Backend Engineer Assessment

Implement a backend server and the corresponding Solidity smart contract that supports two basic methods, *send()* and *get-balance():*

https://<server>/send
- parameter 1: **from** account id (string)
- parameter 2: **to** account id (string)
- parameter 3: **amount** (double)

Response:
- return code ("ok" or "error" + some meaningful message)
- id (see below)

https://<server>/get-balance
- parameter 1: **account** id

Response:
- return code ("ok" or "error")
- amount (double)
- id (see below)

Each request must contain the following fields:
- **id** (int in the interval [0:2.000.000.000] – this is an identifier for the request and will be sent back in the response)
- **timestamp –** current timestamp in milliseconds
- **params** as defined above for each method, with corresponding values
- **key –** public key (base 64)
- **signature –** see below how to generate it.

How to generate the signature:
- use RSA asymmetric keys such that the private key is always kept secret by the client while the public key is on the blockchain (stored via the smart contract), associated with one or multiple account(s)
- make sure the parameters are in the order stated above
- build a string with all the fields such as: **id+timestamp+param1+val1+param2+val2+...+paramN+valN+key**
- use the RSA algorithm with the private key to get the encryption of the string above
- encode the output in base 64 to obtain the final signature

How to test your implementation:
- Write a Solidity smart contract that supports the two methods mentioned above and deploy it to an Ethereum testnet (e.g., on the Sepolia testnet).
- The state database of the smart contract should contain <account, balance> and <account, public key> records. It is up to you how you implement this.
- Write a backend in Java, preferably using Spring Framework, and connect it to the smart contract deployed on the Ethereum testnet.
- Create some accounts and generate some keys to test your implementation

Benchmarking

- Design and implement some benchmarking code to get the maximum performance of your system when performing *send* and *get-balance* operations.
- Present the benchmarking tool and the results during the interview.

Other requirements

Deadline: 1 week. The project is to be presented (code + functionality) in an online interview to be scheduled one week after receiving this assessment. You are required to share the code with us (github or archive) before or during the interview.

Resources

- You could use Remix IDE to implement and deploy you Solidy smart contract: https://remix.ethereum.org
- You could use web3j maven plugin: https://github.com/web3j/web3j-maven-plugin
- As a node provider, you could use Infura: https://www.infura.io/
- https://github.com/web3j/web3j
- https://docs.web3j.io/4.8.7/quickstart/