Práctica 4 - PTI

Blockchain

Arnau Yepes Huguet

Natalia Dai

Grupo 1

04/03/2023

**Índice**

[**1. -Introducción**](#_20f9a3owlw7d) **2**

[**2. -Tareas**](#_n2orlwb6ytr0) **3**

[**3. -Modificaciones**](#_sz9zzol6oufr) **7**

# **1. -Introducción**

En esta sesión experimentamos con una blockchain programada en Python. Este ejemplo presenta las operaciones básicas que se pueden llevar a cabo.

Consta de dos partes:

* En la primera experimentamos con las operaciones básicas y veremos el comportamiento de los datos, nodos y chains.
* En la segunda, a partir del ejemplo dado, añadiremos tres funcionalidades y, en el último caso, veremos como la blockchain detecta manipulaciones por parte de cualquiera.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **2. -Tareas**

En esta primera parte, una vez instaladas las dependencias e inicializar el servidor que usaremos, experimentamos con los ejemplos propuestos en el guión de la práctica.

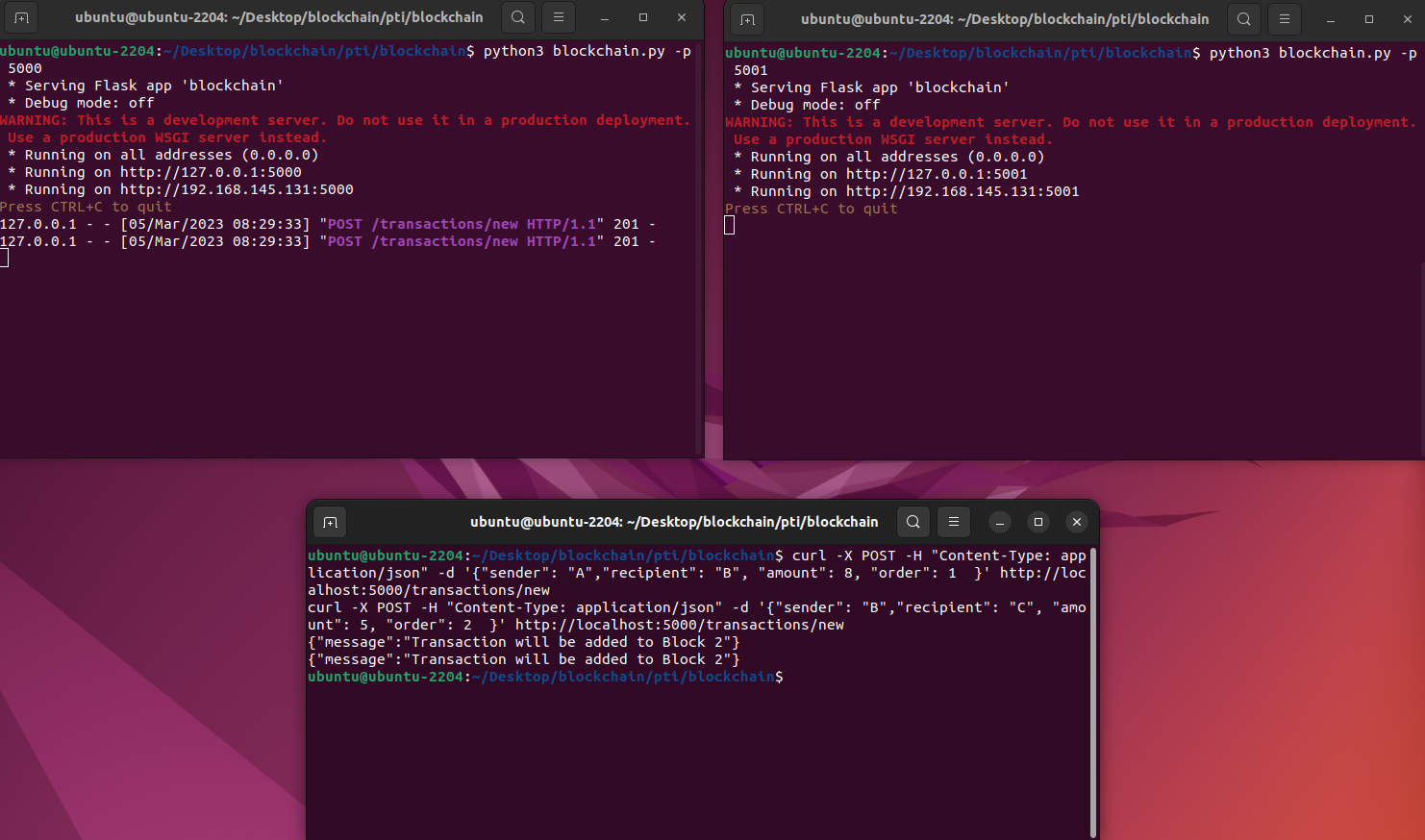


Figura 1 - Escritura de dos transacciones en el nodo 1

En este caso podemos ver como las transacciones se añaden a un bloque que posteriormente minaremos en el nodo para que formen parte de la blockchain.

