

Desarrollo Backend

Bienvenid@s

Documentación

Clase 20



Pon a grabar la clase



Temario

- La importancia de documentar
- Formas de documentar en Node.js
- Markdown
 - Sintaxis
 - agregar imágenes
 - agregar listas de tareas y tablas
 - agregar código
 - agregar diagramas
- Estructura de la documentación API Restful
- Práctica integradora

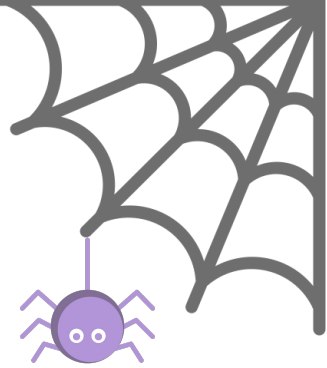


La importancia de documentar

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar



Llegamos al momento más terrorífico para
cualquier desarrollador de software...

¡DOCUMENTAR!



La importancia de documentar

Al igual que sucede con la mayoría de productos que llegan al mercado, donde encontramos que los mismos cuentan con una guía, manual, o documento que explica su uso, el software no es la excepción a la regla.

Como parte de nuestra rutina cotidiana, debemos incluir la documentación de cada pieza de software que realicemos.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar

Algunos de los puntos más importantes que podemos rescatar de la creación de documentación en nuestros proyectos, son:

- Mejorar la comprensión del proyecto
- Facilitar la colaboración
- Reducir errores y retrasos
- Facilitar la transición y/o adopción
- Aportar en la planificación futura



La importancia de documentar

Mejorar la comprensión del proyecto

La documentación detallada del desarrollo de aplicaciones ayuda a los desarrolladores a comprender mejor el proyecto y sus objetivos, así como a identificar cualquier problema o desafío que puedan surgir.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar

Facilitar la colaboración

Una buena documentación hace que sea más fácil para los miembros del equipo colaborar y trabajar juntos en el proyecto. Esto es especialmente importante cuando hay varios equipos de desarrollo trabajando en diferentes aspectos de la aplicación.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar

Reducir errores y retrasos

También ayuda a los equipos de desarrollo a evitar errores y retrasos innecesarios, ya que pueden referirse a la documentación para asegurarse de que están siguiendo pautas y procedimientos adecuados.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar

Facilitar transición de recursos

Ante el constante cambio de recursos vinculados a un equipo de desarrollo o la contratación de nuevos integrantes, la documentación detallada es muy útil para acelerar cualquiera de estos tipos de transiciones.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

La importancia de documentar

Aportar en la planificación futura

La documentación detallada puede ser útil para futuros proyectos similares, ya que puede servir como base para la planificación y el desarrollo de otras futuras aplicaciones, o para incrementar las capacidades de la aplicación actual.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Formas de documentar en Node.js

Formas de documentar en Node.js

Hay varias herramientas que puedes utilizar para documentar aplicaciones Node.js.

Veamos a continuación algunas de las más populares que pueden servirnos para investigar con mucho más tiempo.



Formas de documentar en Node.js



JSDoc

Swagger

YUIDoc

GitBook

Docco

Como todo tipo de herramientas en general, algunas son más permeables que otras pero, en sí, todas requerirán un proceso de investigación y adaptación.

Formas de documentar en Node.js



Markdown es una opción ligera y fácil de aprender...

(y hasta, tal vez, ya domines).

Será el que elegimos para crear documentos bien estructurados y con el formato apropiado, sin preocuparnos por la complejidad de lenguajes de marcado tradicionales como ser HTML, [LaTeX](#) u otras herramientas.

Formas de documentar en Node.js

Vayamos entonces a realizar un repaso de Markdown para estar todas en el mismo nivel, así luego nos dedicamos a profundizar las pautas requeridas para construir documentación backend de calidad.



MarkDown

Markdown

Por si nunca experimentaste con **Markdown**, te contamos que éste es un lenguaje de marcado ligero que se utiliza comúnmente para formatear el texto de una manera sencilla y fácil de leer.

Fue creado por John Gruber en 2004 con el objetivo de que las personas escribieran documentos en formato fácil de leer y fácil de redactar, utilizando un formato de texto plano.



Markdown

Markdown se ha vuelto muy popular entre los desarrolladores de software, los escritores técnicos, y otros usuarios que necesitan crear documentación de una manera rápida y sencilla.

Su sintaxis es muy sencilla y utiliza una serie de etiquetas simples para indicar el formato del texto como: encabezados, listas, enlaces, texto en negrita y cursiva, entre otros.



