**Reto técnico pruebas de servicio con Postman**

**Analista de Pruebas en Formación**

**Ana María Doria Barrera**

**Presentado a:**

**Didier Gerardo Gutiérrez Pérez**

**Choucair S.A.**

**25/05/2022**

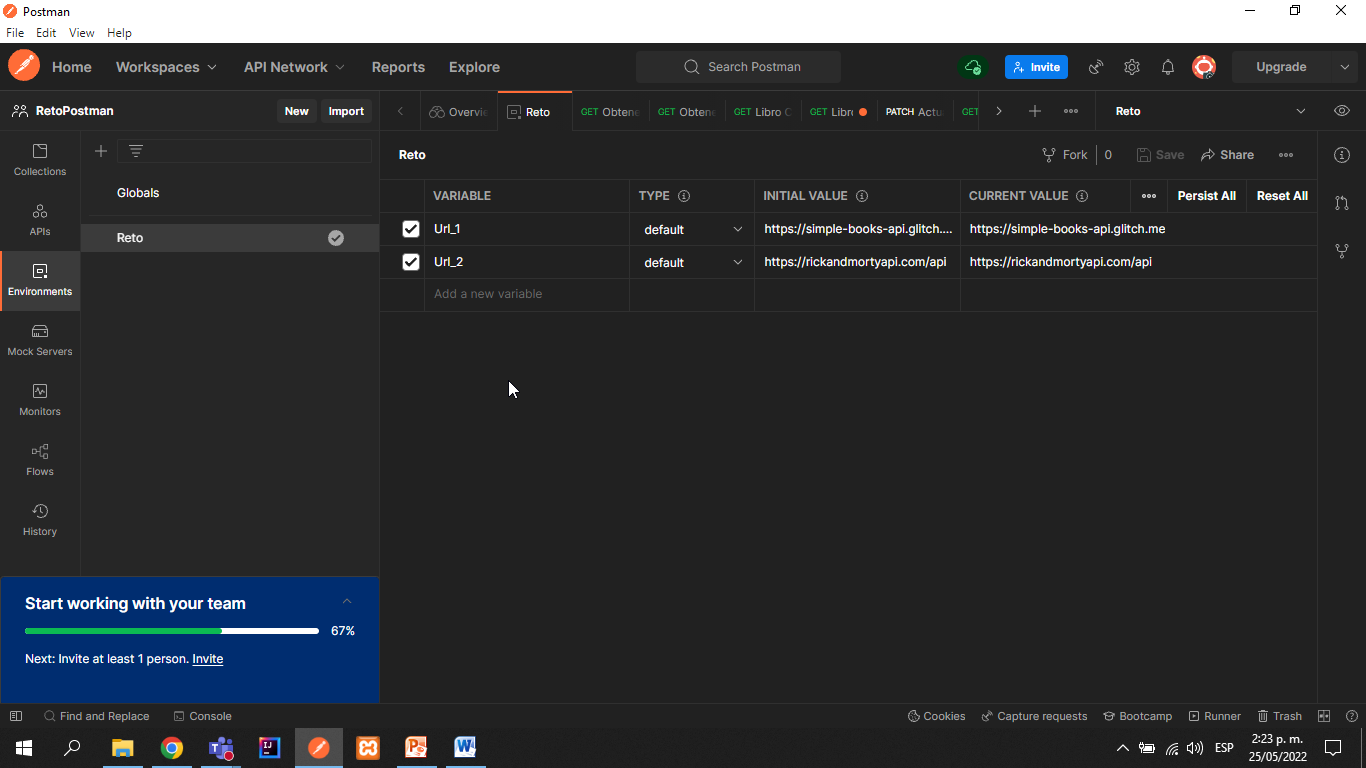
**Primera parte**

Valentin Despa es un desarrollador de software que está trabajando en un proyecto, en el cual hace un servicio Rest, para poder ver libros disponible, así como poder hacer ordenes de libros, así como actualización de estas órdenes, y hacer un borrado de una orden en caso de ser necesario. Por este motivo después de terminado el desarrollo del servicio el desarrollador quiere probar el correcto funcionamiento de este y para esto contrata a la empresa Choucair Testing.

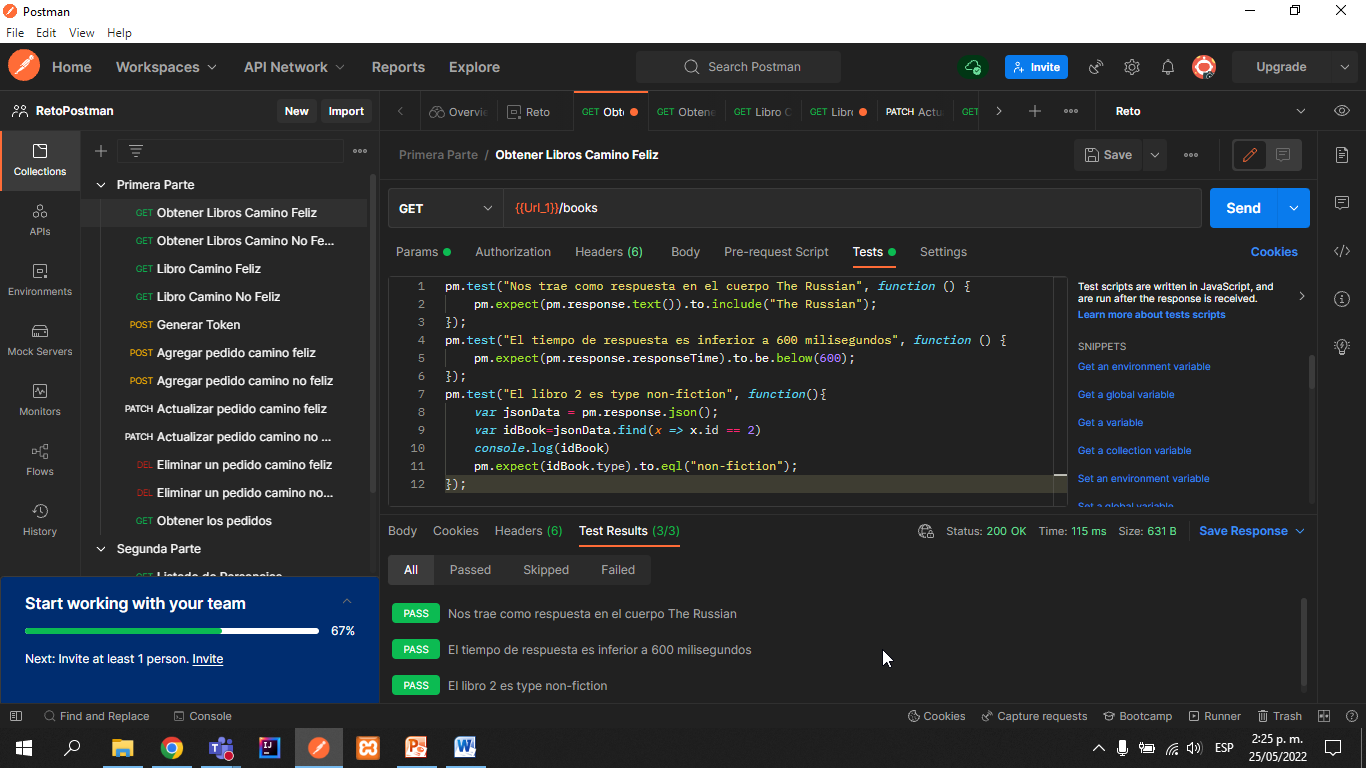
Por este motivo el desarrollador le proporciona a Choucair Testing la documentación de la Api en el siguiente link: <https://github.com/vdespa/introduction-to-postman-course/blob/main/simple-books-api.md>.

Ya después de haber leído la documentación, con el cliente se llega a un acuerdo de que las pruebas realizadas se harán en Postman, y que deben estar organizadas en colecciones que nos permitan ver de mejor manera que endpoint se está probando. Por otra parte lo que el cliente espera que tenga nuestras pruebas en la herramienta de Postman es lo siguiente:

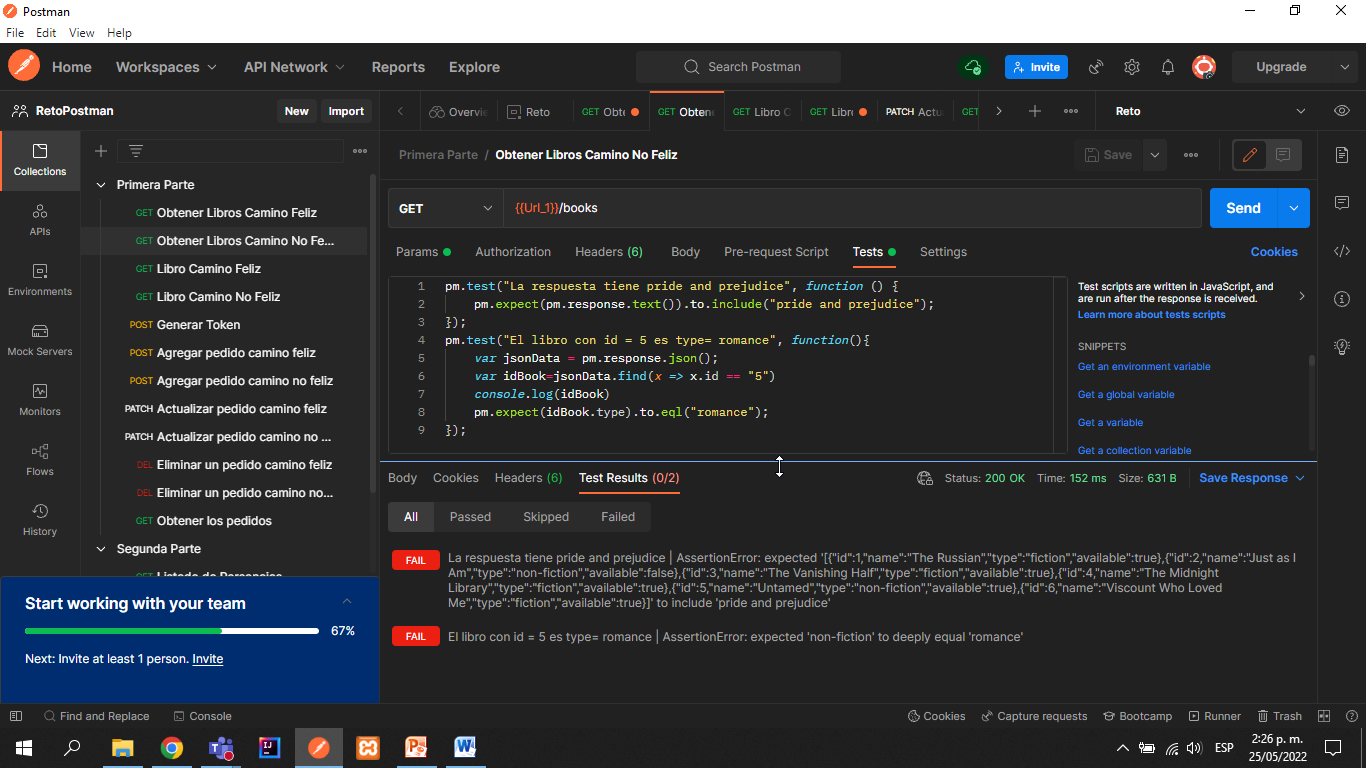
1. Se debe crear un ambiente de pruebas para hacer la ejecución de nuestras pruebas
2. También se hace necesario el uso de variables desde el ambiente creado.
3. Poder hacer un crud con cada uno de los endpoints del servicio (incluye probar caminos felices y caminos no felices)
4. Cada una de las validaciones que se hacen deben estar respaldadas con aserciones que lo justifiquen, no solo validaciones de código de respuesta.

En la siguiente imagen se puede observar la respectiva configuración del Ambiente y creación de variables

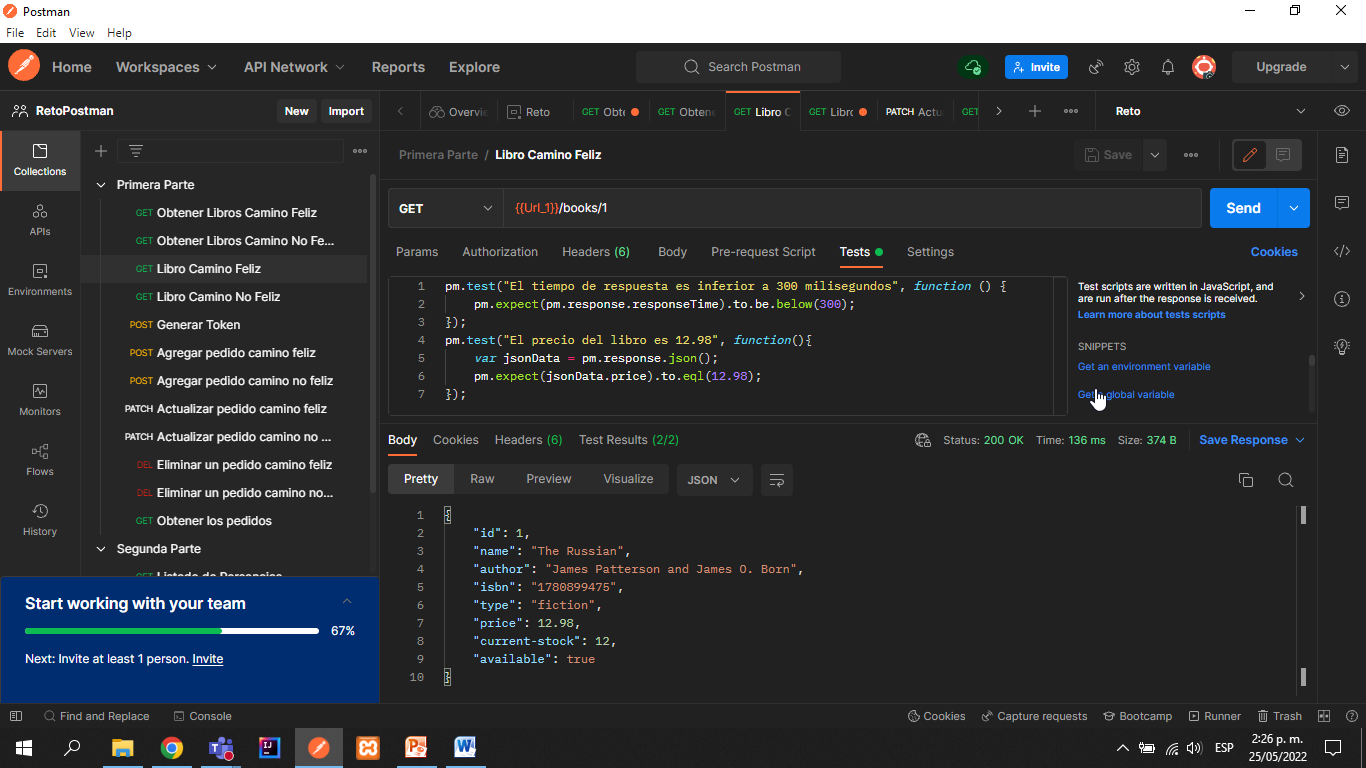
Observamos la petición GET que nos trae los libros (camino feliz), vemos las aserciones que realizamos y el resultado del test



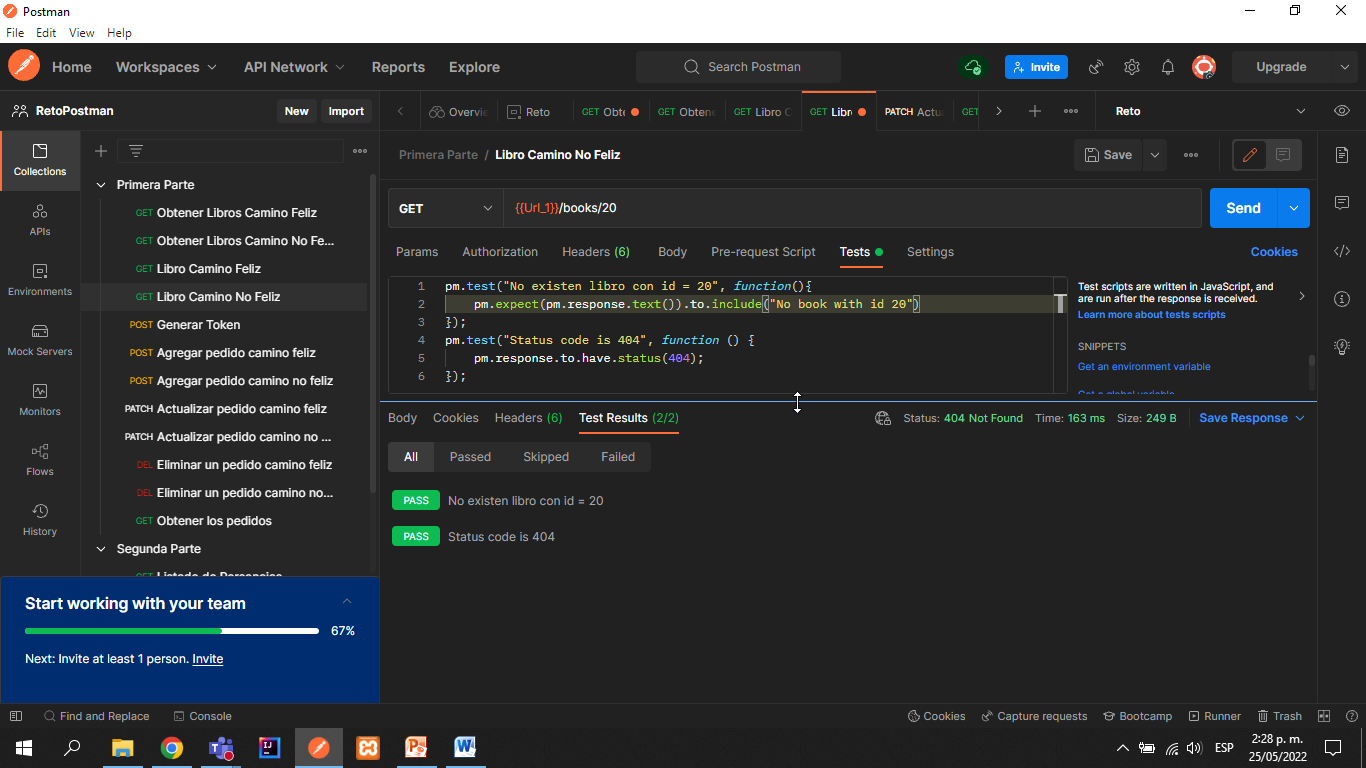
Observamos una lista de los libros (camino no feliz), las aserciones agregadas para confirmar lo consultado y el resultado del test.