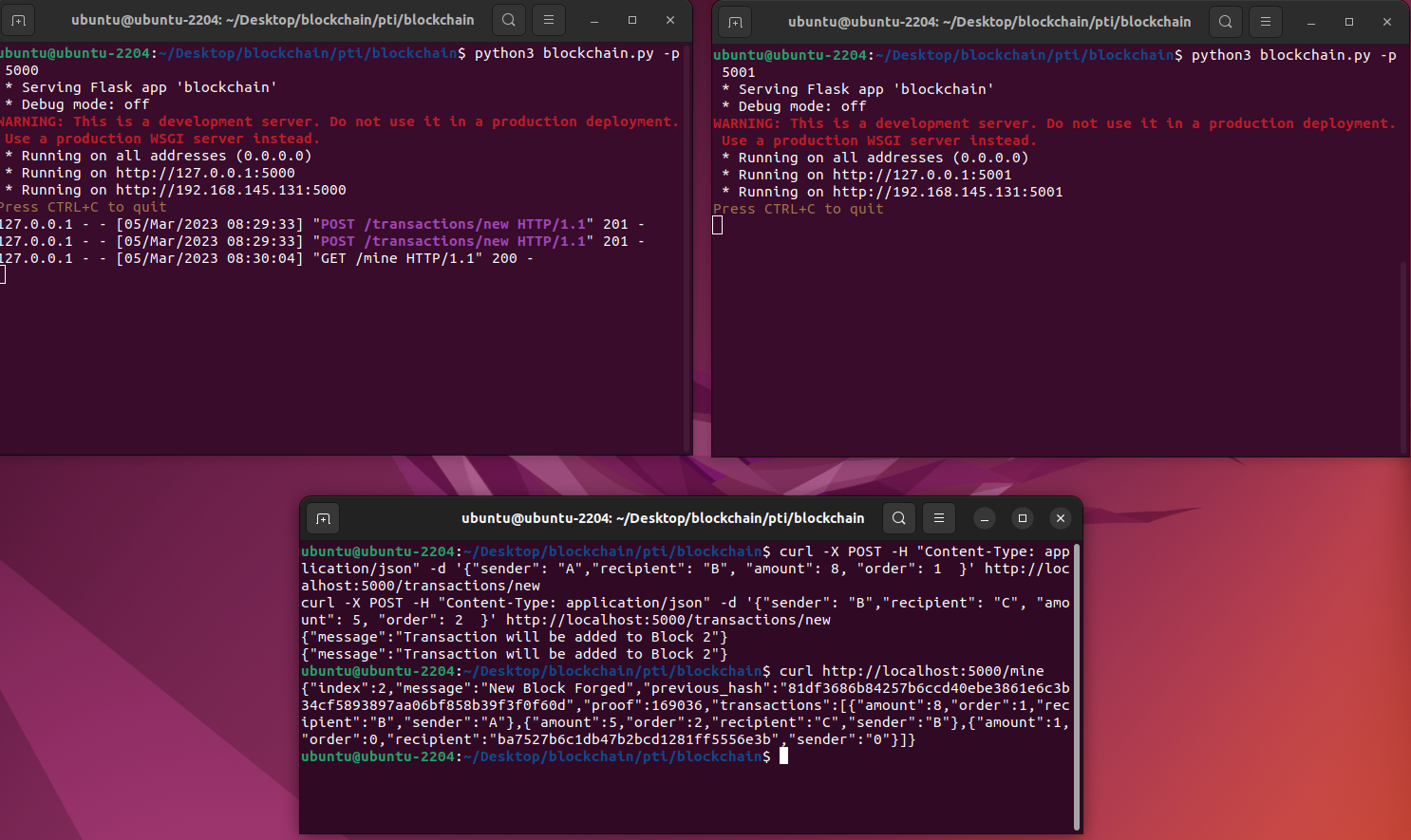


Figura 2 - Minamos el bloque en el nodo 1

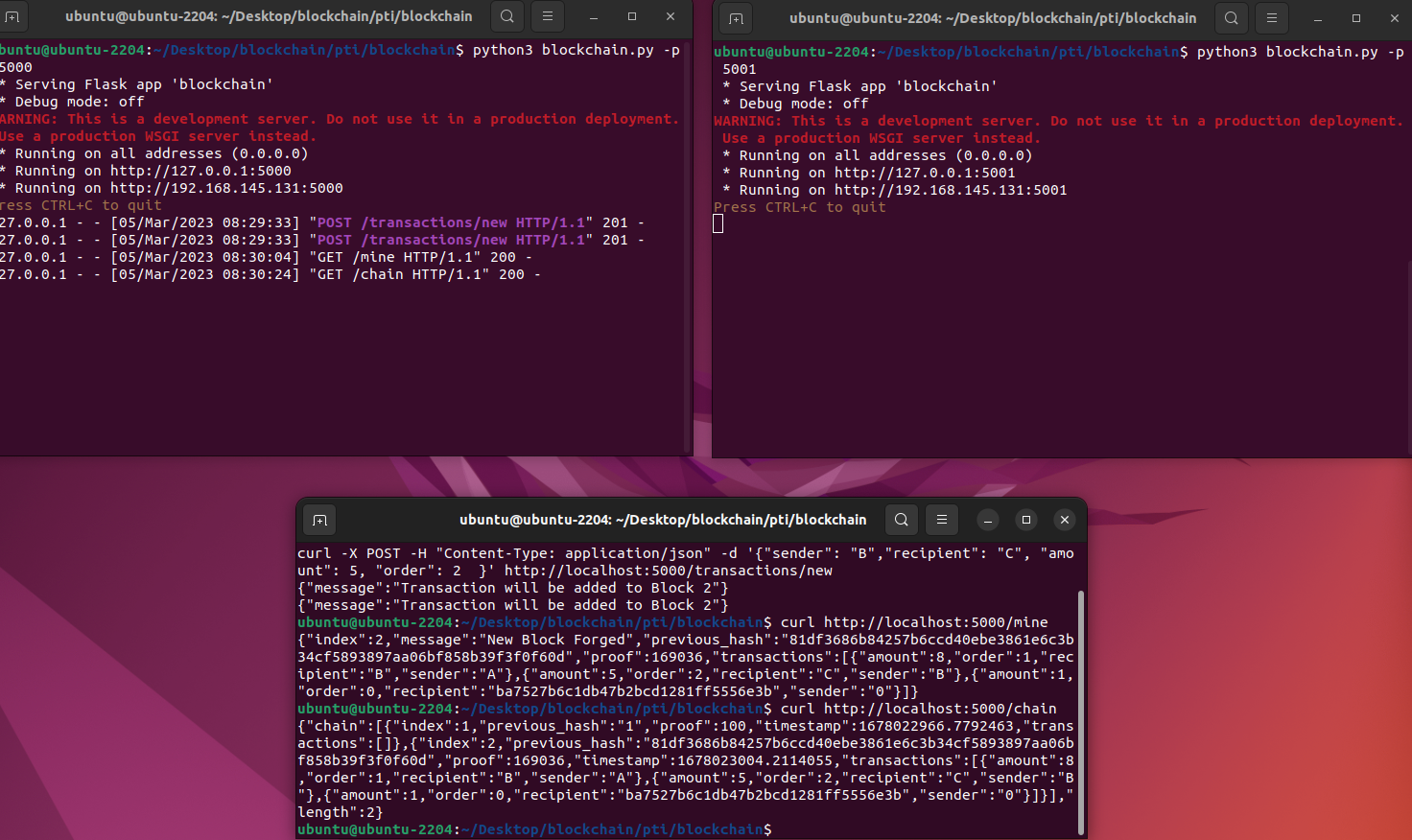


Figura 3 - Pedimos la chain del nodo

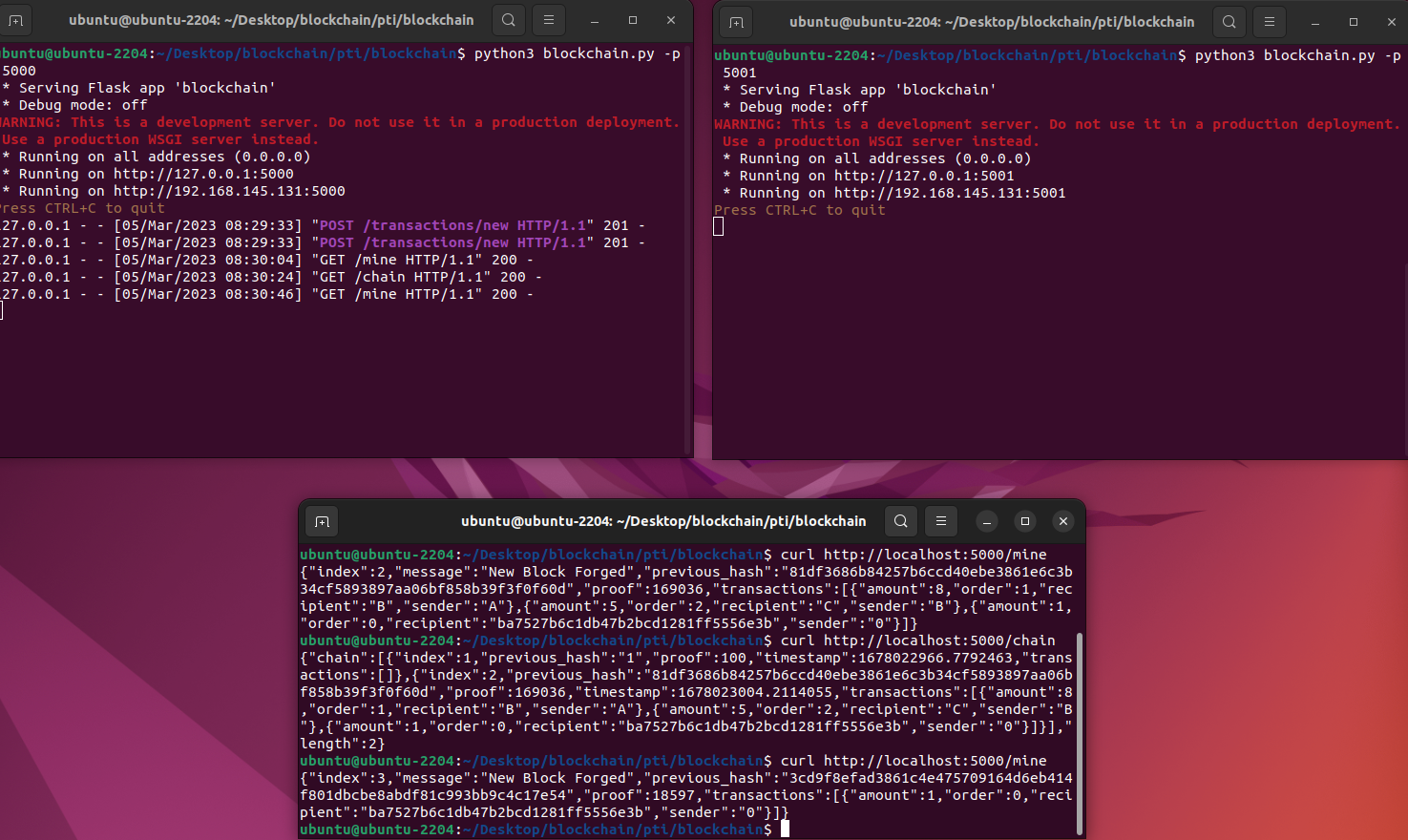


