

Descripción general de la aplicación: [OrangeHRM](#) es un sistema de gestión de recursos humanos (HRMS) que permite a las organizaciones administrar procesos relacionados con sus empleados.

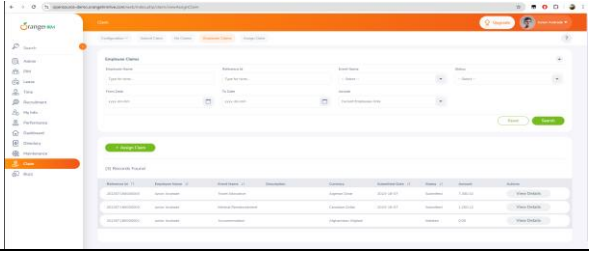
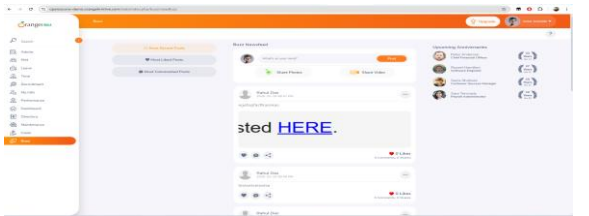
Cuenta con los siguientes módulos:

Módulo	Descripción de la funcionalidad
Admin	 <p>Configuración del sistema, roles, permisos y estructura organizacional</p>
PIM (Personal Information Management)	 <p>Gestión de datos personales y laborales de los empleados.</p>
Leave	 <p>Administración de solicitudes y aprobaciones de licencias</p>
Time	 <p>Control de asistencia y horas trabajadas</p>

Reto - Pruebas funcionales



Recruitment		Proceso de reclutamiento y selección de candidatos
My Info		Información personal del empleado
Performance		Evaluaciones de desempeño
Dashboard		Indicadores generales del sistema.
Directory		Directorio de empleados
Maintenance		Respaldo y

		mantenimiento de datos.
Claim		Reembolsos
Buzz		red social interna (en versiones avanzadas)

Metodología y Estrategia de Pruebas

Para abordar adecuadamente el reto de pruebas funcionales sobre la aplicación OrangeHRM, se definió la siguiente metodología y estrategia de trabajo:

Metodología de Pruebas

Se utilizará un enfoque basado en pruebas funcionales manuales, complementadas con pruebas exploratorias para validar flujos críticos y detectar posibles defectos no contemplados en los casos estándar.

Pruebas de humo para entendimiento funcional

Dado que se cuenta con información limitada sobre la aplicación, se realizaron inicialmente pruebas de humo con el propósito principal de:

- Familiarizar al equipo de QA con la interfaz y funcionalidades básicas del sistema.
- Entender el comportamiento general y flujo de los módulos críticos, como el inicio de sesión, administración y gestión de empleados.
- Verificar la estabilidad básica del sistema para asegurar que está en condiciones de continuar con pruebas más detalladas.
- Estas pruebas permitieron obtener una visión general que facilitó la posterior

planificación y diseño de casos y escenarios de prueba, así como identificar áreas críticas que requerían mayor enfoque.

Estrategia de ejecución

- El proyecto se desarrollará bajo un enfoque práctico, con un plazo de ejecución de tres días. Durante este período, se implementará la solución propuesta de los ítems del reto técnico de pruebas funcionales.
- Se priorizarán las pruebas en módulos de alto riesgo y mayor impacto funcional.
- Se documentarán todos los hallazgos y defectos, clasificándolos según severidad y prioridad.
- Al finalizar, se entregará el enlace al repositorio donde se encuentran la colección de Postman, los scripts SQL.
- Además, se proporcionarán documento con la solución del **Reto pruebas funcionales**

Fuera del Alcance

Con el fin de mantener el enfoque dentro del tiempo y alcance definidos para el reto técnico, se establecen las siguientes exclusiones:

- No se realizarán pruebas automatizadas ni de rendimiento, dado que el enfoque del ejercicio está centrado en la validación funcional manual.
- No se abordarán pruebas de seguridad, integración con sistemas externos o validaciones de correo electrónico.
- No se realizarán pruebas sobre configuraciones avanzadas del sistema ni personalizaciones específicas del entorno de OrangeHRM.
- Las pruebas se limitarán a los módulos críticos priorizados (**Inicio de sesión (Login), Administración (Admin) y Gestión de empleados (PIM)**), sin incluir módulos adicionales como: **Recruitment, Leave, Time, Performance, Directory, Maintenance, Dashboard, Claim y Buzz**.
- No se considerarán pruebas en dispositivos móviles ni validaciones multi-navegador fuera del entorno base de ejecución.

Responsable:

Marta David Gallego, Ingeniera de Sistemas

Con énfasis en: Ingeniera de Calidad de Software

Solución y Ejecución del Reto

PRUEBAS FUNCIONALES II

1. Proceso de pruebas

Ambiente de Pruebas: Para el desarrollo del reto técnico y la validación funcional, se utilizó el entorno web de OrangeHRM accesible mediante la siguiente URL: [OrangeHRM](#)

1.1 Análisis de matriz de riesgos

Objetivo: Identificar, evaluar y priorizar los posibles riesgos funcionales y técnicos asociados a los módulos críticos de la aplicación (**Login, Admin y PIM**) con el fin de anticipar incidentes que puedan comprometer la estabilidad, seguridad o funcionalidad del sistema.

Matriz de Riesgos – Aplicación OrangeHRM (Módulos Críticos: Login, Admin y PIM)

ID	Módulo	Riesgo Identificado	Causa Potencial	Impacto Potencial	Probabilidad	Nivel de Riesgo (I x P)	Acción Preventiva / Mitigación	Responsable
R1	Login	Error al autenticar usuario válido	Falla en la conexión con backend o base de datos	Impide acceso de usuarios autorizados	Media	Alta	Validar integración API login y pruebas de regresión	QA & Dev

R2	Login	Permite acceso con credenciales incorrectas	Falta de validación adecuada del lado servidor	Vulnerabilidad de seguridad	Baja	Alta	Implementar pruebas negativas y validaciones de respuesta 401	QA & Dev
R3	Login	Campos sin validación de formato	Validaciones solo en frontend	Ingreso de datos inválidos	Media	Media	Pruebas con inputs erróneos y caracteres especiales	QA
R4	Login	Sin bloqueo tras varios intentos fallidos	Falta de política de seguridad configurada	Riesgo de ataques de fuerza bruta	Media	Alta	Configurar umbral de intentos y validarlo en QA	QA & Infra
R5	Login	Incompatibilidad en navegador	Librerías o CSS no soportados	Fallos en UI o inoperatividad	Baja	Media	Validar en navegadores base definidos	QA
R6	Login	Recuperación de contraseña inactiva	Enlace o flujo de recuperación roto	Usuario bloqueado sin acceso	Media	Media	Verificar flujo de "Forgot your password?"	QA
R7	Login	Carga lenta del formulario	Problemas en servidor o recursos estáticos	Experiencia de usuario afectada	Media	Media	Monitorear tiempos de carga y recursos	QA & Infra

R8	Login	Mensajes de error incorrectos o confusos	Falta de mensajes claros en UI	Confusión del usuario	Alta	Media	Validar mensajes en escenarios positivos y negativos	QA
R9	Admin	Error al crear o editar usuarios	Falla en validaciones o backend	Pérdida de control de acceso	Alta	Alta	Pruebas CRUD sobre gestión de usuarios	QA
R10	Admin	Permisos mal asignados	Lógica incorrecta de roles y privilegios	Riesgo de acceso indebido	Media	Alta	Pruebas con distintos roles y escenarios	QA & Dev
R11	Admin	No actualización en base de datos	Problemas en persistencia	Inconsistencias en datos	Media	Media	Validar integridad de datos con SQL	QA
R12	Admin	Falla al eliminar usuario	Error de referencia o falta de confirmación	Datos residuales	Baja	Media	Validar flujo de eliminación y dependencias	QA
R13	Admin	Falta de validación de campos obligatorios	Falta de controles en UI o backend	Datos incompletos	Alta	Media	Pruebas de validaciones frontend/backend	QA

R14	Admin	Interfaz no refresca tras editar	Cache o renderizado tardío	Información desactualizada	Media	Media	Validar actualización de UI con datos reales	QA
R15	PIM	Error al registrar nuevo empleado	Falla en conexión o validación de datos	Pérdida de información del empleado	Alta	Alta	Pruebas de creación y persistencia de empleados	QA
R16	PIM	Inconsistencias al editar información	Fallos en actualización de backend	Datos duplicados o incompletos	Media	Alta	Pruebas de edición con verificación en BD	QA
R17	PIM	Carga incorrecta de archivos adjuntos	Formatos no validados o límites incorrectos	Error al subir documentos	Media	Media	Validar formatos y límites en pruebas funcionales	QA
R18	PIM	Falla en búsqueda o filtros de empleados	Errores de consulta o paginación	Dificultad en gestión de empleados	Media	Media	Validar criterios de búsqueda y ordenamiento	QA
R19	PIM	Eliminación no controlada de registros	Falta de confirmación o dependencias rotas	Pérdida de información crítica	Baja	Alta	Agregar confirmación de eliminación y pruebas negativas	QA

R20	PIM	Campos sensibles visibles sin restricción	Error en permisos	Riesgo de exposición de datos	Baja	Alta	Revisar permisos de visualización	QA & Dev
-----	-----	---	-------------------	-------------------------------	------	------	-----------------------------------	----------

1.2 Plan de Pruebas – Proyecto OrangeHRM

Propósito del Plan de Pruebas

El propósito del presente Plan de Pruebas es definir el alcance, los objetivos, el enfoque, los recursos y las actividades necesarias para ejecutar las pruebas funcionales del sistema **OrangeHRM**, garantizando la validación de los flujos críticos del aplicativo bajo los criterios de calidad establecidos para el reto técnico.

Objetivo General

Garantizar que las funcionalidades críticas operen conforme a los requerimientos establecidos y que los flujos principales de negocio no presenten defectos que afecten la experiencia del usuario.

Objetivos específicos:

- **Validar la funcionalidad básica** de los módulos priorizados (**Login, Admin y PIM**), asegurando su correcto funcionamiento bajo condiciones normales de uso.
- **Identificar defectos funcionales** que puedan afectar la experiencia del usuario o la estabilidad del sistema.
- Evaluar el cumplimiento de los criterios de aceptación definidos en los requerimientos funcionales.
- Entregar evidencia clara y documentada del proceso de pruebas

Alcance

Las pruebas se centrarán en la **verificación funcional** de los módulos críticos del sistema:

- **Login (Inicio de sesión)**
- **Admin (Administración de usuarios y roles)**
- **PIM (Gestión de empleados)**

Fuera del Alcance

- No se incluirán pruebas sobre los módulos **Recruitment, Leave, Time, Performance, Directory, Maintenance, Dashboard, Claim y Buzz**.
- No se considerarán pruebas en dispositivos móviles ni validaciones multi-navegador fuera del entorno base de ejecución
- No se realizarán pruebas automatizadas ni de rendimiento

Ambiente de Pruebas

URL del entorno: [OrangeHRM](#)

Navegador base: Google Chrome (versión estable).

Usuarios de prueba: Credenciales suministradas por la plataforma

Metodología de Pruebas

Se aplicará una **metodología de pruebas funcionales manuales** complementada con **pruebas exploratorias** para validar flujos no contemplados.

Las pruebas seguirán un enfoque **basado en riesgos**, priorizando los módulos y funcionalidades de mayor impacto para el negocio.

Tipos de Prueba a Ejecutar

Tipo de Prueba	Descripción
Pruebas de humo	Validan la disponibilidad y estabilidad inicial del sistema.
Pruebas funcionales	Verifican que cada funcionalidad cumpla con su propósito.

Pruebas exploratorias	Permiten identificar defectos no previstos en los casos estándar.
------------------------------	---

Prerrequisitos

Criterios de entrada:

- Entorno accesible y estable.
- Casos de prueba revisados y validados.
- Credenciales de acceso disponibles.

Criterios de salida:

- Ejecución del 100% de los casos planificados.
- Registro y documentación de todos los defectos encontrados.
- Validación de resultados por parte del equipo QA.

Planificación y Entregables

Duración estimada: 3 días hábiles.

Entregables:

- Matriz de riesgos funcionales.
- Diseño de los escenarios en lenguaje gherkin
- Reportar defectos encontrados indicando la prioridad
- Evidencias de ejecución (capturas, reportes).

Gestión de Riesgos

Los riesgos identificados durante la ejecución serán gestionados a través de la **Matriz de Riesgos**, la cual permite priorizar acciones de mitigación sobre los módulos de **Login, Admin y PIM**.

Roles y Responsabilidades

Rol	Responsabilidad
QA (responsable principal)	Planificación, diseño, ejecución y documentación de las pruebas.
QA Lead	Supervisión del proceso de pruebas y control de calidad del entregable.
Developer	Soporte técnico en defectos y validación de correcciones.
Stakeholder técnico	Validación final de resultados y cierre del reto.

Cierre de Pruebas

Al finalizar el proceso, se generará un **informe de resultados** consolidando los defectos encontrados, su nivel de severidad y el cumplimiento de los objetivos del plan.

1.3 Diseño de los escenarios en lenguaje gherkin

- Diseño de Escenarios Módulo Login (OrangeHRM)

Funcionalidad: Validación de inicio de sesión en OrangeHRM

Como usuario del sistema **OrangeHRM**

Quiero poder acceder a la aplicación mediante credenciales válidas

Para realizar las operaciones de administración de empleado

Escenario 01: Validación de elementos visibles en la interfaz

Dado que el usuario accede a la URL [OrangeHRM](#)

Entonces se debe visualizar el logotipo de OrangeHRM

Y el campo " **Username** "

Y el campo " **Password** "

Y el botón " **Login** "

Y el enlace "¿Olvidaste tu contraseña?"

