**1.Proceso de pruebas**

Ingresar a la siguiente url OrangeHRM y realizar todo el proceso de

pruebas con los siguientes items

1.1 Análisis de matriz de riesgos: [Matriz de Riesgos](https://github.com/lejismejia/Satrack-Reto-tecnico/commit/c64c8eb305c2f4cd771892531bd91a72c3bda277)

1.2 Plan de Pruebas URL: [Plan de Pruebas](https://github.com/lejismejia/Satrack-Reto-tecnico/blob/main/Plan%20de%20Pruebas%20(1).docx)

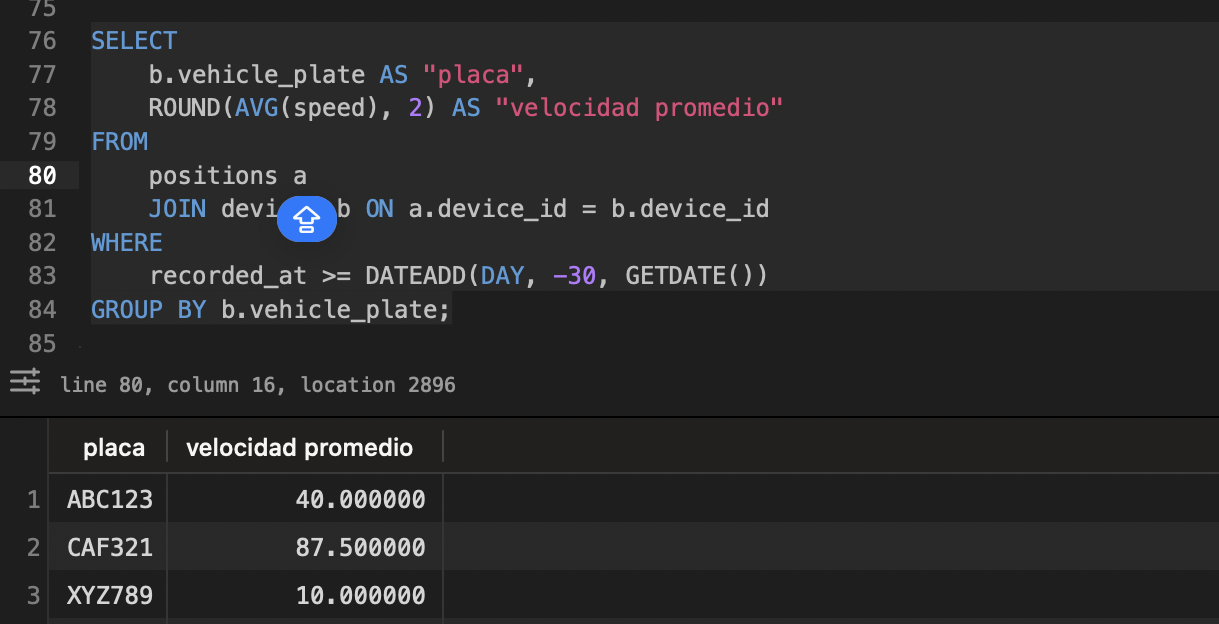
1.3 Diseño de los escenarios en lenguaje gherkin: [Escenarios de prueba](https://github.com/lejismejia/Satrack-Reto-tecnico/blob/main/Escenarios%20de%20prueba.xlsx)

1.4 Reportar defectos encontrados indicando la prioridad [bugs Reportados](https://github.com/lejismejia/Satrack-Reto-tecnico/blob/main/Bugs.xlsx)

**2. Consultas SQL**

2.1 Velocidad promedio por vehículo (últimos 30 días)

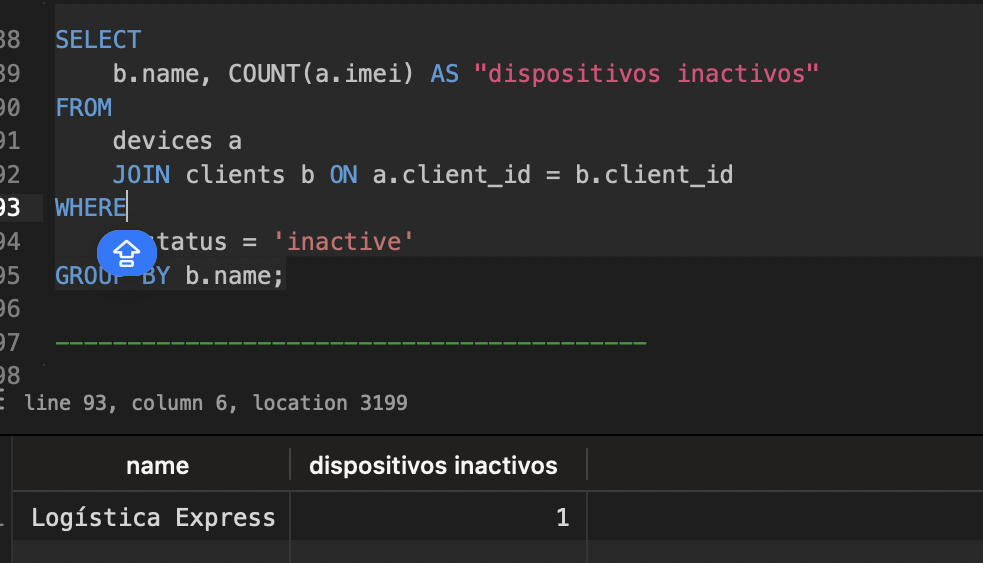
Obtener la velocidad promedio de cada vehículo en el último mes.



2.2 Clientes con dispositivos inactivos

Listar los clientes que tienen al menos un dispositivo inactivo y cuántos

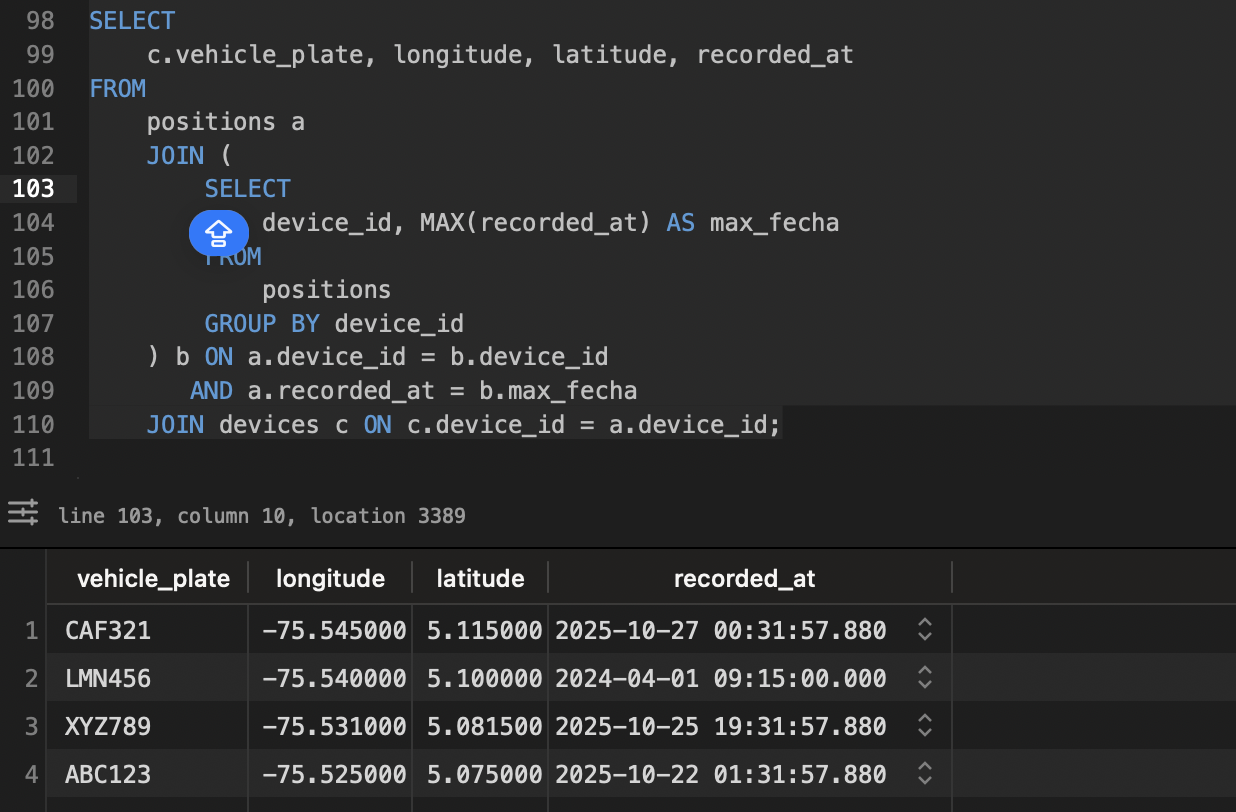
son.