Figura 4 - Minamos otro bloque en el nodo 1

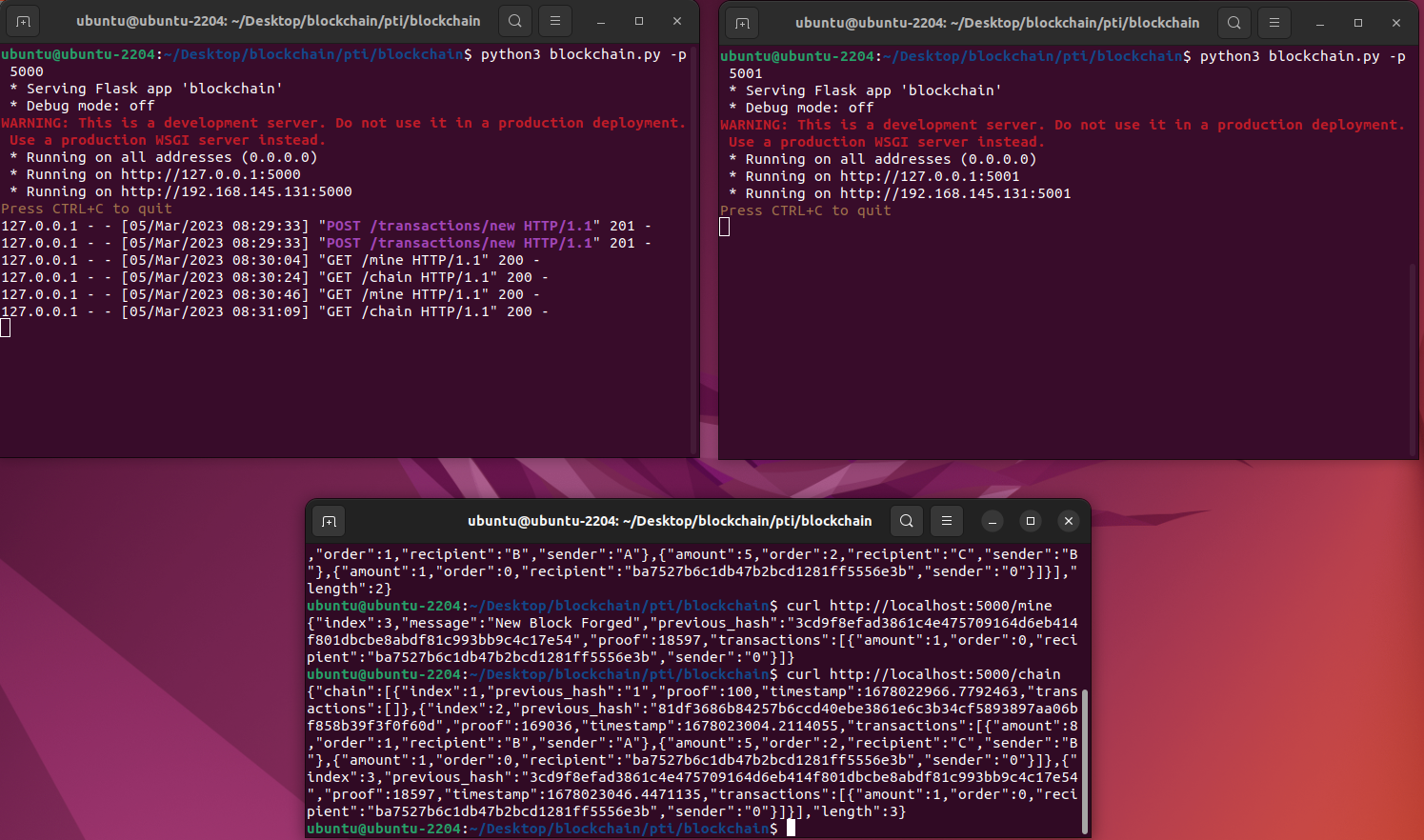


Figura 5 - Pedimos la chain del nodo

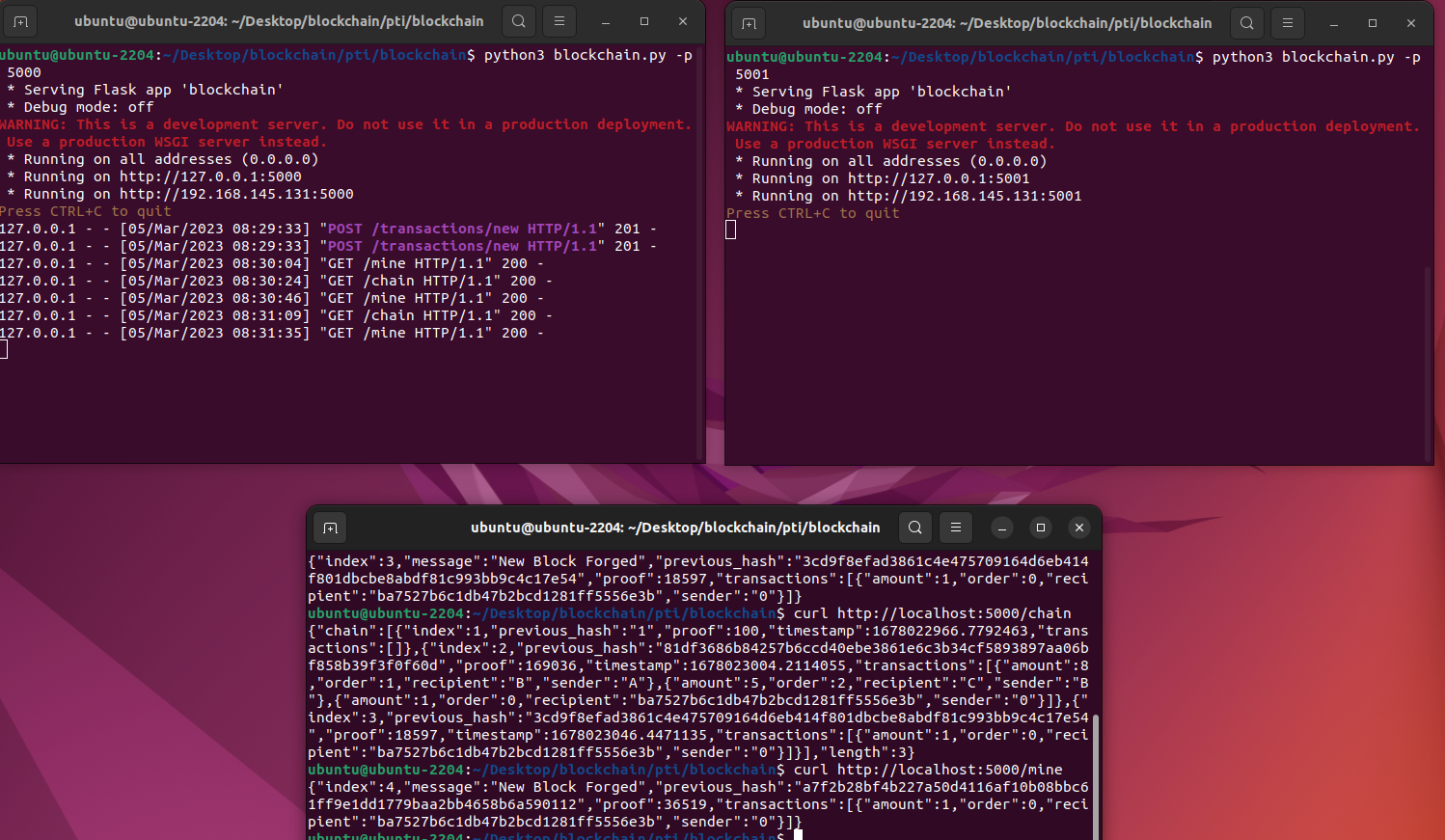


Figura 6 - Minamos otro bloque en el nodo 1

