

Taller Express y otras cosas



Nombre: Francisco Raziel Andalón Aguayo

Matrícula: A01640235

Película favorita y razón: Sueño de fuga es mi película favorita porque combina muy bien una historia sobre esperanza con personajes memorables. Tiene un mensaje sobre la resistencia del espíritu humano y me inspira cada vez que la veo.

El objetivo de la actividad es explorar y conocer más sobre el desarrollo web del lado del backend.

Al final de la actividad deberás hacer un PDF con este documento. El PDF deberá tener un enlace en supertarea.html

- **Ejercicio 0: El pasado**

En las semanas previas vimos la forma de consumir un API desde el cliente (el navegador). Ahora consumirás el **API desde el servidor**. Usa javascript y nodejs.

Selecciona un API que no requiera autenticación.

El servidor deberá consumir el API, procesarlo de alguna manera (ejemplo: seleccionar un dato) y posteriormente entregar ese dato (o datos) al cliente (navegador).

Sube tu archivo a tu repositorio.

Escribe aquí abajo el enlace al archivo en tu repositorio:
<https://FranciscoAndalon.github.io/ejercicio0elpasado.html>

¿Cuál fue el principal problema para resolver este ejercicio? ¿Cuál fue el problema que tuvo una de las personas en tu equipo? Explica cómo resolvió esa persona ese problema.

- **Ejercicio 1: Simplicidad**

¿Recuerdas cómo **instalar un módulo** en node.js? Escribe la instrucción que se usa:
npm install nombre-del-modulo

- **Ejercicio 2: Express es**

Investiga qué es express (en el contexto de desarrollo web en nodejs). Explica a un **niñ@** qué es express, cómo se usa, para qué sirve. Escribe ese texto aquí. Incluye también **las referencias** de tu investigación.

Node.js es como la base sobre la que armas una casa de bloques. Es un programa que le deja a tu computadora entender un lenguaje especial llamado JavaScript, el

cual se usa mucho para hacer que las páginas de internet funcionen y hagan cosas interesantes.

Express es como una caja de herramientas que se construye encima de Node.js y hace que sea mucha más fácil y rápido construir las paredes, las ventanas y las puertas de tu página de internet.

Para usar express, imaginate que quieres que cuando alguien llegue a la puerta principal de tu página de internet, salga un mensaje que diga "Hola". Entonces con express puedes decirle a Node.js que cuando alguien llegue a la puerta principal, le enseñe un mensaje que diga "Hola". Con Express vienen las herramientas para acomodar estas órdenes.

Express sirve para que los que hacemos páginas de internet y aplicaciones las podamos hacer más rápido y más fácil, y si no se usara Express se tendría que trabajar más y estaría más complicado organizar todo.

Dile a una persona de la clase (que no esté en tu mesa) que te explique qué es express y qué ventajas te podría dar su uso. Graba el audio de la explicación.

¿Crees que realizó una buena explicación? ¿Por qué?

• Ejercicio 3: Desde un libro

Investiga una definición en un libro (digital o físico) **de una API**. Incluye la referencia del libro.

"APIs are contracts that define how applications, services, and components communicate."

Referencia:

Geewax, John J. (2021). API Design Patterns. Manning Publications.

Escribe un ejemplo de un endpoint de tu proyecto.
/escuelas/:id/donaciones

• Ejercicio 4: Instala express

Escribe la instrucción que se usa para **instalar el módulo de express**:
npm install express

• Ejercicio 5: Uso de express

Ejecuta y explica que hace el siguiente código:

```
import express from 'express';

const app = express();

app.listen(1984, () => {
  console.log('Up and up');
});
```

Este código importa el framework Express.js, crea una nueva aplicación Express, inicia un servidor web que escucha en el puerto 1984 y cuando el servidor inicia correctamente, muestra el mensaje “Up and up” en la consola. Básicamente, está creando un servidor web básico usando Express.js.

