**Trabajo práctico**

Antes de empezar, te cuento algunas cosas de utilidad para poder realizar el ejercicio.

**¿Conoces la Serie de Fibonacci?**

Si no la conoces, un breve extracto de Wikipedia:

(Fuente [Click aquí para leer más acerca de la serie de Fibonacci](https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci))

En matemáticas, la **sucesión o serie de Fibonacci** es la siguiente sucesión infinita de números naturales:

*1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597*

A los elementos de esta sucesión se les llama **hijos de Fibonacci**. Esta sucesión fue descrita en Europa por Leonardo de Pisa, matemático italiano del siglo xiii también conocido como **Fibonacci**. Tiene numerosas aplicaciones en ciencias de la computación, matemática y teoría de juegos. También aparece en configuraciones biológicas, como por ejemplo en las ramas de los árboles, en la disposición de las hojas en el tallo, en las flores de alcachofas y girasoles, en las inflorescencias del brécol romanesco, en la configuración de las piñas de las coníferas, en la reproducción de los conejos y en cómo el ADN codifica el crecimiento de formas orgánicas complejas. De igual manera, se encuentra en la estructura espiral del caparazón de algunos moluscos, como el nautilus.

**Y, ¿Por qué les cuento esto?**, cuando desarrollamos servicios Restful, no todo es crear la firma del servicio (URL, parámetros, etc), nuestros servicios ¡¡¡Tienen lógica!!!. Acá empieza el desafío.

**¡¡Empecemos!! Utilizando NodeJS y ExpressJS**

1. Crear un proyecto de NodeJS utilizando ExpressJS
2. Instala Nodemon (No olvides que Nodemon es una dependencia de desarrollo).
3. Crea un servicio que retorne una cierta cantidad de números de la serie de Fibonacci.
4. La cantidad de números a retornar debe ser enviada al servicio como parámetro. ¿Cúal crees que es el mejor tipo de parámetro? ¿Que verbo utilizar? 😉
5. Sí, al realizar la petición no se envía ningún parámetro, por defecto retornar los primeros 20 números de la serie.
6. Si como parámetro no se envía un número, retornar un error (un mensaje de error) pero acompáñalo de un status code que creas que encaja con el mal envío del parámetro (si no lo sabes, lo puedes googlear).

Ya entramos en calor. Ahora viene **la segunda parte del ejercicio**.

Asumiendo que queremos crear un backend para gestionar usuarios, y estos usuarios tienen solamente los siguientes atributos: **ID, Nombre, Apellido, DNI**. (atención a las nomenclaturas)

Para este caso, queremos poder hacer operaciones **CRUD (Create, Read, Update, Delete)**. No se asusten! No hace falta utilizar un motor de base de datos. Para realizar el ejercicio es necesario crear un array en nuestra aplicación de manera global y que nuestros servicios utilicen dicho array, el objetivo es modificar el array que está en memoria.

Puntos a tener en cuenta:

1. Se debe crear un proyecto nuevo utilizando NodeJS y ExpressJS
2. El ID no se puede repetir ya que es unívoco para cada elemento del array (puede ser un número incremental).
3. Validar todos los posibles errores:
   1. Al crear un usuario todos los datos previamente mencionados son obligatorios.
   2. Al actualizar o eliminar un registro, se debe hacer por ID.
   3. Se debe contemplar en el servicio que retorna los usuarios, la posibilidad OPCIONAL de recibir un ID y retornar sólo el que coincida con ese parámetro o un error si ese ID no existiese en el array. (¡Pensá bien!, ¿Qué tipo de parámetro crees que es el indicado para esto? y, ¿para actualizar y eliminar?)
   4. El servicio de consulta de usuario debe permitir filtrar por nombre y/o apellido.

**Tips:**

Recuerden cuando se debe utilizar cada verbo, qué tipo de parámetro se utiliza y por qué dependiendo la situación. Es importante conocer los status codes de HTTP y a la hora de emitir una respuesta en nuestro servicio acompañarlo del status que corresponda. Si tienen dudas pueden leer más acerca de los status code de HTTP [**ACÁ.**](https://developer.mozilla.org/es/docs/Web/HTTP/Status)

**Entrega del Trabajo Práctico**

El ejercicio debe subirse a un repositorio de Git (Github por ejemplo, debe ser público).

Puede realizarse de manera **individual o grupal** (en caso de ser grupal, en el archivo Readme indicar los participantes y los mails de cada participante).

Se valora el correcto envío de parámetros utilizando query, path o body params según corresponda y el correcto verbo http dependiendo la acción que se quiere realizar.

Estar atento a la nomenclatura de variables y funciones como así también al correcto uso de Camel Case. Preferentemente, utilicen nomenclatura en Inglés (OPCIONAL).

Prestar atención a los posibles fallos y errores que puedan originarse por mal envío de parámetros, falta de datos, etc. Los mensajes de error los pueden determinar ustedes.

El proyecto lo pueden realizar de la manera que prefieran, en cuanto a organización y estructura de carpetas. Aún así, recomiendo basarse en el curso.

Junto al proyecto se debe subir la colección de Postman que utilizaron para probar sus servicios. (La colección de Postman se puede exportar).

¡Éxitos!