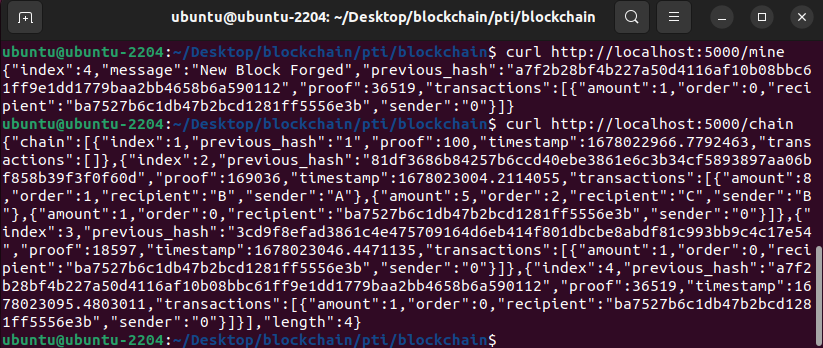


Figura 6 - Pedimos la chain del nodo

Como podemos observar, a medida que minamos bloques, el nodo va expandiendo su chain con los nuevos datos que estamos minando.

*[Pasamos a operar con el nodo 2]*

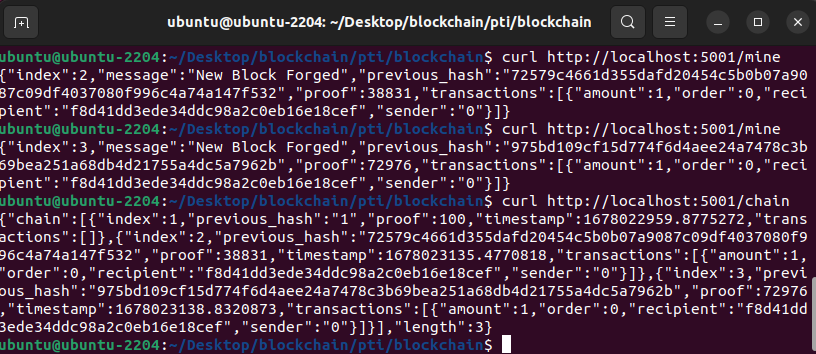


Figura 7 - Minamos dos bloques en el nodo 2 y pedimos su chain

En este caso, como podemos observar en el output de la terminal, el nodo 2 sólo tendrá dos bloques.

