break**

  Es una instrucción que hace que el navegador que se la encuentra, abandone inmediatamente el bucle en el que está inmerso.

- **Continue**

  Es parecida a la anterior, sólo que en lugar de abandonar el bucle, lo que hace el navegador es dejar de leer las siguientes instrucciones del bucle y saltar al principio del mismo.

1.3.8.2. Funciones

Las funciones son una serie de instrucciones que realizan una determinada tarea.

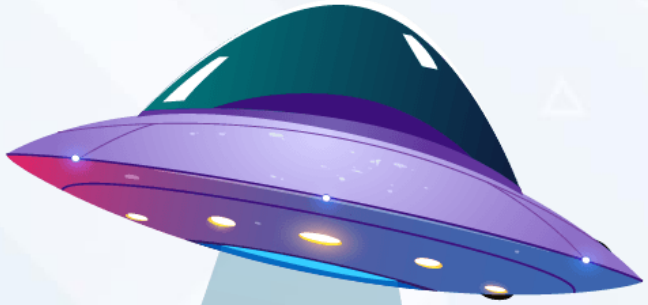
- **¿Cómo se hace la definición?**

Antes de poder usar una función en el código de la página, se la debe definir.

function nombredelafunción**(**argumento1, argumento2,...**) {**

instrucciones que debe realizar la función

}



¿Cómo se llama una función?

Para usar (invocar) una función en el script, basta con poner su nombre seguido de los paréntesis. por ejemplo

```
function error() {

document.write("<b>Ocurrió un error</B><BR>");

}
```

Funciones predefinidas

JavaScript trae consigo muchas funciones predefinidas.

- **eval(textoCódigo):** función hace que el texto sea interpretado como si fuera código normal de JavaScript.
- **eval(**"alert('Hola')"**);**
- **parseInt(textoNúmero, base):** Convierte el texto (que debe tener cifras numéricas) a formato de número.
- **parseFloat(textoNúmero):** Convierte el texto (que debe tener cifras numéricas) a formato de número con decimales**.**
- **escape(texto):** Muestra el código ASCII de los símbolos del texto.
- **unescape(texto):** Hace justo lo inverso del anterior.
- **isNaN(expresión):** Devuelve true si la expresión tiene un contenido no numérico.

1.3.9. Objetos

Un objeto es una agrupación de variables, que en ese caso se llaman propiedades, y de funciones, las cuales se llaman métodos.

- **Propiedades :** los objetos poseen propiedades asociadas, para acceder a ellas se utiliza el punto, en esta forma:

  objeto.propiedad

- **Métodos :** Los métodos son funciones asociadas a los objetos.

  objeto.metodo()

- **Operación new:** new sirve para crear nuevos objetos en el tiempo de ejecución del código JavaScript.

  miordenador = new ordenador("HP", "Pentium III", 64**);**

1.3.10. Objetos predefinidos

- **String:** El objeto string sirve para manejar cadenas de texto. Cada vez que creamos una variable de cadena, en realidad estamos creando una variable de tipo string.
- **Math:** El objeto Math tiene propiedades y métodos que representan valores matemáticos.
- **Objeto Date():** Representa fechas y horas. JavaScript no permite trabajar con fechas anteriores a 1970.
- **Objeto array:** Los arrays (matrices) son elementos indispensables en la programación de ordenadores. Un array es un conjunto de datos agrupados.

nombreArray= **new Array(**tamaño**)**

1.3.11. Objetos del navegador

Son todos aquellos objetos que el navegador pone a nuestra disposición para poder modificar los elementos de las páginas. entre estos tenemos:

- **Navigator: Objeto que representa al navegador con el que el usuario está mostrando la página.**
- **Screen:** Permite acceder a las propiedades de la pantalla del usuario.
- **Window:** Representa a la ventana en la cual se ve la página web.
- **Location:** Objeto incluido dentro del objeto window. Almacena información sobre la localización de la página de la ventana y por tanto permite cambiar dinámicamente la página web.