Escenario 02: Inicio de sesión exitoso con credenciales válidas

Dado que el usuario se encuentra en la página de inicio de sesión de **OrangeHRM**

Cuando ingresa el nombre de usuario "**Admin**" en el campo **Username**

Y la contraseña "**admin123**" en el campo **Password**

Y hace clic en el botón "**Login**"

Entonces el sistema muestra la página principal del panel de administración

Y se visualiza el nombre del usuario en la parte superior derecha

Escenario 03: Intento de inicio de sesión con credenciales inválidas

Dado que el usuario se encuentra en la pantalla de inicio de sesión

Cuando ingresa el nombre de usuario "**Admin**" en el campo **Username**

Y la contraseña "**12345**" en el campo **Password**

Y hace clic en el botón " **Login** "

Entonces el sistema muestra un mensaje de error "**Credenciales inválidas**"

Y el usuario permanece en la pantalla de inicio de sesión

Escenario 04: Inicio de sesión con campos vacíos

Dado que el usuario está en la pantalla de inicio de sesión

Cuando deja los campos " **Username** " o " **Admin** " vacíos

Y hace clic en el botón " **Login** "

Entonces el sistema muestra mensajes de validación indicando que los campos son obligatorios

Escenario 05: Recuperación de contraseña

Dado que el usuario se encuentra en la pantalla de inicio de sesión

Cuando hace clic en el enlace "**¿Olvidaste tu contraseña?**"

Entonces el sistema redirige a la pantalla de recuperación de credenciales

Y muestra el campo para ingresar el nombre de usuario

- **Diseño de Escenarios – Módulo Administración (Admin)**

Funcionalidad: Gestión de usuarios del sistema

Como administrador

Quiero gestionar los usuarios y sus roles dentro del sistema

Para garantizar que solo personal autorizado tenga acceso a las funcionalidades de **OrangeHRM**

Escenario 01: Visualizar la lista de usuarios

Dado que el usuario ha iniciado sesión correctamente

Y se encuentra en el módulo " **Admin** "

Cuando accede a la opción "Usuarios"

Entonces el sistema muestra una lista con los usuarios registrados

Y se visualizan las columnas "Nombre de usuario", "Rol" y "Estado"

Escenario 02: Buscar usuario por nombre

Dado que el usuario está en la pantalla de Administración de Usuarios

Cuando ingresa un nombre de usuario válido en el campo de búsqueda

Y hace clic en el botón “Buscar”

Entonces el sistema muestra los resultados que coinciden con el criterio ingresado

Escenario 03: Agregar un nuevo usuario

Dado que el usuario se encuentra en la sección de Administración

Cuando hace clic en el botón “Agregar”

Y completa los campos requeridos: “Rol”, “Empleado”, “Nombre de usuario”, “Contraseña” y “Confirmar contraseña”

Y guarda la información

Entonces el sistema crea el nuevo usuario correctamente

Y muestra un mensaje de confirmación “Usuario agregado exitosamente”

Escenario 04: Intentar agregar usuario con datos incompletos

Dado que el usuario está en la pantalla para agregar usuario

Cuando deja uno o más campos requeridos sin diligenciar

Y hace clic en “Guardar”

Entonces el sistema muestra mensajes de validación indicando los campos faltantes

Escenario 05: Eliminar un usuario

Dado que el usuario está en la lista de usuarios

Cuando selecciona un registro y hace clic en el ícono de eliminar

Y confirma la acción

Entonces el sistema elimina el usuario y muestra un mensaje “Usuario eliminado correctamente”

○ Diseño de Escenarios – Módulo Gestión de Empleados (PIM)

Funcionalidad: Administración del registro de empleados

Como analista de recursos humanos

Quiero registrar, consultar y actualizar información de empleados

Para mantener la base de datos de personal actualizada y precisa

Escenario 01: Consultar lista de empleados

Dado que el usuario ha iniciado sesión correctamente

Y se encuentra en el módulo “PIM”

Cuando accede a la opción “Lista de empleados”

Entonces el sistema muestra una tabla con los empleados registrados

Y se visualizan las columnas "ID", "Nombre completo", "Cargo" y "Estado"

Escenario 2: Registrar un nuevo empleado

Dado que el usuario está en la pantalla de "Agregar empleado"

Cuando ingresa la información requerida en los campos "Nombre", "Apellido" y "ID de empleado"

Y hace clic en "Guardar"

Entonces el sistema crea el nuevo registro y muestra el mensaje "Empleado agregado exitosamente"

Escenario 3: Editar información de un empleado

Dado que el usuario está en la lista de empleados

Cuando selecciona un empleado existente

Y actualiza la información del campo "Cargo"

Y guarda los cambios

Entonces el sistema actualiza correctamente la información del empleado

Escenario 4: Buscar empleado por nombre

Dado que el usuario está en la lista de empleados

Cuando ingresa el nombre del empleado en el campo de búsqueda

Y hace clic en el botón "Buscar"

Entonces el sistema muestra los registros que coinciden con el criterio ingresado

Escenario 5: Eliminar un empleado

Dado que el usuario está en la lista de empleados

Cuando selecciona un registro

Y hace clic en el ícono de eliminar

Y confirma la acción

Entonces el sistema elimina el registro y muestra el mensaje "Empleado eliminado correctamente"

1.4 Reportar defectos encontrados indicando la prioridad

Nota:

Durante la ejecución del reto técnico, se realizaron las pruebas en diferentes días (viernes, sábado y domingo), horarios y navegadores (Chrome, Edge y modo incógnito) cumpliendo con el proceso de validación solicitado.

Se determinó que el entorno no se encuentra disponible de forma estable, presentando fallos de servidor que impiden el acceso a la aplicación. A pesar de la limitación externa, se cumplió con el análisis técnico y funcional, la obtención de evidencias y la documentación del incidente en un formato formal de reporte de defectos, demostrando el cumplimiento del requerimiento planteado en el ejercicio

Bug 01

Título: [OrangeHRM Demo] – Problemas al cargar la página de login: pantalla en blanco y errores JS

Descripción:

Al acceder al entorno demo de OrangeHRM ([link](#)), la página **no carga** y se muestra una **pantalla en blanco**.

En la consola del navegador aparecen **errores de carga de archivos JavaScript esenciales** para el funcionamiento de la interfaz.

Pasos para reproducir:

Dado que el usuario abre el navegador Google Chrome y Edge o en modo incógnito,

Cuando accede a la URL [OrangeHRM](#)

Y espera a que cargue la página de login,

Entonces la interfaz no se renderiza correctamente

Y en la consola del navegador aparecen errores de carga de archivos JavaScript.

Resultado esperado:

Al acceder a la URL del entorno demo de OrangeHRM, la página de inicio de sesión **debe cargarse correctamente**, mostrando todos los elementos de la interfaz (**Username**, **Password** botón " **Login**", entre otros).

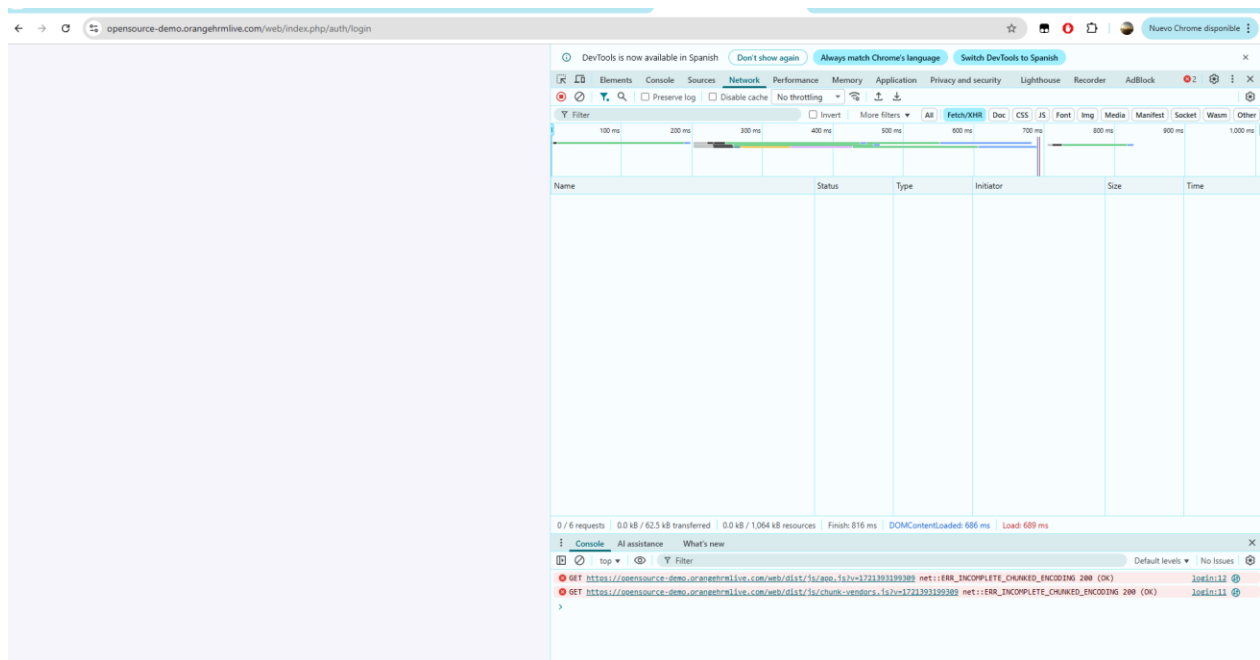
Resultado actual:

Al acceder a la URL del entorno demo de OrangeHRM, la página de inicio de sesión **no se carga correctamente** y se muestra una **pantalla completamente en blanco**.

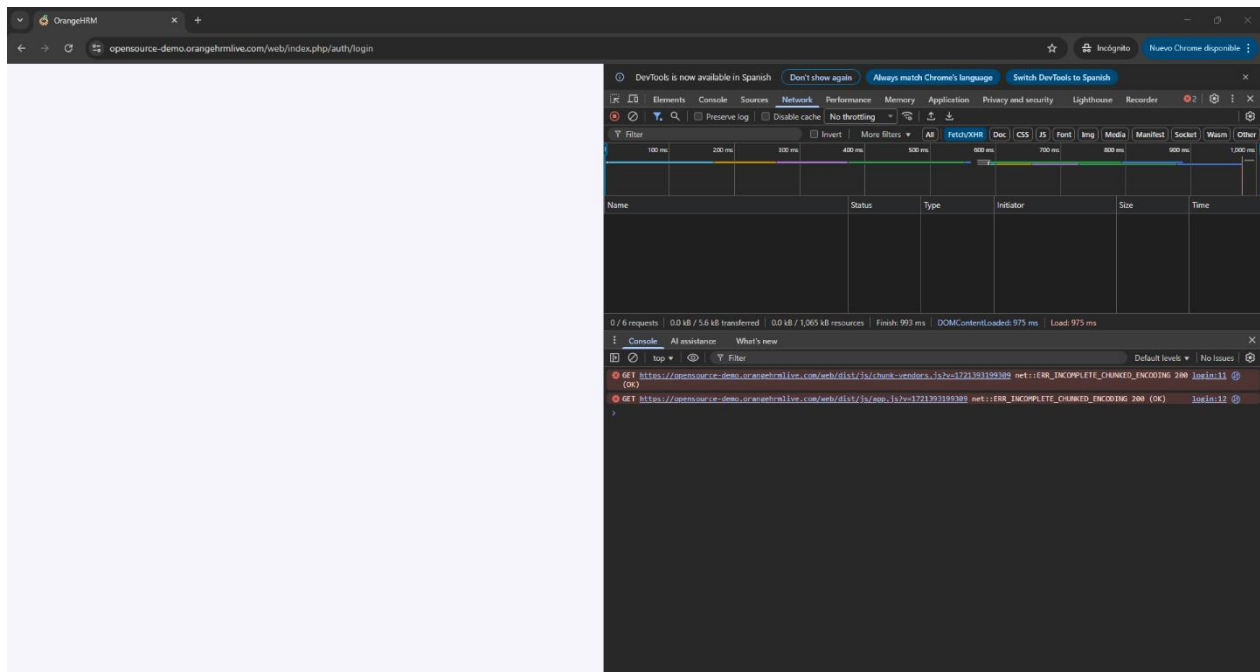
En la consola del navegador aparecen **errores de carga de archivos JavaScript**, impidiendo que la interfaz funcione.

Evidencias:

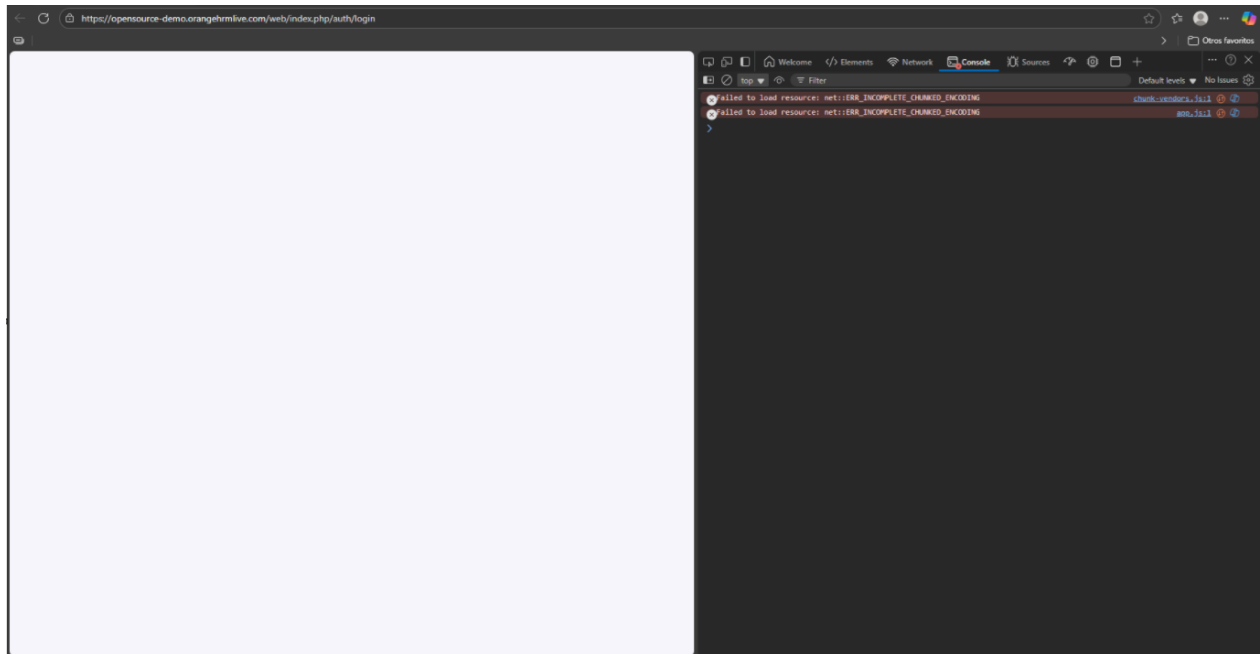
Accediendo a la URL [OrangeHRM](https://opensource-demo.orangehrmlive.com/web/index.php/auth/login) con navegador Google Chrome



Accediendo a la URL [OrangeHRM](https://opensource-demo.orangehrmlive.com/web/index.php/auth/login) con navegador Google Chrome en modo incógnito



Accediendo a la URL [OrangeHRM](https://opensource-demo.orangehrmlive.com/web/index.php/auth/login) navegador Edge en modo incógnito



Evidencia técnica (consola del navegador):

```
login:11 GET https://opensource-demo.orangehrmlive.com/web/dist/js/chunk-vendors.js?v=1721393199309 net::ERR_INCOMPLETE_CHUNKED_ENCODING 200 (OK)
login:12 GET https://opensource-demo.orangehrmlive.com/web/dist/js/app.js?v=1721393199309 net::ERR_INCOMPLETE_CHUNKED_ENCODING 200 (OK)
```

Información adicional:

- Navegadores: Google Chrome (Versión 141.0.7390.122) y Edge (Versión 141.0.3537.92) o en modo incógnito.
- Días de prueba: viernes, sábado y Domingo.
- Horarios: Diferentes franjas del día.
- Sistema operativo: Windows 11.