*[Pasamos a registrar los nodos entre ellos, dando así lugar a la blockchain]*

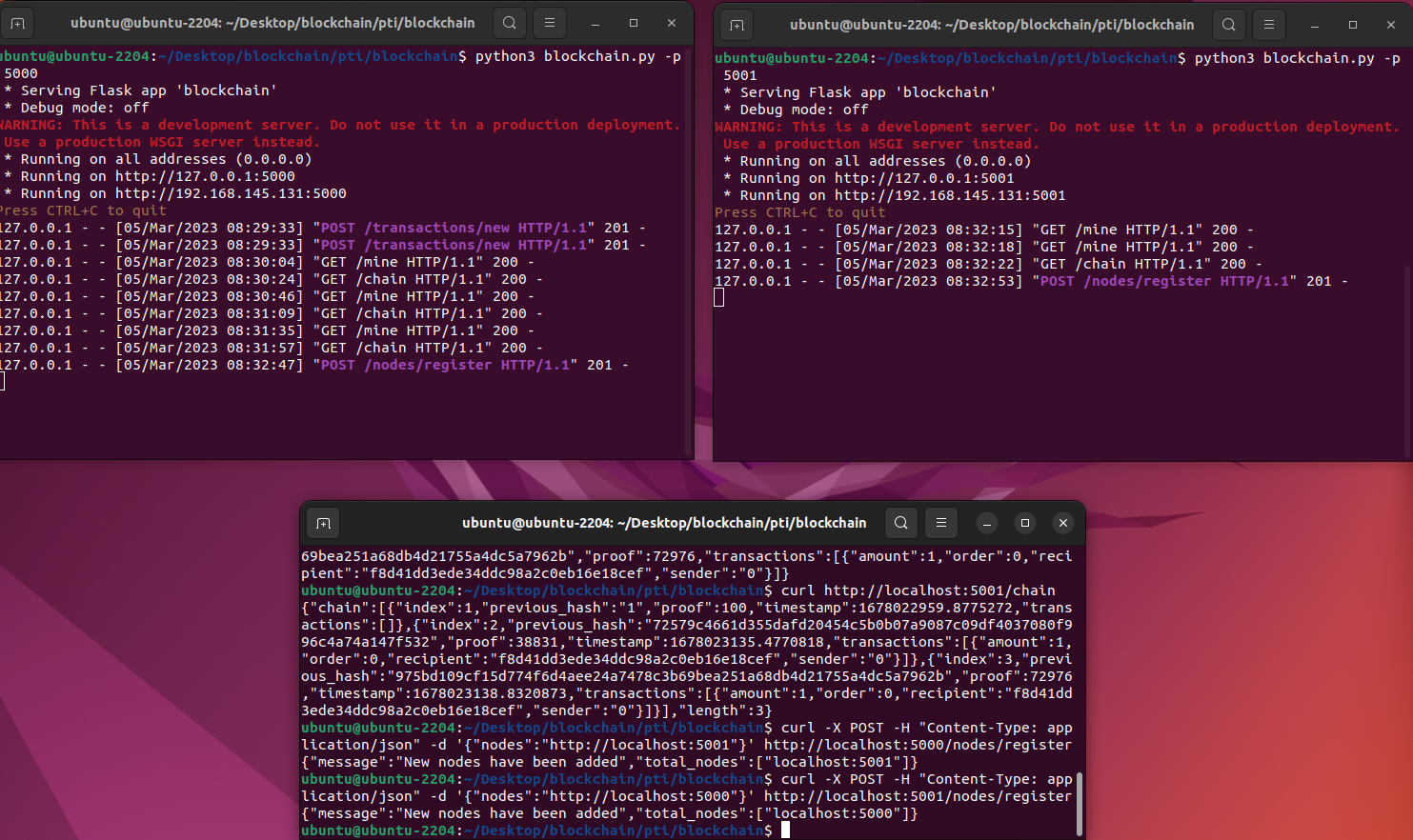


Figura 8 - Registro bidireccional de los nodos

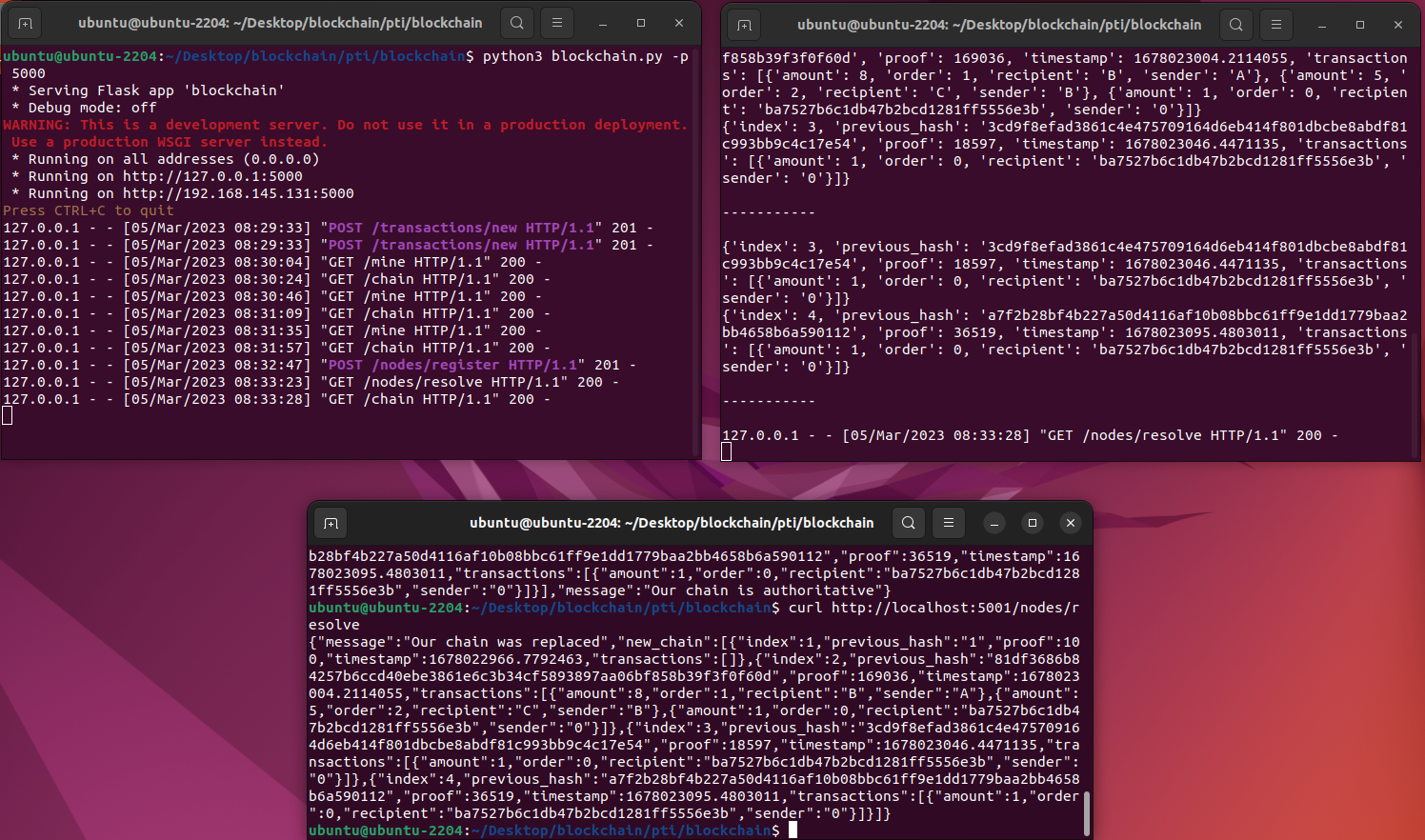


Figura 9 - Resolución entre nodos para formar la blockchain

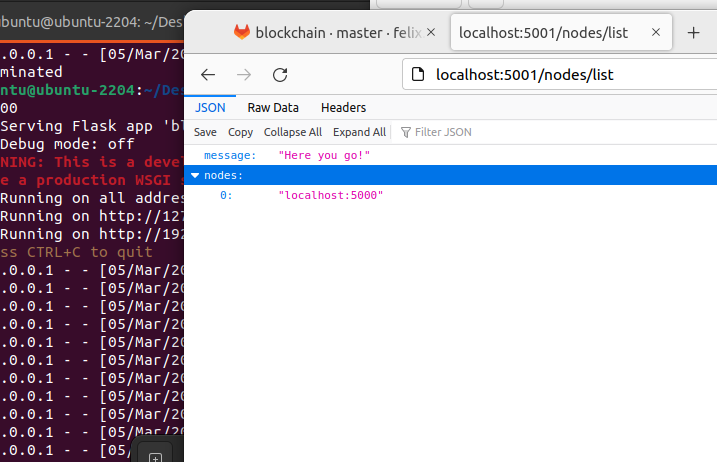
En este punto ya tenemos los dos nodos sincronizados con la información buena. Cómo el nodo 1 tiene la chain más larga y contiene, por tanto, más datos, la blockchain se quedará con el chain del nodo 1.