Markdown

Soporta la inserción de imágenes, hipervínculos internos y externos, y cuenta con una capacidad avanzada para poder crear diagramas de diferentes tipos:

- de Clase
- UML
- de Entidad relacional

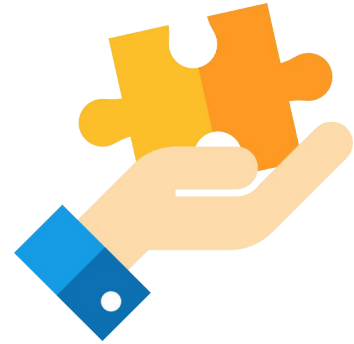
Entre otros. Para estos últimos, necesitaremos extensiones adicionales.



Markdown

Trabajaremos con el lenguaje Markdown directamente en nuestro editor de código.

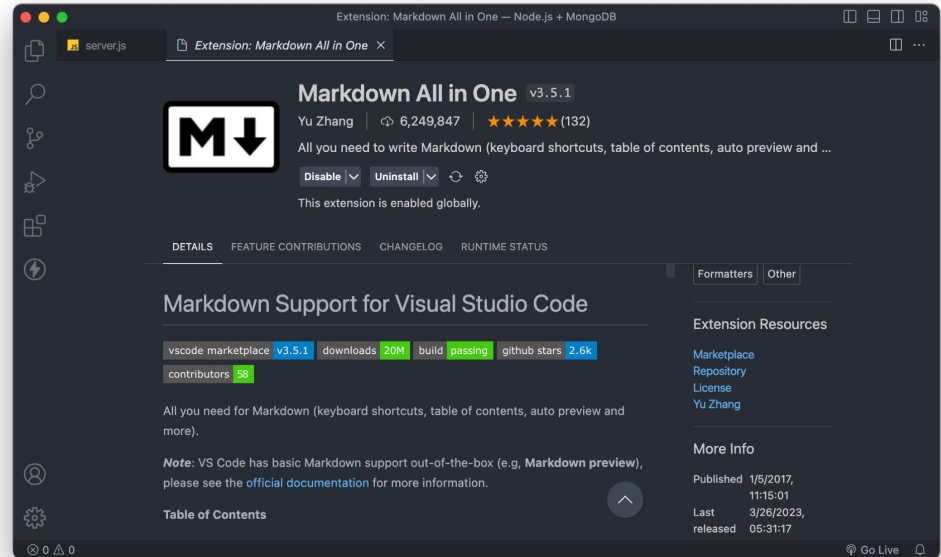
Instalaremos una **extensión en VS Code** llamada: **Markdown All In One**. Ésta integra un visualizador en tiempo real de archivos Markdown dentro de nuestro editor.



Markdown

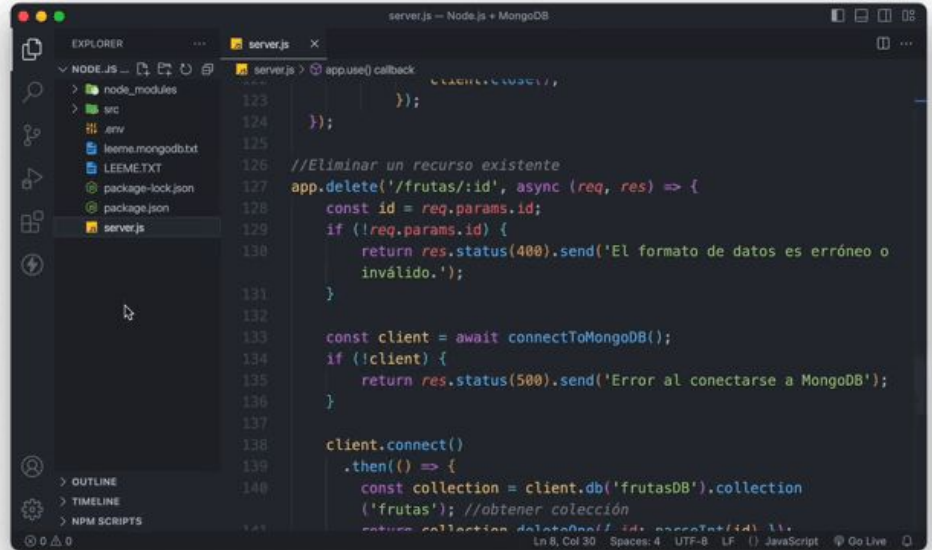
Más allá de nuestra recomendación, existen otras tantas extensiones que realizan lo mismo:

- Markdown Preview Enhanced
- Markdown Preview GitHub Styling
- Markdown Preview



Markdown

Prueba más adelante otras opciones.
Te ayudarán a conocer otras formas de generar documentación, y estar preparada para adoptar prácticas alternativas de equipos de trabajo.



UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Sintaxis

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Sintaxis

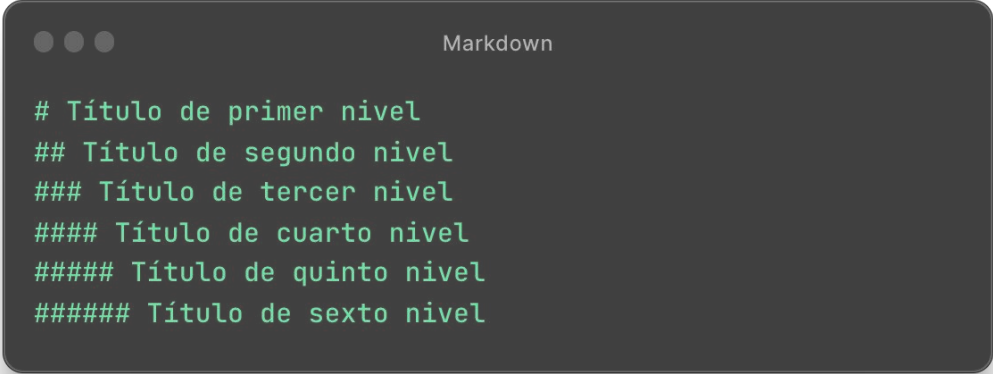


```
# Título de primer nivel
```

Para crear títulos en Markdown ante pondremos el caracter hash **#** al párrafo que deseamos convertir en título. Debemos tener presente respetar el espacio entre este caracter y el título, para que se genere correctamente.

Sintaxis

Cada hash **#** adicional indica un nivel de título adicional. Por lo tanto, el primer ejemplo crea un título de primer nivel y el último ejemplo crea un título de sexto nivel.



```
# Título de primer nivel
## Título de segundo nivel
### Título de tercer nivel
#### Título de cuarto nivel
##### Título de quinto nivel
##### Título de sexto nivel
```

Sintaxis

El manejo de formateo en negrita, cursiva y subrayado, se puede realizar de dos formas diferentes:

- 1) utilizando caracteres underscore _
- 2) utilizando caracteres asterisco *

Este es un texto con formato negrita.

Este es otro texto con *formato cursiva*.

Este es un texto con formato combinado de negrita y cursiva.

Este es un texto con **formato del tipo**.

Este es otro texto con *formato del tipo*.

Este es un texto con ***formato combinado de negrita y cursiva***.

Sintaxis

Dos caracteres underscore o dos asteriscos = **negrita**

Un caracter underscore o un asterisco = *cursiva*

Tres caracteres underscore o tres asteriscos = ***negrita y cursiva***

```
1 # El lenguaje Markdown
```

```
3 Este es un texto con __formato negrita__.
```

```
5 Este es otro texto con _formato cursiva_.
```

```
7 Este es un texto con ___formato combinado de negrita y cursiva___.
```

```
9 Este es un texto con **formato del tipo**.
```

```
11 Este es otro texto con *formato del tipo*.
```

```
13 Este es un texto con ***formato combinado de negrita y cursiva***.
```

El lenguaje Markdown

Este es un texto con formato negrita.

Este es otro texto con *formato cursiva*.

Este es un texto con ***formato combinado de negrita y cursiva***.

Este es un texto con formato del tipo.

Este es otro texto con *formato del tipo*.

Este es un texto con ***formato combinado de negrita y cursiva***.

Sintaxis

El caracter **ñuflo** (o *virgulilla*) por duplicado, permite representar un texto tachado.

Y como Markdown no cuenta con un formato de subrayado, podemos utilizar como alternativa, el antiguo tag HTML `<u>`, para subrayar texto.



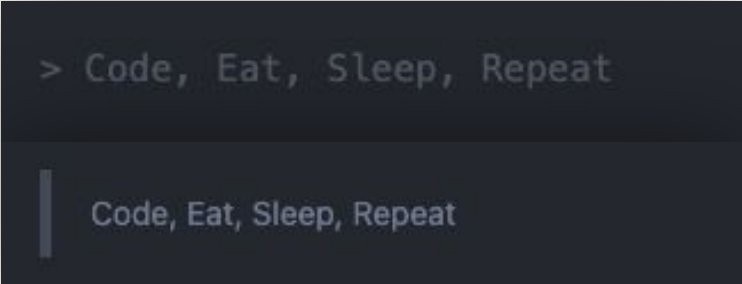
Markdown

```
~~ Texto tachado por defecto ~~
```

```
<u> Texto subrayado por defecto </u>
```

Sintaxis

El uso del símbolo **>** (*mayor que*) se utiliza para representar una cita dentro de un bloque de texto breve.