2.3 Última posición de cada vehículo

Mostrar la última ubicación registrada por cada vehículo (placa, latitud,

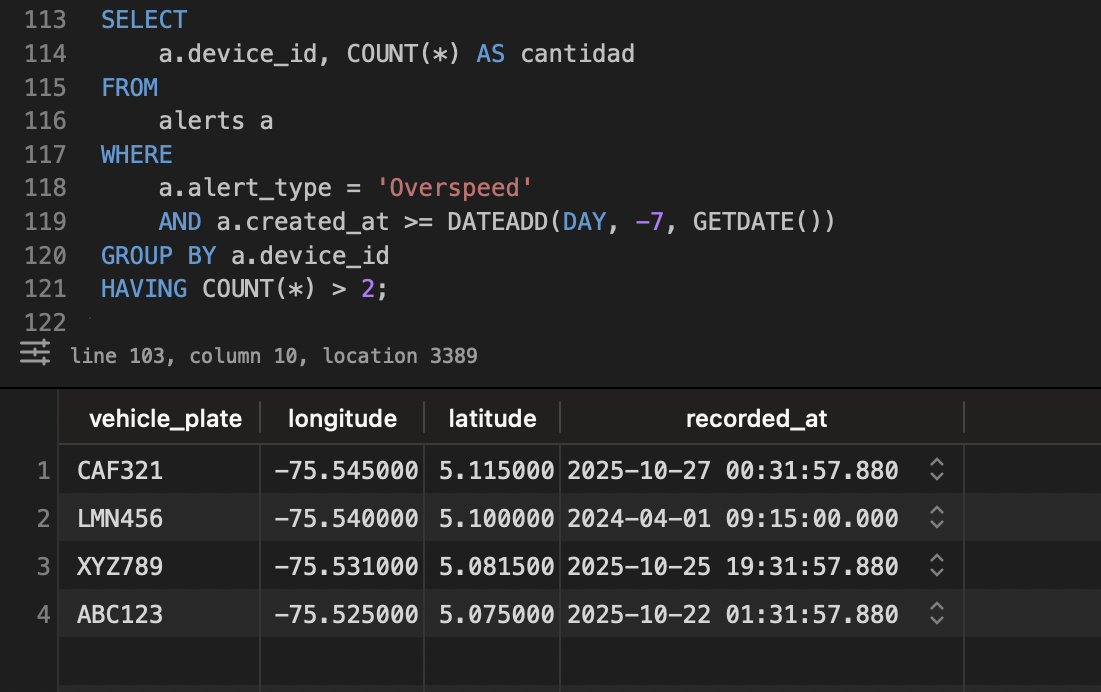
longitud, fecha).



2.4 Vehículos con múltiples alertas de exceso de velocidad

Identificar los vehículos que han generado más de 2 alertas de tipo

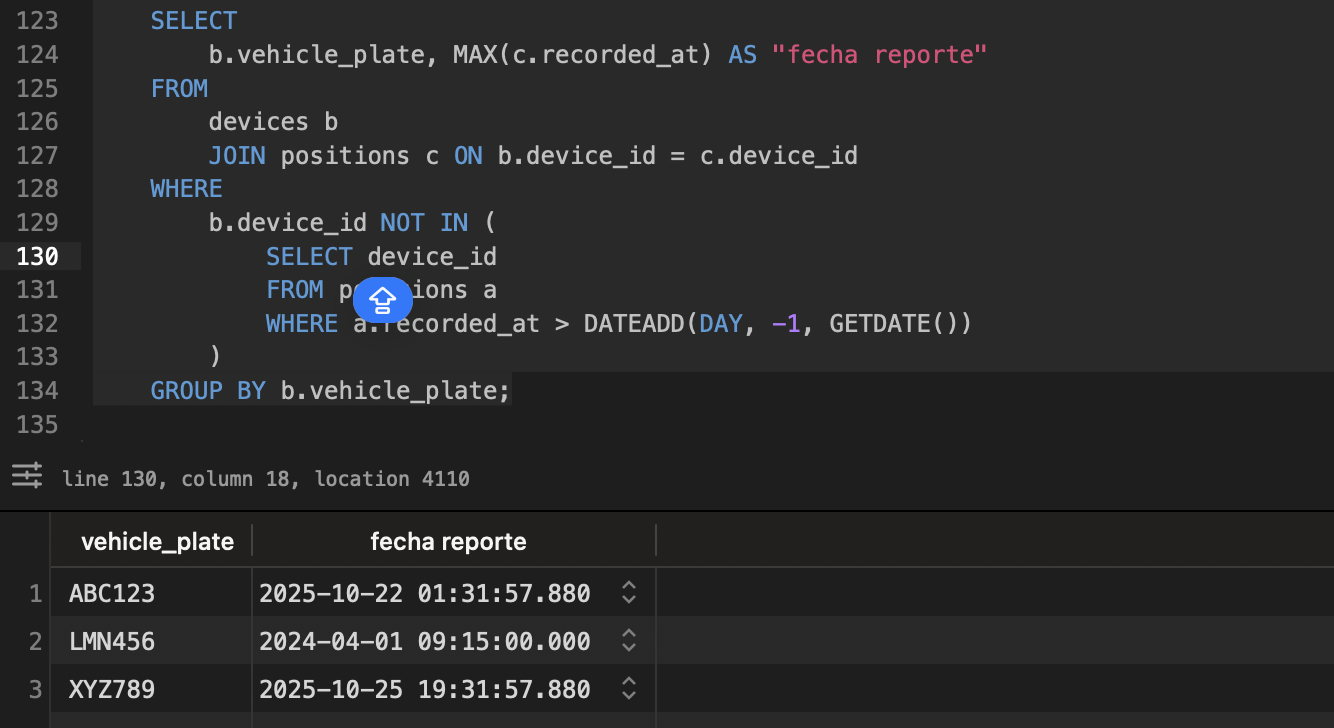
Overspeed en los últimos 7 días.



2.5 Vehículos sin reporte en 24 horas

Listar los vehículos que no han enviado posiciones en las últimas 24 horas

y su último reporte (si existe).



2.6 Tiempo promedio entre reportes

Calcular el tiempo promedio (en minutos) entre posiciones de cada

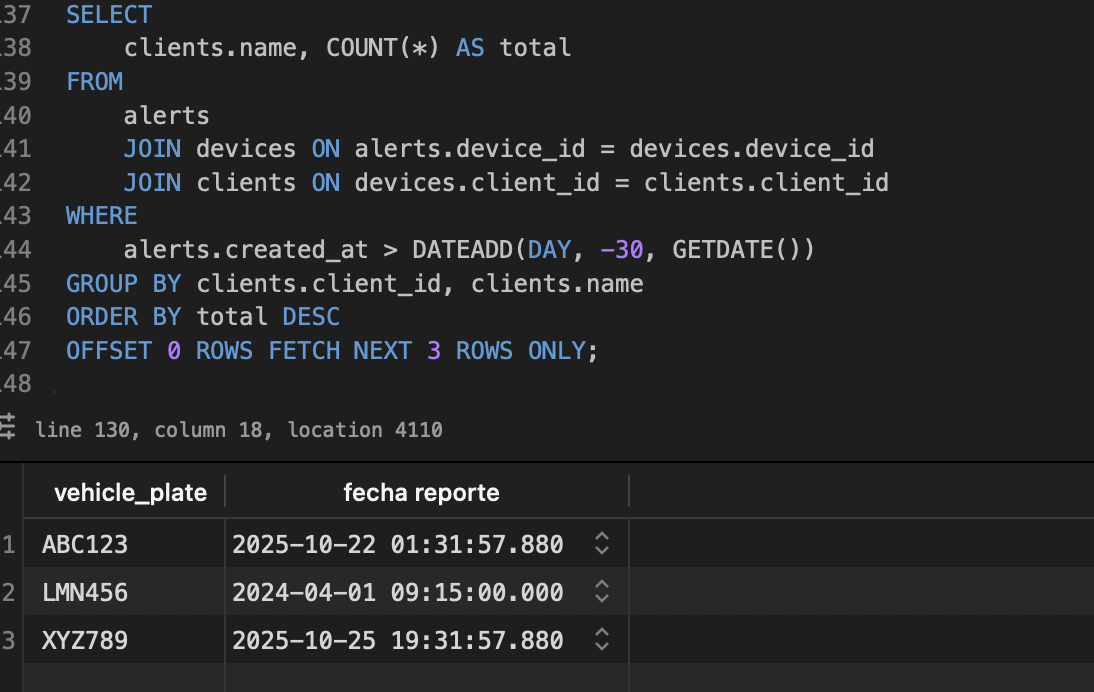
vehículo.