# 

# **3. -Modificaciones**

Ahora que ya hemos visualizado el funcionamiento de la blockchain se nos pide implementar unas simples nuevas funcionalidades:

***@app.route('/nodes/list', methods=['GET'])***



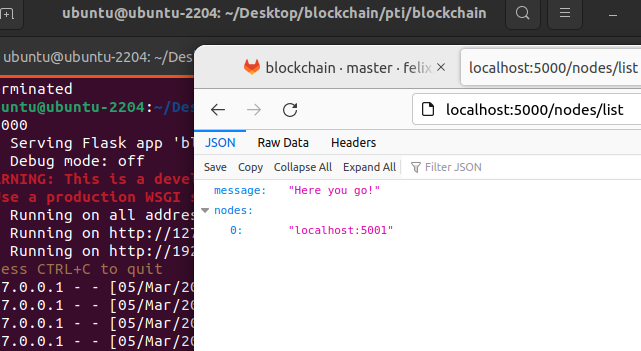


Figura 10 - Tests de la primera función

Como podemos observar estamos listando los nodos de los que el nodo en -p 5000 tiene conocimiento.

Para probar el funcionamiento añadiremos un nuevo nodo y lo registraremos entre los nodos existentes:

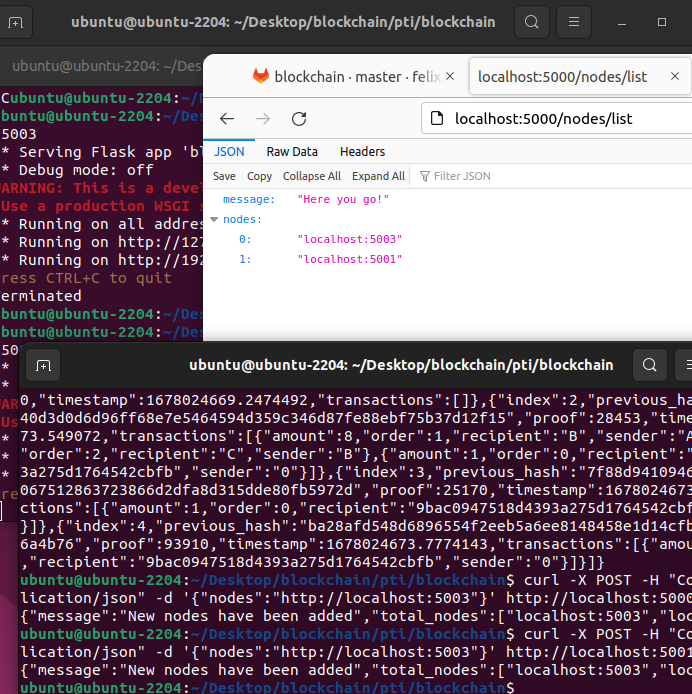


Figura 11 - Prueba final de la primera función

Efectivamente, podemos listar los nodos.

