Servicio Nacional De Aprendizaje - SENA

Tecnólogo en Análisis y Desarrollo de Software - 2834891

API del Proyecto. GA7-220501096-AA5-EV04.

Por

Christofer Jordan Homen

Presentado a

Israel Arbona

Bogotá D.C., 24 - Noviembre – 2024

Introducción

Esta evidencia documenta la implementación y el testeo de una API REST desarrollada para la aplicación "Performance Radio". La API proporciona funcionalidades CRUD (Crear, Leer, Actualizar, Eliminar) para la gestión de oyentes de la aplicación. En este documento se detalla el proceso de prueba realizado utilizando la herramienta Postman, capturando cada paso mediante pantallazos y explicando los objetivos y resultados de cada prueba.

Objetivos

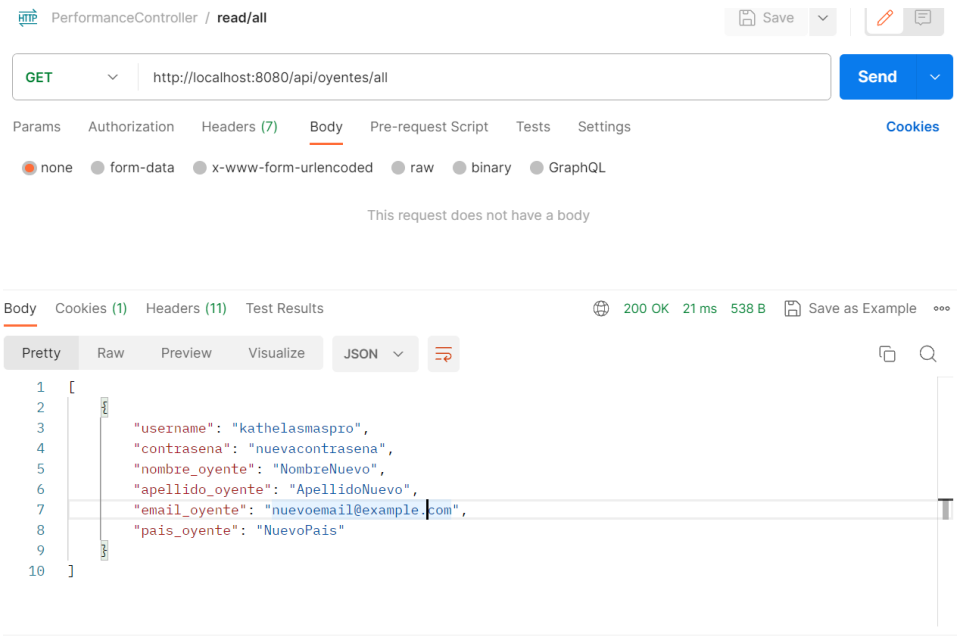
- Probar la funcionalidad de la API para verificar su correcto funcionamiento.
- Asegurar que los endpoints creados gestionen adecuadamente la información de los oyentes.
- Documentar los resultados de cada prueba realizada para evidenciar la estabilidad y funcionalidad de la API.

Endpoints Probados

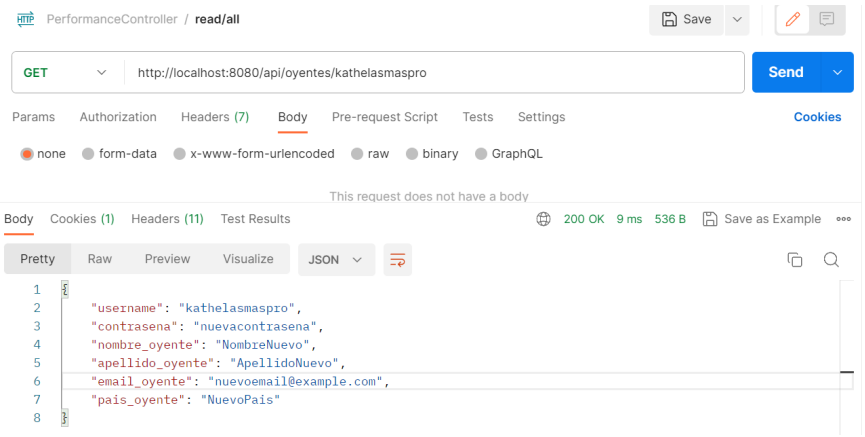
1. **GET /api/oyentes/all**: Devuelve una lista de todos los oyentes.
2. **GET /api/oyentes/{username}**: Devuelve un oyente específico identificado por su nombre de usuario.
3. **POST /api/oyentes/buscar**: Permite buscar un oyente enviando el nombre de usuario en el cuerpo de la solicitud.
4. **POST /api/oyentes**: Crea un nuevo oyente en la base de datos.
5. **PUT /api/oyentes/actualizar**: Actualiza la información de un oyente existente.
6. **DELETE /api/oyentes/eliminar/{username}**: Elimina un oyente especificado por su nombre de usuario.