- **Document:** Representa al contenido de una página web. Está incluido dentro del objeto window.
- **History:** Objeto que representa a las direcciones de las páginas que se almacenan en el historial del navegador. Este objeto está dentro del objeto window.
- **Image:** Objeto que representa una imagen en el documento definida con la etiqueta HTML <IMG> o con el código JavaScript new Image.
- **Link:** Representa cada enlace de la página creado con la etiqueta A. El array document.links contiene todos los enlaces del documento.

1.3.12. Eventos

Un evento es un suceso que ocurre cuando el usuario realiza alguna acción. Por ejemplo, cuando el usuario pasa el ratón encima de un objeto de la página, cuando pulsa una tecla, ... Incluso algunos eventos no los produce el usuario, sino el navegador, como por ejemplo la carga de la página. los eventos que podemos encontrar son:

onClick, onDblClick, onMouseOver, onMouseOut, onMouseDown, onMouseUp, onMouseMove, onKeyDown, onKeyPress, onKeyUp, onLoad, onUnload, onResize, onBlur, onFocus, onAbort, onError, onChange, onSelect, onSubmit y onReset

1.4. Manejo de DOM(Document Object Model)

Modelo de Objeto de Documento, es una interfaz de programación para los documentos HTML y XML, se muestra como una jerarquía de objetos del navegador, una estructura jerárquica donde existen varios objetos y unos dependen de otros.

La representación de la página web en la memoria del navegador, a la que podemos acceder a través de JavaScript. El DOM es un árbol donde cada nodo es un objeto con todas sus propiedades y métodos que nos permiten modificarlo. Estas son algunas funciones que nos permiten acceder y modificar los elementos del DOM, también se puede acceder usando la librería jQuery:

- **Acceso a elementos del DOM**

```
// Obtiene un elemento por id

document.getElementById('someid');

// Obtiene una lista con los elementos que tienen esa clase

document.getElementsByClassName('someclass');

// Obtiene una HTMLCollection con los todos los elementos 'li'
```

```
document.getElementsByTagName('LI');

// Devuelve el primer elemento del documento que cumpla la selección (la notación es como en CSS)

document.querySelector('.someclass');

// Devuelve una lista de elementos que cumplen con la selección (notación como en CSS)

document.querySelectorAll('div.note, div.alert');
```

- ## Acceder a hijos/padres de un elemento

```
// Obtener los hijos de un elemento

var elem = document.getElementById('someid');

var hijos = elem.childNodes;

// Su nodo padre

var padre = elem.parentNode;
```

• Crear nuevos elementos en el DOM

```
var nuevoH1 = document.createElement('h1');

var nuevoParrafo = document.createElement('p');

// Crear nodos de texto para un elemento

var textoH1 = document.createTextNode('Hola mundo!');

var textoParrafo = document.createTextNode('lorem ipsum...');

// Añadir el texto a los elementos

nuevoH1.appendChild(textoH1);

nuevoParrafo.appendChild(textoParrafo);

// también podemos asignar directamente el valor a la propiedad innerHTML

nuevoH1.innerHTML = textoH1

nuevoParrafo.innerHTML = textoParrafo
```

• Añadir elementos al DOM

```
// seleccionamos un elemento

var cabecera = document.getElementById('cabecera');

// Añadir elementos hijos a un elemento

cabecera.appendChild(nuevoH1);

cabecera.appendChild(nuevoParrafo);

// También podemos añadir elementos ANTES del elemento seleccionado

// Tomamos el padre

var padre = cabecera.parentNode;

// Insertamos el h1 antes de la cabecera

padre.insertBefore(nuevoH1, cabecera);
```

1.5. ¿Qué es typescript?

TypeScript, también llamado TS fue creado por Microsoft. La primera versión de TypeScript fue lanzada por Microsoft en octubre de 2012. Es un superconjunto de tipo estricto del lenguaje **JavaScript**. TypeScript es un lenguaje de programación de código abierto que también recibe muchas mejoras de la comunidad. TypeScript se puede transferir a JavaScript, lo que lo hace popular para el desarrollo web

1.5.1. Características typescript

Lenguaje orientado a objetos, trayendo herramientas como la herencia, sobrecarga, etc; igual que lenguajes como Java, C++, C#, etc. Tienen tipado estático. En el caso de TypeScript este tipado estático es opcional, pero obviamente su uso es muy recomendado. Estas opciones presentan las características para el desarrollo de frameworks y librerías de desarrollo web tanto del lado del servidor como del cliente.

