

Taller Express y otras cosas



Nombre: **Eloy Rodríguez Bonilla**

Matrícula: **A00227629**

Película favorita y razón: **Whiplash, porque se puede interpretar de muchas maneras, y es una representación perfecta de la obsesión en cualquier tipo de hobby que tenga alguien**

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.

eloyrb0.github.io/eloy_rodriguez_A00227629/servidor.js

¿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.

Considero que la gran parte de la realización de ese ejercicio es el formateado de los headers y de las estructuras json, para que estas fueran accedidas de manera correcta, y los headers para que el servidor pueda servirlos al usuario como debe de ser. Otros compañeros tenían el problema de que no mostraban adecuadamente el json de la información de los donantes/escuelas, pero esto lo resolvieron acomodando simplemente los corchetes y llaves de manera adecuada.

• Ejercicio 1: Simplicidad

¿ Recuerdas cómo **instalar un módulo** en node.js? Escribe la instrucción que se usa:
Se instala con el comando **npm install <nombre del módulo/paquete>**

• 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.

Express es básicamente unas herramientas para hacerte la vida más fácil cuando haces un sitio web. En vez de tener que escribir todo desde 0, Express hace que ya tengas código hecho para poder hacer conexiones entre cosas más fácilmente. Primero lo descargas de internet, y ya estaría listo para usarse. Solo tendrías que avisar/escribir en tu código que lo vas a usar. Sirve para muchas cosas, como conectarte con otras páginas más fácil y rápido, y muchas cosas más.

Referencias:

<https://expressjs.com/>

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.

 express_explicado.mp3

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

Encapsuló todo de manera muy efectiva, resumiendo los puntos claves, al ser básicamente un facilitador de muchas funciones de node que para un desarrollador web tomarían un tiempo considerable de implementar de 0.

• Ejercicio 3: Desde un libro

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

“Una API es una aplicación con la cual nos podemos comunicar para poder obtener información o realizar ciertas acciones.”

Aprende a Trabajar con Web APIS - Alberto Ramírez Caballero

<https://cosasdedevelopers.com/static/dist/files/guia-para-aprender-a-trabajar-con-apis-versión-1.pdf>

Escribe un ejemplo de un endpoint de tu proyecto.

Un endpoint de un administrador sería por ejemplo (url del sitio/admin/usuarios/...) y sería un DELETE, cuando borra a un usuario de la base de datos, ya sea escuela o aliado. Todavía no está definido el path de esta acción, pero sabemos que va a existir

• Ejercicio 4: Instala express

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

```
$ npm install express --save
```

• 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');
});
```

Básicamente inicializa un servidor express y lo hace escuchar en el puerto 1984, mostrando el mensaje “Up and up” cuando inicializa.

• 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');
});
```



Crea las rutas /bienvenida y /otraBienvenida si se incluye este código sumado al anterior. Así que ahora al entrar al puerto 1984 con la ruta /bienvenida, nos muestra el mensaje “Esto no es una página html”. La otra ruta envía un archivo html llamado bienvenida, pero al no estar creado y guardado en el directorio del proyecto, nos arroja un error. Crear el archivo bienvenida.html es precisamente lo que se necesita hacer en el directorio para poder servirlo al ingresar a esa ruta, sumado a la importación de otras dependencias para poder identificar esa ruta:

```
1  import express from 'express';
2  import path from 'path';
3  import { fileURLToPath } from 'url';
4
5  const app = express();
6
7  const __filename = fileURLToPath(import.meta.url);
8  const __dirname = path.dirname(__filename);
9
10 app.listen(1984, () => {
11   console.log('Up and up');
12 });
13
14 app.get('/bienvenida', (req, res) => {
15   res.send('Esto no es una página html');
16 });
17
18 app.get('/otraBienvenida', (req, res) => {
19   res.sendFile(path.join(__dirname, 'bienvenida.html'));
20 });
21
```

• 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

Según el sitio oficial de Express, express.static es una función de middleware, que tiene la función de servir archivos estáticos, como imágenes, html, css, javascript, etc. Esta función se utiliza de la siguiente manera:

```
app.use(express.static(path.join(__dirname, 'public')));
```

Esto le indica a la aplicación el path en donde van a encontrarse todos los archivos estáticos que queremos servir, y los hace accesibles.