***@app.route('/validate', methods=['GET'])***

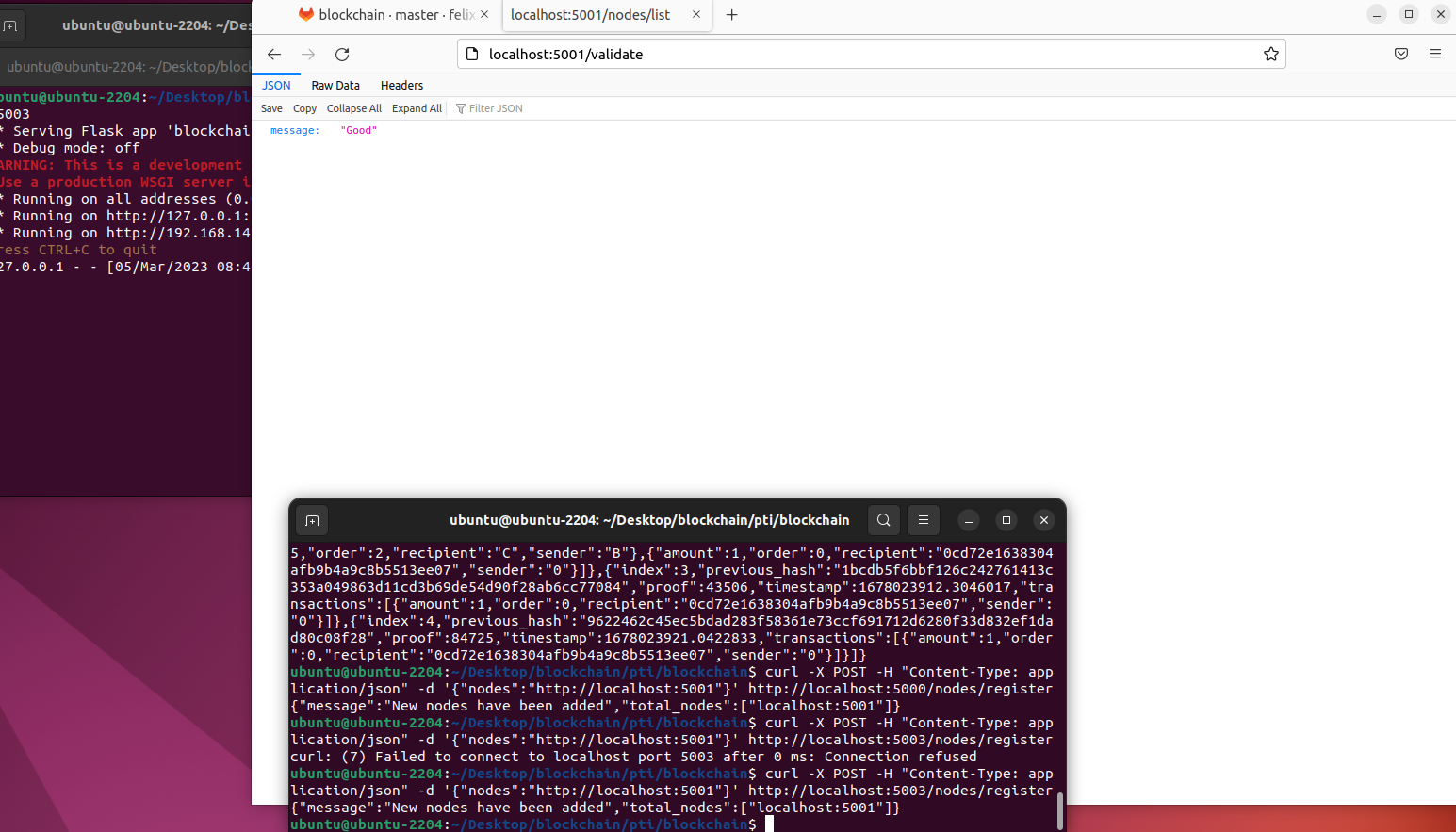


Figura 12 - Test de la segunda función

Como podemos observar, la chain que se crea en el guión de la práctica no presenta fallos. Para probar esta funcionalidad vamos a implementar una función para añadir una transacción a la blockchain sin validar.

***@app.route('/nodes/manipulate', methods=['POST'])***

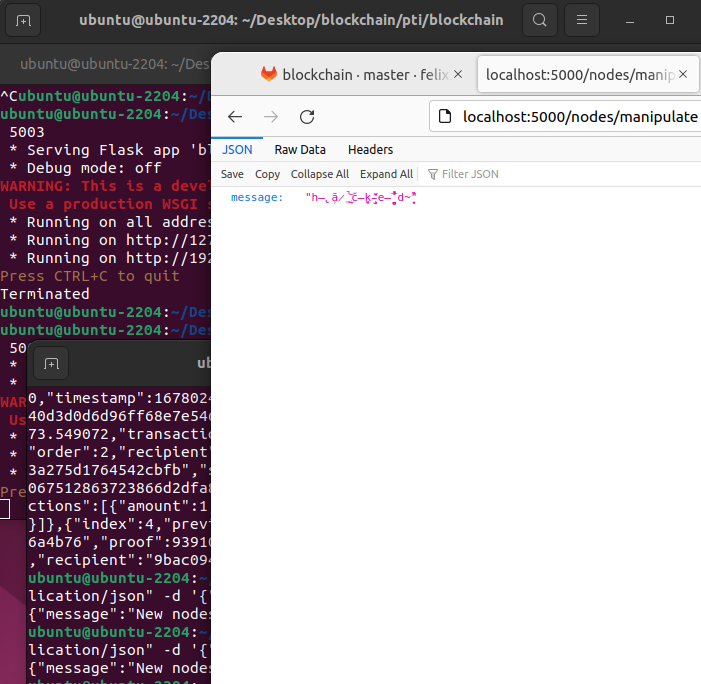
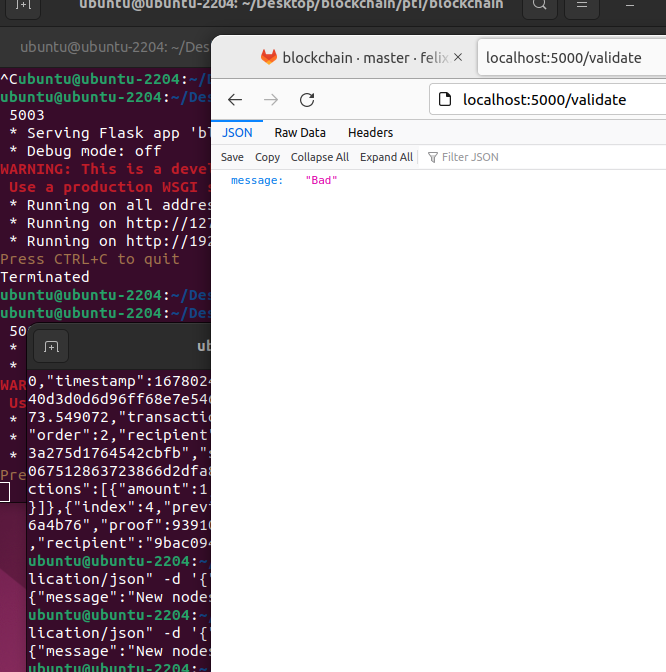


Figura 13 - Test de la tercera función

Figura 14 - Test final validate

Podemos observar que cuando la función valid\_chain analiza el bloque que hemos modificado retorna falso y retornamos Bad chain en nuestra funcionalidad.