• Ejercicio 6: Uso de express++

Ejecuta y explica que hace el siguiente código:

```
app.get('/bienvenida', (req, res) => {
  res.send('Esto no es una página html');
});

app.get('/otraBienvenida', (req, res) => {
  res.sendFile('bienvenida.html');
});
```

Este código define dos rutas en el servidor Express, la primera responde a solicitudes GET en la URL “/bienvenida” enviando el texto “Esto no es una página html” y la segunda responde a solicitudes GET en la URL “/otraBienvenida” intentando mandar un archivo que se llama “bienvenida.html”

En caso de que alguno de estos te marque error, soluciona el problema y explica la forma de solucionarlo:

En el segundo bloque de código hay un problema con la segunda ruta (
`res.sendFile('bienvenida.html');`) porque ocupa una ruta absoluta al archivo, no solamente el nombre del archivo. Express no puede ubicar un archivo si no sabe su ubicación completa.

Se puede solucionar el problema usando
app.get('/otraBienvenida', (req, res) => {

```
res.sendFile(__dirname + '/bienvenida.html');
});
```

• Ejercicio 7: Recuerdo de imagen

Recuerdas que la imagen no se podía ver en <https://github.com/sgiomatec/act2025>. Un servidor entrega respuestas a solicitudes realizadas por un cliente. Cuando el cliente solicita una imagen (u otro tipo de archivo) el servidor tiene que saber responder.

Express nos propone crear un directorio para los archivos que queremos usar, por ejemplo imágenes o archivos de hojas de estilo.

Investiga qué es express.static. Escribe tu investigación.

Escribe tu investigación

Express.static es un middleware integrado en Express.js que deja servir archivos estáticos como imágenes, CSS, JavaScript y archivos html directamente desde el servidor. Cuando se configura nada más hay que especificar un directorio en el sistema de archivos y Express automáticamente hace disponibles todos los archivos de ese directorio por medio de la web sin tener que hacer rutas individuales para cada recurso. Así es mucho más fácil crear sitios web completos porque se pueden organizar los recursos en carpetas y Express los sirve eficientemente con los encabezados HTTP adecuados.

• Ejercicio 8: Película

Pregunta a una persona de otra mesa su película favorita y sus razones.

¿Ya viste esa película?. ¿Te gusta? ¿La verías si no la has visto? ¿Por qué?

• Ejercicio 9: Transformación

Usa express para transformar todo el código de servidor.js, nombra el archivo ahora servidor_express.js

Escribe aquí abajo el enlace al archivo en tu repositorio:

<https://FranciscoAndalon.github.io/ejercicio9transformacion.html>

• Ejercicio 10: Verbos

En el contexto de desarrollo web, explica los siguientes verbos GET, POST, PUT, DELETE

GET se usa para obtener información de un recurso sin cambiarlo.

POST se usa para mandar datos al servidor para crear un nuevo recurso.

PUT se usa para actualizar o crear un recurso en una ubicación específica.

DELETE se usa para borrar un recurso que exista.

Explica los siguientes códigos, en el contexto de desarrollo web:

1xx Indica que la solicitud se recibió y el servidor sigue procesándola.

2xx La solicitud se procesó con éxito.

3xx Indica que el cliente tiene que hacer más acciones para completar la solicitud.

4xx Indica que hay un problema con la solicitud del usuario.

5xx Indica que el servidor no pudo procesar la solicitud.