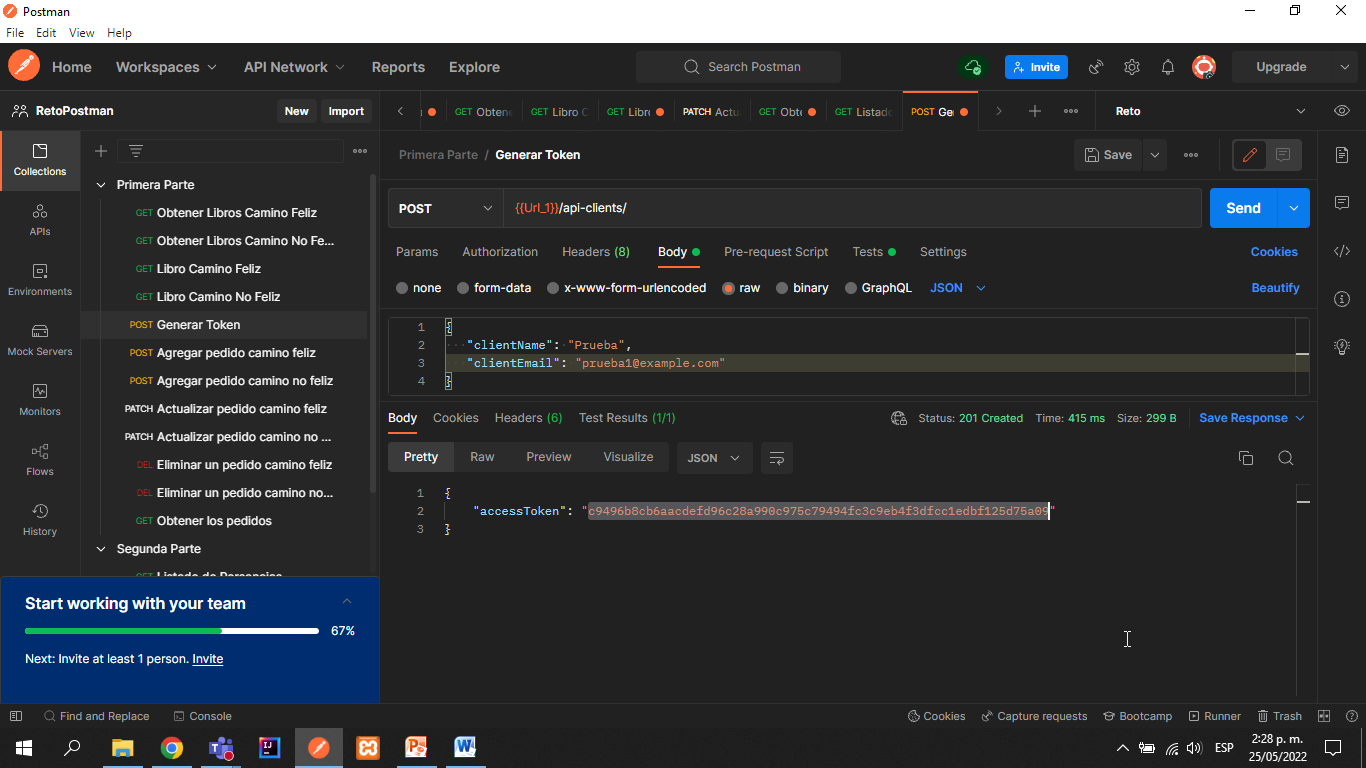
Obtenemos un libro con un Id especifico (camino feliz) y se confirma por medio de las aserciones si lo encontró, lo cual lo podemos observar en los resultados del test.



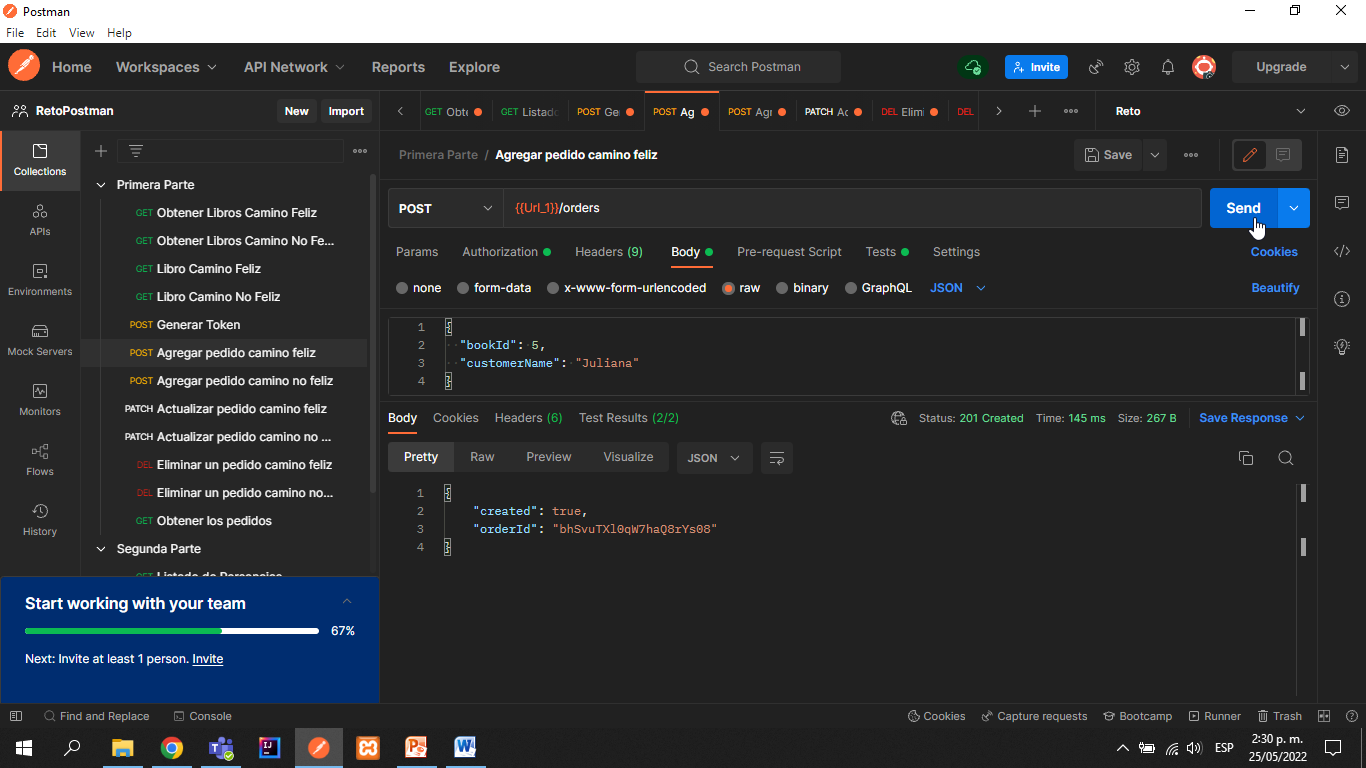
Obtenemos un libro con un id especifico (camino no feliz), observamos que se genera un error debido a que no existe este libro, lo cual confirmamos con las aserciones añadidas en el test.



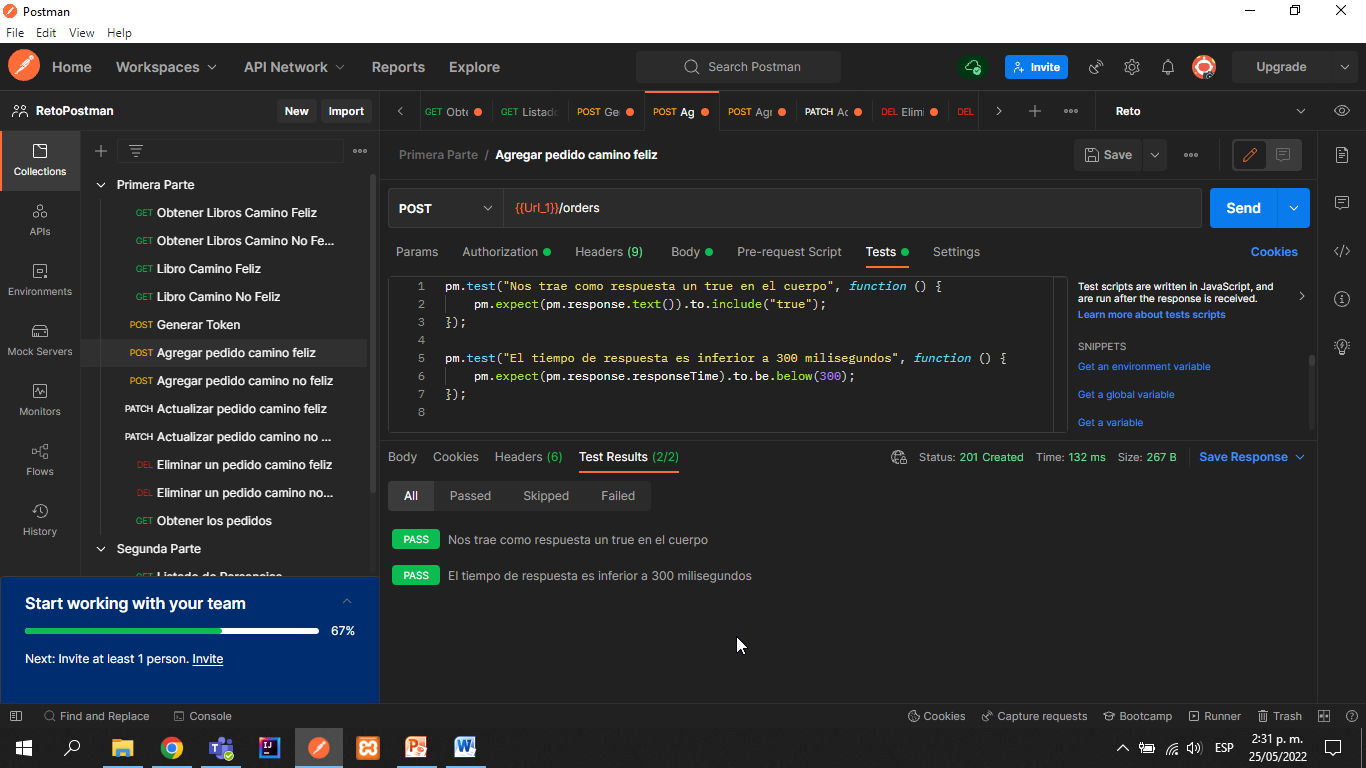
Se genera un token para poder realizar las demás peticiones ya que estas requieren Autenticación para poderlas llevar acabo.