A dark-themed code block with a light blue border. The top line contains the text "> Code, Eat, Sleep, Repeat" in a light blue font. The bottom line contains the text "Code, Eat, Sleep, Repeat" in a light blue font, preceded by a vertical light blue bar.

```
> Code, Eat, Sleep, Repeat
```

```
Code, Eat, Sleep, Repeat
```

Listas de tareas y tablas

Sintaxis

La integración de checklist también es posible dentro de Markdown. Esto es muy útil para contar con un resumen rápido dentro de la documentación, sobre qué funcionalidades están implementadas y cuáles no.



Markdown

Listado de tareas 21-03-2023

- [x] Construcción del servidor web Express
- [x] Definición de las rutas y métodos HTTP
- [x] Integración con la base de datos MongoDB
- [x] Desarrollo de la lógica de cada Método HTTP
- [] Control de errores al utilizar cada método HTTP
- [] Integración de Token a las peticiones HTTP

Sintaxis

Estos checkboxes sólo se utilizan en modo de sólo lectura. No es posible tildarlos para cambiar su estado directamente desde el archivo Markdown como si fuese un formulario HTML.

```
Markdown

### Listado de tareas 21-03-2023

- [x] Construcción del servidor web Express
- [x] Definición de las rutas y métodos HTTP
- [x] Integración con la base de datos MongoDB
- [x] Desarrollo de la lógica de cada Método HTTP
- [ ] Control de errores al utilizar cada método HTTP
- [ ] Integración de Token a las peticiones HTTP
```

Listado de tareas 21-03-2023

- ☒ Construcción del servidor web Express
- ☒ Definición de las rutas y métodos HTTP
- ☒ Integración con la base de datos MongoDB
- ☒ Desarrollo de la lógica de cada Método HTTP
- ☐ Control de errores al utilizar cada método HTTP
- ☐ Integración de Token a las peticiones HTTP

Sintaxis

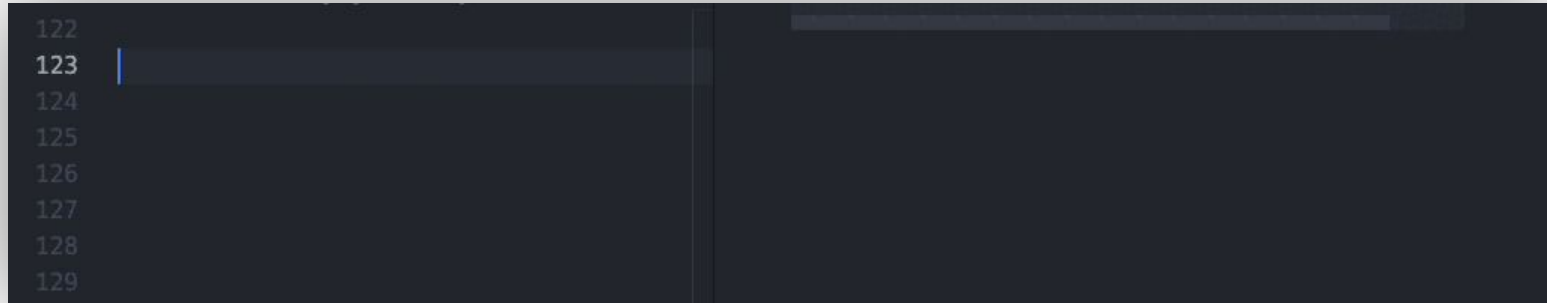
Markdown nos permite estructurar tablas dentro de nuestros documentos. Su forma de construirlas es combinando los guiones - para trazar las líneas horizontales y los caracteres Pipe | para establecer la estructura de columnas.

Como podemos ver en el código de ejemplo, podemos intercalar hipervínculos, como también cualquier otro juego de caracteres que formatee el texto a utilizar.

```
| PETICIÓN | URL | DESCRIPCIÓN |  
|:-----:|-----|-----|  
| GET | [/frutas](/frutas) | Obtener todas las frutas |  
| POST | [/frutas](/frutas) | Grabar una nueva fruta |
```

Sintaxis

Además, contamos con la posibilidad de centrar el contenido de determinadas celdas, si así lo necesitáramos, integrando en los extremos de los guiones correspondientes a la celda a centrar, el caracter : (*dos puntos*).



