ssv.network

Full Stack - Home Assignment

Congratulations on advancing to the next stage for the Software Engineer position in the SSV.Network Fullstack team. This assignment is designed to evaluate your software engineering abilities. It doesn't require domain knowledge and should take about 4 hours.

Feel free to use any programming language. Don’t hesitate to contact phoebe@blox.io or ilya@blox.io for any questions.

Task Overview:

This task involves two entities: Validators and Operators. Validators register with Operators who perform duties for registered validators. The task tracks Validators to Operators registrations.

Assignment Details:

The provided `blocks.json` file sample simulates Ethereum block generation. Each block includes:

- Id: Unique identifier for the block.
- Transactions: Contain id, Ethereum addresses. Some transactions include a 'register' field, indicating the address is registering as a Validator with 4 Operators.

Your task is to develop a system that processes blocks and maintains the Validator and Operator entities registration according to the rules detailed below. You should support processing blocks quickly for example if the network is already at block 10,000,000, and you want to quickly resync the system from block 0.
Create a GET endpoint that takes a file name and block number, returning the state of Operators and Validators post-processing that block.

Key Rules:

- Blocks and transactions should be processed in ascending order of block id and transaction id.
- You should create a unique numeric id for each Validator.
- Validators register with 4 Operators.

- Operators can serve multiple Validators.
- Validators require at least 3 Operators to fulfill duties per block.
- Live blocks are expected to be generated every 12 seconds, forever.
- You can assume the blocks.json format and data is correct, no need to validate it.
- You are not expected to use databases, message queues, datastores, etc. Use the app memory, but explain your considerations and add a short and high level suggestion of what are some possible points that need to be addresses and what you would do if you would design an actual production system.

Example:

`blocks.json` -

block 1

address 0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045 does a Validator registration transaction, The new validator is registered with operators 101, 102, 103, 104.

block 2

address 0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae does a Validator registration transaction, The new validator is registered with operators 105, 106, 107, 108.

block 3

an address 0x1Db3439a222C519ab44bb1144fC28167b4Fa6EE6 creates a transaction.
address 0x999c49fE1c1FC19292e9895d877BB8715040A609 does a Validator registration transaction, The new validator is registered with operators 101, 102, 103, 107

endpoint

GET /state?fileName=blocks.json&blockNumber=2 should return the content of state.json file.

Notes:

Include a README.md with setup and run instructions, and explain your implementation choices.