Explica con un ejemplo (relacionado con tu proyecto/reto) cómo se atendería una solicitud de cada uno de esos verbos usando express. Incluye al menos una respuesta 404, 201 y 200.

```
const express = require('express');
```

```
const app = express();
```

```
app.use(express.json());
```

```
let donadores = [];
```

```
// GET: Obtener información de un donador en específico
```

```
app.get('/new/get/donador/:id', (req, res) => {
  const donador = donadores.find(d => d.id === parseInt(req.params.id));
  if (!donador) {
    return res.status(404).json({ mensaje: "Donador no encontrado" });
  }
  res.status(200).json(donador);
});
```

```

// POST: Verificar login de un donador
app.post('/login/donador', (req, res) => {
  const { usuario, contraseña } = req.body;
  // Lógica de autenticación aquí
  const autenticado = usuario === "ejemplo" && contraseña === "1234";
  if (autenticado) {
    return res.status(200).json({ mensaje: "Login exitoso" });
  }
  res.status(401).json({ mensaje: "Credenciales incorrectas" });
});

// PUT: Registrar un nuevo donador
app.put('/new/donador', (req, res) => {
  const nuevoDonador = { id: donadores.length + 1, ...req.body };
  donadores.push(nuevoDonador);
  res.status(201).json({ mensaje: "Donador creado exitosamente", donador: nuevoDonador });
});

// DELETE: Eliminar una cuenta de usuario
app.delete('/eliminar/cuenta/:id', (req, res) => {
  const index = donadores.findIndex(d => d.id === parseInt(req.params.id));
  if (index === -1) {
    return res.status(404).json({ mensaje: "Cuenta no encontrada" });
  }
  donadores.splice(index, 1);
  res.status(200).json({ mensaje: "Cuenta eliminada correctamente" });
});

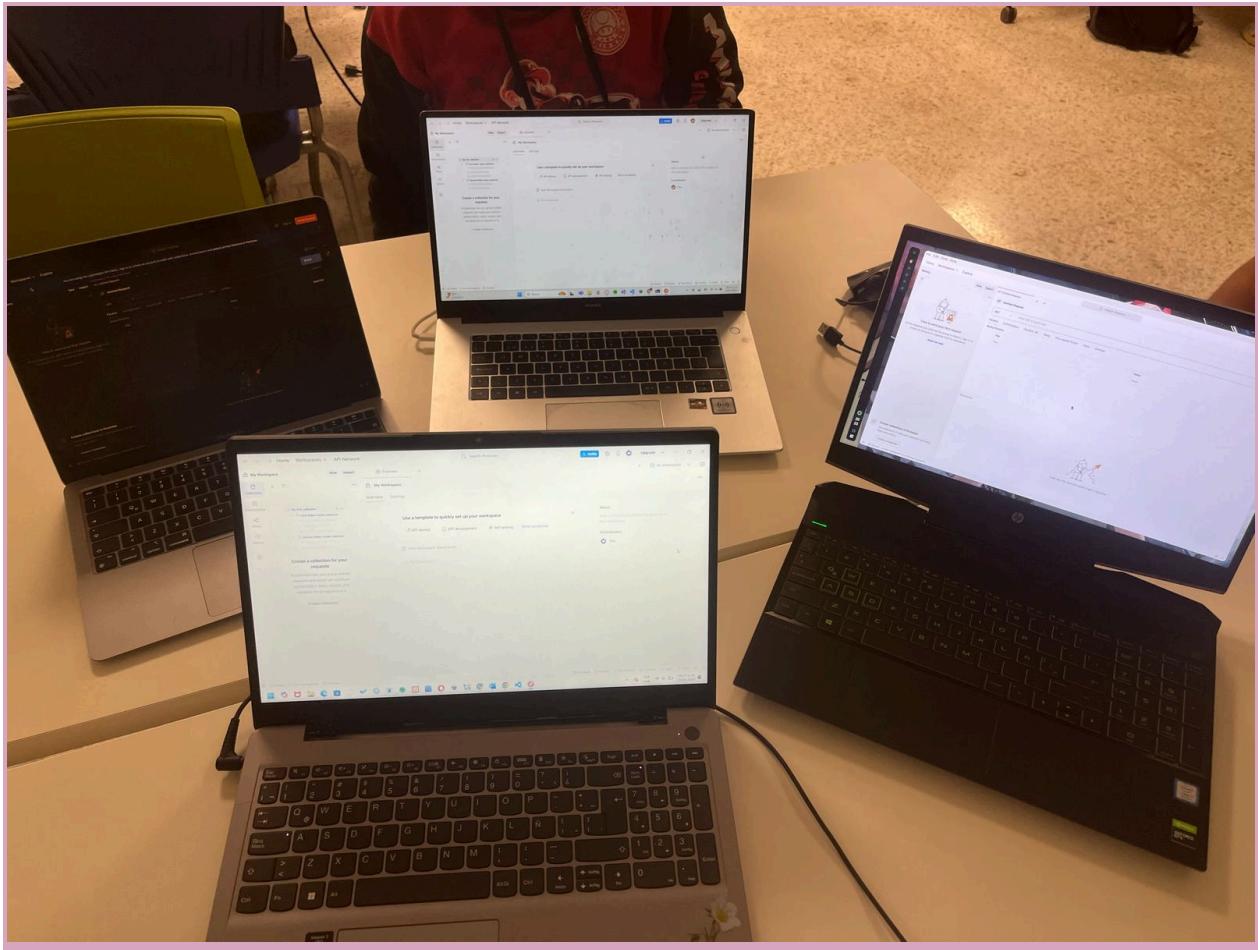
app.listen(3000, () => console.log("Servidor en ejecución en puerto 3000"));