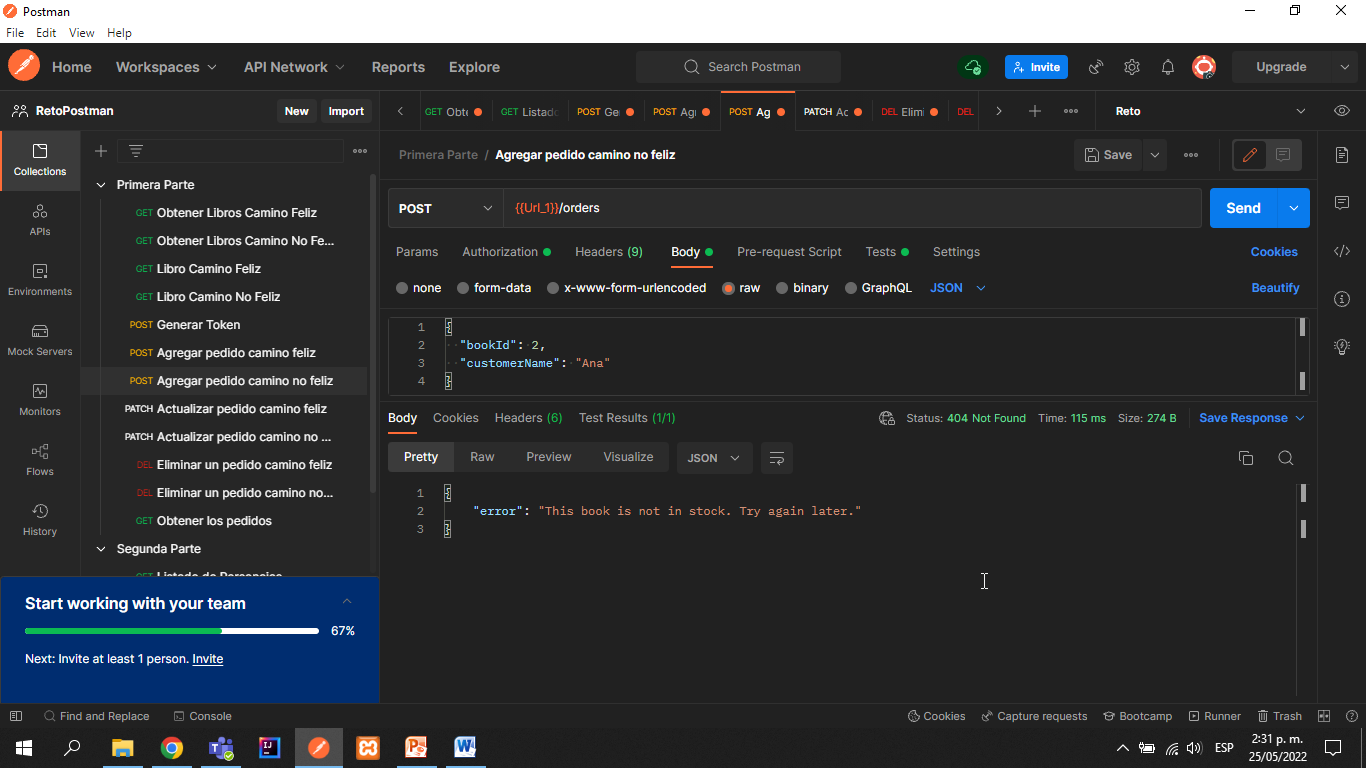
Se agrega un pedido utilizando el método POST (Camino feliz), en el cual mediante el Body anexamos los campos que deseamos agregar en formato JSON y en el Body podemos observar la respuesta a esta solicitud.



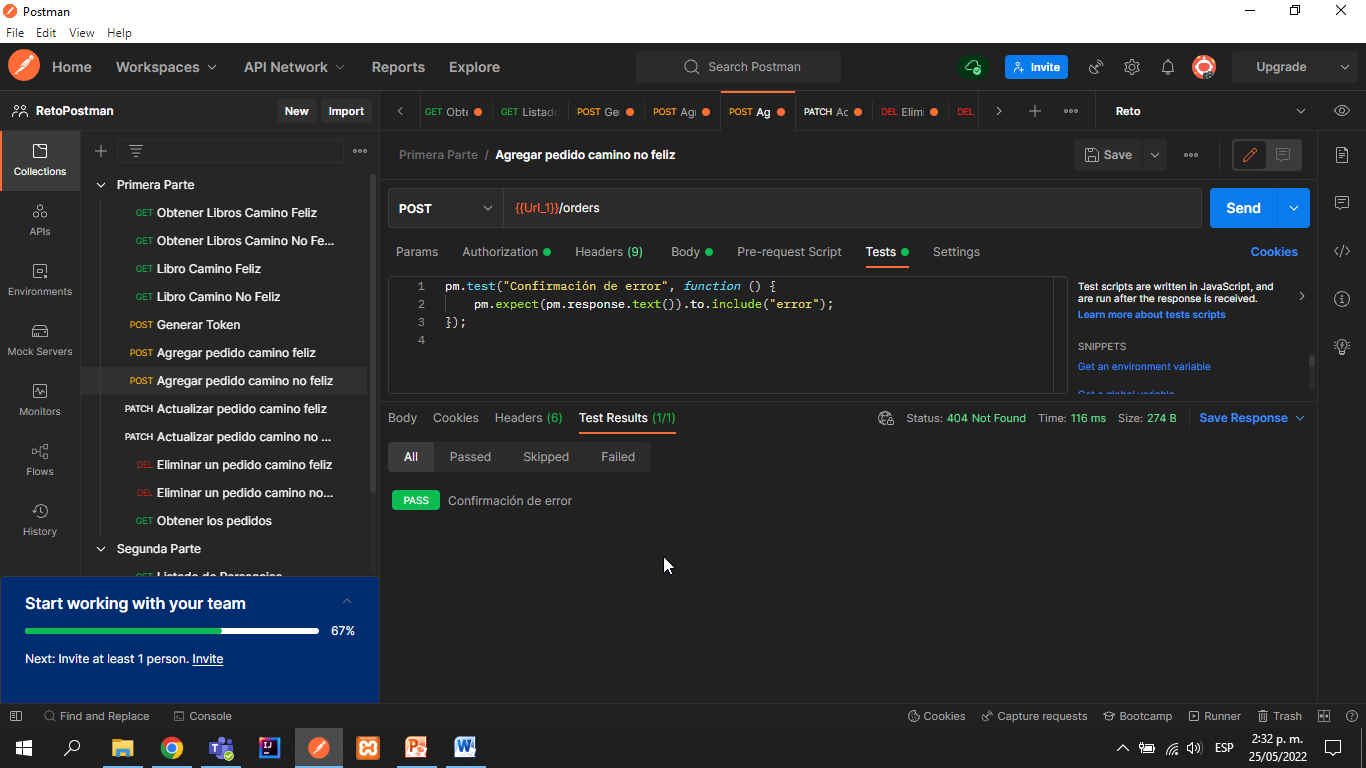
Podemos observar las aserciones añadidas para confirmar que si se agregó de forma correcta verificando la respuesta.