Agregar código

Agregar código

También es posible representar bloques de código dentro de nuestra aplicación. Y también contamos con dos formas de realizarlo:

- 1) dejando cuatro espacios delante de cada línea de código
- 2) Encerrando el o las líneas de código entre tres caracteres backtick

1



Markdown

```
frutas.forEach(fruta => console.log(fruta))
```

```
...
```

```
const resultado = frutas.filter(fruta => fruta.stock > 50)
```

```
...
```

2



Preview readme.md x

```
frutas.forEach(fruta => console.log(fruta))
```

```
const resultado = frutas.filter(fruta => fruta.stock > 50)
```

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Agregar código

```
Markdown

```javascript
{
 imagen: '🐉',
 nombre: 'Dragon Fruit',
 precio: 1400,
 stock: 95
}

// El id no es necesario informarlo.
// El servidor se ocupa de resolverlo.
...
```
```

Incluso, agregando un nombre de lenguaje contiguo a la declaración inicial de backtick, lograremos que el bloque de código coloree su sintaxis, tal cual en los editores de texto.

```
Método POST

{
  imagen: '🐉',
  nombre: 'Dragon Fruit',
  precio: 1400,
  stock: 95
}

// El id no es necesario informarlo.
// El servidor se ocupa de resolverlo.
```

referenciar hipervínculos

Referenciar hipervínculos

```
Markdown

![Aprender Markdown](ruta-o-carpeta/nombre-de-imagen.jpg)

![Aprender Markdown](ruta-o-carpeta/nombre-de-imagen.jpg
'Texto o Tooltip sobre la imagen')
```

La **inserción de imágenes** dentro de nuestro documento es posible combinando corchetes y paréntesis consecutivos, y anteponiendo el caracter !

Dentro de los corchetes definimos el texto alternativo para la imagen (*accesibilidad*), y dentro de los paréntesis, la ruta hacia la imagen en cuestión.

Referenciar hipervínculos

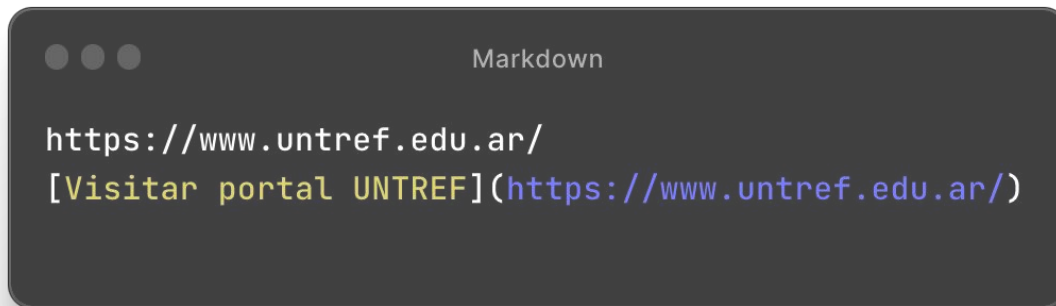


También, contiguo a la imagen definida, podemos sumar un comentario o texto del tipo **Tooltip** ó **Alt**, el cual aparecerá sobre la imagen al posicionar el mouse en esta.

Para ello, debemos encerrarlo entre comillas simples.



Referenciar hipervínculos

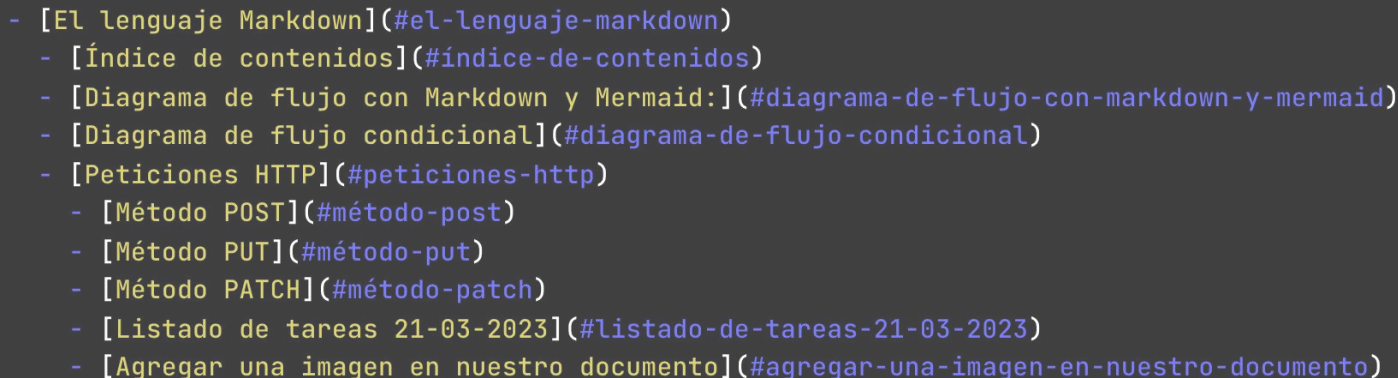


```
https://www.untref.edu.ar/  
[Visitar portal UNTREF](https://www.untref.edu.ar/)
```

La inserción de hipervínculos externos también es posible realizar. Contamos con la opción de referencia directa del hipervínculo donde vemos la URL natural, y también la opción de referenciar un hipervínculo dentro de un texto o párrafo.