----------------------------------------------

2.7 – Top 3 clientes con más alertas en 30 días

Determinar los tres clientes que más alertas han generado en el último

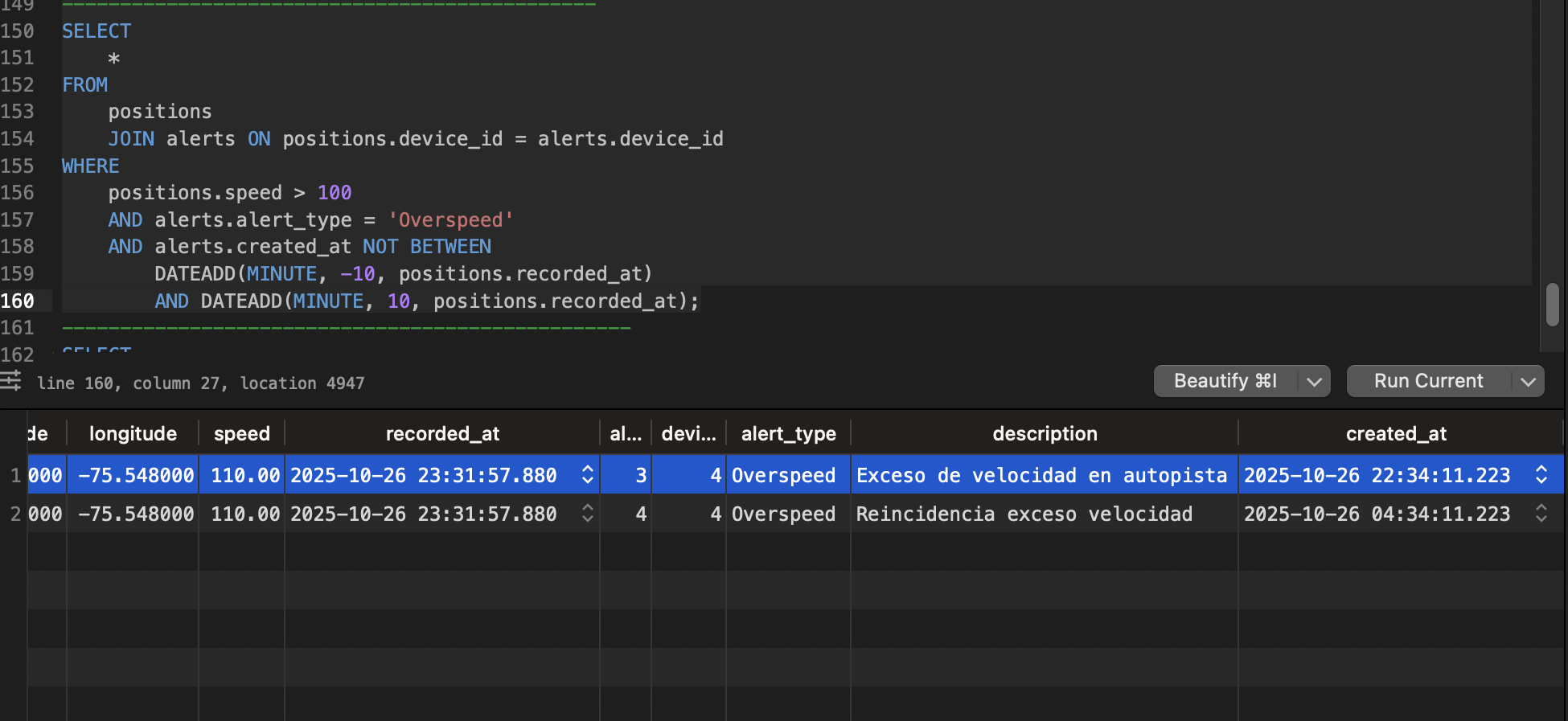
mes.



2.8 Velocidades sin alerta correspondiente

Detectar posiciones con velocidad mayor a 100 km/h que no tengan una

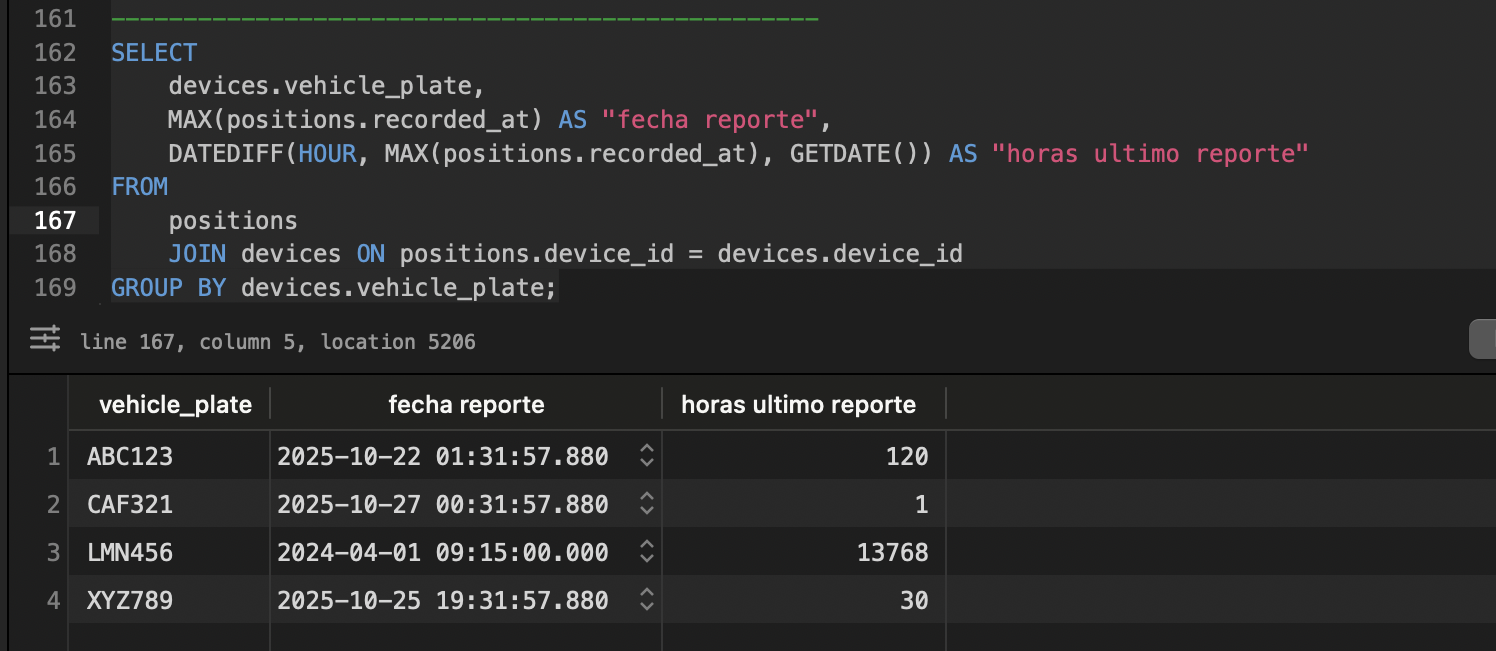
alerta Overspeed ±10 minutos alrededor.



2.9 Tiempo desde el último reporte

Calcular cuántas horas han pasado desde la última posición registrada por

cada vehículo.



2.10 Vehículo más activo por cliente

Para cada cliente, identificar el vehículo que más posiciones ha reportado

en los últimos 30 días.