Impacto:

Moderado / Medio

No detiene toda la operación del entorno demo de OrangeHRM, pero impide cumplir con un requerimiento funcional específico del reto técnico (validación de login y ejecución de pruebas

funcionales).

Recomendable asignarlo al equipo de desarrollo o soporte del entorno para la revisión y corrección de la entrega de archivos JavaScript (chunk-vendors.js y app.js), o la actualización de la configuración del servidor.

Estado:

New

Asignado a: DEV

NOTA: En contexto funcional real

El impacto sería **Alto/Crítico**, porque el entorno **bloquea completamente la ejecución de pruebas funcionales**.



2. Prueba Consulta SQL_Empresa_de_Monitoreo_Satelital_GeoTrack

Descripción:

La empresa **GeoTrack** ofrece servicios de **rastreo satelital a compañías de transporte**. Cada cliente tiene dispositivos instalados en sus vehículos, los cuales reportan periódicamente su posición geográfica (latitud, longitud, velocidad) y generan alertas cuando ocurre un evento relevante (exceso de velocidad, pérdida de señal, salida de zona permitida). La dirección de GeoTrack necesita respuestas de negocio confiables a partir de la base de datos.

Requerimiento: Es escribir consultas SQL en Microsoft SQL Server que respondan a las preguntas planteadas.

Estructura de Base de Datos

Tabla: clients

100 %	No se encontraron problemas.
Resultados	Mensajes
	client_idnameemailcreated_at
1	1Transporte Andinocontacto@andino.com2024-01-10
2	2Logística Expressinfo@express.com2024-02-15
3	3Rutas del Caférutas@cafe.com2024-03-01

Tabla: devices

100 %	No se encontraron problemas.
Resultados	Mensajes
	device_idclient_idimeivehicle_platestatus
1	1112345678901ABC123active
2	22198765432109XYZ789active
3	33255566677788LMN456inactive
4	44300011122233CAF321active

Tabla: positions

Resultados	Mensajes
	position_iddevice_idlatitudelongitude speedsrecorded_at
1	115.070000-75.52000060.52024-04-01 08:30:00.000
2	215.072000-75.52200082.02024-04-01 08:45:00.000
3	315.075000-75.52500040.02025-10-20 01:54:00.447
4	425.080000-75.5300000.02024-04-01 09:00:00.000
5	525.081500-75.53100010.02025-10-23 19:54:00.447
6	635.100000-75.54000045.02024-04-01 09:15:00.000
7	745.115000-75.54500065.02025-10-25 00:54:00.447
8	845.118000-75.548000110.02025-10-24 23:54:00.447

Tabla: alerts

Resultados	Mensajes
	alert_iddevice_idalert_typedescriptioncreated_at
1	11OverspeedVelocidad mayor a 80 km/h2024-04-01 08:46:00.000
2	22NoSignalDispositivo sin señal por 30 min2024-04-01 09:30:00.000
3	34OverspeedExceso de velocidad en autopista2025-10-24 23:54:00.453
4	44OverspeedReincidencia exceso velocidad2025-10-24 05:54:00.453
5	54GeofenceSalida de zona permitida2025-10-23 01:54:00.453
6	61NoSignalSin reporte 24h2025-10-22 01:54:00.453

Requerimiento 01: Velocidad promedio por vehículo (últimos 30 días)

Objetivo: Obtener la velocidad promedio de cada vehículo en el último mes.

Script SQL

```
SELECT
    d.vehicle_plate,
    AVG(p.speed) AS Velocidad_promedio
FROM positions p
JOIN devices d ON p.device_id = d.device_id
WHERE p.recorded_at >= DATEADD(DAY, -30, GETDATE())
GROUP BY d.vehicle_plate
ORDER BY velocidad_promedio DESC;
```

Resultado:

	vehicle_plate	Velocidad promedio
1	CAF321	87.500000
2	ABC123	40.000000
3	XYZ789	10.000000

Requerimiento 02: Clientes con dispositivos inactivos

Objetivo: Listar los clientes que tienen al menos un dispositivo inactivo y cuántos son.

Script SQL

```
SELECT
    c.name AS Nombre_cliente,
    COUNT(d.device_id) AS Dispositivos_inactivos
FROM clients c
JOIN devices d ON c.client_id = d.client_id
WHERE d.status = 'inactive'
GROUP BY c.name
HAVING COUNT(d.device_id) >= 1;
```

Resultado:

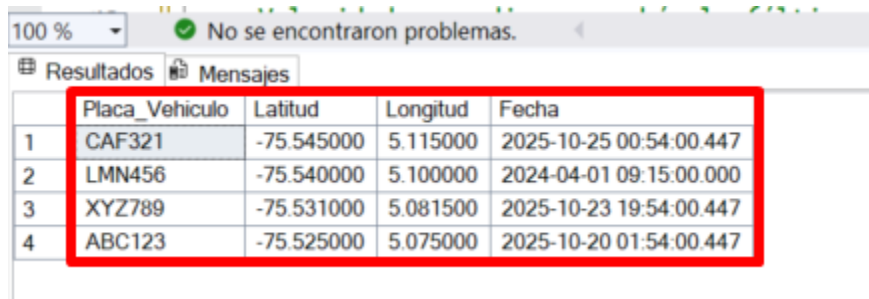
100 %	✓ No se encontraron problemas.
Resultados	Mensajes
	</

Requerimiento 03: Última posición de cada vehículo

Objetivo: Mostrar la última ubicación registrada por cada vehículo, incluyendo placa, latitud, longitud y fecha.

Script SQL

```
SELECT
    d.vehicle_plate AS Placa_Vehiculo,
    p.longitude AS Latitud,
    p.latitude AS Longitud,
    p.recorded_at AS Fecha
FROM devices d
JOIN positions p ON d.device_id = p.device_id
WHERE p.recorded_at = (
    SELECT MAX(p2.recorded_at)
    FROM positions p2
    WHERE p2.device_id = d.device_id
);
```

Resultado:

	Placa_Vehiculo	Latitud	Longitud	Fecha
1	CAF321	-75.545000	5.115000	2025-10-25 00:54:00.447
2	LMN456	-75.540000	5.100000	2024-04-01 09:15:00.000
3	XYZ789	-75.531000	5.081500	2025-10-23 19:54:00.447
4	ABC123	-75.525000	5.075000	2025-10-20 01:54:00.447

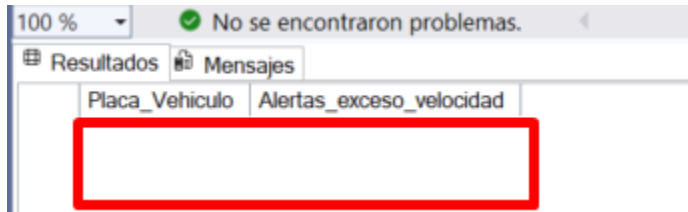
Requerimiento 04: Vehículos con múltiples alertas de exceso de velocidad

Objetivo: Identificar los vehículos que han generado más de 2 alertas de tipo Overspeed en los últimos 7 días.

Script SQL

```
SELECT
    d.vehicle_plate AS Placa_Vehiculo,
    COUNT(a.alert_id) AS Alertas_exceso_velocidad
FROM alerts a
JOIN devices d ON a.device_id = d.device_id
WHERE a.alert_type = 'Overspeed'
    AND a.created_at >= DATEADD(DAY, -7, GETDATE())
GROUP BY d.vehicle_plate
HAVING COUNT(a.alert_id) > 2;
```

Resultado: No se encontraron vehículos que cumplan la condición de tener más de 2 alertas de exceso de velocidad en los últimos 7 días.



Placa_Vehiculo	Alertas_exceso_velocidad
----------------	--------------------------

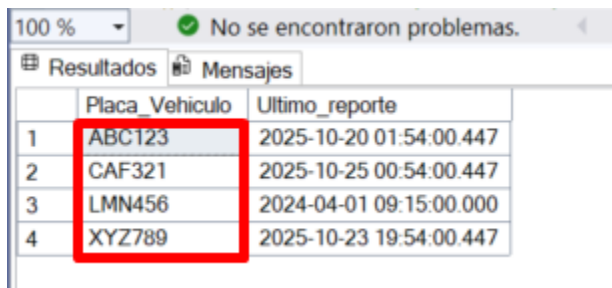
Requerimiento 05: Vehículos sin reporte en 24 horas

Objetivo: Listar los vehículos que no han enviado posiciones en las últimas 24 horas y su último reporte (si existe).

Script SQL

```
SELECT
    d.vehicle_plate AS Placa_Vehiculo,
    MAX(p.recorded_at) AS Ultimo_reporte
FROM devices d
LEFT JOIN positions p ON d.device_id = p.device_id
GROUP BY d.vehicle_plate
HAVING MAX(p.recorded_at) < DATEADD(HOUR, -24, GETDATE())
OR MAX(p.recorded_at) IS NULL;
```

Resultado: Los vehículos están inactivos o sin comunicación en las últimas 24 horas son:



	Placa_Vehiculo	Ultimo_reporte
1	ABC123	2025-10-20 01:54:00.447
2	CAF321	2025-10-25 00:54:00.447
3	LMN456	2024-04-01 09:15:00.000
4	XYZ789	2025-10-23 19:54:00.447

Requerimiento 06: Tiempo promedio entre reportes

Objetivo: Calcular el tiempo promedio (en minutos) entre posiciones consecutivas de cada vehículo.

Script SQL

```

WITH diffs AS (
    SELECT
        device_id,
        DATEDIFF(MINUTE,
            LAG(recorded_at) OVER(PARTITION BY device_id ORDER BY recorded_at),
            recorded_at) AS diferencia_minutos
    FROM positions
)
SELECT
    d.vehicle_plate AS Placa_Vehiculo,
    AVG(diffs.diferencia_minutos) AS promedio_de_minutos_entre_informes
FROM diffs
JOIN devices d ON diffs.device_id = d.device_id
WHERE diffs.diferencia_minutos IS NOT NULL
GROUP BY d.vehicle_plate ;
    
```

Resultado

100 % No se encontraron problemas.

	Placa_Vehiculo	promedio_de_minutos_entre_informes
1	ABC123	408042
2	CAF321	60
3	XYZ789	821454

Requerimiento 07: Top 3 clientes con más alertas en 30 días

Objetivo: Determinar los tres clientes que más alertas han generado en el último mes.

Script SQL

```

SELECT TOP 3
    c.name AS Nombre_del_cliente,
    COUNT(a.alert_id) AS Cantidad_de_alertas
FROM alerts a
JOIN devices d ON a.device_id = d.device_id
JOIN clients c ON d.client_id = c.client_id
WHERE a.created_at >= DATEADD(DAY, -30, GETDATE())
GROUP BY c.name
ORDER BY Cantidad_de_alertas DESC;
    
```

Resultado:

100 % No se encontraron problemas.

	Nombre_del_cliente	Cantidad_de_alertas
1	Rutas del Café	3
2	Transporte Andino	1

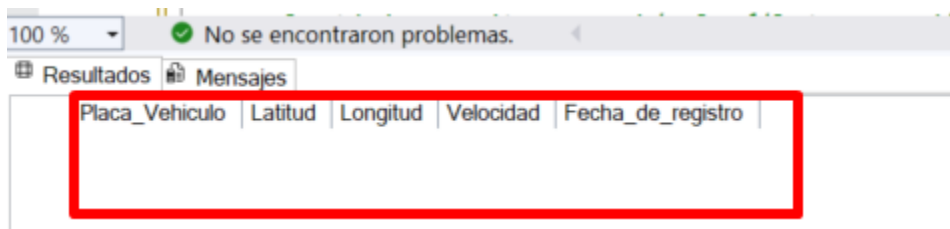
Requerimiento 08: Velocidades sin alerta correspondiente

Objetivo: Detectar posiciones con velocidad mayor a 100 km/h que no tengan una alerta Overspeed ± 10 minutos alrededor.

Script SQL

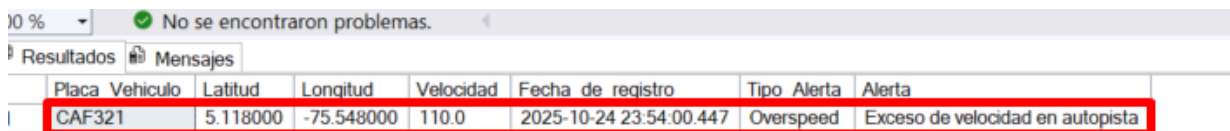
```
SELECT  
  
    d.vehicle_plate AS Placa_Vehiculo,  
    p.longitude AS Latitud,  
    p.latitude AS Longitud,  
    p.speed AS Velocidad,  
    p.recorded_at As Fecha_de_registro  
FROM positions p  
JOIN devices d ON p.device_id = d.device_id  
WHERE p.speed > 100  
    AND NOT EXISTS (  
        SELECT 1  
        FROM alerts a  
        WHERE a.device_id = p.device_id  
            AND a.alert_type = 'Overspeed'  
            AND a.created_at BETWEEN DATEADD(MINUTE, -10, p.recorded_at)  
                                AND DATEADD(MINUTE, 10, p.recorded_at)  
    );
```

Resultado: No se encontraron posiciones con velocidad superior a **100 km/h** que carezcan de una alerta **Overspeed** en el rango de **± 10 minutos**,



Placa_Vehiculo	Latitud	Longitud	Velocidad	Fecha_de_registro
----------------	---------	----------	-----------	-------------------

Dado que todas las posiciones con velocidad elevada ya cuentan con su alerta correspondiente



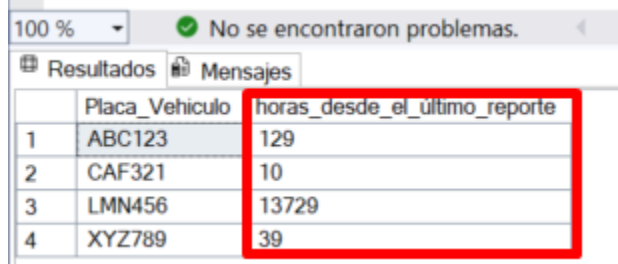
Placa_Vehiculo	Latitud	Longitud	Velocidad	Fecha de registro	Tipo Alerta	Alerta
CAF321	5.118000	-75.548000	110.0	2025-10-24 23:54:00.447	Overspeed	Exceso de velocidad en autopista

Requerimiento 09: Tiempo desde el último reporte

Objetivo: Calcular cuántas horas han pasado desde la última posición registrada por cada vehículo.