```

• Ejercicio 11: Postman

En equipo con tu mesa, instalen postman en cada una de sus computadoras. Tomen una misma foto en la que aparezcan todas las pantallas con postman abierto.

Foto aquí:



Explica para qué sirve Postman

Postman es una plataforma de colaboración para el desarrollo de API que deja diseñar, probar, documentar y compartir interfaces de programación de aplicaciones eficientemente. Funciona como un cliente HTTP avanzado que hace más fácil crear y enviar solicitudes a servidores web sin tener que escribir código, permitiendo elegir diferentes métodos, añadir encabezados, cuerpos de mensaje y parámetros.

Prueba lo que hiciste en el ejercicio 10 con Postman. Pon una captura de pantalla usando el DELETE.

• Ejercicio 12: Parámetros

Inventa un ejercicio (que esté relacionado con escuelas/donantes) para que uses parámetros en las rutas (por ejemplo /getEscuelas)

Crear una ruta en Express que permita obtener información sobre escuelas dependiendo de su nivel educativo (primaria, secundaria, preparatoria). La información se tiene que obtener por medio de un parámetro en la URL, por ejemplo:

/getEscuelas/primaria

/getEscuelas/secundaria

/getEscuelas/preparatoria

Si el nivel educativo no existe, la ruta tiene que regresar un código 404 (Not Found).

[Soluciona ese ejercicio](#)

Escribe aquí abajo el enlace al archivo en tu repositorio:

<https://FranciscoAndalon.github.io/ejercicio12parametros.html>

Compara ese ejercicio con el de otra persona de la clase. ¿Qué ejercicio ayuda mejor a comprender el uso de parámetros y por qué?

• Ejercicio 13: npx

Investiga con alguien más lo siguiente: npx express-generator. Importante haz un directorio diferente si quieres ejecutar el comando.

Escribe el nombre de ese alguien más:

Explica lo que genera y hace ese comando.

El comando crea automáticamente la estructura básica de una aplicación web Express.js. Cuando se ejecuta genera un proyecto con una estructura de directorios y archivos preconfigurados que siguen las mejores prácticas para desarrollar aplicaciones con Express.

Explica de manera básica y conceptualmente qué archivos se generaron.

app.js: El archivo principal de la aplicación que configura Express y conecta todos los componentes.

package.json: Define las dependencias del proyecto y los scripts para ejecutar la aplicación.

bin/www: Script que inicia el servidor HTTP y configura el puerto de escucha.

public/: Carpeta para almacenar archivos estáticos como CSS, JavaScript e imágenes que serán accesibles directamente desde el navegador.

routes/: Contiene archivos de JavaScript que definen las rutas de la aplicación.

views/: Almacena las plantillas para generar HTML dinámico.

node_modules/: Se crea cuando se instalan las dependencias con npm install.

• Ejercicio 14: Explicar

Explica a una niña o un niño de 10 años para qué sirven frameworks como Angular, React, Vue, o Svelte.

Hacer una página de internet es como armar un juguete muy complicado, pero los frameworks son como cajas de herramientas especiales que sirven para que armar ese juguete sea mucho más fácil. Entonces en lugar de tener que dibujar cada pieza desde cero, estas herramientas ya vienen con muchas piezas listas para usar. Te ayudan a hacer páginas de internet que cambian y responden cuando las usas, como cuando das clic en un botón y aparece algo, sin tener que escribir tantas instrucciones complicadas. Básicamente es como tener bloques de construcción mágicos que hacen que tu página cobre vida rápidamente.