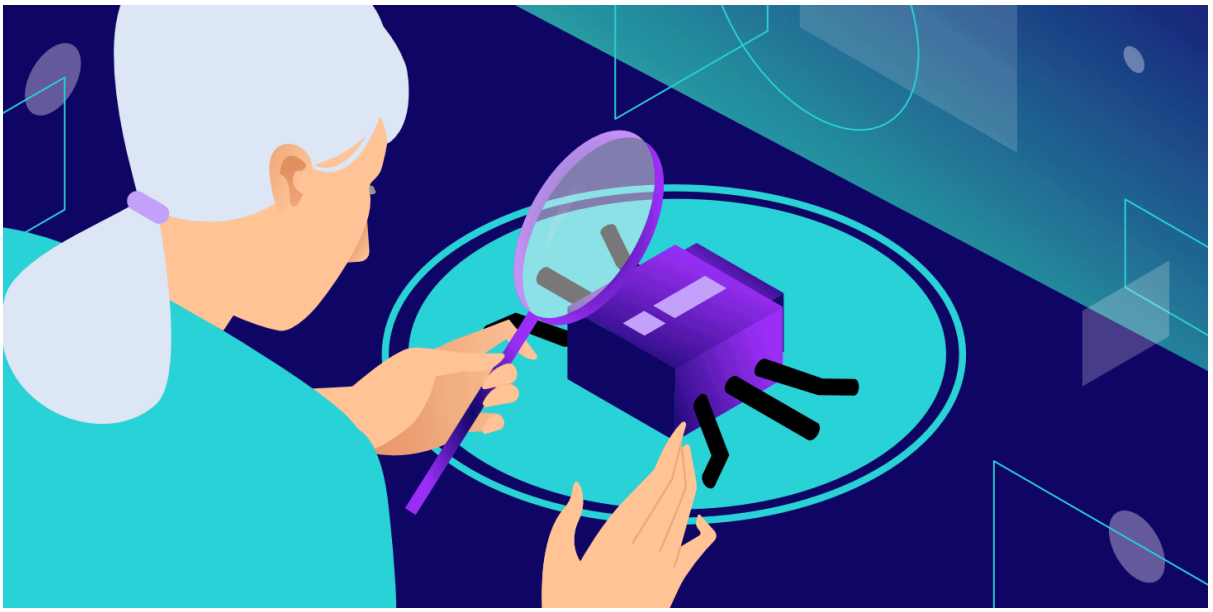
Análisis de la Aplicación: Proceso de Depuración



A continuación, se detallan las pruebas realizadas para depurar la aplicación, en particular en el proceso de autenticación de usuarios a través de la API.

1. Revisión de la Solicitud en el Cliente (Frontend)

El objetivo es verificar que los datos enviados desde el frontend sean correctos y estén bien formateados para que el servidor pueda procesarlos sin errores.

Para cumplir con eso, se revisó el cuerpo de la solicitud para asegurarse de que los campos requeridos, como email y password, estuvieran correctamente incluidos y en el formato esperado por la API.
Se validó que el tipo de contenido en las cabeceras de la solicitud fuera el adecuado (application/json) para indicar que los datos se estaban enviando en formato JSON

Para revisar eso, se imprimieron los datos enviados desde el frontend para confirmar que la estructura de la solicitud fuera correcta:

```
console.log(formData);
```

2. Inspección de la Respuesta del Servidor (Backend)

El objetivo en este caso es obtener información más detallada sobre la respuesta del servidor para entender la causa del error 400 - Bad Request por ejemplo

Para ello se imprimió el mensaje de error detallado para obtener más contexto sobre el fallo:

```
} catch (error) {
  setMessage(error.response.data.message);
  console.error('Error al hacer la solicitud:', error.response.data);
  console.error('Código de estado:', error.response.status);
  console.error('Mensaje:', error.response.statusText);
}
```

3. Verificación de las Validaciones en el Backend

El objetivo es asegurarse de que el servidor esté validando correctamente los datos de entrada y rechazando solicitudes que no cumplan con los criterios esperados.

Para asegurarse de que la entrada sea correcta se revisaron las validaciones para el formato del email, longitud de la password, que existan los datos y cualquier otro dato requerido.

```
if (!nombre || !correo || !password) {
  return res.status(400).json({ message: 'El nombre, correo y contraseña son obligatorios' });
}
```

Pruebas Realizadas al Usar MongoDB Atlas.

1. Conexión a MongoDB Atlas

El objetivo es verificar que la aplicación pueda conectarse correctamente a la base de datos en MongoDB Atlas desde el entorno local o desde el servidor de producción.

Para ello se probó la conexión utilizando las credenciales correctas del clúster de MongoDB Atlas (usuario, contraseña y URI de conexión).

```
PORT = 4000
MONGODB_URI = mongodb+srv://eliaserperezyaque:LuVNVtCastSPn6wt@clusterprueba.lusk0.mongodb.net/?retryWrites=true&w=majority&appName=ClusterPrueba
```

```
// Cadena de conexión
const URI = process.env.MONGODB_URI || 'mongodb://localhost/dbtest';
// const URI = process.env.MONGODB_URI ? process.env.MONGODB_URI : 'mongodb://localhost/dbtest';


mongoose.connect(URI)
    .then(() => console.log('La base de datos ha sido conectada: ', URI))
    .catch(error => console.error('Error al conectar con la base de datos:', error));
```

3. Comprobación de la Integridad de la Base de Datos.

El objetivo es asegurarse de que los datos estén correctamente insertados y estructurados en la base de datos de MongoDB Atlas.

Para ello se subieron datos de prueba a la base de datos y se verificó que se almacenarán correctamente.


```
_id: ObjectId('67cb0f1076c947623081e3d8')
nombre : "elia"
correo : "elia@gmail.com"
password : "$2b$10$b9oTj3gwBP.HJ3w5Kp23.uipfVIYtcAgL0fdxteCmmkaf.O.Q8Xm6"
foto : "noFoto.png"
createdAt : 2025-03-07T15:21:52.190+00:00
updatedAt : 2025-03-07T15:21:52.190+00:00
__v : 0
```


Documentación del Proceso de Optimización y Despliegue

1. Optimización del Código

Antes de realizar el despliegue, es crucial asegurarse de que el código de la aplicación sea eficiente y escalable. Esto incluye tanto el código backend como las interacciones con la base de datos.

Se ha comprobado que el código funciona de manera esperada realizando pruebas y para su optimización por si sucede algún error, se han usado try catch para evitar que la aplicación pare su ejecución debido al error, por otro lado también se han utilizado métodos async/await en el momento de realizar las llamadas a la api.

1.2 Optimización del Frontend.

Para que el frontEnd sea optimizado se deben llevar a cabo ciertos puntos extra como la compresión de imágenes para que no ocupen demasiado espacio en servidor, también se implementa lazy loading para que se carguen los componentes sólo cuando sea necesario y así no saturar la aplicación.

2. Despliegue de la Aplicación

El despliegue de esta aplicación ha sido sencilla ya que solo se necesitaba usar algo como MongoAtlas para que cumpliera con ese requisito, pero aun así, se ha vigilado que se cumplan todos los requisitos necesarios a la hora de usar la propia base de datos usando archivos de configuración para que los datos que no se deben ver en el código, no se vean y aun asi cumplan con su función.

En resumen, el proceso de optimización y despliegue de una aplicación incluye varias etapas clave que van desde la optimización del código y el rendimiento hasta la configuración de un entorno de producción escalable.