

# Módulo 4: Ejercicio de evaluación final

---

Antes de empezar a realizar el ejercicio, hay que crear un nuevo repositorio desde GitHub Classroom usando [este enlace](#). Una vez creado, hay que clonarlo en nuestro ordenador y en la carpeta creada empezaremos a trabajar en el ejercicio. El ejercicio consiste en desarrollar un API que permita **insertar, modificar, listar y eliminar** información utilizando **ExpressJS, Nodes** y una base de datos **MySQL**.

## Instrucciones

1. Selecciona un tema de tu interés para crear una API. Podría ser una API para gestionar una librería, un sistema de inventario, una aplicación para manejar eventos, un chat, una enciclopedia de animales, etc.
2. Desarrolla una API REST que permita realizar operaciones CRUD sobre una entidad principal del tema seleccionado.
3. Asegúrate de utilizar **ExpressJS** para el servidor, **Node** para el manejo del backend, y **MySQL** para la base de datos.
4. Proporciona la documentación básica de cómo un cliente podría consumir el API en el fichero **README .md** o usando una biblioteca como **Swagger**.

### Sección 1: Diseño de la Base de Datos

Crea el esquema de la base de datos para su API, incluyendo tablas, columnas y relaciones. Debes incluir el diagrama de Entidad-Relación y el código SQL para crear este esquema en tu repositorio (también puedes incluir algunas filas de datos).

### Sección 2: Configuración del Servidor

- Escribe el código para configurar un servidor Express.js y conectarse a la base de datos.
- Implementa las funciones necesarias para el manejo de JSON y para cualquier otra funcionalidad que considere necesaria.

### Sección 3: API RESTful

Define las rutas para las siguientes operaciones y escribe los endpoints correspondientes:

- Insertar una entrada en su entidad principal.
- Leer>Listar todas las entradas existentes.
- Actualizar una entrada existente.
- Eliminar una entrada existente.

## BONUS

Si te sobra tiempo, te proponemos otros endpoints para realizar que **no son obligatorios**. Consiste en implementar un sistema de autenticación con JWT (JSON Web Tokens) que incluya las funcionalidades de registro y inicio de sesión.

Te dejamos algunas pistas...

## Instrucciones sobre el bonus

1. Crea una tabla llamada **usuarias** que contenga la siguiente información:
  - "**usuarias  - "**id  - "**email  - "**nombre  - "**password**********
2. Los endpoints a desarrollar son:
  - **Registro de usuaria (POST /registro)**: esta ruta debe permitir el registro de una nueva usuaria. El cuerpo de la solicitud debe incluir el nombre, el correo electrónico y la contraseña de la usuaria:

```
{  
  "user": "usuaria",  
  "pass": "contraseña1234"  
}
```

La contraseña debe ser encriptada antes de almacenarla en la base de datos. Genera y devuelve un token JWT al usuario registrado:

```
{  
  "success": true,  
  "token": token  
}
```

En caso de error, devolverá un json con el error:

```
{  
  "success": false,  
  "error": error  
}
```

- **Inicio de sesión (POST /login)**: esta ruta debe permitir que una usuaria existente inicie sesión. El cuerpo de la solicitud debe incluir el correo electrónico y la contraseña de la usuaria:

```
{  
  "user": "usuaria",  
  "pass": "contraseña1234"  
}
```

Verifica las credenciales proporcionadas con los registros de la base de datos. Si son válidas, genera y devuelve un token JWT para autenticar a la usuaria.

3. Implementa un middleware de autenticación que verifique el token JWT en cada solicitud del API. Si el token es válido, permite que la solicitud continúe; de lo contrario, devuelve un error de autenticación.

## Otros bonus

Algunas ideas que os proponemos como ejercicios de repaso para cuando os pongáis a repasar el módulo de programación backend que visten y completan vuestro proyecto:

1. Instala alguna librería como **dotenv** para gestionar la contraseña y datos de acceso a la base de datos con variables de entorno.
2. Puedes subir el servidor de la API a algún **servicio como Render** para que esté disponible en Internet.
3. Puedes agregar un carpeta configurada con **un servidor de estáticos** en el que haya una pequeña aplicación Frontend que permita consultar alguno de los endpoint del API.
4. Puedes instalar y configurar la **librería Swagger** para generar una página web con la documentación de los endpoints de vuestra API.

## Entrega

Hemos pautado 5 horas de dedicación al ejercicio (**¡¡¡sin contar los bonus!!!**), el límite de entrega es el

**29 de diciembre a las 14:00 horas**

Solo debéis hacer commits y merges en la rama main/master de vuestro repositorio hasta la fecha límite (pero podéis usar libremente otras ramas en el repositorio). Si después del ejercicio queréis seguir trabajando sobre el ejercicio, lo podéis hacer en otra rama y no debéis hacer merge hasta que vuestra tutora os envíe las calificaciones.

## Normas

Este ejercicio está pensado para que lo realices de forma individual en clase, pero podrás consultar tus dudas con la profesora y tus compañeras si lo consideras necesario. Ellas no te darán directamente la solución de tu duda, pero sí pistas para poder solucionarla. Aún facilitando la comunicación entre compañeras, durante la prueba no debes copiar código de otra persona ni acceder a su portátil. Confiamos en tu responsabilidad.

La evaluación es una buena oportunidad para conocer cómo estás progresando, saber qué temas debes reforzar durante las siguientes semanas y cuáles dominas. Te recomendamos que te sientas cómoda con el ejercicio que entregues y no envíes cosas copiadas que no entiendas.

Si detectamos que has entregado código que no es tuyo, no entiendes y no lo puedes defender, suspendes el módulo. Tu objetivo no debería ser pasar la evaluación sino convertirte en programadora, y esto debes tenerlo claro en todo momento.

Al final tendrás un feedback sobre aspectos a destacar y a mejorar en tu ejercicio, y sabrás qué objetivos de aprendizaje has superado.

**¡Al turrón de pistacho!!**