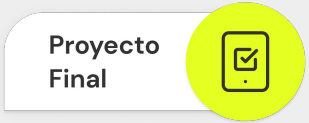
PRIMERA ENTREGA DEL PROYECTO FINAL

Deberás entregar el avance de tu aplicación eCommerce Backend correspondiente a la primera entrega de tu proyecto final.

CODER HOUSE

PRIMERA ENTREGA DEL PROYECTO FINAL

Formato: link a un repositorio en Github con el proyecto cargado.
Sugerencia: no incluir los node_modules



>>Consigna: Deberás entregar el estado de avance de tu aplicación eCommerce Backend, que implemente un servidor de aplicación basado en la plataforma Node.js y el módulo Express. El servidor implementará dos conjuntos de rutas agrupadas en routers, uno con la url base '/api/productos' y el otro con '/api/carrito'. El puerto de escucha será el 8080 para desarrollo y process.env.PORT para producción en glitch.com

>>Aspectos a incluir en el entregable:

1. El **router base '/api/productos'** implementará cinco funcionalidades:
    a. GET: '/' - Me permite listar todos los productos disponibles (disponible para todes)
    b. GET: '/:id' - Me permite listar un producto por su id (disponible para todes)
    c. POST: '/' - Para incorporar productos al listado (disponible solo para administradores)
    d. PUT: '/:id' - Actualiza un producto por su id (disponible solo para administradores)
    e. DELETE: '/:id' - Borra un producto por su id (disponible solo para administradores)

CODER HOUSE

PRIMERA ENTREGA DEL PROYECTO FINAL

Formato: link a un repositorio en Github con el proyecto cargado.
Sugerencia: no incluir los node_modules

2. El **router base '/api/carritos'** implementará cinco rutas, disponibles para usuarios y administradores:
   a. POST: '/' - Crea un carrito y devuelve su id.
   b. POST: '/:id_carrito/productos' - Para incorporar productos al carrito, enviando el id de producto en el cuerpo de la petición.
   c. GET: '/:id_carrito/productos' - Me permite listar todos los productos guardados en el carrito
   d. DELETE: '/:id_carrito/productos/:id_prod' - Eliminar un producto del carrito por su id de carrito y de producto.
   e. DELETE: '/:id_carrito' - Vacía un carrito.

3. Crear una variable booleana administrador, cuyo valor configuraremos más adelante con el sistema de login. Según su valor (true ó false) me permitirá alcanzar o no las rutas indicadas. En el caso de recibir un request a una ruta no permitida por el perfil, devolver un objeto de error. Ejemplo: *{ error : -1, descripcion: ruta 'x' método 'y' no autorizada }*

PRIMERA ENTREGA DEL PROYECTO FINAL

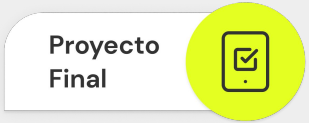
Formato: link a un repositorio en Github con el proyecto cargado.
Sugerencia: no incluir los node_modules

5. Un **producto** dispondrá de los siguientes campos: id, nombre, descripcion, foto (url), precio.
6. El **carrito de compras** tendrá la siguiente estructura:
   { id, productos: [{ id, nombre, descripcion, foto (url), precio }] }
   ó
   { id, productos: [idProd, …] }
7. Realizar la persistencia de productos y del carrito de compras en el File System.

CODER HOUSE

PRIMERA ENTREGA DEL PROYECTO FINAL

Formato: link a un repositorio en Github con el proyecto cargado.
Sugerencia: no incluir los node_modules

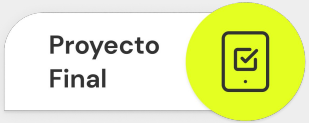


>>A tener en cuenta:

1. Para realizar la **prueba de funcionalidad** hay dos opciones:
   a. Probar con postman cada uno de los endpoints (productos y carrito) y su operación en conjunto.
   b. Realizar una aplicación frontend sencilla, utilizando HTML/CSS/JS ó algún framework de preferencia, que represente el listado de productos en forma de tarjetas. En cada tarjeta figuran los datos del producto, que, en el caso de ser administradores, podremos editar su información. Para este último caso incorporar los botones *actualizar* y *eliminar*. También tendremos un formulario de ingreso de productos nuevos con los campos correspondientes y un botón *enviar*. Asimismo, construir la vista del carrito donde se podrán ver los productos agregados e incorporar productos a comprar por su id de producto. Esta aplicación de frontend debe enviar los requests *get*, *post*, *put* y *delete* al servidor utilizando fetch y debe estar ofrecida en su espacio público.

PRIMERA ENTREGA DEL PROYECTO FINAL

Formato: link a un repositorio en Github con el proyecto cargado.
Sugerencia: no incluir los node_modules



2. En todos los casos, el diálogo entre el frontend y el backend debe ser en formato JSON. El servidor no debe generar ninguna vista.
3. En el caso de requerir una ruta no implementada en el servidor, este debe contestar un objeto de error: ej { error : -2, descripcion: ruta 'x' método 'y' no implementada}
4. La estructura de programación será ECMAScript, separada tres en módulos básicos (router, lógica de negocio/api y persistencia). Más adelante implementaremos el desarrollo en capas. Utilizar preferentemente clases, constructores de variables let y const y arrow function.
5. Realizar la prueba de funcionalidad completa en el ámbito local (puerto 8080) y en glitch.com