-----------------------------------------------------------------------

**3. Códigos de error protocolo web**

|  |
| --- |
| **• Un cliente reporta que al intentar acceder a un recurso obtiene un 401. ¿Qué significa y cómo lo resolverías?** |
| En primer lugar es importante revisar si el cliente está ingresando mal sus datos de ingreso o si su cuenta ha sido bloqueada por alguna razón, si requiere algún token de ingreso o si puede ser el caso de que sean software internos y requiera de VPN, tratar de esclarecer cual es la razón porla que no se le está permitiendo el acceso, para poder gestionarlo y solucionar su impedimento |
| **• En producción aparece un 503 intermitente. ¿Qué hipótesis plantearías y qué pasos seguirías para diagnosticar?** |
| Es importante conocer cuales son los escenarios en los que está sucediendo cada vez que se presenta, lo anterior para determinar si hay una casuística específica en donde sucede con mayor frecuencia,y enfocar los recursos de pruebas allí, en ese tipo de escenarios, porque cuando encontremos bien el escenario en el cual se replica, vamos a poder analizar a fondo porque sucede y buscar la mejor manera de corregirlo |
| **• Explique la diferencia práctica entre un 401 Unauthorized y un 403 Forbidden** |
| un 401 es un user que no ha conseguido el ingreso específico al servidor y un 403 es un user que no ha conseguido el acceso a un recurso específico, por ejemplo no tiene permiso para ver el módulo de facturas, un caso hipotético |
| **• ¿En qué se diferencia un 500 Internal Server Error de un 503 Service Unavailable en cuanto a causa y responsabilidad?** |
| un error 500 es un error inesperado o no controlado en donde el servidor de manera abrupta no está disponible,  un error como este puede ser por ejemplo un tema de concurrencia, cuando se hacen peticiones en donde el servidor tiene que procesar algo y de repente tenemos por ejemplo un null pointer y allí se genera un error inesperado generando la interrupción inmediata del servidor, por lo general son errores de código y un 503 es que el servidor no puede atender la solicitud de manera momentánea, temas de devops por ejemplo, porque hubo algo que no pudo procesar, ambos códigos de respuesta hacen referencia a temas con el servidor, |
| **• Cuéntanos un caso en el que hayas enfrentado un 500 en producción. ¿Cómo lo detectaste y qué acciones tomaste?** |
| Teníamos un error de concurrencia en producción, cuando se procesaban los objetos asegurables de un seguro, como tenía que recibir 5000 direcciones distintas al mismo tiempo, el sistema arrojaba un error 500, lo manejamos analizando cuáles eran las direcciones en donde especialmente fallaba, y asi nos dimos cuenta que habian procesos que al hacerse al mismo tiempo causaban la caída del servidor, siempre indagamos mucho en los posibles escenarios, longitudes de las direcciones etc |
| **• Un usuario autenticado intenta acceder a un recurso y recibe un 401. ¿Es correcto o debería ser otro código?** |
| debería ser un código 403, porque el usuario ya está autenticado |
| **• Una API devuelve 200 con un cuerpo vacío. ¿Sería más correcto devolver 204?** |
| Cuando se tiene un cuerpo vacío debe responder un 204, es más adecuado |
| **• Un balanceador devuelve 502. ¿Qué diferencia hay con un 504?** |
| con el 502 el proxy recibió una respuesta mala o inválida del backend, mientras que en el 504 el proxy no recibió ninguna respuesta a tiempo del backend. |
| **• Un recurso fue eliminado permanentemente. ¿Qué código usarías: 404 o 410?** |
| En este caso sería más acorde usar el 410, dado que se tiene certeza de que el recurso existió y fue eliminado, mientras que con el 404, quizás no sabemos qué sucedió con el recurso específicamente, quizás no existió |
| **• Una petición POST con JSON válida falla porque falta un campo obligatorio. ¿Devolverías 400 o 422?** |
| El error 400 puede ser también que la petición está mal fundamentada, podría ser temas de sintaxis, mientras que en el 422 se sugiere que sea cuando un campo falta, pero la sintaxis de la petición es correcta, al ser un json válido se sugiere el 422 |

**4. Conceptos**

| **Tipo de prueba** | **Descripción** | **Implicaciones no ejecutarlas** | **Responsable** |
| --- | --- | --- | --- |
|  |  |  |  |
| Unit test | Para el ISTQB, las pruebas unitarias y las pruebas de componente son equivalentes. Representan el nivel más bajo de la pirámide de Cohn y garantizan la ejecución de pruebas sobre pequeños fragmentos del sistema, las pruebas unitarias garantizan que fragmentos de código o métodos específicos o clases funcionen como se espera,  Su objetivo es verificar que cada unidad realmente cumpla con el resultado para el cual fue creada y que ejecute correctamente su responsabilidad única. | Cuando no se garantiza este nivel que corresponde al primero en el proceso de prueba, la calidad en los niveles superiores se vuelve difícil de asegurar, ya que las pruebas se sobrecargan en etapas posteriores. En realidad, debería concentrarse un mayor volumen de pruebas en este nivel.  Tener una mayor cantidad de pruebas en esta etapa mejora considerablemente la calidad del producto y reduce los costos de tiempo y dinero.  Es fundamental que el equipo de desarrollo tenga plena conciencia de cómo mejorar su trabajo día a día, y parte de ello consiste en mantener una excelente calidad en lo que desarrolla y hacerse cargo de las pruebas unitarias | Dev |
| Component test | Su objetivo principal es validar que un componente se comporte de acuerdo a como está estipulado, un ejemplo de ellos podría ser un microservicio, pero como lo mencionó anteriormente, en el glosario istqb,las pruebas de componente y unitarias son equivalentes | Muy similar a las pruebas unitarias, las pruebas de componente, hacen parte del primer nivel de la pirámide, por lo cual es fundamental que se realicen, para garantizar una mejor calidad en cada uno de los desarrollos, cuando no se ejecutan las pruebas de componentes esto se ve reflejado en la mala calidad con la que se entregan los desarrollos a QA, teniendo reprocesos de tiempo y por ende dinero | Dev |
| Contract test | Como siguiendo una hoja de ruta, precisamente es algo similar a las pruebas de contrato, dado que en ellas se tiene lo que espera el sistema que se integra o comunica con nuestro sistema y deben poder comunicarse sin problema, un ejemplo de ello es cuando se produce una integración con algún servicio de la registraduría, que datos esperan que nosotros les enviemos y viceversa, que está estipulado en el contrato para poder comunicarnos | Cuando estas pruebas no se hacen, lo mas probable es que tengamos muchos problemas con la calidad, dado que si un sistema no es capaz de comunicarse con otro de manera correcta, y con buenos tiempos de respuesta, no podríamos garantizar que las api hagan o funcionen para lo cual están diseñadas, es vital que manejemos los errores de comunicación y excelente sintaxis en nuestras peticiones, las pruebas de integración garantizan que estamos en la capacidad de actualizar, consultar, crear, eliminar datos que viajan en nuestras API o API Externas. | Dev, QA, Arquitectura,devops |
| Load test | Las pruebas de carga, son pruebas específicas que hacen parte del conjunto de pruebas de performance, que busca para cargas esperadas ver las respuestas de las aplicaciones, ver como son de resistentes estas cargas, con cargas se puede entender cantidad de usuarios esperada en la aplicación, como por ejemplo un dia normal de trabajo, que pasa cuando la aplicación se somete a estas cargas, cuales son sus tiempos de respuesta y si el servidor soporta o no dichas transacciones | Cuando un sistema está expuesto a las cargas comunes que podría tener, y dichas pruebas no se realizan o no son tenidas en cuenta, lo más probable es que el sistema se caiga y se generan grandes pérdidas de dinero, por esto es mejor invertir en este tipo de pruebas, siempre siendo preventivos y no reactivos | Dev, QA, Arquitectura,devops |
| Stress test | Como la palabra estrés lo indica, es cuando un sistema es llevado a su punto de quiebre, son pruebas en donde se busca conocer cuáles son las capacidades máximas del sistema, un ejemplo de ello es la simulación de cargas muy pesadas, en donde sabemos que por lo general los sistemas fallarían, esto lo hacemos para saber cual es su punto de ruptura | No conocer hasta dónde puede soportar nuestro sistema nos lleva a un porcentaje de incertidumbre alto, dado que en un evento donde se requiera tener esa confianza plena, no vamos a saber en qué momento nuestra aplicación va fallar en la mitad de algo importante, un ejemplo de ello son los eventos masivos como la venta de boletería etc | Dev, QA, Arquitectura,devops |
| Endurance test | También hacen parte del conjunto de pruebas de performance, estas enfocándose en periodos largos de tiempo, por ejemplo horas o días, que pasa si tenemos a los usuarios por 36 horas seguidas en el sistema, un ejemplo de ellos puede ser un software como el de las clínicas, que siempre tiene servicio 24 horas, que pasa con el comportamiento de este software | Cuando estas pruebas en softwares que son de uso recurrente y constante no se hacen, podríamos tener errores en software que siempre deben estar activos y funcionando de manera perfecta | Dev, QA, Arquitectura,devops |
| Scalability test | Las pruebas de escalabilidad, hacen referencia a cómo se comporta el sistema, cuando se escalan los niveles o volúmenes de datos, por ejemplo, tenemos 100, luego escalamos a 200, y luego seguimos escalando, que podría pasar en cada uno de esos escalamientos. | Cuando estas pruebas no se hacen, cuando la aplicación pasa de tener 100 a 500 usuarios y no sabemos cual es el comportamiento esperado, la aplicación podría inclusive caerse y no tolerar los aumentos que se van produciendo | Dev, QA, Arquitectura,devops |
| Volume test | Estas pruebas que también hacen parte de las pruebas de performance, consisten en ver cómo se comporta por ejemplo la base de datos con un gran volumen de datos, cual es el procedimiento a seguir cuando se tienen estas cargas, temas de rendimiento y saturación de la base de datos y el procesamiento de los datos | Siempre es importante saber cuanto es el volumen de datos que podemos respaldar, y que pasa cuando no es posible,aunque cabe aclarar que hay softwares que son más críticos para este tipo de pruebas por el negocio en el que se enfocan y porque una caída de la base de datos implica dejar de funcionar, todo el negocio como tal, | Dev, QA, Arquitectura,devops |
| Smoke test | El smoke test o llamada prueba de humo, es una prueba básica que se hace para saber si el software que estamos recibiendo posee una estabilidad mínima para iniciar el proceso de pruebas de manera mas rigurosa, son unos casos diseñados con mucho cuidado para garantizar un flujo mínimo de cobertura | Es de vital importancia realizar el diagnostico de cómo se recibe un desarrollo, esto porque si el equipo empieza a hacer pruebas y desde el inicio se observa una gran cantidad de errores y vemos que no hay estabilidad mínima,perderíamos tiempo probando algo que se debe devolver al team de desarrollo para que se mejore la calidad | Dev, QA, Arquitectura,devops |
| Exploratory test | Hacen parte de un conjunto de pruebas basadas en la experiencia, esto hace que el tester recorra la aplicación conociéndola y a su vez realizando las pruebas, tienen la ventaja de que la intuición del tester juega un papel muy importante, pero también si hay un desconocimiento de la plataforma esta práctica podría tener desventajas | Cuando una persona llega al proyecto, llega con una visión fresca y cuando hace exploratory testing quizás ve cosas que los devs por su sesgo de confirmación no verían, u inclusive otros compañeros de qa, las pruebas exploratorias se recomiendan para temas de agilidad por lo que se va probando y se va ejecutando, quizás no son tan documentadas, sugiero que en el día a día, podríamos obtener una combinación de estas pruebas quizás con unas más estructuradas, para sacarle provecho a ambas cosas | tester, users |
| Regression test | Las pruebas de regresión hacen parte de las pruebas asociadas al cambio, son vitales para garantizar que el deploy realizado no dañó cosas que ya estaban estables, o incluso la solución de muchos bugs en conjunto pudo afectar algún módulo que estaba funcionando correctamente | Es vital que siempre se realicen las pruebas de regresión, y que se tengan automatizadas, dado que se convierten en pruebas muy repetitivas, peo es vital saber que no afectamos cosas que en realidad si estaban funcionando de manera correcta. | tester, devs |