1.5.2. Tipos básicos

En TypeScript las variables se declaran de la siguiente manera:

variable: Tipovariable = valor

- **Number:** Valor numéricos, todos los números en TypeScript son valores de coma flotante

  let decimal: number = 6;

- **String:** Valor de cadena o plantilla, que pueden abarcar varias líneas y tener expresiones incrustadas.

  let fullName: string = `Bob Bobbington`;

- **Boolean:** true o false

- **Object:** Un objeto es una instancia que contiene un conjunto de pares de valores clave. Los valores pueden ser valores o funciones escalares o incluso una matriz de otros objetos.

```
var object_name = {

  key1: “value1”, //scalar value

  key2: “value”,

  key3: function() {

   //functions

  },

  key4:[“content1”, “content2”] //collection

};
```

- **Interface:** La interfaz es una estructura que define el contrato en su aplicación. Define la sintaxis para las clases a seguir. Las clases que se derivan de una interfaz deben seguir la estructura proporcionada por su interfaz.

```
interface IEmployee {
  empCode: number;
  empName: string;
  getSalary: (number) => number; // arrow function
  getManagerName(number): string;
}
```

- **Any:** Es posible que necesitemos describir el tipo de variables que no sabemos cuando estamos escribiendo una solicitud.

```
let notSure: any = 4;
```

- **Array:** TypeScript, como JavaScript, le permite trabajar con matrices de valores.

```
let list: number[] = [1, 2, 3];
```

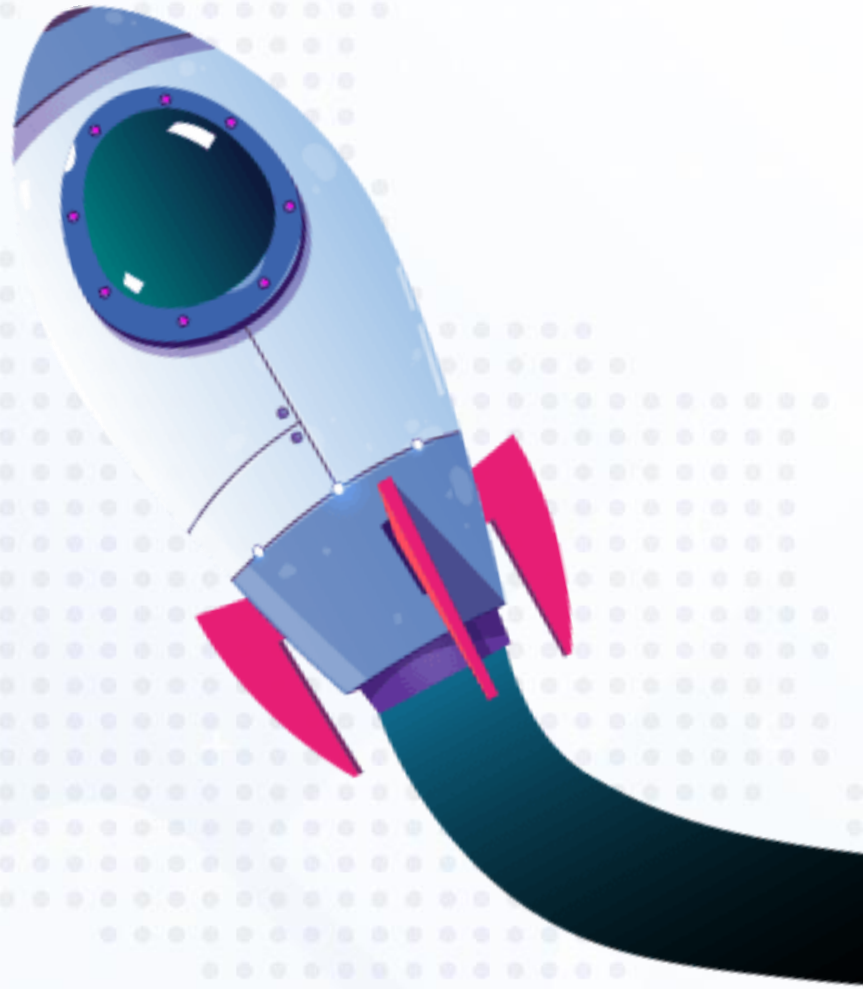
La segunda forma usa un tipo de matriz genérico Array<elemType>:

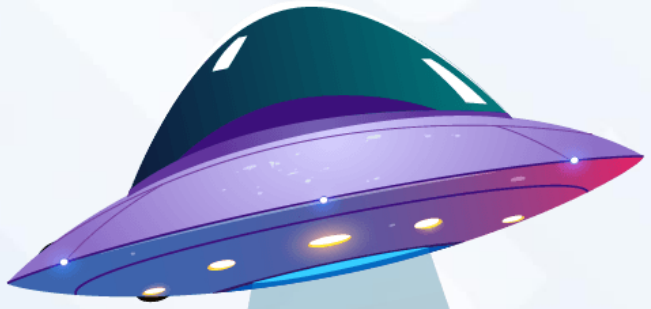
```
let list: Array<number> = [1, 2, 3];
```

1.5.3. Ventajas de TypeScript

- Es fácil de aprender
- Convierte a JavaScript en un lenguaje fuerte.
- Permite desarrollar sin preocuparse por el soporte de ciertas características
- Mejora la ayuda contextual.
- Permite crear código estandarizado.

Funciona bien con las librerías y *frameworks* de Front-End.





1.6. MongoDB

Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto y orientado a documentos. MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico.

1.6.1. Características

- MongoDB guarda estructuras de datos en documentos tipo JSON (JavaScript Object Notation) con un esquema dinámico.

- Internamente MongoDB almacena los datos en formato BSON (Binary JavaScript Object Notation).

BSON está diseñado para tener un almacenamiento y velocidad más eficiente.

1.6.2. Historia

MongoDb empezó con la empresa de software 10gen en el 2007, cuando estaban desarrollando una plataforma como servicio (PAAS) similar al conocido Google App Engine. En el 2009, MongoDB fue lanzado como un producto independiente y publicado bajo licencia de código libre AGPL. En el 2011 se lanzó la versión 1.4 y se consideró como una BD lista para su uso en producción.

1.6.3. Comparaciones

RDBMS	MongoDB
Database instance	MongoDB instance
Database / Schema	Database
Table	Collection
Row	Document
Rowid	_id
Join	Dbref

1.6.4. ¿En qué casos usarlas?

- **Logging de Eventos:** las bases de datos basadas en documentos pueden loguear cualquier clase de eventos y almacenarlos con sus diferentes estructuras. Pueden funcionar como un repositorio central de logueo de eventos.
- **CMS, blogging:** su falta de estructura predefinida hace que funcionen bien para este tipo de aplicaciones.
- **Web-analytics / Real-Time analytics:** Almacenar cantidad de vistas a una página o visitantes únicos.
- **Commerce:** A menudo requieren tener esquemas flexibles para los productos y órdenes

Referencias Bibliográficas

Libros

Scrum Manager Gestión de Proyectos. Juan Palacio, Claudia Ruata.

Flexibilidad con Scrum. Juan Palacio.

Agile Project Management with Scrum. Ren Schaner.

Aprendiendo JavaScript, Carlos Azaustre.

Comprendiendo ECMAScript 6, Nicholas Zakas

Practical Web Design for Absolute Beginners, Adrian W. West

TypeScript: Curso Práctico (Spanish Edition), CARLOS SERRANO.

MongoDB en Español, Yohan Graterol.

MongoDB en Español: MongoDB : Aprendamos a usar bases de datos NoSQL, Anthony Watkins.

Web

Scrum the Home of Scrum, https://www.scrum.org/resources/scrum-guide

«JavaScript». Mozilla Developer Network, https://developer.mozilla.org/es/docs/Web/JavaScript

JavaScript Tutorial, w3schools, https://www.w3schools.com/js/

typescript, Microsoft, https://www.typescriptlang.org/

MongoDB Atlas, https://www.mongodb.com