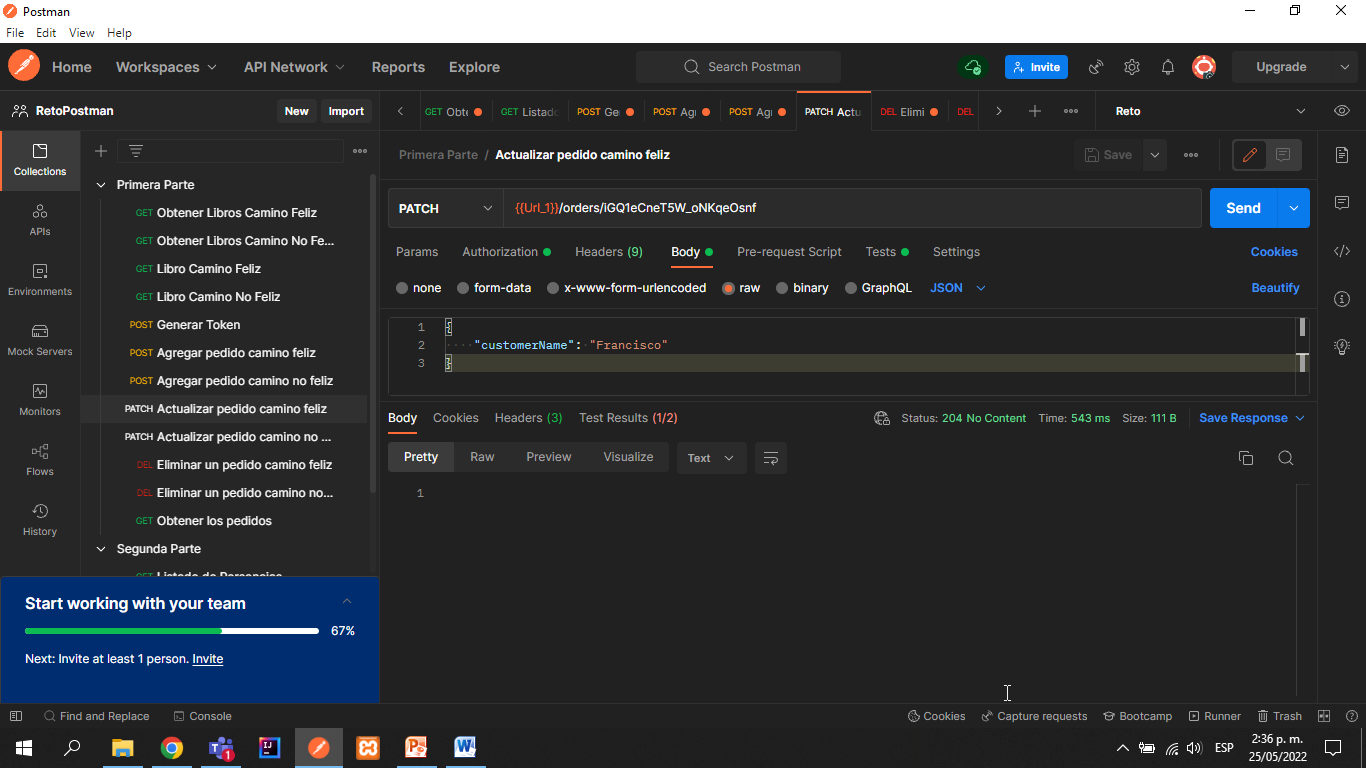
Se agrega un pedido (camino no feliz), pero nos muestra como respuesta un error, que nos notifica que no se pudo hacer el pedido ya que el libro no está en stock.



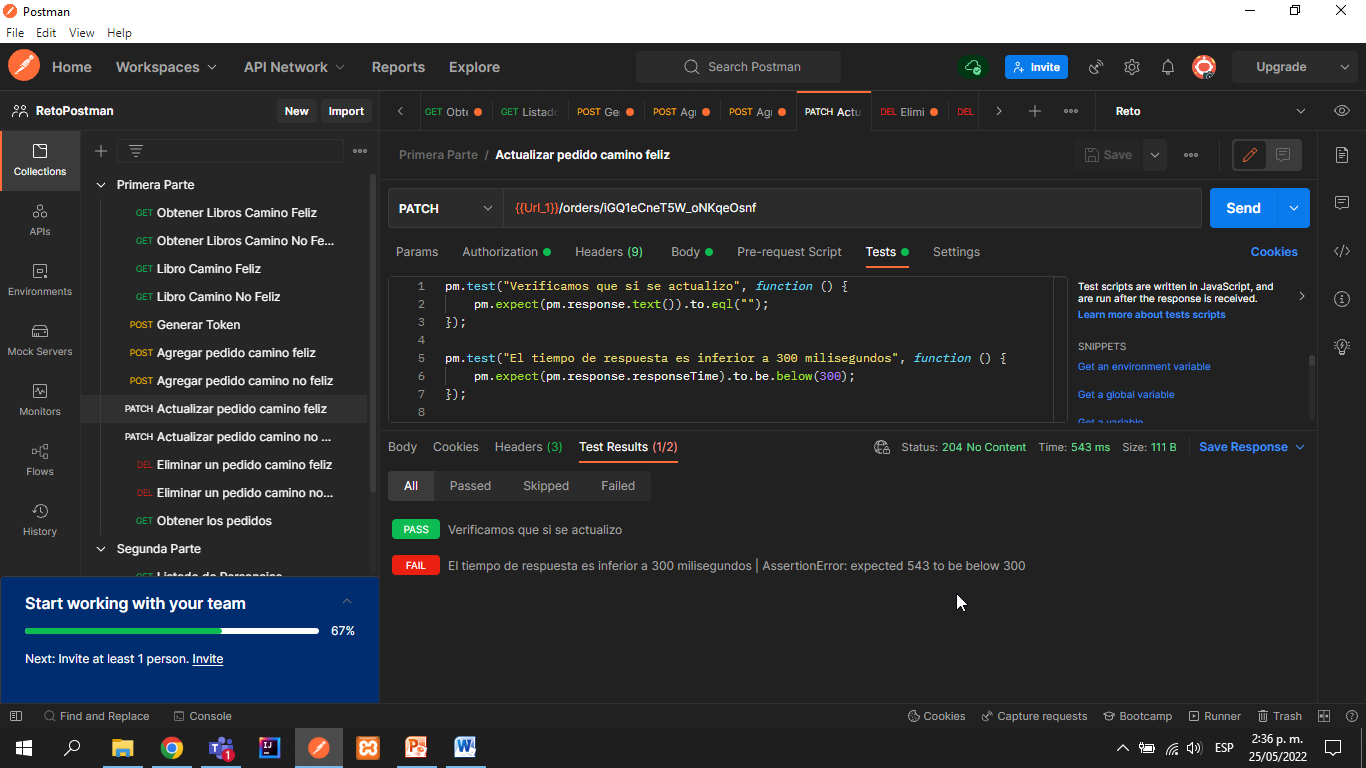
Confirmamos la respuesta por medio de un código de respuesta en este caso 404 que nos indica que no existe este libro.



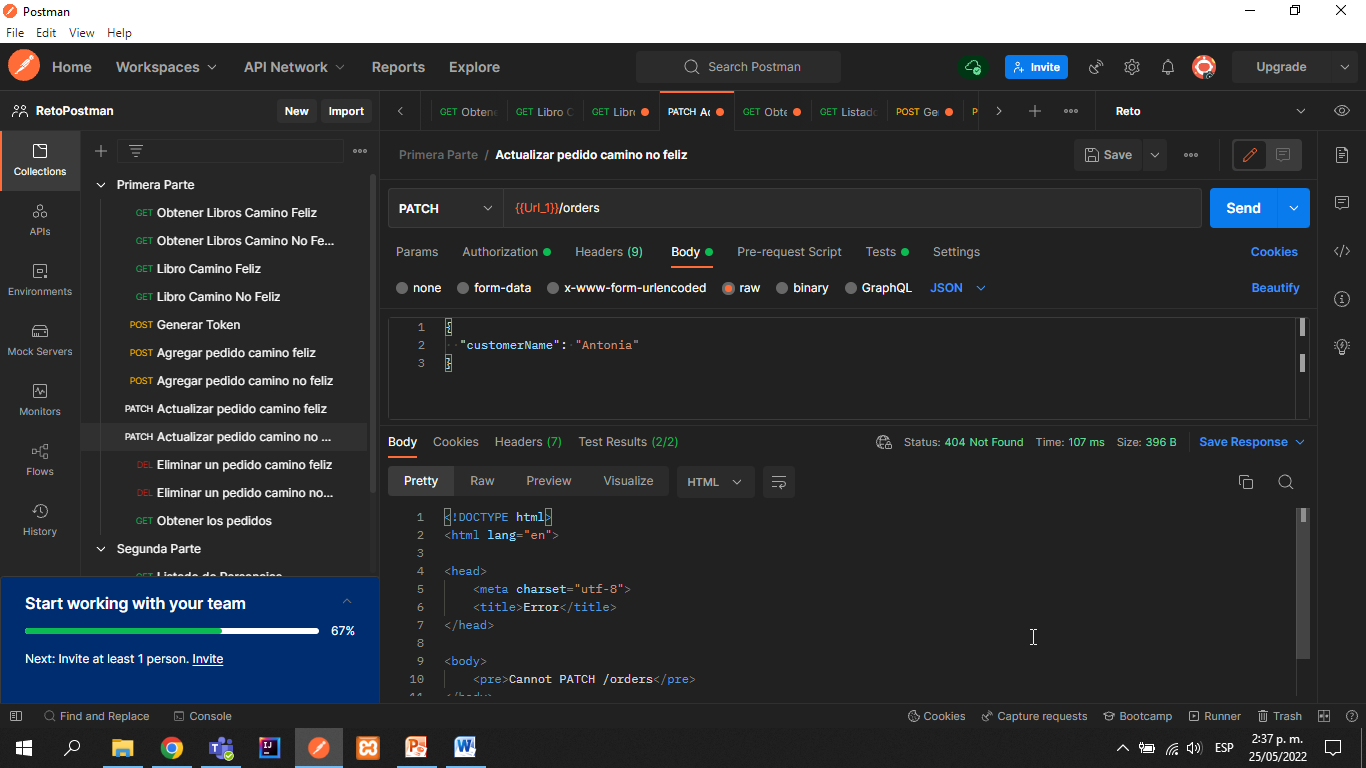
Mediante el método PATCH realizamos una actualización de un pedido (camino feliz) agregando el id correspondiente al pedido a actualizar, anexar el campo que se va a actualizar en formato JSON en el Body y nos envía de respuesta un código 204 el cual nos confirma que es exitoso y no retorna nada.