Script SQL

```
SELECT
    d.vehicle_plate AS Placa_Vehiculo,
    DATEDIFF(HOUR, MAX(p.recorded_at), GETDATE()) AS horas_desde_el_ultimo_reporte
FROM devices d
LEFT JOIN positions p ON d.device_id = p.device_id
GROUP BY d.vehicle_plate;
```

Resultado

	Placa_Vehiculo	horas_desde_el_ultimo_reporte
1	ABC123	129
2	CAF321	10
3	LMN456	13729
4	XYZ789	39

Requerimiento 10: Vehículo más activo por cliente

Objetivo: Para cada cliente, identificar el vehículo que más posiciones ha reportado en los últimos 30 días.

Script SQL

```
WITH counts AS (
    SELECT
        d.client_id,
        d.vehicle_plate,
        COUNT(p.position_id) AS cantidad_posiciones
    FROM positions p
    JOIN devices d ON p.device_id = d.device_id
    WHERE p.recorded_at >= DATEADD(DAY, -30, GETDATE())
    GROUP BY d.client_id, d.vehicle_plate
)
SELECT
    c.name AS nombre_cliente,
    cts.vehicle_plate AS placa_vehiculo,
    cts.cantidad_posiciones
FROM (
    SELECT
        client_id,
        vehicle_plate,
        cantidad_posiciones,
        ROW_NUMBER() OVER(PARTITION BY client_id ORDER BY cantidad_posiciones DESC) AS rn
    FROM counts
) cts
JOIN clients c ON cts.client_id = c.client_id
WHERE cts.rn = 1;
```

Resultado: Para cada cliente, se contabilizó el número de posiciones reportadas por cada vehículo en los 30 días, permite conocer la actividad de cada vehículo dentro del período y el vehículo más activo es:

100 % ✓ No se encontraron problemas.			
<div> <div>Resultados</div> <div>Mensajes</div> </div>			
	nombre_cliente	placa_vehiculo	cantidad_posiciones
1	Transporte Andino	ABC123	1
2	Rutas del Café	CAF321	2

3. Códigos de error protocolo web

Descripción: El requerimiento busca analizar los códigos de error del **protocolo HTTP** desde la perspectiva funcional, con el fin de identificar cómo impactan los procesos de negocio, la experiencia del usuario y las validaciones en los flujos funcionales.

El analista funcional debe interpretar los códigos, comunicar el incidente al área técnica y validar las condiciones de negocio implicadas.

Análisis y manejo de errores HTTP

Caso 01: Un cliente reporta que al intentar acceder a un recurso obtiene un **401**. ¿Qué significa y cómo lo resolverías?

- **Error 401 – Unauthorized:** Indica que la solicitud no ha sido completada porque el sistema no reconoce la autenticación del usuario o las credenciales proporcionadas no son válidas.
- **Posibles causas:** El usuario no tiene permisos suficientes para ese recurso, Vencimiento de credenciales, vencimiento de token, roles inactivos, Fallo en el proceso de login o manejo de sesiones entre otros.

Acción QA:

- Recopila información del usuario para comprender el origen del problema de autenticación, validando sus permisos, credenciales y el estado de su cuenta. Se consulta si el incidente ha ocurrido anteriormente y se registran posibles cambios recientes, como actualizaciones de contraseña o cambios de dispositivo. Posteriormente, se reproduce el escenario en el mismo entorno, utilizando las mismas credenciales y pasos realizados por el usuario, con el fin de confirmar y analizar el error en condiciones controladas y determinar su causa.
- Si el usuario cumple con todos los requisitos para acceder; es decir, cuenta con credenciales válidas, una cuenta activa y los permisos correctos; y aun así no puede acceder al recurso, se debe reportar un bug indicando:

BUG_# Caso 01

Título: **Acceso denegado (401) para usuario con permisos válidos en recurso X**

Descripción: “El sistema retorna código 401 al intentar acceder al recurso X con un usuario autenticado. Según el flujo funcional, el acceso debería estar permitido.”

Comportamiento observado: descripción clara del error que ocurre al intentar acceder (Según las la recopilación de información).

Comportamiento esperado: explicación de cómo debería funcionar el sistema según el flujo funcional.

Datos del usuario: nombre de usuario, rol, permisos y estado de la cuenta.

Pasos para reproducir el error: detalle paso a paso de cómo se reproduce el fallo en el mismo entorno y condiciones del usuario.

Entorno y plataforma: sistema, versión, navegador o dispositivo donde se presentó el error.

Evidencias: capturas de pantalla, logs, códigos de error, video o cualquier registro que permita al equipo de desarrollo comprender y replicar el problema.

Responsable: Equipo responsable.

- En caso contrario, si se identifica que el problema **se debe a una causa válida del lado del usuario**, como:
 - Credenciales incorrectas,
 - Cuenta bloqueada o inactiva,
 - Permisos insuficientes para el recurso,

Entonces **no se reporta como bug**, sino que se **registra como incidencia de usuario o error de configuración**. En estos casos, se notifica al usuario la causa del problema y, si corresponde, se **orienta en el procedimiento para restablecer el acceso** (por ejemplo, desbloquear la cuenta o solicitar permisos al administrador).

Caso 02: En producción aparece un **503 intermitente**. ¿Qué hipótesis plantearías y qué pasos seguirías para diagnosticarlo?

503 Service Unavailable – Intermitente

Significado: El servidor no puede atender la solicitud temporalmente.

Posibles causas:

Sobrecarga del servidor o aplicación.

Mantenimiento o reinicios de servicios.

Problemas entre servicios (DB, microservicios etc).

Recursos limitados (CPU, memoria, conexiones).

Reportar a DevOps con toda la información relevante si es un problema de infraestructura.

Acción QA:

- Registrar evidencia (captura, hora, request/response si es API).
- Confirmar si el error afecta pasos funcionales críticos (por ejemplo, registro, pago, consulta).
- Comunicar a infraestructura o desarrollo indicando que la prueba no es estable por **intermitencia del servicio**, sin levantar un bug de aplicación (ya que no es error funcional, sino técnico).

Casos 03: Explique la diferencia práctica entre un **401 Unauthorized** y un **403 Forbidden**

401 Unauthorized: indica que existen problemas con la autenticación del usuario.

403 Forbidden: indica que existen problemas con los permisos del usuario para ejecutar acciones o acceder a los recursos del sistema.

Caso 04: ¿En qué se diferencia un **500 Internal Server Error** de un **503 Service Unavailable** en cuanto a causa y responsabilidad?

500 – Internal Server Error: Ocurre por un **fallo interno en la aplicación o el sistema**, generalmente debido a errores en el código, la lógica o el manejo de datos. La responsabilidad recae en el **equipo de desarrollo (DEV)**.

503 – Service Unavailable: Ocurre cuando el **servicio no está disponible**, generalmente por problemas de infraestructura, sobrecarga o mantenimiento. La responsabilidad recae en el equipo infraestructura.

Casos 05: Cuéntanos un caso en el que hayas enfrentado un **500** en producción. ¿Cómo lo detectaste y qué acciones tomaste?

Reporte de incidente de usuario:

Un usuario reportó que, al intentar **aprobar una factura** en la funcionalidad de **aprobación y rechazo**, al presionar el botón **“Aprobar”**, el sistema mostraba un **mensaje de alerta** indicando que se habían presentado inconvenientes que **no permitieron completar el proceso**.

Acciones tomadas:

Se Contactó al usuario se le pidió reproducir el error, se le dieron indicaciones para poder revisar **URL** con el fin de identificar el **Endpoint** que presentó la afectación, se confirmó error **500**, se revisa **payload y Response**, con esta información inicial, se solicitó inmediato el apoyo del equipo de soporte, dado que, la funcionalidad afectada hace parte de los flujos importantes de la operación.

Se creo **Issue**, incluyendo:

- Pasos exactos para reproducirlo.
- Mensaje de alerta mostrado.
- Endpoint, payload y Response
- Datos del usuario y entorno.
- Evidencias (capturas de pantalla y registros relevantes).
- Resultado obtenido y resultado esperado

Se creo mesa de trabajo para soporte técnico, se corrigió el error, se realizaron las pruebas de comprobación y regresión y se cierra **Issue**

Casos 06: Un usuario autenticado intenta acceder a un recurso y recibe un 401. ¿Es correcto o debería ser otro código?

No es correcto. Se debería mostrar el **error 403 Forbidden**, ya que el usuario sí tiene una sesión válida, pero su rol o permisos no le permiten acceder al recurso. Es decir, el sistema reconoce quién es el usuario, pero este no tiene autorización para acceder al contenido solicitado.

Caso 07: Una API devuelve **200** con un cuerpo vacío. ¿Sería más correcto devolver **204**?

Depende del contexto

Devolver **204** es más correcto cuando se procesa correctamente la solicitud, pero **no necesita devolver nada** ejemplo actualizar o eliminar información.

Devolver **200** es más correcto cuando se procesa correctamente la solicitud, aunque no haya datos que devolver ese cuerpo vacío es un resultado valido ejemplo una petición, GET al **endpoint** /facturas?estado = aprobadas, devuelve 200 OK con [] (lista vacía).

Esto es correcto, porque la solicitud fue exitosa y el cuerpo tiene significado: “**no hay facturas aprobadas**”.

Caso 08: Un balanceador devuelve **502**. ¿Qué diferencia hay con un **504**?

502 Bad Gateway: La respuesta **llegó, pero está mal es incorrecta o inesperada**, es como cuando llamas a alguien y preguntas algo y te responde con otra cosa que no tiene nada que ver con la pregunta.

504 Gateway Timeout: Esperas largas por parte del servidor, **tarda demasiado y no llega**. Es como si llamaras a alguien y nadie contestara antes de colgar.

Caso 09: Un recurso fue eliminado permanentemente. ¿Qué código usarías: **404** o **410**?

El código **410**, porque indica que el recurso fue **eliminado permanentemente** y no volverá a estar disponible en el servidor, es decir que, el servidor reconoce que ese recurso existió en algún momento, pero ya no existe y no se debe volver a solicitar

Caso 09: Una petición POST con JSON válido falla porque falta un campo obligatorio. ¿Devolverías **400** o **422**?

400 Bad Request: Indica que la petición no se puede procesar debido a un **error de sintaxis o estructura inválida**. Por ejemplo, el payload puede estar incompleto o contener errores de forma.

422 Unprocessable Entity: Indica que la petición es **sintácticamente correcta**, pero **no cumple con las reglas de negocio o validación del servidor**. Por ejemplo, falta un campo obligatorio en un JSON válido

4. Describir los siguientes tipos de pruebas, indicar qué implicaciones tiene no hacerlas durante un proyecto y quién sería responsable de ejecutarlas (desde su experiencia y conocimiento).

Tipo de prueba	Descripción	Implicaciones no ejecutarlas	Responsable
Unit test	Validan que cada parte pequeña del sistema (una función o módulo) funcione correctamente antes de integrarse.	Se pueden acumular errores desde etapas tempranas, afectando otras partes del sistema.	Desarrollador
Component test	Verifican que un módulo completo del sistema (por ejemplo, login o registro) funcione bien por sí solo.	Si no se hace, los módulos podrían fallar al integrarse con otros.	Desarrollador / QA tecnico
Contract test	Aseguran que los sistemas que se comunican entre sí (como APIs) intercambien la información correcta.	Pueden generarse errores de conexión o respuestas incorrectas entre servicios.	Equipo de integración/ Desarrollador
Load test	Comprueban si el sistema soporta la cantidad de usuarios o transacciones esperadas.	El sistema podría volverse lento o caerse con muchos usuarios activos	Performance/ Infraestructura
Stress test	Evalúan cómo responde el sistema cuando se exige más de lo normal.	No saber el límite del sistema puede causar fallas en momentos críticos.	Performance
Endurance test	Miden si el sistema se mantiene estable cuando se usa por mucho tiempo.	Pueden aparecer errores o lentitud después de horas o días de uso continuo.	performance
Scalability test	Prueban si el sistema puede crecer (más	Si no se prueba, el sistema podría no	Performance / Arquitectode

	usuarios o más datos) sin perder rendimiento	adaptarse al aumento real de demanda.	soluciones
Volume test	Verifican cómo se comporta el sistema con una gran cantidad de datos guardados o procesados.	Puede haber lentitud o errores al manejar información masiva.	Performance/Analista BD
Smoke test	Pruebas rápidas para confirmar que las funciones principales del sistema están operativas tras una actualización.	Podría pasarse a pruebas más complejas sobre una versión que ni siquiera inicia correctamente	QA funcional
Exploratory test	El QA prueba libremente la aplicación para descubrir errores que no están en los casos de prueba	Se pueden escapar errores no previstos o comportamientos extraños del sistema.	QA funcional
Regression test	Aseguran que los cambios o mejoras no dañen funciones que ya estaban correctas.	Pueden reaparecer errores antiguos o fallar partes del sistema que antes funcionaban.	QA funcional / Automatizador