• 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?

The Batman, ya la he visto varias veces y me gusta mucho como película, porque tiene muy buena dirección artística y cinematografía.

• 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:
eloyrb0.github.io/eloy_rodriguez_A00227629/servidor_express.js

• Ejercicio 10: Verbos

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

GET: Se usa comúnmente para recuperar datos del servidor

POST: Se usa para enviar datos al servidor, como cuando se registra algo en una base de datos

PUT: Se usa para actualizar un recurso en el servidor

DELETE: Se usa para eliminar un recurso en el servidor

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

1xx: Son informativos, por ejemplo 101 es cambio de protocolo, 100 es que el servidor recibió todo correctamente y el cliente puede seguir enviando datos

2xx: Significan éxito, como 200 que es OK, 201 que es que se creó algo, etc.

3xx: Significan redireccionamiento, 301 significa que una URL migró, 302 es una redirección temporal, etc.

4xx: Errores del cliente, como 400 que es una solicitud incorrecta, 401 es no autorizado, 404 es no encontrado, etc.

5xx: Errores del servidor, 500 es error interno, 503 es servicio no disponible

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.

En el contexto del proyecto que estamos haciendo:

GET /escuelas

- Descripción: Obtiene una lista de todas las escuelas activas en la app.
- Mensaje de confirmación: Retorna una lista de escuelas con estado aceptado mostrando solamente datos públicos.
- Mensajes de error:
 - 404 Not Found: Escuelas no encontradas.
 - 500 Internal Server Error: Error en el servidor.

POST /registro-perfil

- Descripción: Registra un nuevo perfil de usuario en la base de datos.
- Mensaje de confirmación: 201 Created: Perfil registrado exitosamente.
- Mensajes de error:
 - 400 Bad Request: Datos inválidos.

PUT usuario/{id}/actualizar-notificaciones

- Descripción: Actualiza las notificaciones de un usuario específico por el id.
- Mensaje de confirmación: 200 OK: Notificaciones actualizadas correctamente.
- Mensajes de error:
 - 404 Not Found: No se pudo actualizar correctamente.

DELETE /eliminar-perfil/{id}

- Descripción: Permite eliminar un perfil de usuario.
- Mensaje de confirmación: 200 OK: Perfil eliminado exitosamente.
- Mensajes de error:
 - 403 Forbidden: No está autorizado para eliminar el perfil.
 - 404 Not Found: Perfil no encontrado.

Usando express el código se vería más o menos así:

```

import express from 'express';
const app = express();
app.use(express.json()); // Para manejar JSON en solicitudes POST y PUT

// GET /escuelas
app.get('/escuelas', (req, res) => {
  const escuelas = [] // Simula una base de datos
  if (escuelas.length === 0) return res.status(404).send('Escuelas no encontradas');
  res.status(200).json(escuelas);
});

// POST /registro-perfil
app.post('/registro-perfil', (req, res) => {
  if (!req.body.nombre) return res.status(400).send('Datos inválidos');
  res.status(201).send('Perfil registrado exitosamente');
});

// PUT /usuario/:id/actualizar-notificaciones
app.put('/usuario/:id/actualizar-notificaciones', (req, res) => {
  const usuarioExiste = true; // Simulación
  if (!usuarioExiste) return res.status(404).send('No se pudo actualizar
correctamente');
  res.status(200).send('Notificaciones actualizadas correctamente');
});

// DELETE /eliminar-perfil/:id
app.delete('/eliminar-perfil/:id', (req, res) => {
  const autorizado = true; // Simulación
  if (!autorizado) return res.status(403).send('No está autorizado para eliminar el
perfil');

  const perfilExiste = true; // Simulación
  if (!perfilExiste) return res.status(404).send('Perfil no encontrado');

  res.status(200).send('Perfil eliminado exitosamente');
});

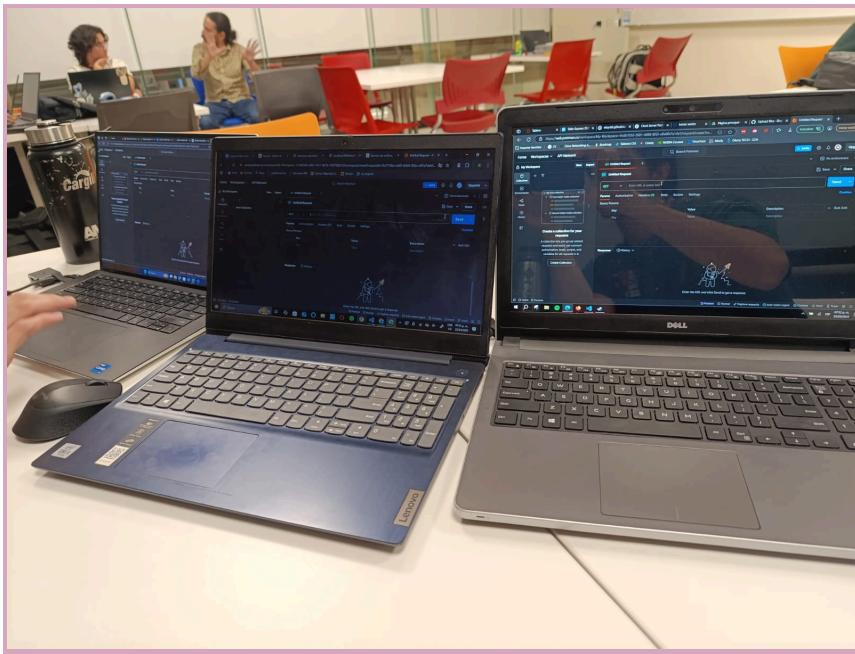
```

```
});  
  
// Iniciar servidor  
app.listen(1984, () => {  
  console.log('Servidor corriendo en el puerto 1984');  
});
```

