

## Desarrollo Web Full Stack Node

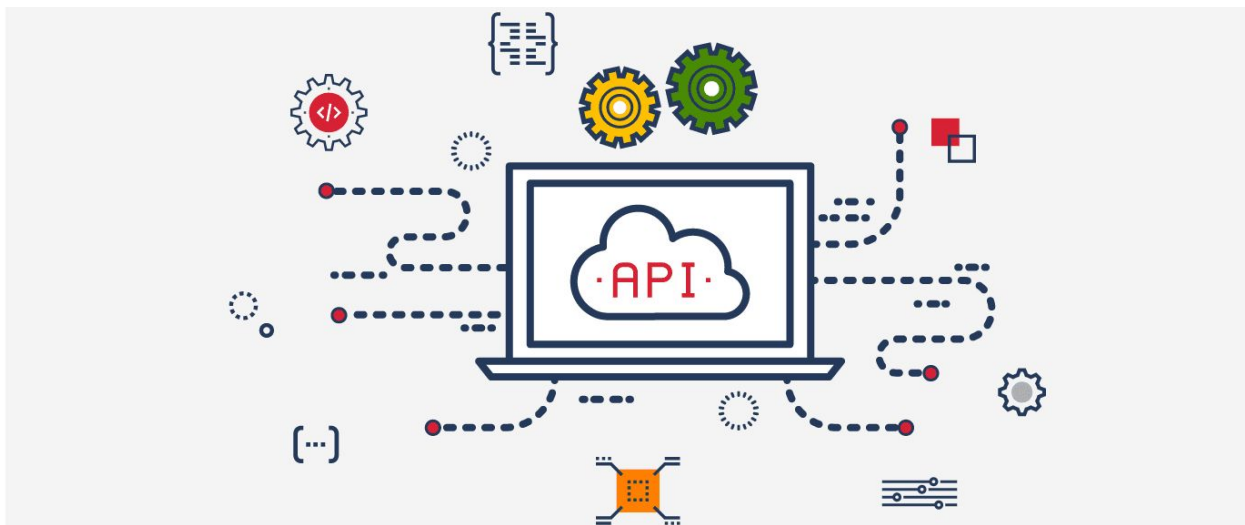
Trabajo integrador - M07C01

# Ejercitación - APIs 1

### Introducción

Hasta ahora toda la comunicación entre el cliente y el navegador sigue el esquema habitual de pedido y respuesta, donde Node entrega la vista terminada al navegador. Si el cliente quiere actualizar una parte de ese contenido, es necesario enviar todo de nuevo.

Vamos a ver cómo podemos usar las APIs para crear componentes dinámicos dentro de nuestras páginas.



## Consignas

Se acuerdan del viejo y querido **Mercado Liebre**® pues les vamos a estar entregando un proyecto funcionando con todo el BREAD de productos terminado y sus vistas funcionando.

Su desafío sería el de instalar, configurar y poner en funcionamiento el proyecto como primera tarea y luego crear APIs de productos que permitan dinamizar las vistas existentes.

Si bien para esta ejercitación no es estrictamente necesario, les recomendamos que si no lo hicieron aún, se instalen algún cliente REST como [Postman](#) o [Insomnia](#).

Algunas sugerencias de cómo encarar el ejercicio:


- Revisen el proyecto para determinar cómo está armado. La estructura es la de MVC que ya vimos, pero algunas cosas pueden estar programadas de una manera diferente 😊.
- Recuerden que por lo general la lógica de las APIs será muy similar a la que ya está programada en las rutas y controladores existentes.
- Es importante probar cada endpoint que creen antes de implementarlo para detectar cualquier falla, ahí es donde los clientes REST van a brillar ✨🤖✨.

Bueno, basta de tanta intro y tanto consejo, vamos con esos desafíos 🤓👉✨.

## Desafío 1 – La puesta en marcha

Muchas veces les va a tocar trabajar con código existente, especialmente si entran a un equipo de desarrollo que ya tiene trabajo en progreso. Este es uno de esos casos, donde les toca hacer funcionar el proyecto de alguien más.

