Examen Segunda Evaluación

Introducción

Este proyecto tiene como objetivo crear una API REST con MongoDB como única base de datos y una aplicación cliente programada únicamente con HTML, CSS y JavaScript (sin Angular).

La API gestionará:

- Registro y autenticación de usuarios.
- Registro de logs y errores dentro de *la propia colección MongoDB*
- Gestión de productos asociados a usuarios
- Diferenciación entre dos tipos de usuarios: usuarios normales y admins

La aplicación cliente (HTML+CSS+JS) deberá consumir esta API, mostrar los datos obtenidos y permitir la interacción según permisos del usuario.

API MongoDB

Toda la información (usuarios, productos y logs) se almacenará en MongoDB, cada uno en su colección correspondiente. Habrá peticiones autenticadas, para hacerlo tendréis que poner en la cabecera x-authentication el token y x-admin si es administrador o no.

```
// Token y admin normalmente los guardas tras el login:
// localStorage.setItem("token", token);
// localStorage.setItem("admin", admin); // true/false
const token = localStorage.getItem("token");
const isAdmin = localStorage.getItem("admin"); // "true" o "false"

fetch("http://localhost:3000/products", {
  method: "GET",
  headers: {
      "x-authentication": token,
      "x-admin": isAdmin
  }
})
  .then(res => {
      if (!res.ok) throw new Error("Error al cargar productos: " +
res.status);
      return res.json();
  })
  .then(data => {
      console.log("Productos:", data);
      // TODO ALUMNADO: pintar productos en el DOM
  })
  .catch(err => {
      console.error(err);
      // TODO ALUMNADO: enviar err.message a POST /log
  });
```

En el API tendréis que guardar esos tokens generados y si el usuario es administrador o no, por ejemplo lo puedes hacer en una lista, o en BBDD, como tú consideres.

Colección: Logs

La API deberá registrar logs y errores enviados desde el frontend.

Endpoints

POST /log

Request Body:

```
{
  "message": "string",
  "level": "string"
}
```

GET /log

Response Body:

```
[
  {
    "message": "string",
    "level": "string",
    "time": "date"
  }
]
```

GET /log/date/:date

Devuelve todos los logs a partir de la fecha indicada.\ Response Body:

```
[
  {
    "message": "string",
    "level": "string",
    "time": "date"
  }
]
```

Colección: Products

GET /products

Requiere autenticación.\ Devuelve todos los productos creados por el usuario autenticado o todos si es administrador.

Response Body:

```
[
  {
    "id": "string",
    "name": "string",
```

```
    "description": "string",
    "price": "number",
    "user": {
      "name": "string",
      "id": "string",
      "dateCreation": "date"
    }
  }
]
```

POST /products

Requiere autenticación.\ Request Body:

```
{
  "name": "string",
  "description": "string",
  "price": "number"
}
```

Response Body:

```
{
  "id": "string",
  "name": "string",
  "description": "string",
  "price": "number",
  "user": {
    "name": "string",
    "id": "string",
    "dateCreation": "date"
  }
}
```

GET /products/:id

Requiere autenticación.

DELETE /products/:id

```
Solo accesible para administradores.\ Response Body:
{
  "message": "Producto eliminado correctamente"
}
```

Colección: Users

Autenticación

POST /register

Request Body:

```
{
  "name": "string",
  "password": "string",
  "admin": "boolean"
}
```

``

POST /login

```
Request Body:
{
  "name": "string",
  "password": "string"
}
```

Response Body:

```
{
  "token": "string",
  "admin": "boolean"
}
```

Aplicación Cliente (HTML + CSS + JavaScript)

La aplicación se desarrollará únicamente con HTML, CSS y JavaScript, sin frameworks.

Por defecto aparecerá una pantalla de login.\ En la base de datos existirán dos usuarios precreados: un usuario normal y un admin.

Login Page

El usuario deberá iniciar sesión introduciendo:

- Nombre
- Contraseña

Si el login es correcto, se redirigirá a Product List Page.

Product List Page

Mostrará todos los productos creados por el usuario logueado.\ Campos:

- Nombre del producto
- Descripción
- Precio

Si el usuario es administrador, deberá ver todos los productos, no solo los suyos.

Admin Panel Page

Solo visible para administradores.

Debe incluir:

- Nombre del producto
- Descripción
- Precio
- Usuario creador
- Botón para eliminar productos

Product Form Page

Formulario disponible para todos los usuarios.\ Permite crear un producto enviándolo al endpoint correspondiente de la API.

Gestión de Errores

Cuando ocurra cualquier error en el frontend (peticiones fallidas, validaciones, excepciones JS…), deberá enviarse a la API: POST /log

Rúbrica del Examen

Elemento	Puntuación
API Logs (MongoDB)	1
API Products	2
Login	1
Product List	1
Admin Panel	1.5
Product Form	1.5
Gestión de errores en cliente	1.5
SOLID / KISS / DRY	0.5
TOTAL	10

Entregable

Debéis entregar la URL del repositorio de GitHub, incluyendo:

- Código de la API (Node.js + MongoDB)
- HTML, CSS y JavaScript del cliente