Referenciar hipervínculos

De igual forma podemos definir **hipervínculos internos**, utilizando listas ordenadas o desordenadas, según nuestra necesidad. Dentro de los paréntesis indicamos el título o subtítulo del documento Markdown, reemplazando los espacios del título por guiones, usando minúsculas, y anteponiendo el caracter hash.



```
- [El lenguaje Markdown](#el-lenguaje-markdown)
- [Índice de contenidos](#índice-de-contenidos)
- [Diagrama de flujo con Markdown y Mermaid:](#diagrama-de-flujo-con-markdown-y-mermaid)
- [Diagrama de flujo condicional](#diagrama-de-flujo-condicional)
- [Peticiones HTTP](#peticiones-http)
  - [Método POST](#método-post)
  - [Método PUT](#método-put)
  - [Método PATCH](#método-patch)
- [Listado de tareas 21-03-2023](#listado-de-tareas-21-03-2023)
- [Agregar una imagen en nuestro documento](#agregar-una-imagen-en-nuestro-documento)
```

Referenciar hipervínculos

El comportamiento de un índice de contenidos se adaptará automáticamente dentro de su estructura interna, al nivel del título referenciado, generando la correspondiente tabulación de nivel.

Esto es una gran ayuda para no tener que luchar nosotras mismas con estos detalles.

El lenguaje Markdown

Índice de contenidos

- El lenguaje Markdown
 - Índice de contenidos
 - Diagrama de flujo con Markdown y Mermaid:
 - Diagrama de flujo condicional
 - Peticiones HTTP
 - Método POST
 - Método PUT
 - Método PATCH
 - Listado de tareas 21-03-2023
 - Agregar una imagen en nuestro documento

UNTREF

UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

agregar diagramas

Agregar diagramas

Markdown cuenta con la opción de integrar Diagramas de diferentes tipos.

En VS Code contamos con la extensión [Markdown Preview Mermaid Support](#) la cual nos facilita la integración de estos dentro del documento .md.

The screenshot shows the Visual Studio Marketplace page for the 'Markdown Preview Mermaid Support' extension. The extension is developed by Matt Bierner (mattbierner.com) and has a version of v1.18.1. It has 786,588 downloads and a 4.5-star rating from 37 reviews. The extension adds Mermaid diagram and flowchart support to VS Code's builtin markdown preview and to Markdown cells in notebooks. The page includes tabs for DETAILS, FEATURE CONTRIBUTIONS, CHANGELOG, and RUNTIME STATUS. A preview window shows a sample Mermaid diagram being rendered from a markdown file. The diagram is a flowchart with nodes 'Alice', 'Bob', and 'John' and edges representing interactions. The right sidebar shows categories (Other), extension resources (Marketplace, Repository, License, Matt Bierner), and more info (Published: 8/10/2017, Last released: 4/10/2023, Last updated: 4/29/2023, Identifier: bierner.markdown-mermaid).

Agregar diagramas

La sintaxis para utilizar diagramas debe encerrarse dentro de los caracteres backtick, como si se tratase de código.

Además, debemos indicar como lenguaje de programación '**mermaid**', y definir a continuación el tipo de gráfico a construir.

```
Markdown

## Diagrama de flujo con Markdown y Mermaid:



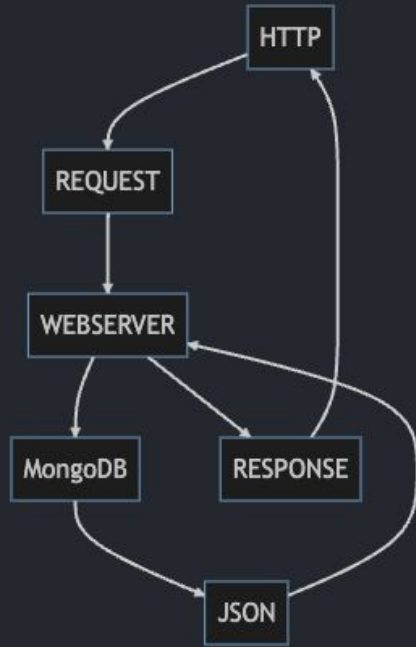
```
mermaid
graph TD;
 HTTP-->REQUEST;
 REQUEST-->WEBSERVER;
 WEBSERVER-->MongoDB;
 MongoDB-->JSON;
 JSON-->WEBSERVER;
 WEBSERVER-->RESPONSE;
 RESPONSE-->HTTP;

```