5. Pruebas en Postman - parte 1

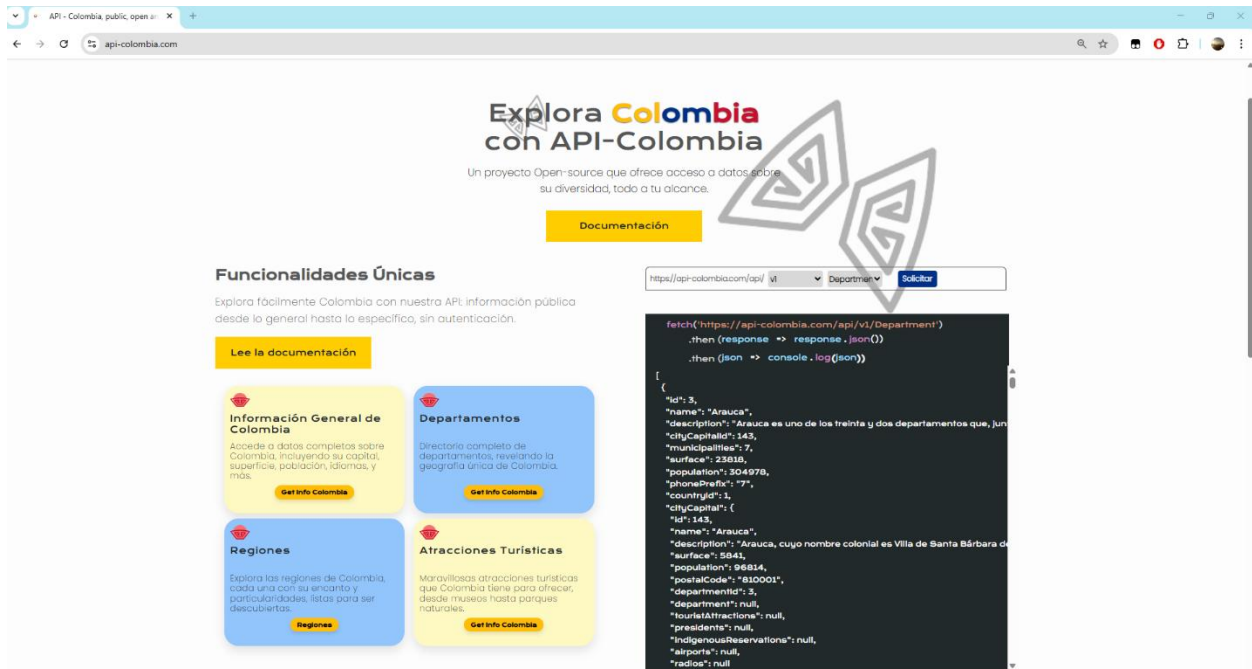
Objetivo

Usar la API pública de Colombia (<https://api-colombia.com/>) para realizar consultas sobre departamentos, ciudades y atracciones turísticas, todo desde Postman.

Antes de realizar la consulta del departamento, se ingresó a la **interfaz del API de Colombia** (<https://api-colombia.com/>) con el propósito de explorar su estructura y verificar los recursos disponibles.

A través de la interfaz se identificaron los **endpoints principales**, entre ellos **/api/v1/Department/{nombre}**, el cual permite consultar la información de un departamento específico.

Asimismo, se observó la **estructura de la respuesta en formato JSON**, donde se visualizaron los campos principales: id, name, description, municipalities, population y cityCapital, entre otros. Con base en esta información, se procedió a **validar los datos correspondientes al departamento seleccionado**.



Explora Colombia con API-Colombia

Un proyecto Open-source que ofrece acceso a datos sobre su diversidad, todo a tu alcance.

Documentación

Funcionalidades Únicas

Explora fácilmente Colombia con nuestra API: información pública desde lo general hasta lo específico, sin autenticación.

Lee la documentación

Información General de Colombia
Accede a datos completos sobre Colombia, incluyendo su capital, superficie, población, idiomas, y más.
[Get Info Colombia](#)

Departamentos
Directorio completo de departamentos, resumiendo la geografía única de Colombia.
[Get Info Colombia](#)

Regiones
Explora las regiones de Colombia, cada una con su encanto y particularidades, listas para ser descubiertas.
[Regiones](#)

Atracciones Turísticas
Maravillosas atracciones turísticas que Colombia tiene para ofrecer, desde museos hasta parques naturales.
[Get Info Colombia](#)

```
fetch('https://api-colombia.com/api/v1/Department')
  .then(response => response.json())
  .then(json => console.log(json))
```

```
{
  "id": 3,
  "name": "Arauca",
  "description": "Arauca es uno de los treinta y dos departamentos que, junto con Bogotá, conforman la República de Colombia. Su capital es el municipio de Arauca.",
  "cityCapital": 143,
  "municipalities": 7,
  "surface": 23810,
  "population": 104970,
  "phonePrefix": "77",
  "countryId": 1,
  "cityCapital": {
    "id": 143,
    "name": "Arauca",
    "description": "Arauca, cuyo nombre colonial es Villa de Santa Bárbara de Arauca, es la capital del departamento de Arauca.",
    "surface": 5841,
    "population": 94814,
    "postalCode": "810001",
    "department": 3,
    "department": null,
    "touristAttractions": null,
    "presidents": null,
    "indigenousReservations": null,
    "airports": null,
    "rivers": null
  }
}
```

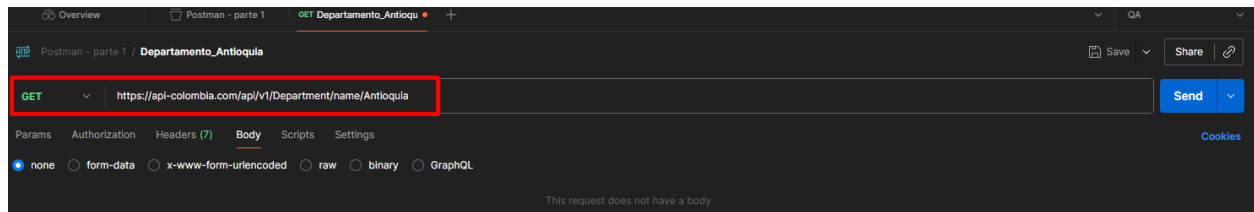
Requerimientos para la práctica con la API de pública de Colombia

Requerimiento 01: Capital y número de municipios del departamento de Antioquia

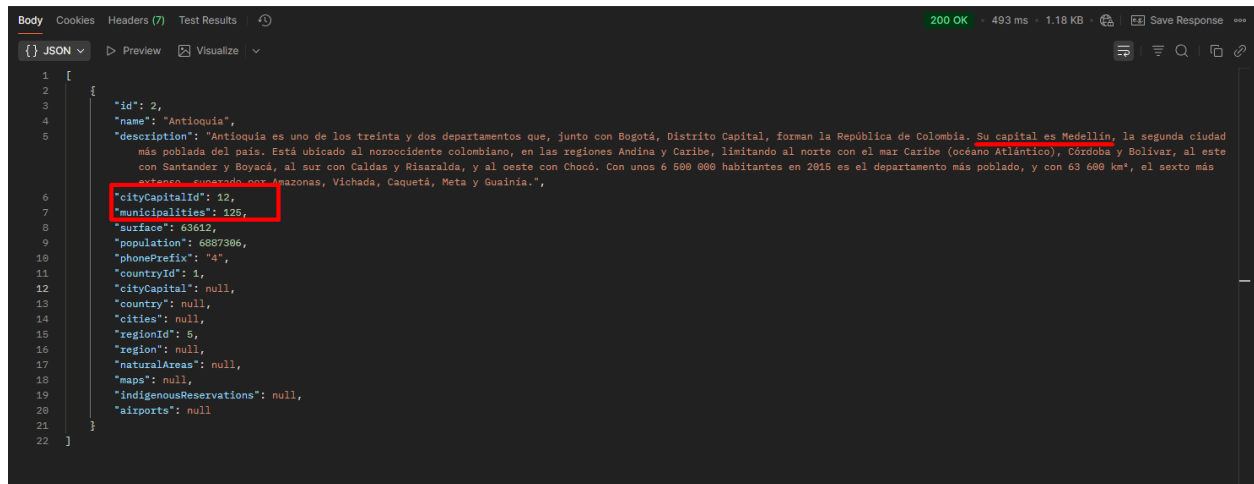
URL base: <https://api-colombia.com/api/v1>

Endpoint: /Department/name/Antioquia

Método: GET



Respuesta

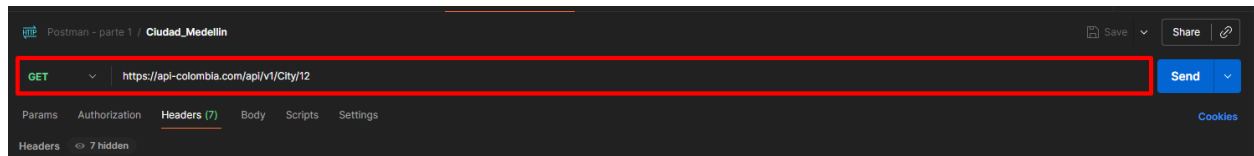
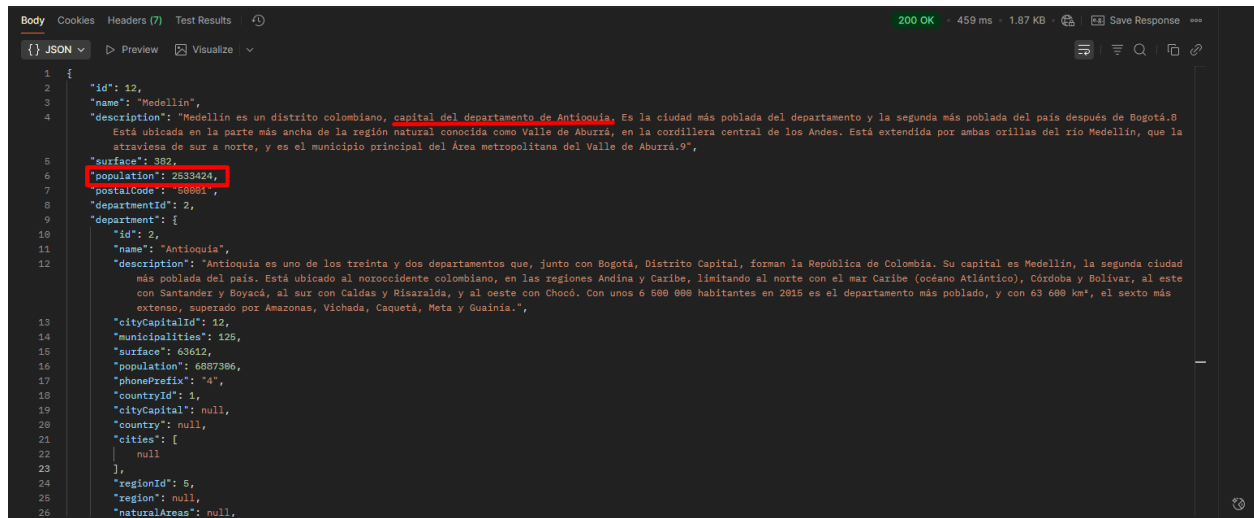


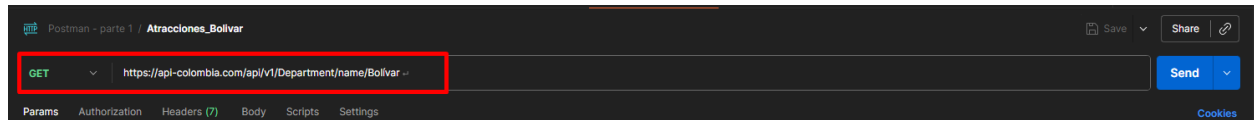
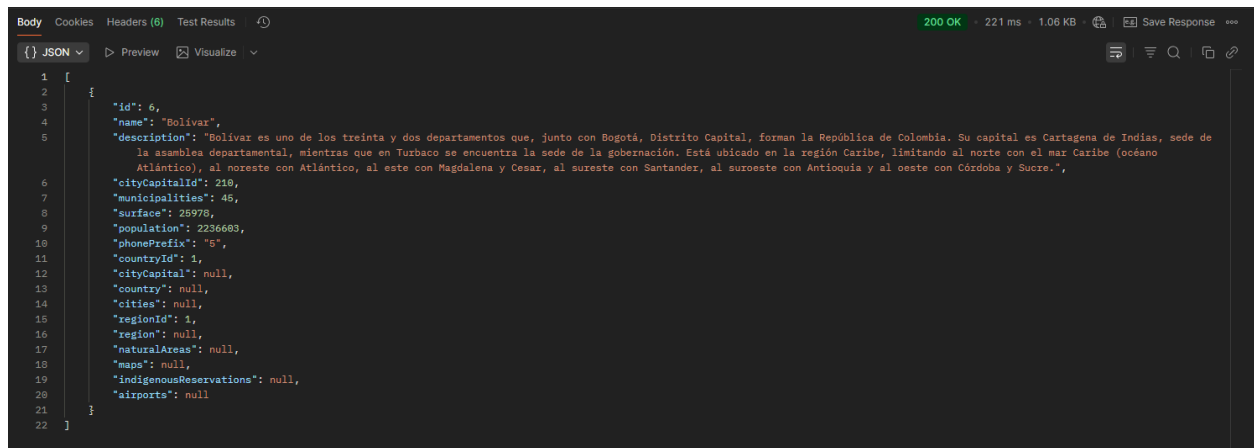
Código de estado de la petición HTTP: **200 OK**

Capital De Antioquia: **Medellín**

ID de la capital: **12**

Cantidad de Municipios: **125**

Requerimiento 02: Verificar si Medellín pertenece a Antioquia y obtener su población**URL base:** <https://api-colombia.com/api/v1>**Endpoint:** /City/12**Método:** GET**Respuesta****Código de estado de la petición HTTP: 200 OK****Departamento al que pertenece: Antioquia****Población: 2533424 habitantes**

Requerimiento 03: Listar 3 atracciones turísticas en el departamento de Bolívar**URL base:** <https://api-colombia.com/api/v1>**Endpoint:** /Department/name/Bolivar**Método:** GET**Respuesta****Código de estado de la petición HTTP: 200 OK**Se observó que el campo **touristAttractions** no está presente en la respuesta.

NOTA: El recurso de atracciones turísticas (touristAttractions) no está disponible en la versión actual del API, por lo que no es posible listar las atracciones directamente desde la consulta del departamento.

Sin embargo, al intentar realizar una segunda petición al endpoint **/Department/6/touristicAttractions**, la API devuelve un **código 404 (Not Found)**, lo que confirma que este recurso no está habilitado o no existe en la versión actual del servicio.