• 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

Básicamente Postman sirve para probar APIs y sus endpoints haciendo requests directamente con esa herramienta, sin tener que abrir servidores webs o haciendo requests con url o con programación.

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

The screenshot shows a Postman request for a DELETE operation at `http://localhost:1984/eliminar-perfil/1`. The response status is `200 OK` with a response time of 63 ms and a body size of 257 B. The response body contains the message: `1 Perfil eliminado exitosamente`.

• Ejercicio 12: Parámetros

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

El ejercicio será una ruta que te permita obtener la información de una escuela poniendo su nombre en la URL

Soluciona ese ejercicio

The screenshot shows a Postman request for a GET operation at `http://localhost:1984/api/escuelas/Escuela%20Agustin%20turbide`. The response status is `404 Not Found` with a response time of 8 ms and a body size of 277 B. The response body contains the message: `1 { 2 | "mensaje": "Escuela no encontrada" 3 }`.

Escribe aquí abajo el enlace al archivo en tu repositorio:

eloyrb0.github.io/eloy_rodriguez_A00227629/servidor_ejercicio.js

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

El ejercicio de Alejandro Gutiérrez es similar, siento que ambos tienen una efectividad muy comparable como ejercicio demostrativo del uso de parámetros en rutas. Igual son ejercicios muy sencillos y se entiende en los dos que estamos recibiendo algo dando parámetros

• Ejercicio 13: npx

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

Escribe el nombre de ese alguien más:

Alejandro Gutiérrez

Explica lo que genera y hace ese comando.

Genera una estructura estándar, como plantilla de un proyecto express.js, con carpetas y archivos ya organizados.

Explica de manera básica y conceptualmente qué archivos se generaron.

app.js es el archivo principal, y ahí se van a configurar todas las rutas del proyecto. routes/index.js y routes/users.js son diferentes rutas base del servidor, bin/www es el archivo que inicia el servidor en el puerto que se defina. views/ son plantillas HTML que pueden ser renderizadas con Pug o algún otro motor. /public es para archivos estáticos, como imágenes o más elementos que el frontend puede servir.

• Ejercicio 14: Explicar

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

El código y la programación es difícil de entender. Para gente que programa, a veces ver código de alguien más se ve como si estuviera en otro idioma, no entiendes, porque todos tienen su manera de arreglar y resolver problemas. Entonces qué pasa cuando alguien quiere ayudarlos, pero no entienden qué está haciendo el otro. Para eso sirven en gran parte los “frameworks”, para que todos programen en el “mismo lenguaje” y todos se entiendan. Simplifica mucho las cosas, sobre todo porque casi siempre estos “frameworks” ya tienen varios problemas resueltos que los programadores normalmente enfrentan por su cuenta.