**5. Pruebas en Postman - parte 1**

**Consumir el siguiente api de Colombia https://api-colombia.com/**

**Utilizando el api Colombia, obtener la siguiente información:**

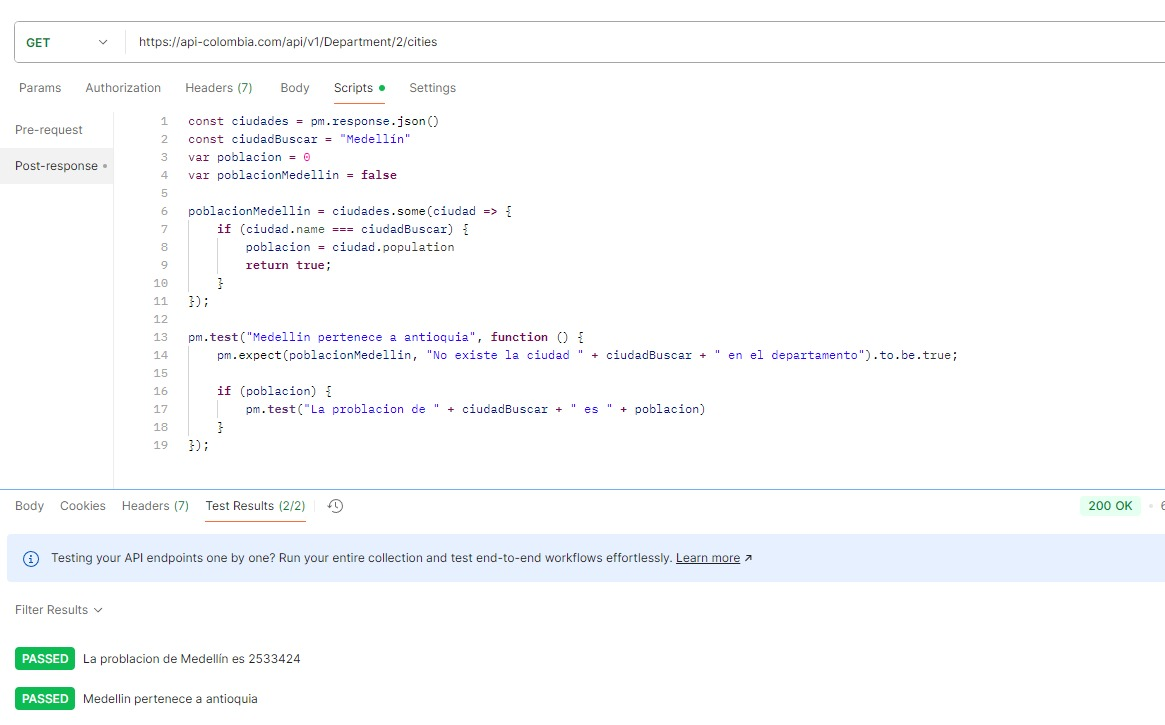
**• ¿Cuál es la capital y cuantos municipios tiene el departamento de**