Esto limita la obtención de información completa requerida para el análisis o cumplimiento del requerimiento correspondiente.

R3_BUG-01

Título: Endpoint `/Department/6/touristicAttractions` devuelve **404** y no permite obtener `touristAttractions` para departamentos

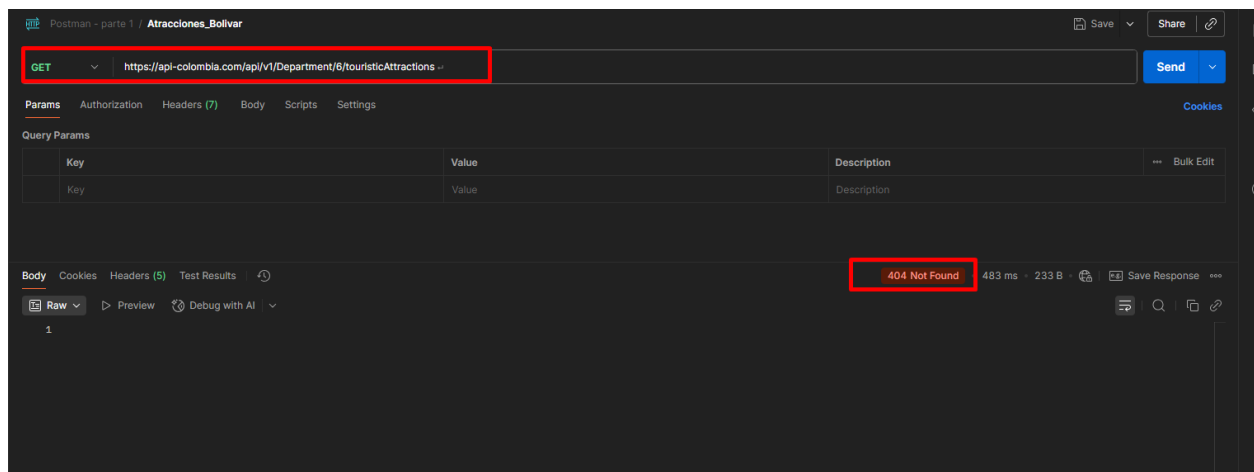
Descripción:

Al intentar obtener las atracciones turísticas de un departamento específico (por ejemplo, Bolívar), el endpoint devuelve un **código 404 Not Found**. Esto impide cumplir con el **Requerimiento 03**, ya que no es posible acceder a la información de atracciones turísticas directamente desde la API.

Reproducción del error 404 en touristAttractions

Dado que se realiza una petición GET al endpoint `/Department/6/touristicAttractions`,
Cuando se ejecuta la solicitud,
Entonces la API responde con código **404 (Not Found)**, lo que indica que el recurso no está disponible o fue removido de la versión actual del servicio.

Evidencia:



Resultado esperado:

- Código HTTP: 200 OK
- Respuesta JSON con el campo `touristAttractions` que contenga una lista de atracciones turísticas del departamento

Resultado actual:

- Código HTTP: 404 Not Found
- El recurso no existe o no está disponible en la versión actual del API.

Información adicional:

URL base: <https://api-colombia.com/api/v1>

Endpoint: /Department/6/touristicAttractions

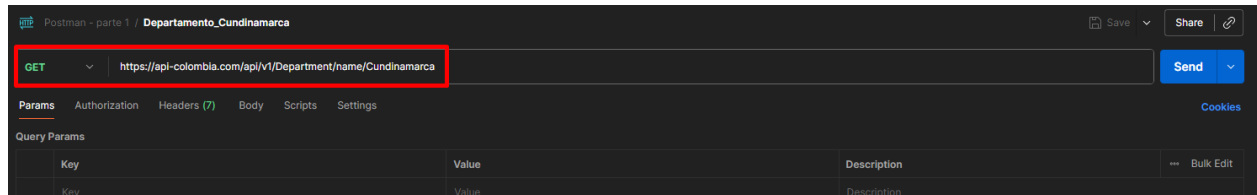
Método: GET

Impacto: Moderado / Medio

- No detiene toda la operación del API, pero impide cumplir con un requerimiento funcional específico.
- Recomendable asignarlo a desarrollo para corrección o actualización de documentación del endpoint.

Estado: New

Asignado: DEV

Requerimiento 04: Buscar el departamento de Cundinamarca.**URL base:** <https://api-colombia.com/api/v1>**Endpoint:** /Department/name/Cundinamarca**Método:** GET**Respuesta****Código de estado de la petición HTTP: 200 OK****Capital De Antioquia: Bogotá****ID de la capital: 167****Cantidad de Municipios: 116****Población: 2473634 habitantes**

6. Pruebas en Postman - parte 2

Objetivo

Usar la **API pública de Rick and Morty** (<https://rickandmortyapi.com/>) para realizar consultas sobre personajes, episodios y ubicaciones, todo desde Postman.

Antes de realizar las consultas específicas, se ingresó a la interfaz de la API de Rick and Morty con el propósito de explorar su estructura y verificar los recursos disponibles.

A través de la interfaz se identificaron los **endpoints principales**, entre ellos:

- `/api/character/{id}`: permite consultar información detallada de un personaje específico.
- `/api/episode/{id}`: permite consultar datos de un episodio determinado.
- `/api/location/{id}`: permite consultar información sobre una ubicación específica.

Asimismo, se observó la **estructura de la respuesta en formato JSON**, donde se visualizaron los campos principales:

- **Para personajes:** id, name, status, species, gender, origin, location, episode.
- **Para episodios:** id, name, air_date, episode, characters.
- **Para ubicaciones:** id, name, type, dimension, residents.

Con base en esta información, se procedió a **validar los datos correspondientes** a los personajes, episodios y ubicaciones seleccionados, verificando aspectos como el estado de los personajes, la aparición en episodios y la ubicación actual de cada personaje.



URL base: <https://rickandmortyapi.com/api>

La URL base contiene información sobre todos los recursos de la API disponibles. Todas las solicitudes son GET solicitudes y se envían a través de [nombre del archivo] [https](https://rickandmortyapi.com/api). Todas las respuestas devolverán datos en [nombre del archivo] `json`.

```
GET https://rickandmortyapi.com/api
```

```
{
  "characters": "https://rickandmortyapi.com/api/character",
  "locations": "https://rickandmortyapi.com/api/location",
  "episodes": "https://rickandmortyapi.com/api/episode"
}
```

Actualmente hay tres recursos disponibles:

- **Personaje**: se utiliza para obtener todos los personajes.
- **Ubicación**: se utiliza para obtener todas las ubicaciones.
- **Episodio**: se utiliza para obtener todos los episodios.

Requerimientos para la práctica con la API de Rick and Morty

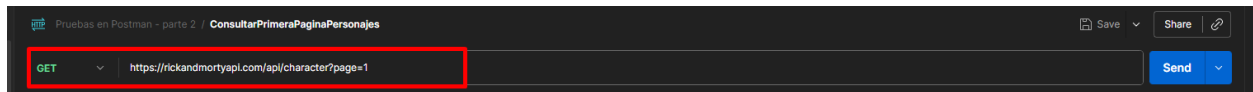
Requerimiento 01: Consultar la primera página de personajes y obtener el código de estado HTTP y la cantidad de personajes en la primera página.

URL base: <https://rickandmortyapi.com/api>

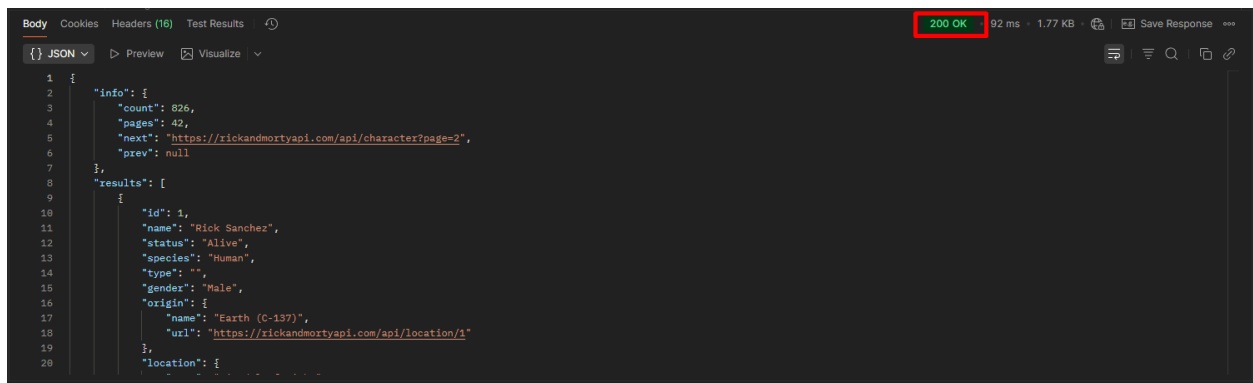
Endpoint: /carácter

Parámetros de consulta: ? page=1

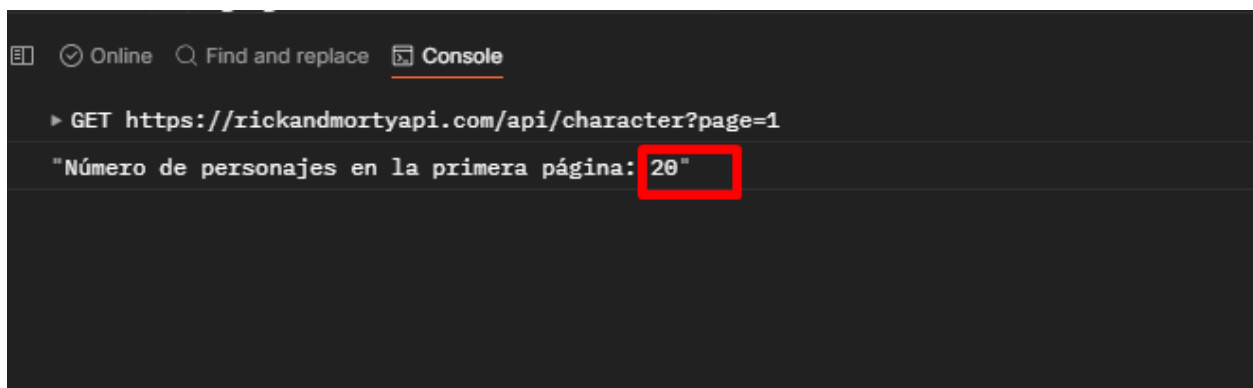
Método: GET



Respuesta:



Código de estado de la petición HTTP: 200 OK



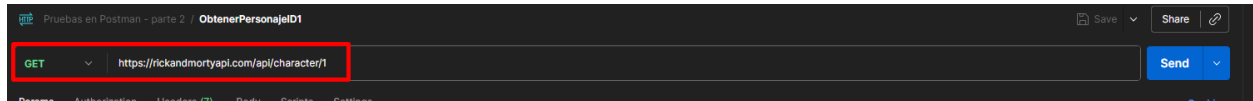
Cantidad de personajes en la primera página: 20

Requerimiento 02: Obtener el nombre, estado y especie del personaje con ID = 1.

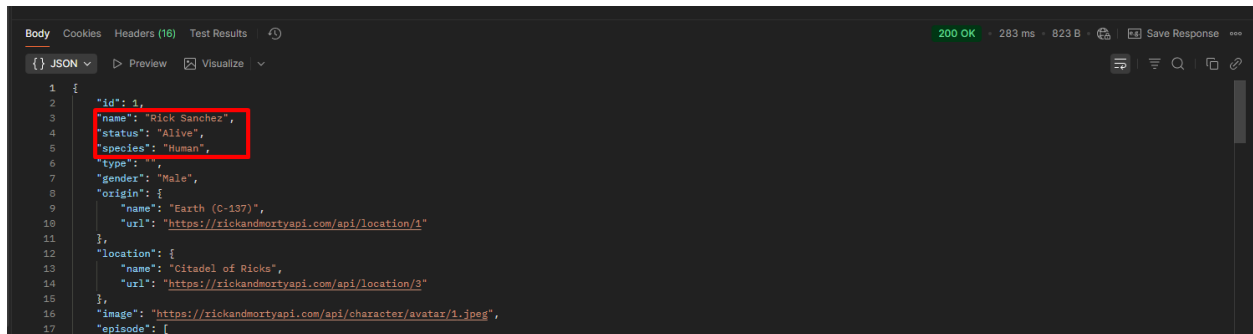
URL base: <https://rickandmortyapi.com/api>

Endpoint: /character/1

Método: GET



Respuesta:



Nombre: Rick Sanchez

Estado: Alive (Vivo)

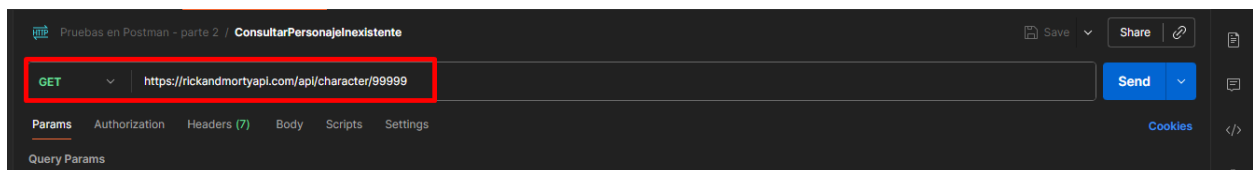
Especie: Human

Requerimiento 03: Consultar el personaje con ID = 99999 y determinar el código de estado HTTP y su significado.

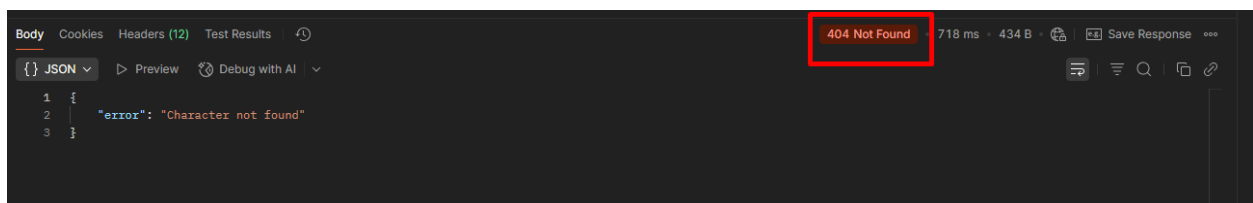
URL base: <https://rickandmortyapi.com/api>

Endpoint: /character/99999

Método: GET



Respuesta



Código de estado: La API devuelve un código de estado HTTP **404 Not Found**, cuando se consulta el personaje con ID=99999 no existe, indica que el recurso (personaje) solicitado no se encuentra disponible en el servidor o no existe.