```


Agregar diagramas

Diagrama de flujo con Markdown y Mermaid:

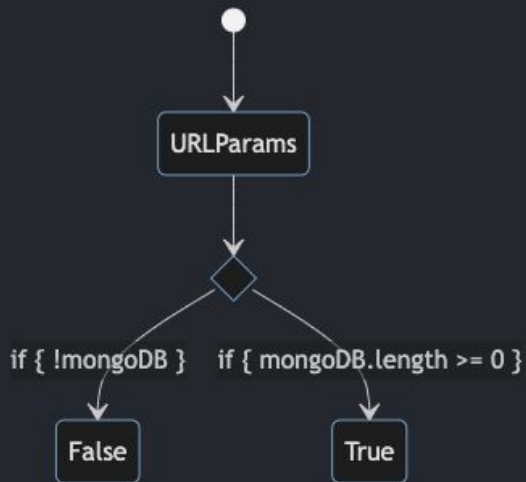


Leyendo un poco la documentación y experimentando otro tanto, lograremos construir de a poco diferentes tipos de diagramas.

Aquí un ejemplo del resultado de la diapositiva anterior.

Agregar diagramas

Diagrama de flujo condicional



```
Markdown

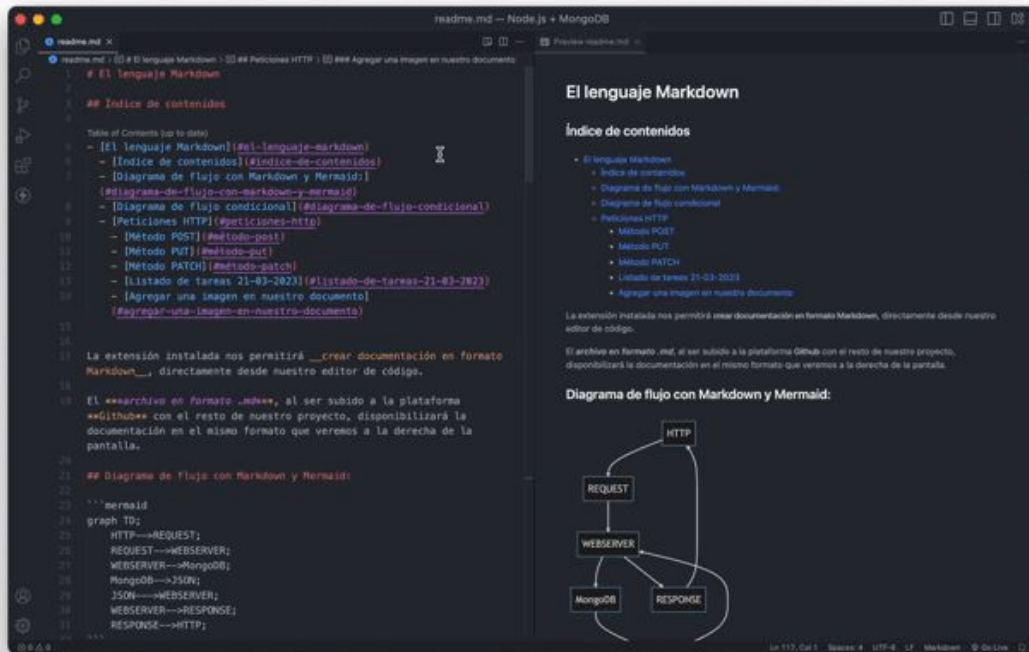
## Diagrama de flujo condicional
```mermaid
stateDiagram-v2
 state if_state <<choice>>
 [*] --> URLParams
 URLParams --> if_state
 if_state --> False: if { !mongoDB }
 if_state --> True : if { mongoDB.length >= 0 }

```

**Existen diferentes tipos de diagrama que podemos encarar:**

*Diagramas de Flujo, Entidad-Relación, de Gantt, de Secuencia, de Redes, de Clases, entre otros.*

# Agregar diagramas



Como podemos apreciar en este repaso, Markdown es un lenguaje de marcado ideal para construir documentación efectiva y completa.

Va mucho más allá de poder estilizar un texto, y referenciar algún hipervínculo y/o imagen simple.

# estructura de la documentación API RESTFUL

# estructura de la documentación API RESTFUL

Cuando hablamos de aplicaciones Backend, debemos pensar en estructurar la documentación de éstas a través de un modelo que describa:

- El objetivo de la documentación
- Cuántos endpoints posee la API
- Detallar el rol de cada endpoint
- Proporcionar ejemplos de uso
- Detallar información adicional
- Actualizar la documentación



# estructura de la documentación API RESTFUL

## El objetivo de la documentación

Antes de crear la documentación, debemos tener claro el objetivo de la misma:

- ¿Qué información necesita el equipo frontend para utilizar los endpoints de nuestra aplicación backend?
- ¿Qué nivel de detalle debemos incluir en la documentación?
- ¿Qué tipo de herramienta utilizaremos para crearla?



# estructura de la documentación API RESTFUL

## **Cuántos endpoints posee la API**

El siguiente paso es que podamos identificar todos los endpoints de la API, y sus correspondientes métodos HTTP: (GET, POST, PUT, DELETE, etc.). Qué parámetros se requieren y qué respuesta obtendrán con cada uno de ellos.

Básicamente, detallar el camino feliz de la funcionalidad de cada endpoint.



# estructura de la documentación API RESTFUL

## Detallar el rol de cada endpoint

Una vez especificados los endpoints de la API, es importante documentar cada uno de ellos.

Para cada endpoint, es importante que incluyamos una descripción detallada de qué hacen, los parámetros necesarios para que respondan correctamente, el formato en el cual nos retornan cada respuesta, como también los posibles errores y condiciones bajo las cuales se devuelven estos errores.





# estructura de la documentación API RESTFUL

## Proporcionar ejemplos de uso

Es recomendable incluir ejemplos de uso para cada endpoint. El objetivo de dichos ejemplos es que muestren cómo utilizarlos, con diferentes combinaciones de parámetros.