Recuerden los pasos que ya vimos durante las clases:

- Descargar el proyecto → [Link de descarga](#) 
- Instalar las dependencias de Node
- Crear la base de datos
- Configurar el acceso a la base de datos (ojo que este proyecto usa archivos **.env**).
- Iniciar el proyecto

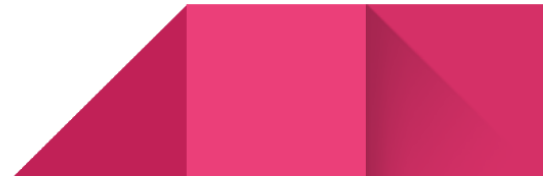
## Desafío 2 – Los endpoints de la home

Si se fijan, tenemos un controlador principal para la home. Este controlador está haciendo dos consultas a la base de datos, una para los **productos recientes** y otra para los **productos con descuento**.

Su tarea será la de crear dos endpoints que entreguen esos mismos datos. Les dejamos un paso a paso para ayudarlos en la tarea 🤖🔗

- Crear el controlador de productos de la api → **/controllers/api/productsController.js**
- Crear las rutas de productos de la api → **/routes/api/productsRouter.js**
- Agregar las rutas al archivo **app.js**
- Programar el endpoint de últimos productos → **/api/products/latest**
- Programar el endpoint de productos con descuento → **/api/products/offers**
- Probar que ambos endpoints funcionen correctamente

Para facilitarles la tarea les pasamos la estructura de cómo deberían entregar los datos de los productos.

[illegible]

## Desafío 3 – Consumiendo los endpoints en la home

Ya tenemos los endpoints listos y funcionando, ahora queda implementarlos.

Para esto vamos a tener que hacer un pedido desde la home y en función de los datos que nos lleguen mostrar los productos en los listados.

Vamos con otro paso a paso para ayudarlos en la tarea 🤖💡

- Agregar un script a la home → **public/js/products.js**
  - Cuando termine de cargar la página, hacer el pedido a la API de últimos productos.
  - Cuando nos llegue el resultado, modificar el contenedor de últimos productos para agregar los datos que llegaron → </api/products/latest>
  - Hacer lo mismo para el endpoint de productos en oferta → </api/products/offers>



BOSS LEVEL!

## Desafío 4 (opcional) – Los endpoints de categorías

Si hacés click en el link de categorías del header, vas a ver un listado de productos y categorías.

La idea del desafío es crear un endpoint de productos que pueda tomar una categoría como parámetro opcional. Si nos llega una categoría, vamos a traer los productos de esa categoría, si no nos llega, traemos todos los productos.

La segunda parte será implementar este pedido al hacer click en cada uno de los botones de categorías.

Vamos con ese paso a paso para este nivel final 🤓👉

### Creando el endpoint

- Agregar la nueva ruta en el ruteador → `/routes/api/productsRouter.js`
- Agregar la nueva acción en el controlador → `/controllers/api/productsController.js`
- Programar el endpoint de últimos productos → `/api/products/{category}`
- Probar que el endpoint funcione correctamente

### Consumiendo el endpoint

- Agregar un script a la página de categorías → `public/js/categories.js`
  - Cuando termine de cargar la página, hacer el pedido a la API de todos los productos.
  - Cuando nos llegue el resultado, modificar el contenedor productos para agregar los datos que llegaron → `/api/products/latest`
- Agregar un evento a cada uno de los botones de categorías
  - Al hacer click, llamar a la ruta con la categoría correspondiente.
  - Cuando vuelvan los resultados actualizar el listado y el título para que diga en qué categoría estamos.

## Cierre

Tal vez no lo vean ahora, pero las APIs son una manera muy poderosa de conectar aplicaciones. Ya hablamos muchas veces de reutilizar código ¿verdad 🤔?

Supongamos que desarrollamos una API de productos completa... entonces la lógica de creación, listado, búsqueda, edición y borrado ya está encapsulada. Ahora esa misma funcionalidad la pueden usar:

- El frontend que ve el cliente
- El backend que ve el administrador
- **[Spoiler alert 🚫]** El dashboard que vamos a desarrollar con React

7

- Una APP para dispositivos móviles
- Otros sitios que quieran listar nuestros productos
- [Inserte infinitos ejemplos aquí]

Así que les toca ponerse a pensar qué APIs van a desarrollar en su proyecto integrador 🤖🚀.