Requerimiento 04: Comparar la primera y segunda página de personajes y describir la diferencia.

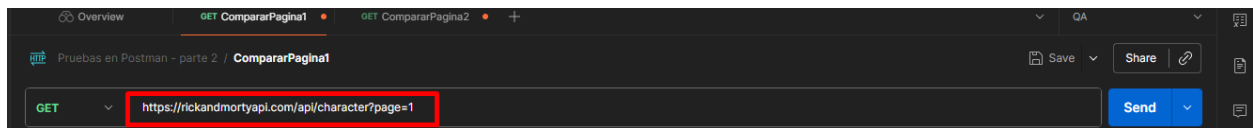
Petición Página 1

URL base: <https://rickandmortyapi.com/api>

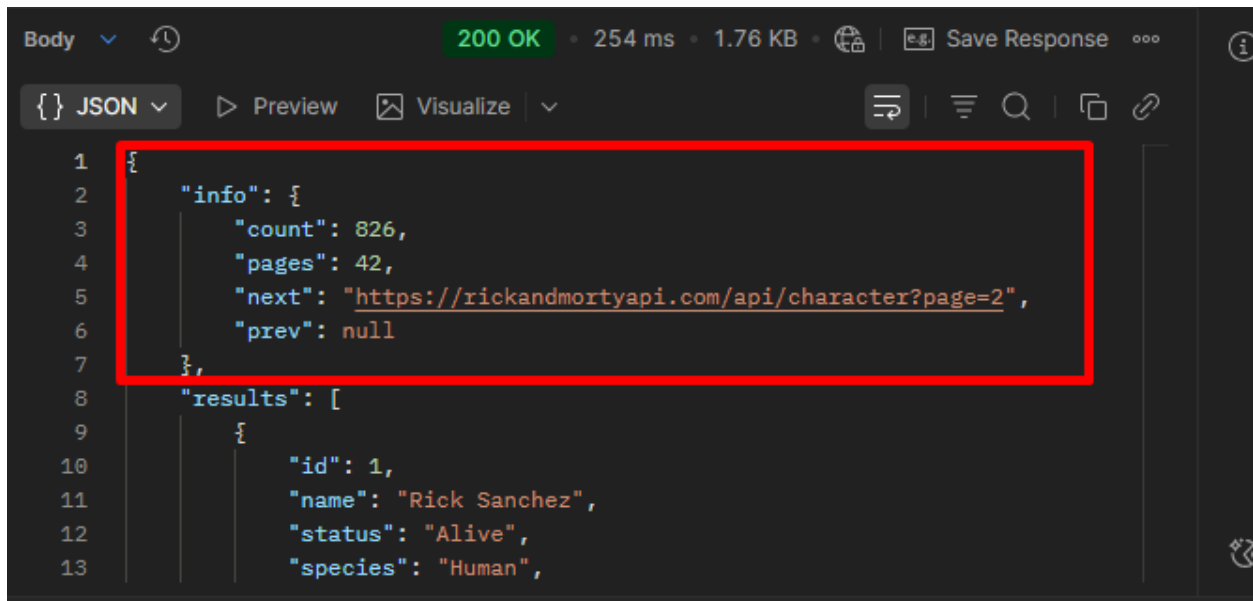
Endpoint: /carácter

Parámetros de consulta: /? page=1

Método: GET



Respuesta



Total, de personajes existentes en toda la API: 826

Total, de páginas disponibles: 42

URL de la siguiente página: pagina 2

URL de la página anterior: null

Cantidad total de personajes por página: 20

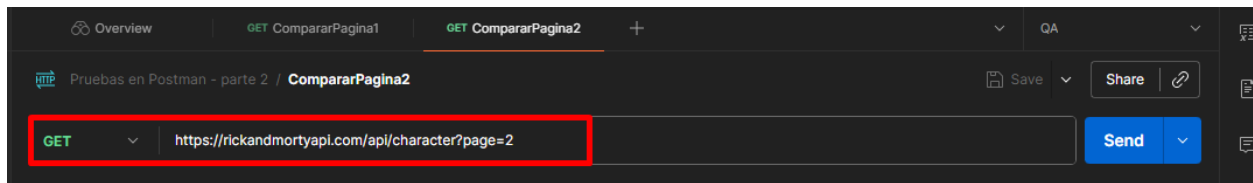
Petición Página 2

URL base: <https://rickandmortyapi.com/api>

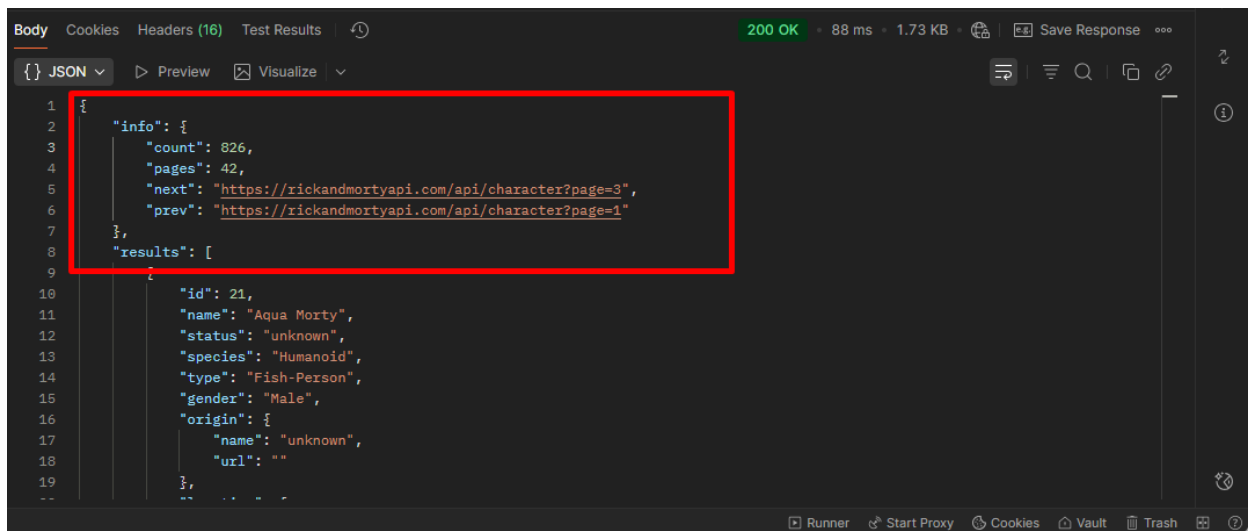
Endpoint: /carácter

Parámetros de consulta: /? page=2

Método: GET



Respuesta



Total, de personajes existentes en toda la API: 826

Total, de páginas disponibles: 42

URL de la siguiente página: página 3

URL de la página anterior: página 1

Cantidad total de personajes por página: 20

Se consultan los personajes por nombre, y la cantidad de personajes por página

```
Online Find and replace Console
> GET https://rickandmortyapi.com/api/character?page=1
(20) ["Rick Sanchez", "Morty Smith", "Summer Smith", "Beth Smith", "Jerry Smith", ...]
0: "Rick Sanchez"
1: "Morty Smith"
2: "Summer Smith"
3: "Beth Smith"
4: "Jerry Smith"
5: "Abadango Cluster Princess"
6: "Abradolf Lincler"
7: "Adjudicator Rick"
8: "Agency Director"
9: "Alan Rails"
10: "Albert Einstein"
11: "Alexander"
12: "Alien Googah"
13: "Alien Morty"
14: "Alien Rick"
15: "Amish Cyborg"
16: "Annie"
17: "Antenna Morty"
18: "Antenna Rick"
19: "Ants in my Eyes Johnson"
"Cantidad total de personajes pag.1:" 20
> GET https://rickandmortyapi.com/api/character?page=2
(20) ["Aqua Morty", "Aqua Rick", "Arcade Alien", "Armageddon", "Armothy", ...]
0: "Aqua Morty"
1: "Aqua Rick"
2: "Arcade Alien"
3: "Armageddon"
4: "Armothy"
5: "Arthricia"
6: "Artist Morty"
7: "Attila Starwar"
8: "Baby Legs"
9: "Baby Poopybutthole"
10: "Baby Wizard"
11: "Bearded Lady"
12: "Beebo"
13: "Benjamin"
14: "Bepisian"
15: "Beta-Seven"
16: "Beth Sanchez"
17: "Beth Smith"
18: "Beth Smith"
19: "Beth's Mytholog"
"Cantidad total de personajes pag.2:" 20
```

Después de comprar la pagina 1 y 2 se observa que, la diferencia principal está en en **los personajes que se presentan en cada página** y en **cómo se gestionan los enlaces de paginación entre páginas**; Ambas páginas tienen la misma cantidad de personajes, pero la paginación difiere: la **primera** no tiene página anterior (**prev es null**), mientras que la **segunda** sí apunta a la página 1 y a la página 3 (**prev y next**).

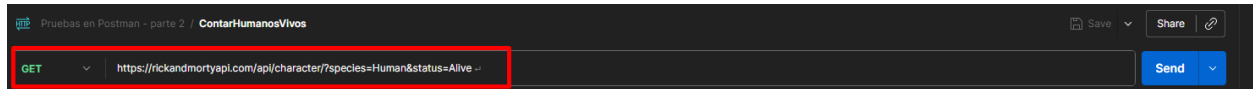
Requerimiento 05: Contar cuántos personajes humanos vivos aparecen en la API.

URL base: <https://rickandmortyapi.com/api>

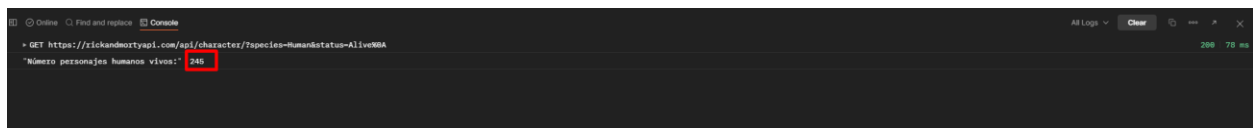
Endpoint: /carácter

Parámetros de consulta /?species=Human&status=Alive

Método: GET



Respuesta



La cantidad total de personajes registrados en la API, **245 son humanos y están vivos**

Requerimiento 06: Listar los personajes que aparecen en el primer episodio.

El requerimiento se ejecuta en dos pasos

Paso 1: Obtener los IDs de los personajes del episodio

URL base: <https://rickandmortyapi.com/api>

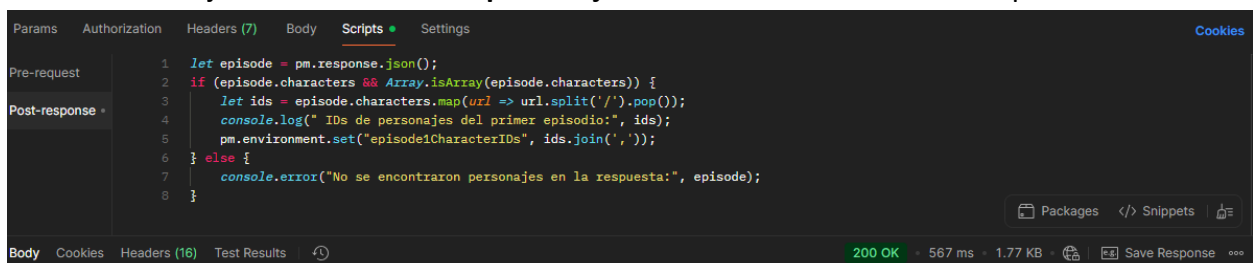
Endpoint: /episode/1

Método: GET



En la respuesta JSON, la propiedad characters contiene las URLs de todos los personajes que aparecen en ese episodio.

- Se extrajeron los **IDs de cada personaje** desde las URLs usando un script



Respuesta

```
► GET https://rickandmortyapi.com/api/episode/1

"IDs de personajes del primer episodio:" ▾ (19) ["1", "2", "35", "38", "62", ...]
  0: "1"
  1: "2"
  2: "35"
  3: "38"
  4: "62"
  5: "92"
  6: "127"
  7: "144"
  8: "158"
  9: "175"
 10: "179"
 11: "181"
 12: "239"
 13: "249"
 14: "271"
 15: "338"
 16: "394"
 17: "395"
 18: "435"
```

Paso 2: Obtener los nombres de los personajes

URL base: <https://rickandmortyapi.com/api>

Endpoint: / character / 1,2,35...

Método: GET

Se realizó una petición a /character pasando todos los IDs en la misma URL

```
GET https://rickandmortyapi.com/api/character/1,2,35,38,62,92,127,144,158,175,179,181,239,249,271,338,394,395,435
```

Esto devuelve un JSON con todos los datos de los personajes especificados.

De esta manera, se pudo **listar los nombres de todos los personajes que aparecen en el episodio 1** con el siguiente script

```
Params Authorization Headers (7) Body Scripts Settings Cookies
Pre-request
Post-response
1 let characters = pm.response.json();
2
3 // Normalizar a array
4 if (!Array.isArray(characters)) {
5   characters = [characters];
6 }
7
8 // Extraer nombres
9 let names = characters.map(c => c.name);
10
11 // Mostrar nombres en consola
12 console.log("Nombres de personajes del episodio 1:", names);
13
14 // Guardar en variable de entorno si quieres
15 pm.environment.set("episode1CharacterNames", JSON.stringify(names));

Body Cookies Headers (16) Test Results 200 OK 317 ms 1.68 KB Save Response
JSON Preview Visualize
```

El primer episodio incluye a los siguientes personajes:

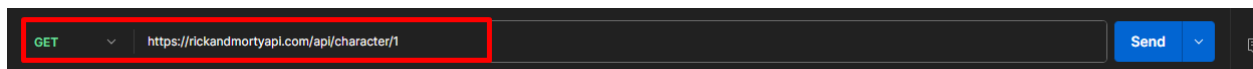
```
Online Find and replace Console All Logs Clear
> GET https://rickandmortyapi.com/api/episode/1 200 82
  "Ids de personajes del primer episodio:" -> (19) ["1", "2", "35", "38", "62", ...]
> GET https://rickandmortyapi.com/api/character/1,2,35,38,62,92,127,144,158,176,179,181,239,249,271,338,394,399,435MMMA 200 317
  "Nombres de personajes del episodio 1:" -> (19) ["Rick Sanchez", "Morty Smith", "Hershel", "Beth Smith", "Canklankz Thom", ...]
    0: "Rick Sanchez"
    1: "Morty Smith"
    2: "Hershel"
    3: "Beth Smith"
    4: "Canklankz Thom"
    5: "Davin"
    6: "Frank Plalicky"
    7: "Glenn"
    8: "Hookah Allen"
    9: "Jerry Smith"
   10: "Jessica"
   11: "Jessica's Friend"
   12: "Mr. Goldenfold"
   13: "Mrs. Sanchez"
   14: "Principal Vagina"
   15: "Summer Smith"
   16: "Davin"
   17: "Greenybabe"
   18: "Principal"
  
```

Requerimiento 07: Configurar una prueba que valide que el campo status del personaje Rick Sanchez sea "Alive".