Documentación de los Tests Realizados

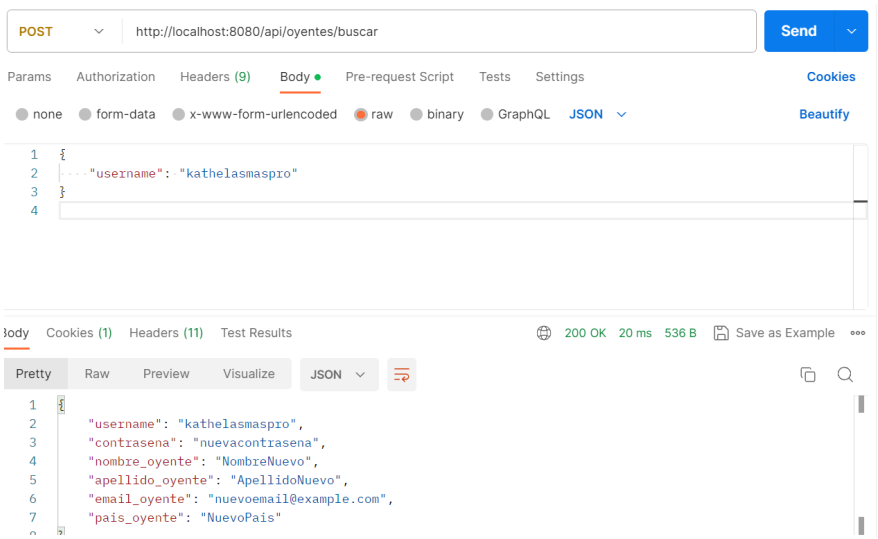
1. **Test GET /api/oyentes/all**
   - **Objetivo**: Verificar que el endpoint devuelva todos los oyentes almacenados en la base de datos.
   - **Resultado**: Solicitud exitosa, lista de oyentes mostrada correctamente.
   - **Pantallazo**: Captura mostrando la lista completa de oyentes devuelta por Postman.



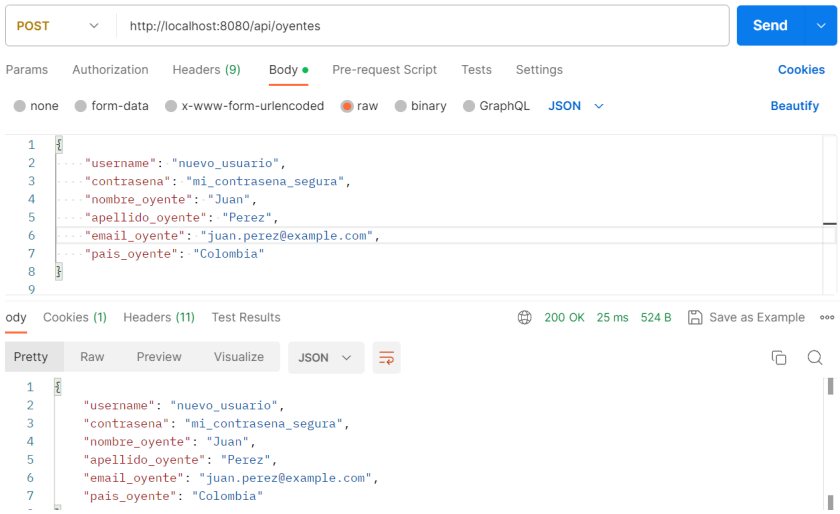
2. **Test GET /api/oyentes/{username}**
   - **Objetivo**: Obtener la información de un oyente específico mediante su nombre de usuario.
   - **Resultado**: Solicitud exitosa, los datos del oyente se devolvieron sin problemas.
   - **Pantallazo**: Captura mostrando el oyente encontrado con el nombre de usuario especificado.