También es clave detallar dentro de la documentación cómo interpretar la respuesta del endpoint.



# estructura de la documentación API RESTFUL

## Detallar información adicional

Puedes incluir información adicional en la documentación, como la versión de la API, la autenticación requerida para utilizar la API, los límites de velocidad de la API, las políticas de privacidad y seguridad, etc.

También es importante aclarar cualquier limitación de URLs para acceder a la API, entre otros puntos claves relacionados a la seguridad; algo de suma importancia hoy en día.



# estructura de la documentación API RESTFUL

## Actualizar la documentación

Es importante mantener la documentación actualizada, ya que cualquier cambio en la API debe ser reflejado en la documentación para evitar confusiones o errores.

Lo ideal es evolucionar la documentación, a la par de cada etapa de construcción de los endpoints. Esto hace más fácil su elaboración, y menos pesado nuestro día a día dado que diversificamos nuestras tareas cotidianas.



# Espacio de trabajo

# Espacio de trabajo

Para poder desarrollar un trabajo 100% efectivo, te pedimos que reflotes el proyecto de Express JS y MongoDB, para construir la documentación de cada uno de los endpoints que lo componen.

Para poder apreciar el trabajo completo del desarrollo de nuestra API RESTFUL de frutas, te invitamos a poner en práctica el desarrollo de la documentación asociada...

**UNTREF**

UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO

# Espacio de trabajo

## 1. Construir una introducción al proyecto backend de frutas:

- a. debes detallar al inicio de la documentación la URL base <http://localhost:3008/api/v1/>
- b. deberás crear un ejemplo de uso de cada uno de los métodos GET - POST - PUT - DELETE
- c. incluye un ejemplo de código del cuerpo del mensaje para los métodos POST - PUT
- d. no incluyas el método PATCH si es que lo construiste oportunamente
- e. incluye un ejemplo del archivo .env para explicar cómo definir la conexión a MongoDB
- f. no es necesario incluir el nivel de seguridad de la API vía JWT

## 2. debes integrar en la creación de esta documentación:

- a. títulos y subtítulos, más formateo estándar de párrafos con Markdown
- b. integrar un sumario en el encabezado del documento Markdown
- c. una tabla general que represente los endpoint a utilizar y la ruta base de cada uno
- d. utilizar el formato código, para representar los bloques de código de ejemplo
- e. al menos un gráfico construido con Markdown, en la sección que consideres apropiada

Postear todo en Github de forma pública. Compartir el link de acceso a tu trabajo práctico.

**UNTREF**

UNIVERSIDAD NACIONAL  
DE TRES DE FEBRERO

```
const questions = ['dudas', 'consultas', '🧐']
```



```
> node gracias.js
```