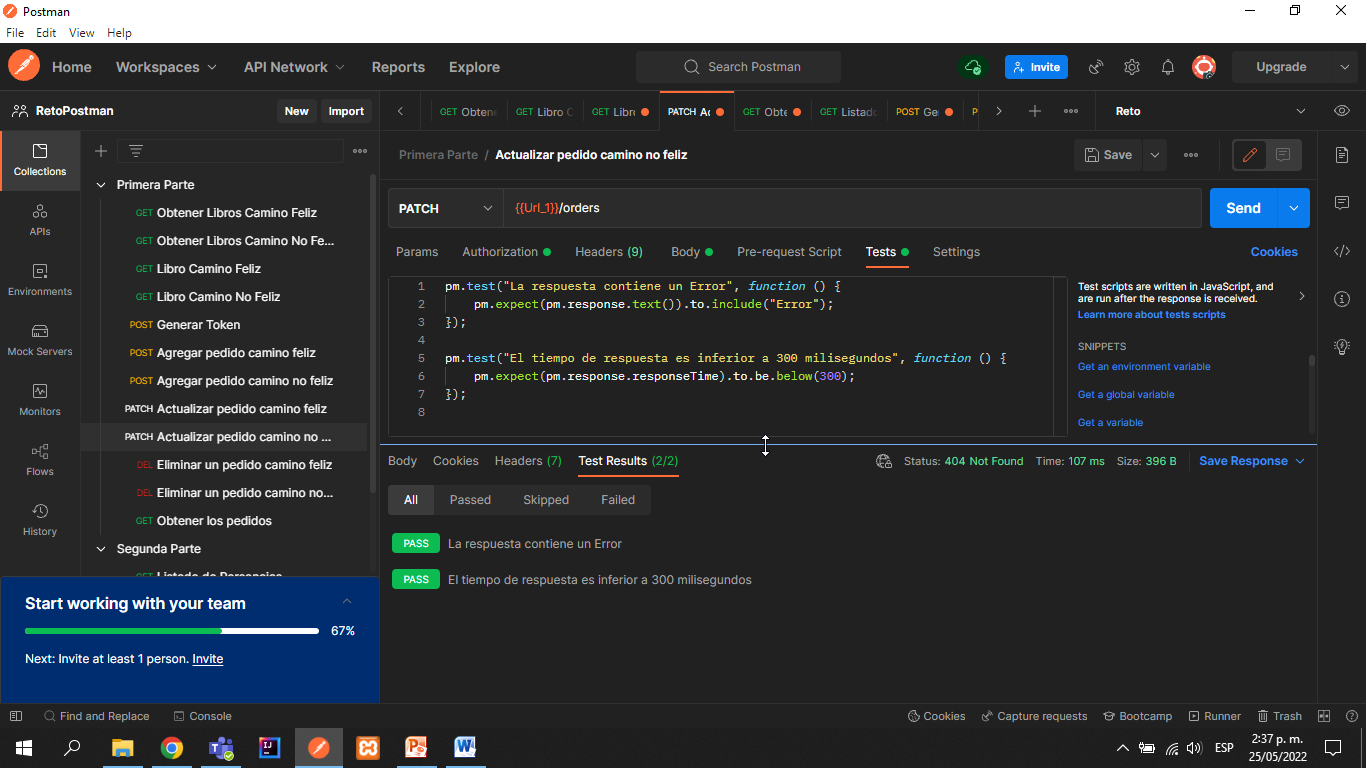
En el test se agregan las aserciones para verificar que se realizó la actualización en este caso que no retornara nada y la otra aserción añadida nos verifica el tiempo de respuesta.



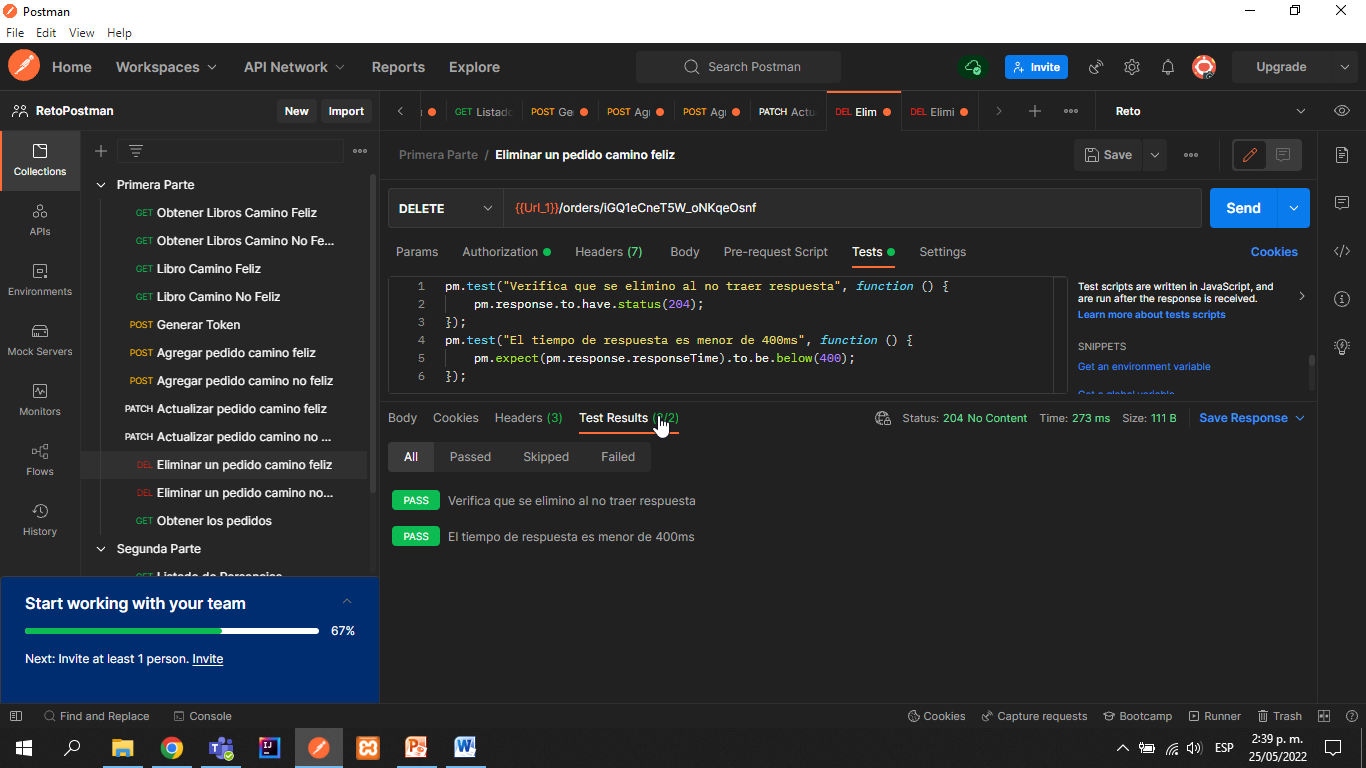
Actualizar un pedido (camino no feliz), nos muestra un error ya que no se realizó la actualización, puesto que no se le añadió el id correspondiente del pedido el cual se le realizaría una actualización.



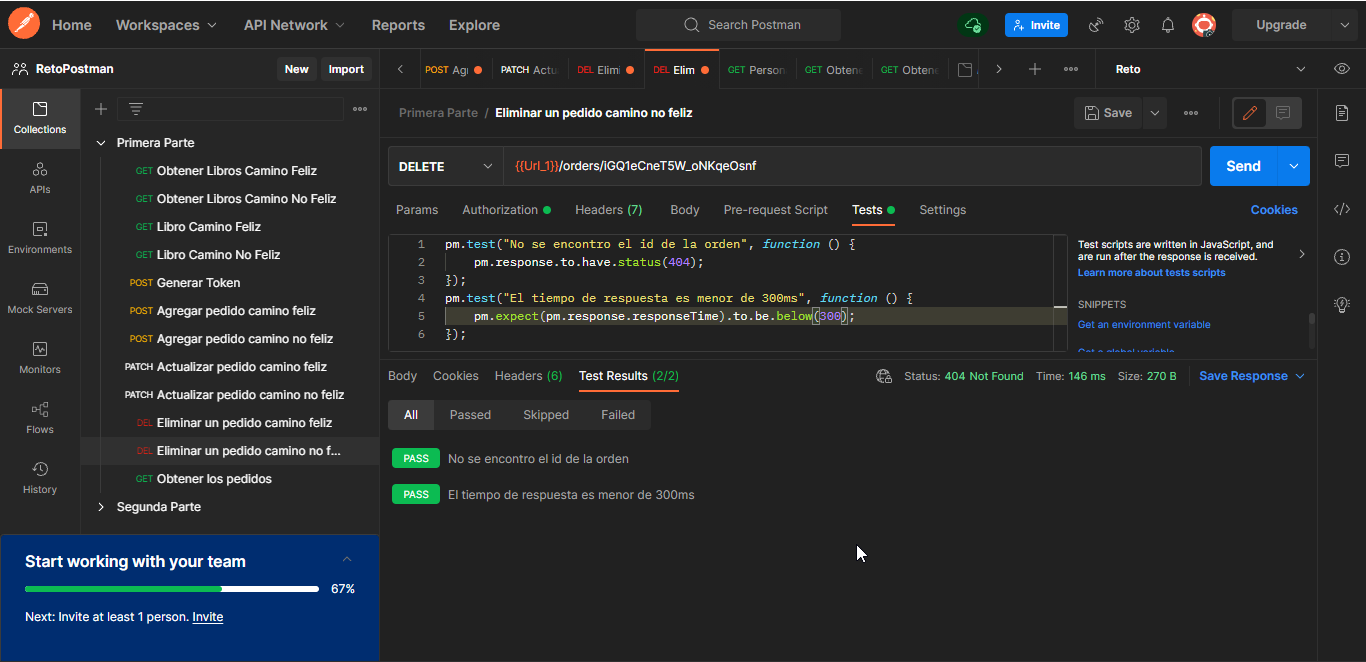
Se verificó que no se realizó la actualización por medio de la aserción añadida en el test.