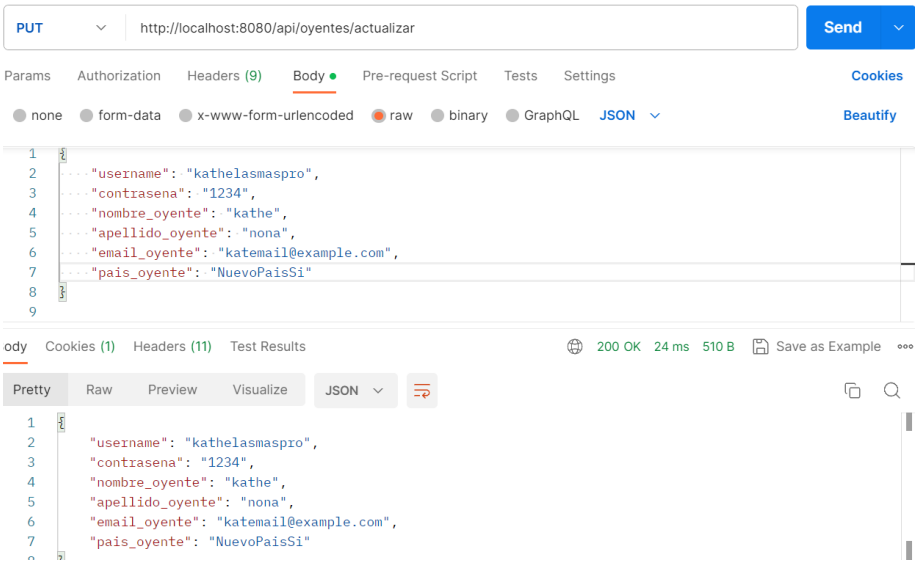
3. **Test POST /api/oyentes/buscar**
    - **Objetivo**: Comprobar la búsqueda de un oyente a partir del nombre de usuario enviado en el cuerpo de la solicitud.
    - **Resultado**: Solicitud exitosa, se encontró el oyente con el nombre de usuario proporcionado.
    - **Pantallazo**: Captura del resultado de la búsqueda mostrando la información del oyente.



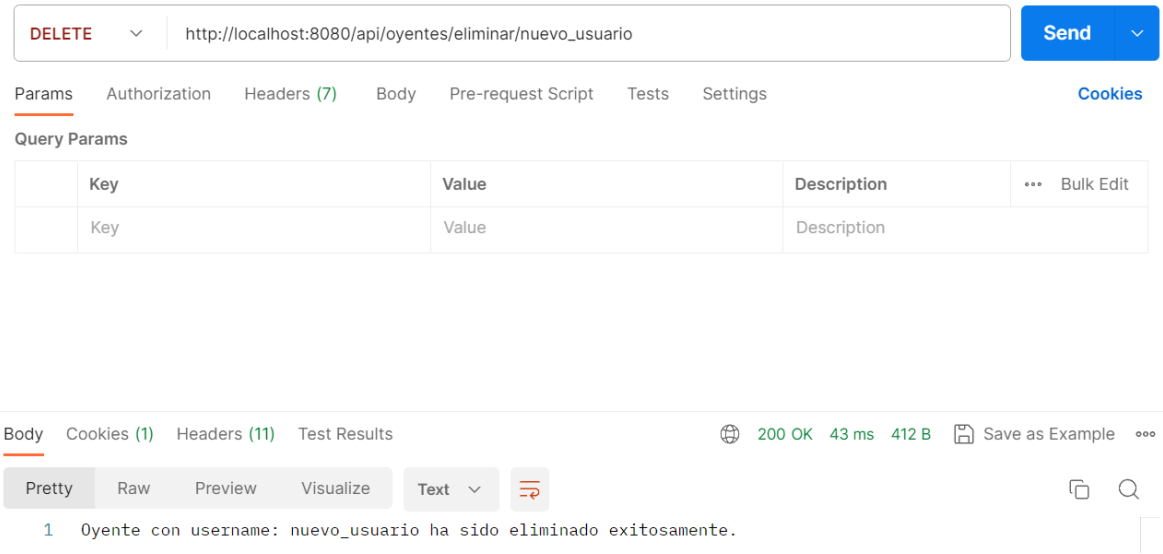
4. **Test POST /api/oyentes**
    - **Objetivo**: Crear un nuevo oyente en la base de datos.
    - **Resultado**: Solicitud exitosa, el oyente fue creado y almacenado correctamente.
    - **Pantallazo**: Captura del cuerpo de la solicitud con los datos del nuevo oyente y la respuesta confirmando su creación.



