Overview:
The Transaction Broadcaster Service is responsible for receiving transaction requests through an internal API, signing the transaction data, broadcasting it to an EVM-compatible blockchain network, and providing a status page to view the list of transactions that have passed or failed. The service should automatically retry failed broadcasts, and an admin should have the ability to manually retry failed transactions.

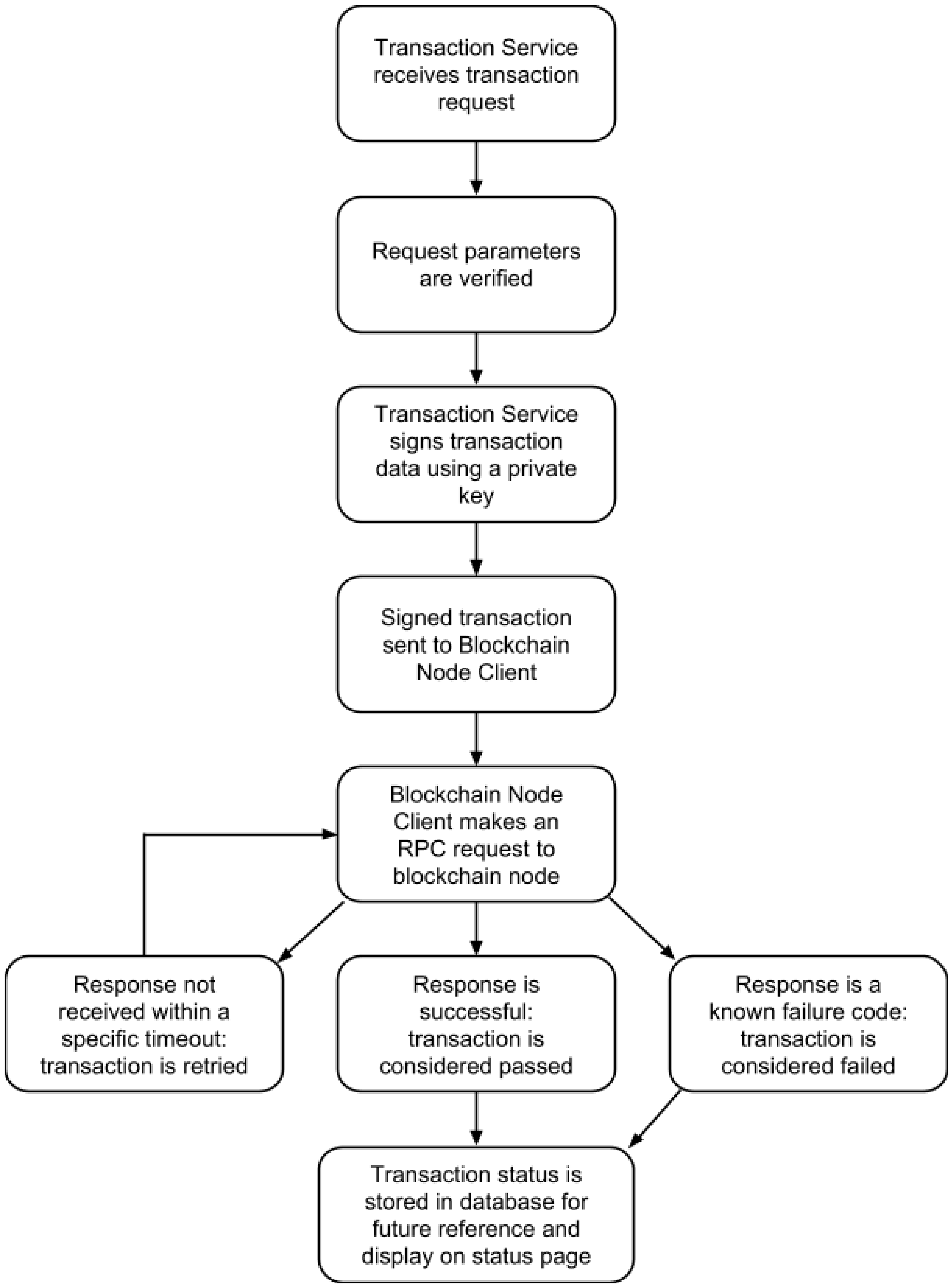
Architecture:
The transaction Broadcaster Service can be designed using a microservice architecture with the following components:

- API Gateway
    - Handle incoming requests and forward them to the appropriate service
- Transaction Service
    - Receive transaction requests, signs the data, and initiates broadcasting process
- Retry Service
    - Periodically checks for failed transactions and retries them automatically
- Blockchain Node Client
    - Communicates with the blockchain node to broadcast signed transactions and retrieve response codes

API Design: (mainly for my personal understanding)
The API Design serves as the interface for communication between the client applications or users and the Transaction Broadcaster Service. It defines the endpoints and parameters for submitting transaction requests and receiving responses.
The internal API endpoint ‘ /broadcast_transaction ‘ receives a POST request with the parameters:

- *message_type:* which specifies the type of message or transaction to be performed
- *data:* which contains the transaction data in a specific format

Transaction Processing Flow:


Transaction Service receives transaction request
Request parameters are verified
Transaction Service signs transaction data using a private key
Signed transaction sent to Blockchain Node Client
Blockchain Node Client makes an RPC request to blockchain node
Response not received within a specific timeout: transaction is retried
Response is successful: transaction is considered passed
Response is a known failure code: transaction is considered failed
Transaction status is stored in database for future reference and display on status page

Status Page:
The status page provides a user interface to view the list of transactions that have passed or failed.It displays relevant information such as the transaction ID, status (passed/failed), timestamp, and any additional details. It also provides an option for the admin to manually retry failed transactions. Such a page has to be very user-friendly and easy to understand. Some examples that exist at the moment are those on Binance and Coinbase.

Additional Requirements:

1. *If POST /broadcast_transction returns HTTP 200 OK, it is assumed that the transaction will eventually be broadcasted successfully. If the broadcaster service restarts unexpectedly, it should still fulfill them.*
    - By implementing a reliable database or storage system to store the transaction data. When the service restarts, it can retrieve the stored transactions and resume the broadcasting process.
2. *An admin is able to, at any point in time, retry a failed broadcast.*
    - We can add an administrative interface such that the admin can use this interface to view the list of failed transactions, select specific transactions for retry, and trigger the retry process. A retry service can be implemented to periodically check for failed transactions, automatically retrying them and marking them with a specific status to retry, but the admin interface provides manual control for immediate retries.