**Antioquia?**

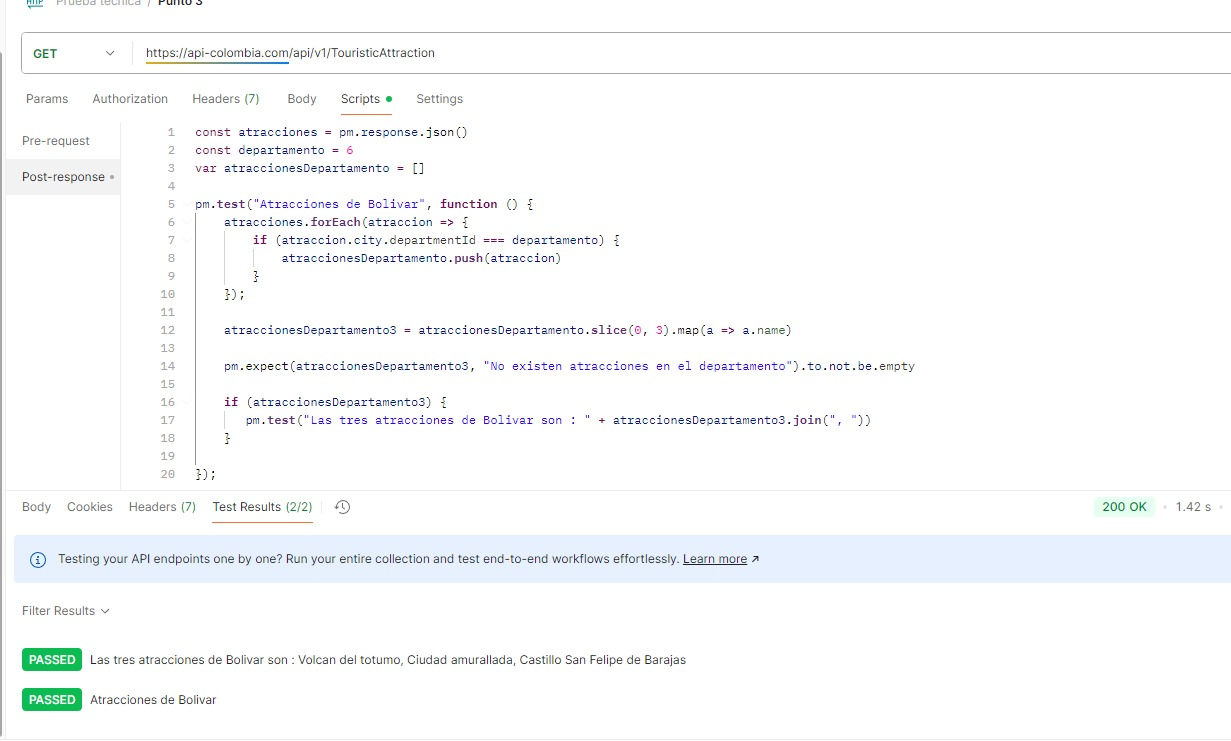
****

**• Verifica si la ciudad de Medellín pertenece al departamento de**

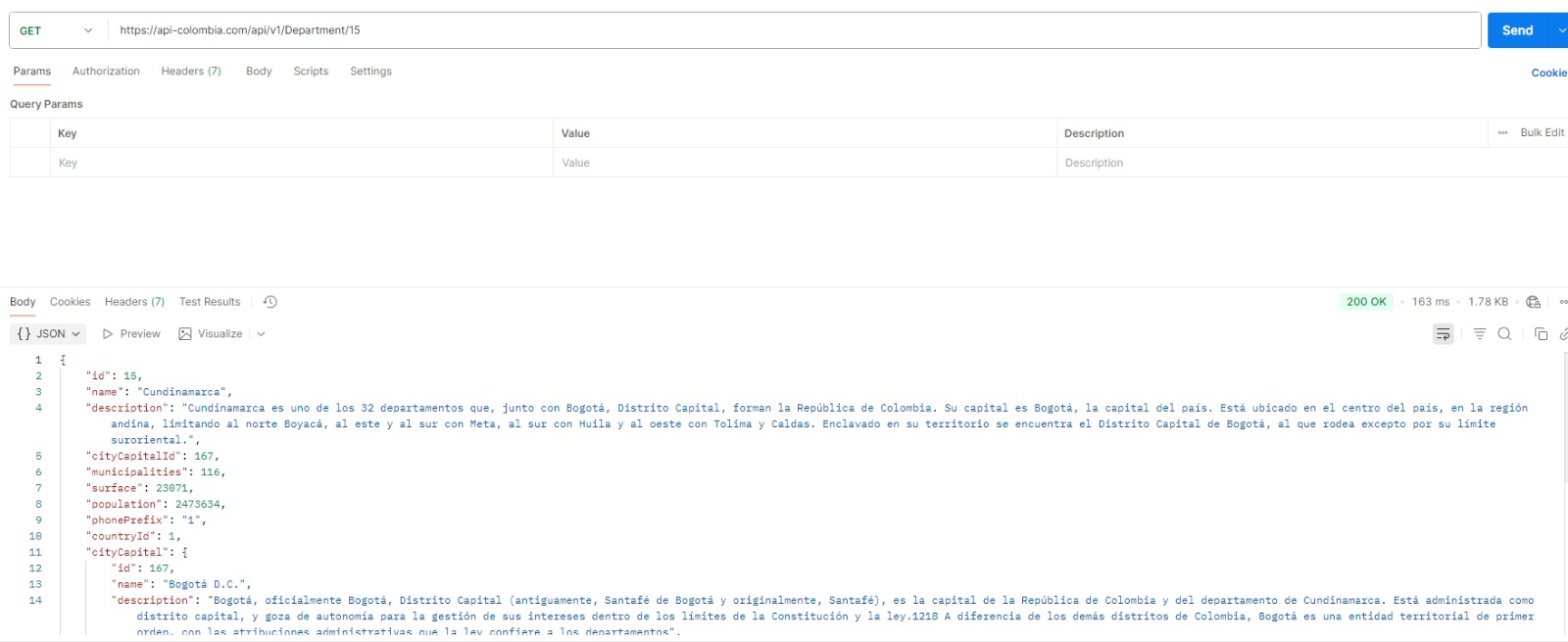
**Antioquia y proporciona su población**

****

**• Lista 3 atracciones turísticas en el departamento de Bolívar**

****

**• Buscar el departamento de Cundinamarca.**

****

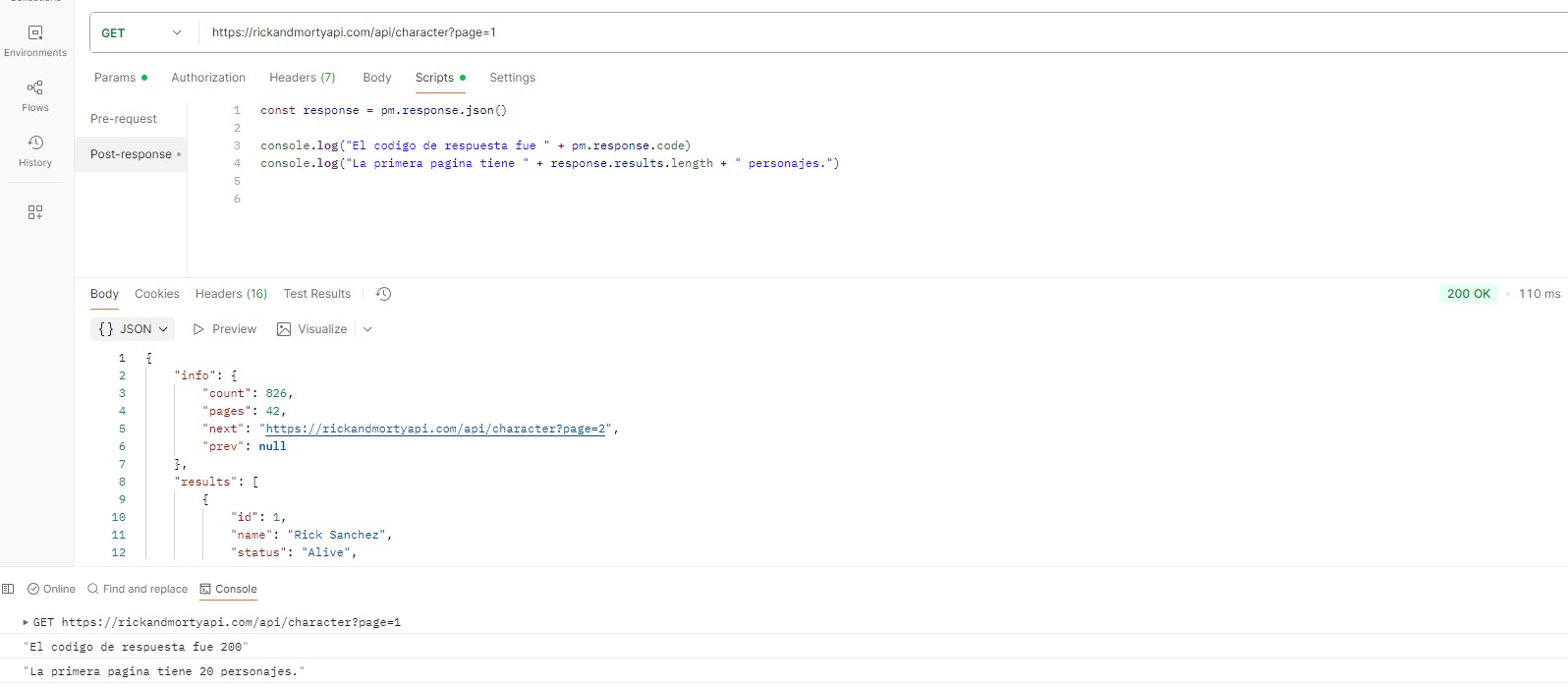
**6. Pruebas en Postman - parte 2**

Consumir el siguiente api de Rick and Morty https://api-

colombia.com/

1. ¿Qué código de estado devuelve? ¿Cuántos personajes aparecen

en la primera página?



El código de estado es un 200 que corresponde a un ok con respecto a la consulta, y debido a que está limitado a 20 resultados por página, la página 1 una muestra los primeros 20 resultados.

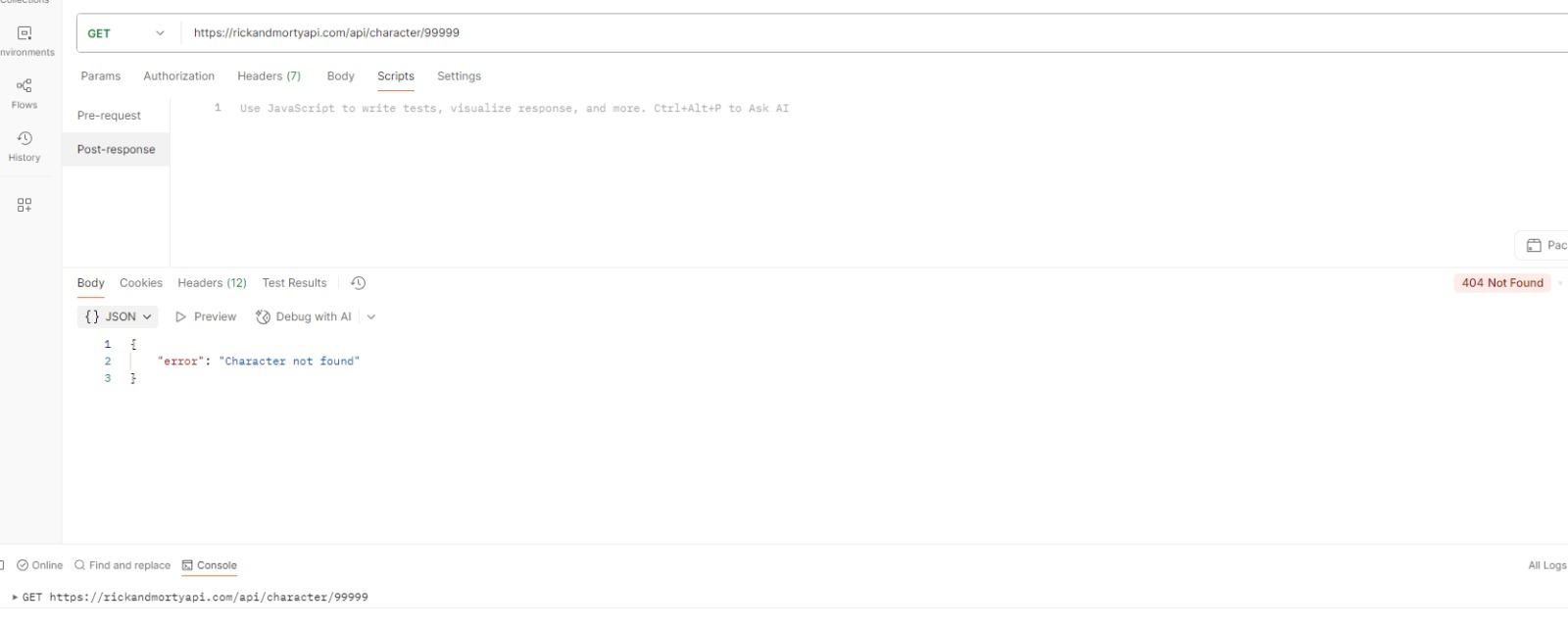
1. ¿Cuál es el nombre, estado y especie del personaje con ID=1?