5. **Test PUT /api/oyentes/actualizar**
   - **Objetivo**: Actualizar los datos de un oyente ya existente.
   - **Resultado**: Solicitud exitosa, los datos del oyente fueron actualizados satisfactoriamente.
   - **Pantallazo**: Captura del cuerpo de la solicitud con los nuevos datos del oyente y la respuesta de la API confirmando la actualización.



6. **Test DELETE /api/oyentes/eliminar/{username}**
   - **Objetivo**: Eliminar un oyente específico de la base de datos usando su nombre de usuario.
   - **Resultado**: Solicitud exitosa, el oyente fue eliminado correctamente.
   - **Pantallazo**: Captura mostrando la confirmación de la eliminación del oyente.



Descripción de los Servicios Web Adicionales que se usaran

Tener en cuenta que estos no se pueden probar en postman ya que no son un crud sino un servicio multimedia, los js están en la evidencia anterior

1. **contactForm.js**: Este servicio permite gestionar el envío de un formulario de contacto mediante AJAX, evitando la recarga de la página y mejorando la experiencia del usuario al mantener la interacción fluida. Además, muestra un indicador de carga para informar al usuario sobre el estado de su solicitud.
2. **locutorInfo.js**: Este script permite que los usuarios obtengan más información sobre los locutores de la emisora. Al hacer clic en un botón de información, se muestra un pop-up con detalles sobre el locutor, lo cual enriquece la experiencia al conocer mejor a quienes están detras de los micrófonos.
3. **login.js**: Gestiona el inicio de sesión de los usuarios mediante la captura y envío de datos desde el formulario de inicio de sesión. Este servicio es esencial para la autenticación de usuarios registrados y permite el acceso a contenido personalizado.
4. **peli.js**: Integra una funcionalidad adicional de entretenimiento, obteniendo y mostrando información sobre películas populares a través de la API de TMDb. Esto le da al sitio web una característica diferenciadora al ofrecer recomendaciones de cine.
5. **playbtn.js**: Controla la reproducción y la pausa de la emisora. Este servicio permite a los usuarios manejar la transmisión de audio de manera intuitiva mediante un único botón.
6. **playimagen2.js**: Actualiza de manera dinámica la imagen del reproductor para reflejar la información actual de la transmisión. Este servicio proporciona un elemento visual importante que contribuye a la identidad del programa que está al aire.
7. **playvolume.js**: Permite a los usuarios ajustar el volumen del audio según sus preferencias. Con esto, se asegura una experiencia de escucha personalizada y cómoda.
8. **programs.js**: Muestra las imágenes de los programas de la emisora de manera aleatoria, lo cual agrega dinamismo a la página web y hace que la interacción visual sea más atractiva para el usuario.
9. **signup.js**: Similar a `login.js`, este servicio permite que los usuarios se registren en la plataforma. Envía la información del formulario de registro al servidor para su procesamiento.
10. **top5.js**: Utiliza la API de Spotify para obtener y mostrar las cinco canciones más populares. Esto agrega un componente interactivo y actualizado constantemente, ayudando a mantener el interés de la audiencia.
11. **whatsapp.js**: Muestra un botón de WhatsApp que permite a los usuarios ponerse en contacto rápidamente con la emisora para consultas, sugerencias o solicitudes. El botón aparece en función del desplazamiento en la página, asegurando que esté visible cuando el usuario lo necesite.
12. **album.js**: Similar a `top5.js`, este servicio utiliza la API de Spotify para mostrar detalles sobre un álbum específico, brindando información adicional que enriquece la experiencia musical del usuario.

13. **Integración con la API de Stripe**: La página de los locutores también incluye opciones para que los usuarios adquieran servicios como cuñas radiales y promociones en vivo mediante la integración con la API de Stripe. Esta funcionalidad permite a los usuarios realizar compras seguras de los servicios ofrecidos por los locutores directamente desde la web. Cada locutor cuenta con botones específicos que, al ser clicados, dirigen al usuario a la pasarela de pagos de Stripe, facilitando así la contratación de servicios.

Conclusiones

- Todos los endpoints de la API fueron probados y cumplieron con las expectativas funcionales. La API maneja correctamente las solicitudes de creación, lectura, actualización y eliminación de oyentes.
- La herramienta Postman facilitó la prueba de los distintos endpoints, permitiendo verificar la correcta comunicación con la base de datos y la gestión de la información.
- La documentación y los pantallazos de cada prueba realizada permiten evidenciar que la API está lista para ser utilizada en entornos reales.