Por medio del método DELETE se eliminó un pedido (camino feliz) se anexo el id correspondiente del pedido que se desea eliminar y se agregaron aserciones para verificar que si se realizó lo solicitado, mediante el código de respuesta 204.



Se elimina un pedido que ya no existe (camino no feliz), se agregaron aserciones por medio de las cuales se puede verificar si ocurre un error como en este caso, que no se puede eliminar ya que no encuentra ese id respectivo y nos envía un código 404.

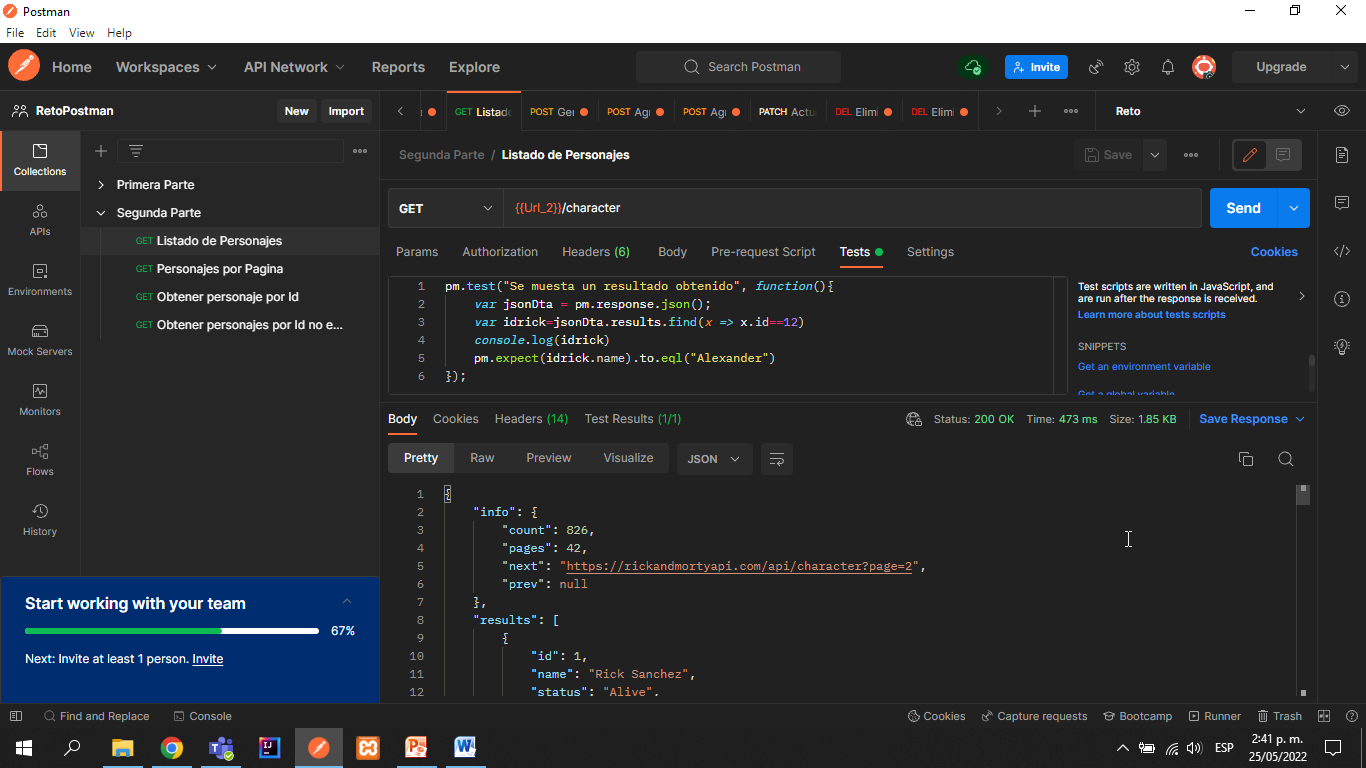


**Segunda parte**

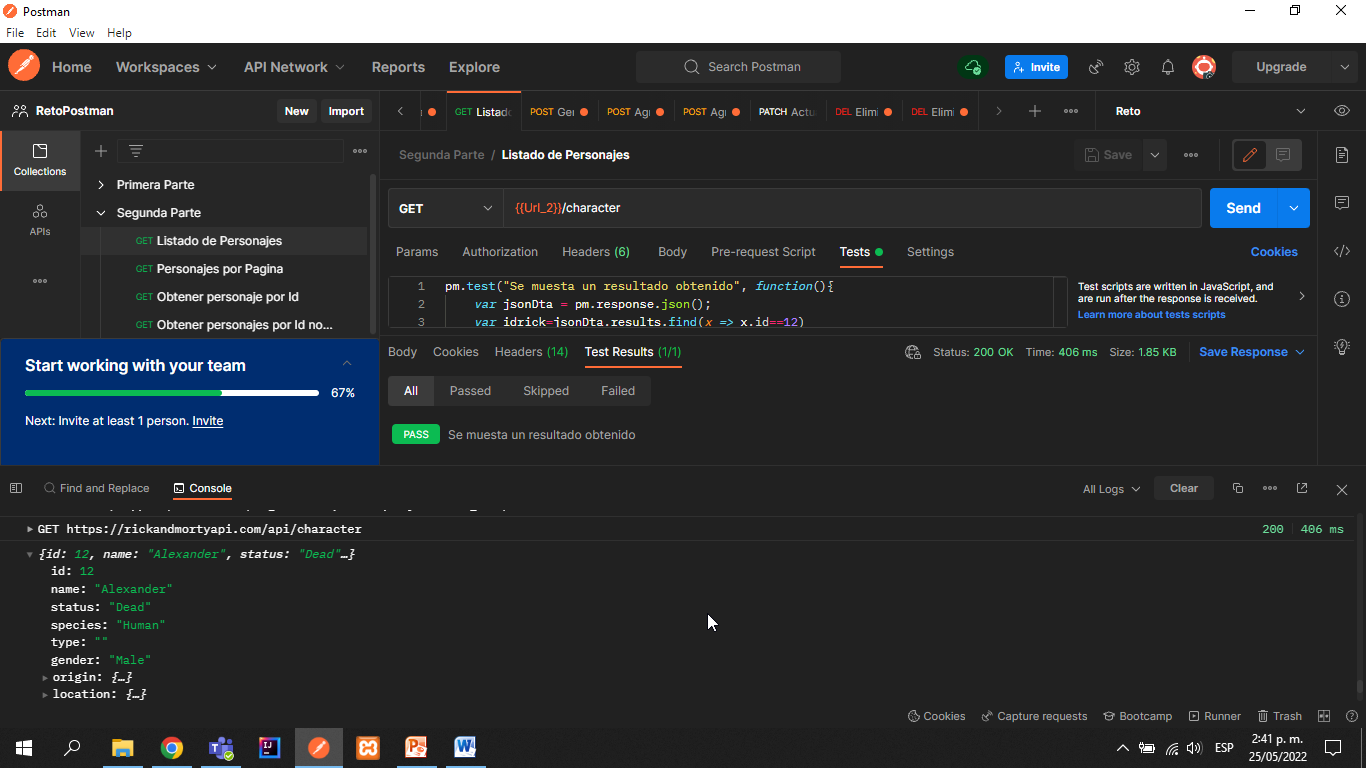
Para esta segunda vamos a trabajar sobre la siguiente Api <https://rickandmortyapi.com/documentation> , la cual es muy usada para hacer pruebas de servicio, para este ejercicio se plantea hacer solicitudes sobre el endpoint de “Character” y se espera puedan hacer lo siguiente con este endpoint:

1. Hacer un listado de todos los personajes, y hacer un test con la información de un personaje que aparezca en esa respuesta (Tener en cuenta que deben hacer una búsqueda entre los diferentes objetos que hay embebidos en la respuesta)
2. Hacer un listado de personajes por página (en la documentación se explica cómo hacer paginación), y hacer un test por aparición de personaje en esta página.
3. Traer un personaje por id, y hacer un test correspondiente.
4. Al momento de buscar múltiples Id, ingresando id’s que no existen nos da una respuesta, la cual es un array vacío, hacer un test en donde se valide que se tiene una respuesta vacía.