El nombre del personaje es Rick Sanchez, con especie Human y su estado es Alive

1. ¿Cuál es el código que devuelve y que significa al consultar el

personaje con ID= 99999?

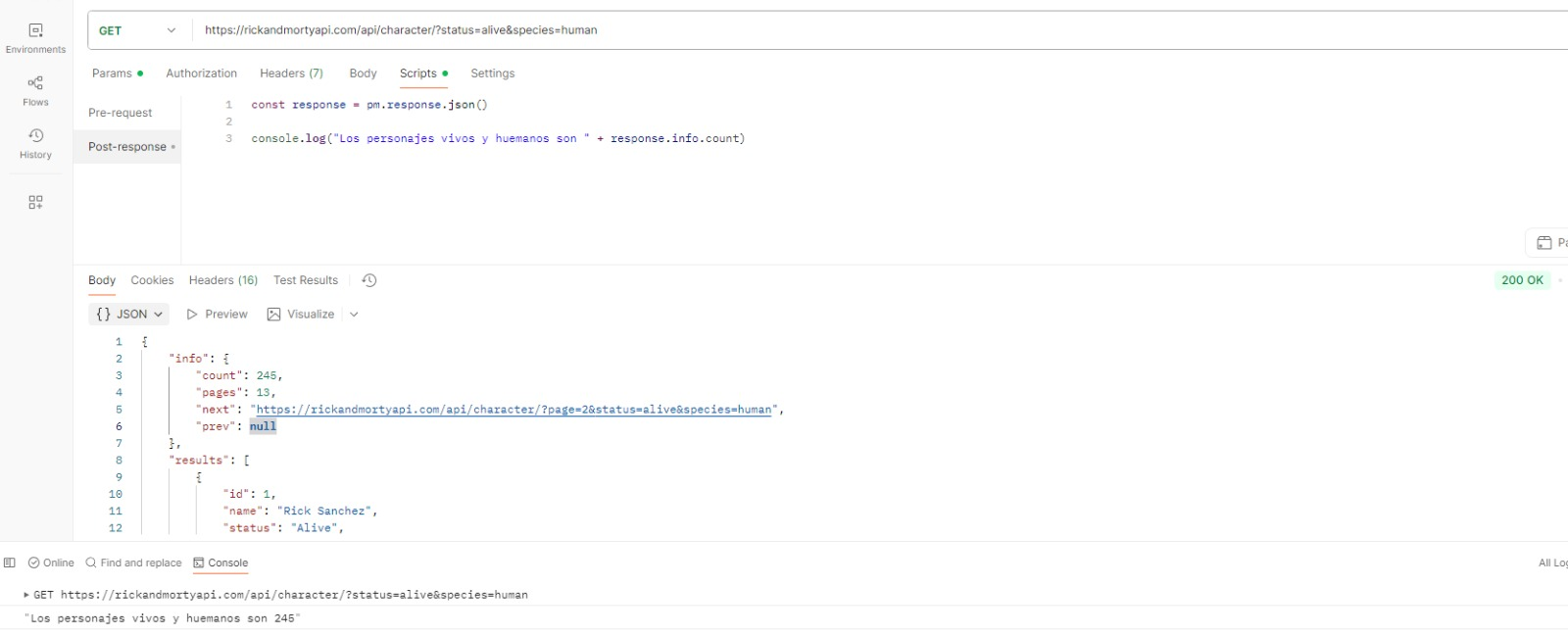


Devuelve un código 404, dado que no existe un registro con este ID

1. ¿Qué diferencia hay entre la página 1 y la página 2?

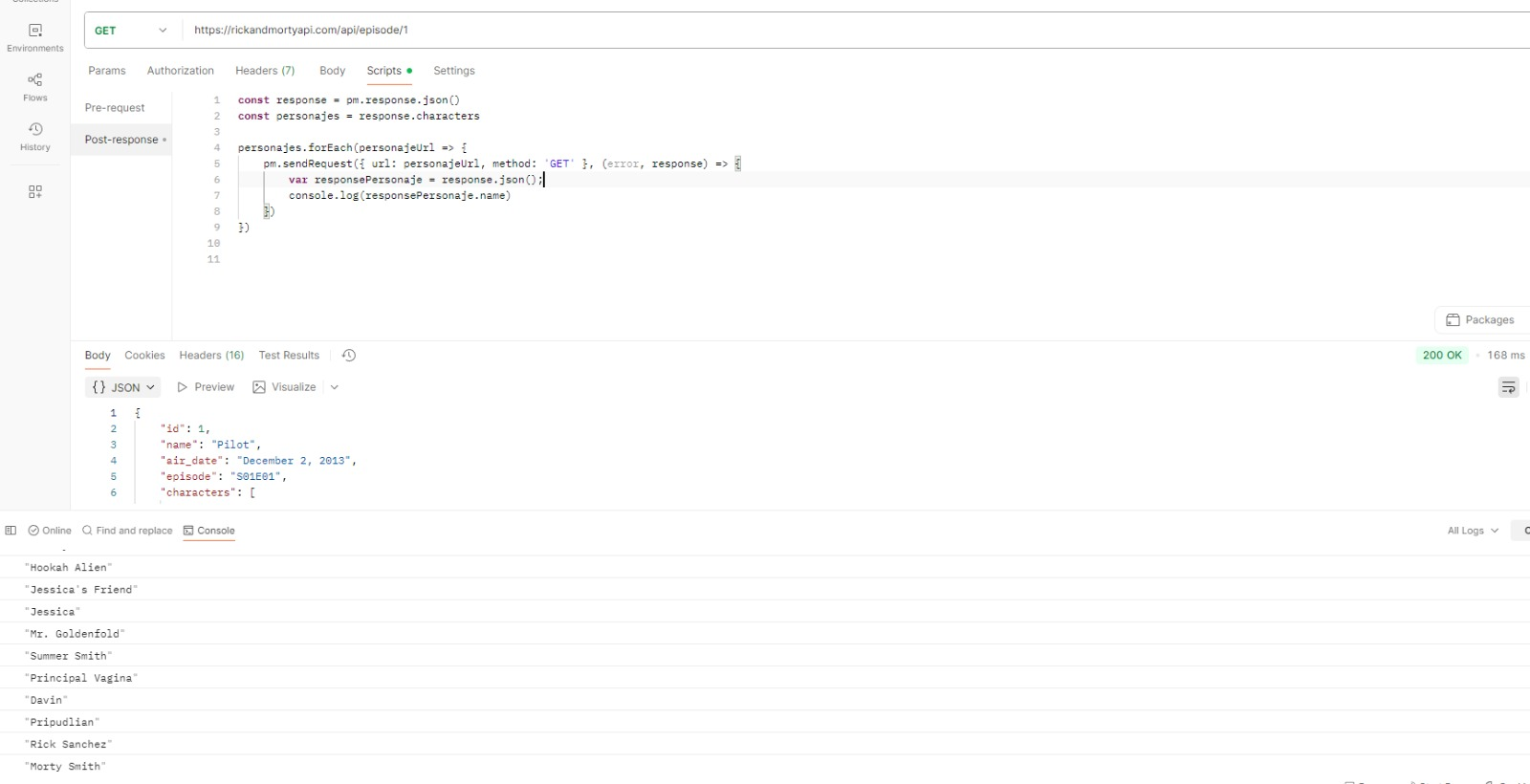
La diferencia entre la página 1 y la 2, los los preview y next que obedece a la página que sigue según el orden, ambas páginas que muestran 20 registros que corresponden a los id de manera ordenada

1. ¿Cuántos personajes humanos vivos devuelve?