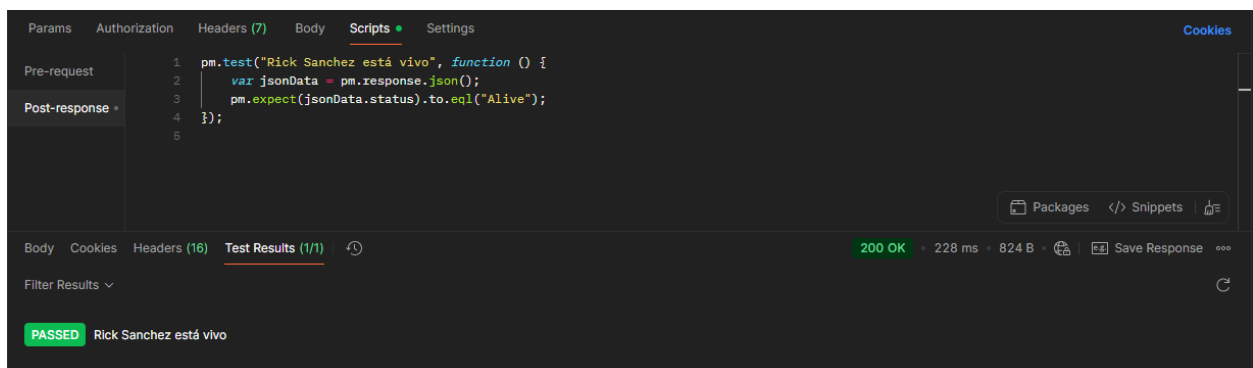
URL base: <https://rickandmortyapi.com/api>

Endpoint: /character/1

Método: GET



Resultado del test



Requerimiento 08: Obtener los datos del personaje Morty Smith.

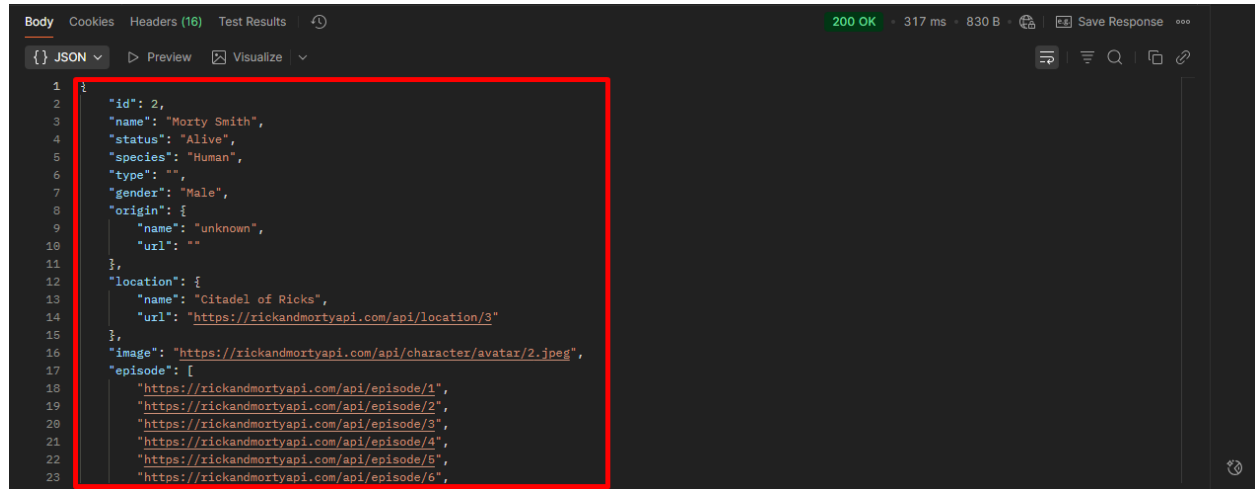
URL base: <https://rickandmortyapi.com/api>

Endpoint: /character/2

Método: GET



Respuesta

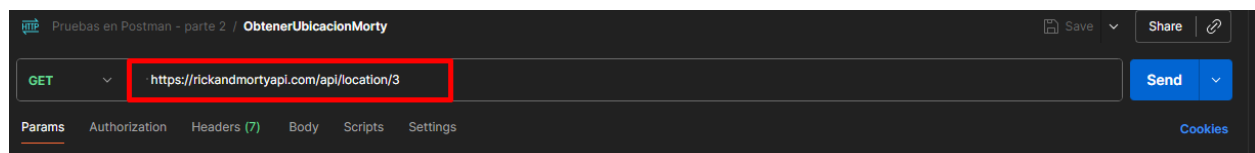


Requerimiento 09: A partir del campo location.url de Morty Smith, realizar un nuevo request para obtener información de su ubicación actual.

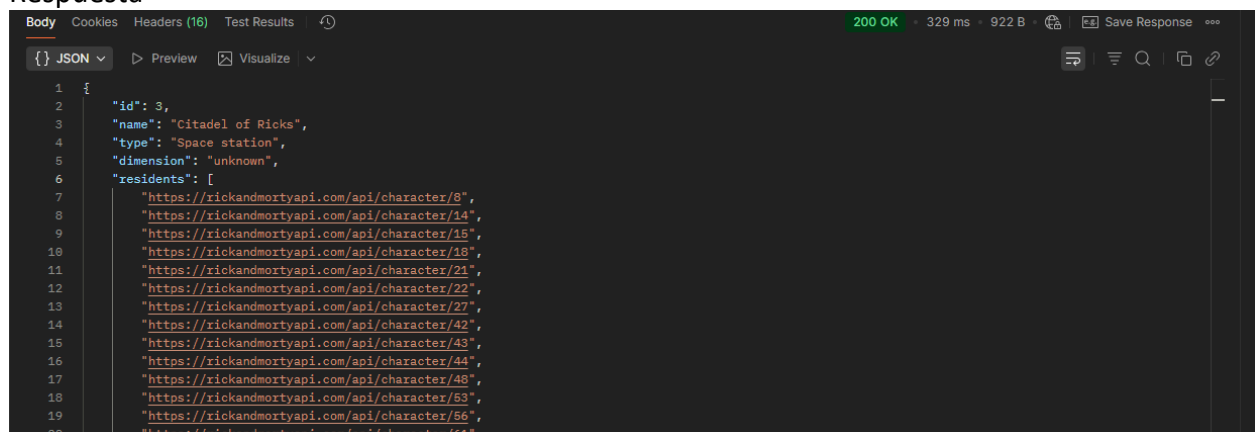
URL base: https://rickandmortyapi.com/api

Endpoint: /location/3

Método: GET



Respuesta



Se visualiza la información completa de la ubicación actual de Morty Smith con la siguiente información:

- **name:** Nombre de la ubicación.
- **type:** Tipo de lugar.
- **dimension:** Dimensión en la que se encuentra.
- **residents:** Lista de URLs de los personajes que viven ahí.

Requerimiento 10: Determinar el nombre de la ubicación actual de Morty Smith.

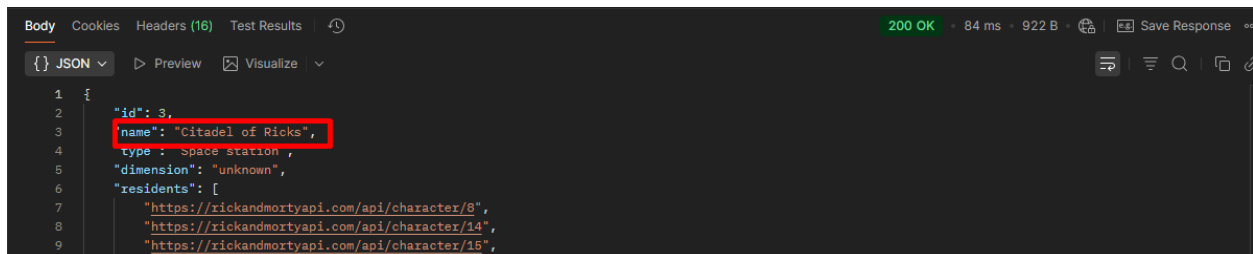
URL base: <https://rickandmortyapi.com/api>

Endpoint: `/location/3`

Método: GET



El nombre de la ubicación actual de Morty Smith es: **Citadel of Ricks**,



Requerimiento 11: Obtener los datos de Rick Sanchez y Summer Smith

URL base: <https://rickandmortyapi.com/api>

Endpoint: `/character/1,3`

Método: GET



Respuesta

Datos de Rick Sanchez

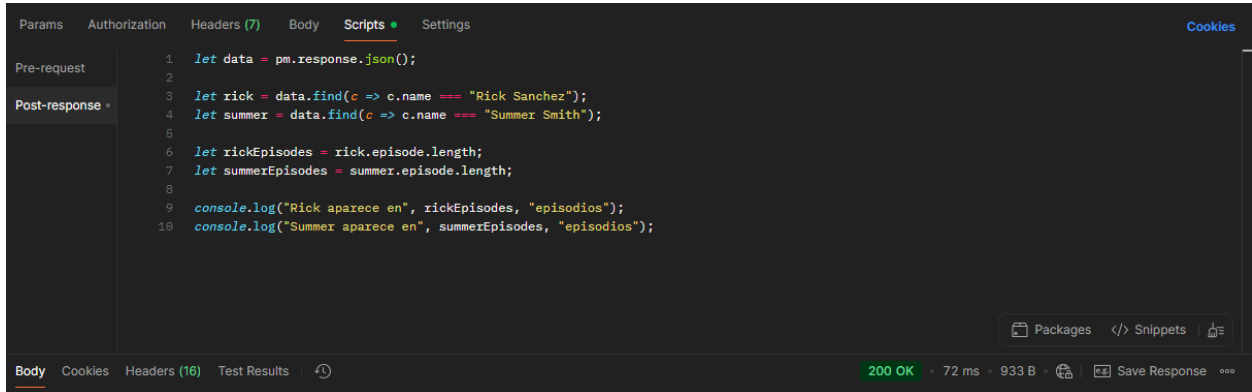
```
1  [
2    {
3      "id": 1,
4      "name": "Rick Sanchez",
5      "status": "Alive",
6      "species": "Human",
7      "type": "",
8      "gender": "Male",
9      "origin": {
10       "name": "Earth (C-137)",
11       "url": "https://rickandmortyapi.com/api/location/1"
12     },
13     "location": {
14       "name": "Citadel of Ricks",
15       "url": "https://rickandmortyapi.com/api/location/3"
16     },
17     "image": "https://rickandmortyapi.com/api/character/avatar/1.jpeg",
18     "episode": [
19       "https://rickandmortyapi.com/api/episode/1",
20       "https://rickandmortyapi.com/api/episode/2",
21       "https://rickandmortyapi.com/api/episode/3",
22     ]
23   }
24 ]
```

Datos de Summer Smith

```
{ JSON } Preview Visualize
73  {
74    {
75      "id": 3,
76      "name": "Summer Smith",
77      "status": "Alive",
78      "species": "Human",
79      "type": "",
80      "gender": "Female",
81      "origin": {
82       "name": "Earth (Replacement Dimension)",
83       "url": "https://rickandmortyapi.com/api/location/28"
84     },
85     "location": {
86       "name": "Earth (Replacement Dimension)",
87       "url": "https://rickandmortyapi.com/api/location/28"
88     },
89     "image": "https://rickandmortyapi.com/api/character/avatar/3.jpeg",
90     "episode": [
91       "https://rickandmortyapi.com/api/episode/6",
92       "https://rickandmortyapi.com/api/episode/7",
93       "https://rickandmortyapi.com/api/episode/8",
94       "https://rickandmortyapi.com/api/episode/9",
95       "https://rickandmortyapi.com/api/episode/10",
96       "https://rickandmortyapi.com/api/episode/11",
97       "https://rickandmortyapi.com/api/episode/12",
98       "https://rickandmortyapi.com/api/episode/14",
99       "https://rickandmortyapi.com/api/episode/15",
100     ]
101   }
102 }
```


Requerimiento 12: ¿Quién aparece en más episodios según el campo episode?

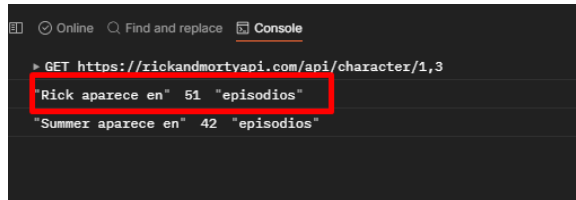
A partir de la request a la API de personajes usando sus IDs, la cantidad de episodios en los que aparece cada personaje se obtiene con el siguiente script



```
1 let data = pm.response.json();
2
3 let rick = data.find(c => c.name === "Rick Sanchez");
4 let summer = data.find(c => c.name === "Summer Smith");
5
6 let rickEpisodes = rick.episode.length;
7 let summerEpisodes = summer.episode.length;
8
9 console.log("Rick aparece en", rickEpisodes, "episodios");
10 console.log("Summer aparece en", summerEpisodes, "episodios");
```

The screenshot shows the 'Scripts' tab in a browser's developer tools. The script is a post-response handler that finds Rick Sanchez and Summer Smith in the API response, counts their episodes, and logs the results. The status bar at the bottom indicates a 200 OK response.

Lo que nos permite identificar que el personaje que **aparece en más episodios según el campo episode es Rick Sanchez con una cantidad total de 51 episodios**



```
> GET https://rickandmortyapi.com/api/character/1,3
"Rick aparece en" 51 "episodios"
"Summer aparece en" 42 "episodios"
```

The screenshot shows the browser's console output. The first log message, "Rick aparece en" 51 "episodios", is highlighted with a red box. The second log message is "Summer aparece en" 42 "episodios".