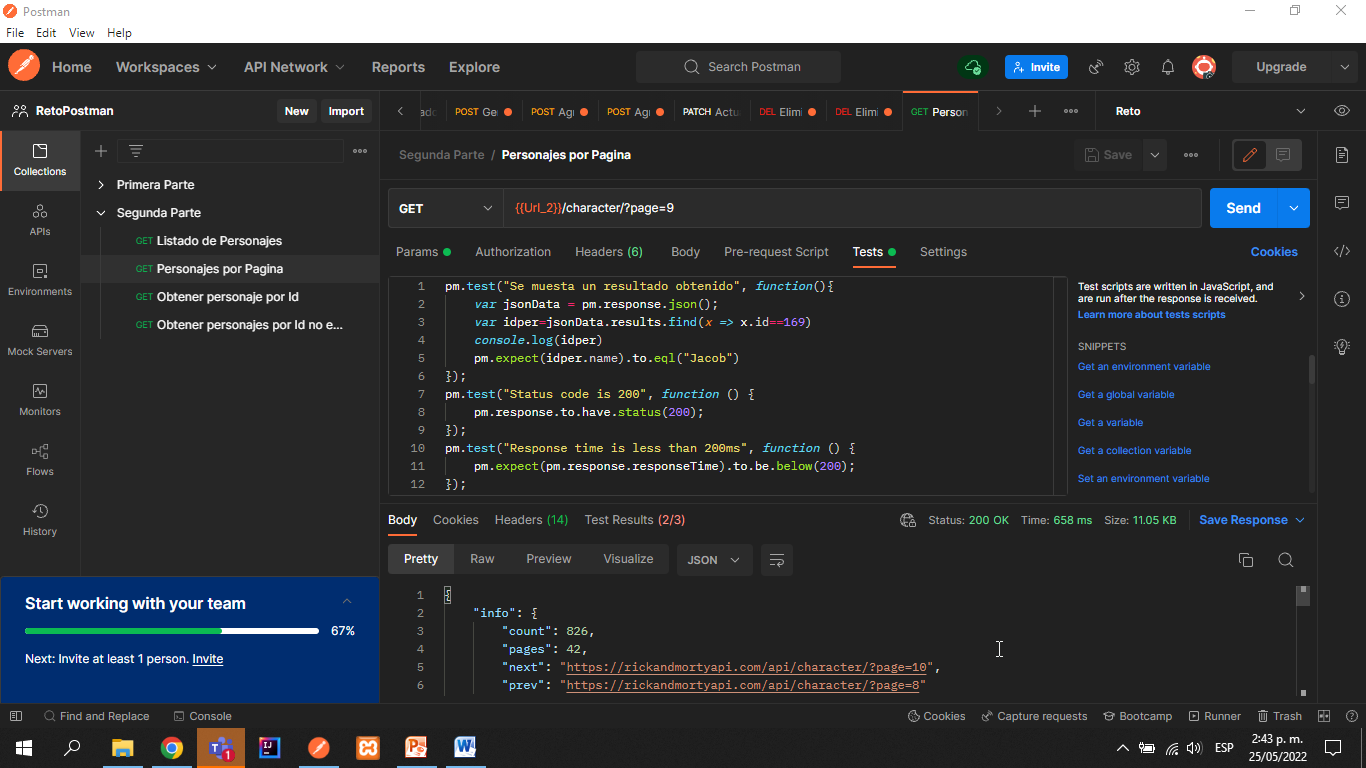
Se realizó una petición GET listar los personajes y se añadió en el test aserciones para verificar un personaje.



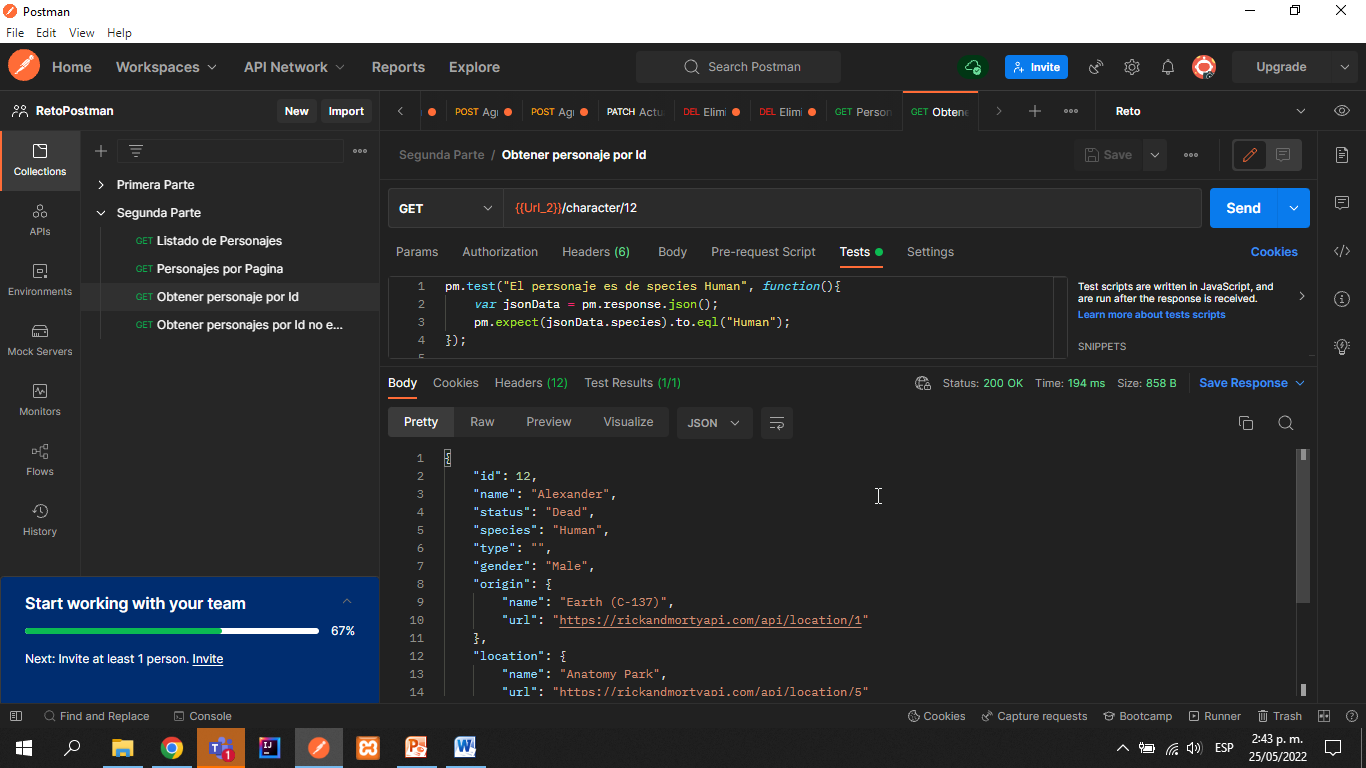
Se muestra en la consola la información del personaje seleccionado por medio del id.



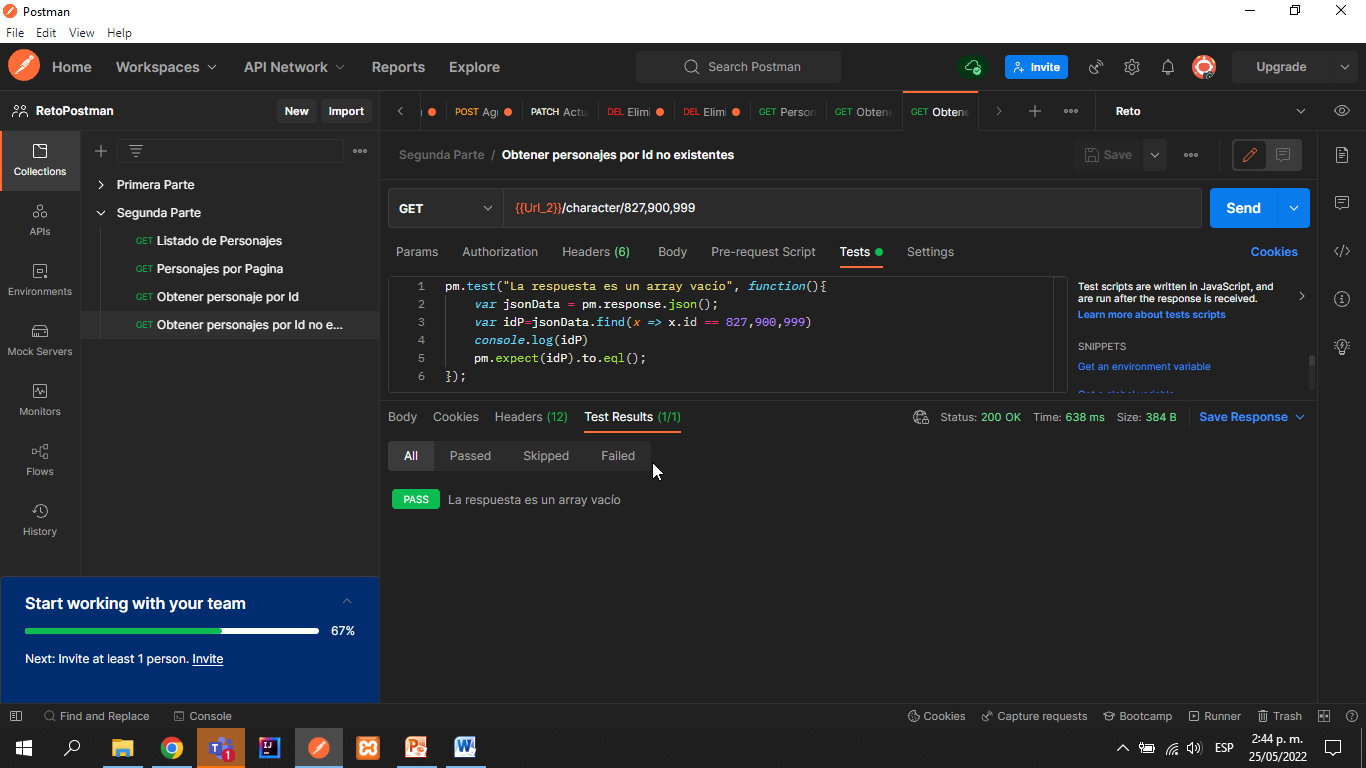
Listado de personajes por página en específico y se agregaron aserciones donde se verifica uno de los personajes que aparece en la pagina seleccionada.



Se Obtuvo personaje por id y se verifica que la species de este personaje corresponda a la que nos trae como respuesta en el body.



Se obtuvo un Array vacío ya que se anexo id de personajes no existentes y se verifica por medio de la aserción que retorna un array vacío



Se puede observar que en el Body retorna un array vacio.