Los personajes son 245.

1. ¿Qué personajes aparecen en el primer episodio?



Se muestran los siguientes :

Rick Sanchez

Canklanker Thom

Bepisian

Davin

Glenn

Morty Smith

Summer Smith

Frank Palicky

Jerry Smith

Hookah Alien

Jessica's Friend

Mrs. Sanchez

Mr. Goldenfold

Davin

Jessica

Principal Vagina

Greebybobe

Beth Smith

Pripudlian

1. Configura una prueba que valide que el campo status de Rick

Sanchez sea "Alive".



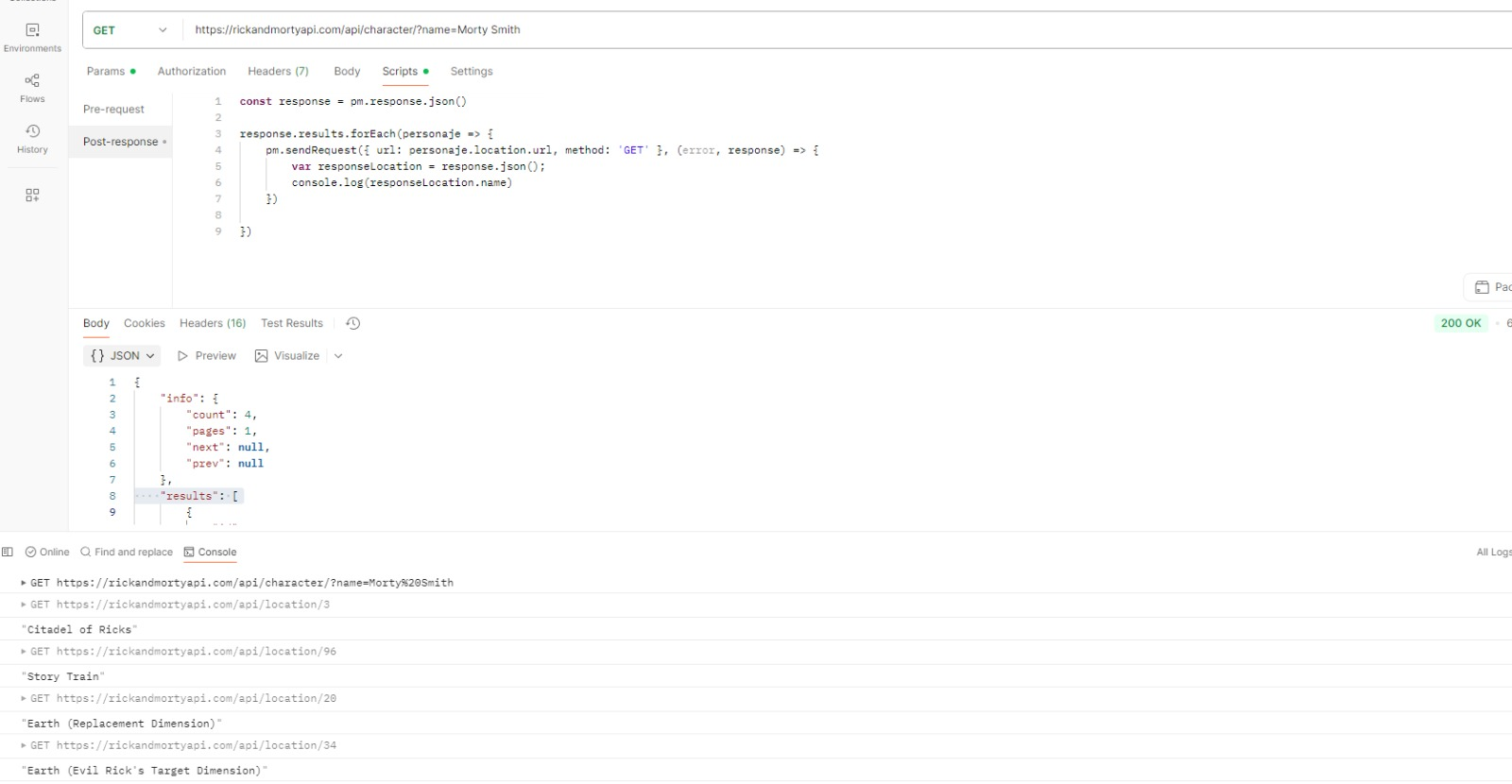
En está prueba se encontró que habian varios registros con el mismo nombre, por tal motivo se verificó que al menos uno de los registros estuviera vivo

• Obtén el personaje Morty Smith.

• De su campo location.url, haz un nuevo request.

Se obtiene el personaje y se encuentran las ubicaciones disponibles

• ¿Cuál es el nombre de la ubicación actual de Morty?

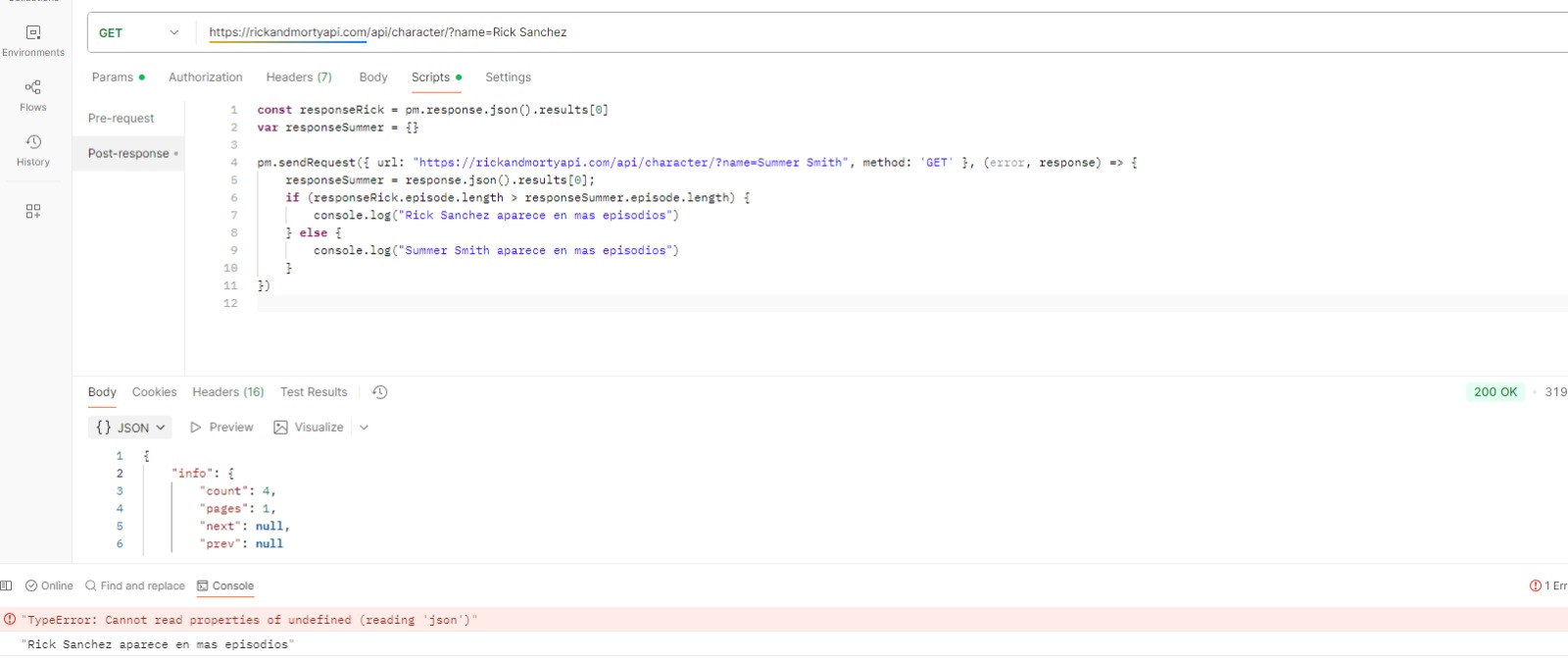


en este test se recopila, el personaje morty smith,se hace un nuevo test buscando la ubicación y también cada uno de los nombres de sus ubicaciones

• Obtén los datos de Rick Sanchez y Summer Smith.

• Pregunta: ¿Quién aparece en más episodios según el campo

episode?



Aparece más Rick Sanchez