UDACITY

‹ Back to Blockchain Developer

Build a Private Blockchain Notary Service

REVIEW

CODE REVIEW 4

HISTORY

▾ app.js 4

```

// link for non-Electrum digital signing
//https://medium.com/@alexpanas/udacity-blockchain-nanodegree-using-nod

const express = require('express');
const app = express();
const port = 8000;

const http = require('http');
const path = require('path');

const bodyParser = require('body-parser');

const Blockchain = require('./blockchainClass');
const Block = require('./blockClass');

const Response = require('./responseClass');

const validationWindow = 300; //5 minutes per project requirements (30

const bitcoin = require('bitcoinjs-lib');
const bitcoinMessage = require('bitcoinjs-message');

// http://expressjs.com/en/guide/writing-middleware.html


// Syntax
// app.get('/', (req, res) => res.send('Hello World!'))
// '/' is the path on host
// req - client request
// res - server response


let blockchain = new Blockchain;


app.get('/', (req, res) => res.sendFile(path.join(__dirname + '/home.ht

app.get('/block/:id', async (req, res) => {
    const blockRes = await blockchain.getBlock(req.params.id);
```

REQUIRED

❌ The `storyDecoded` is missing from the JSON response. Please add it.

```
    if (blockRes) {
        res.send(blockRes) // server response
    } else {
        res.status(404).send("Block Not Found")
    }
});

//body parser allows form data to be available in req.body
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));










app.post('/block', async (req, res) => {
    console.log('----------------------------');

    //NOTE: when using postman, do \\' to escape apostrophe, not the sa
    //console.log('Received star registration request object: ' + JSON.
    if (!req.body.address || !req.body.star) {
        res.status(400).json({
            "status": 400,
            message: "Address and requested star must both be present"
```

SUGGESTION

Use error payloads

All exceptions should be mapped in an error payload. Here is an example how a JSON pa

```
{
  "errors": [
   {
    "userMessage": "Sorry, the requested resource does not exist",
    "internalMessage": "No car found in the database",
    "code": 34,
    "more info": "http://dev.mwaysolutions.com/blog/api/v1/errors/12345"
   }
  ]
}
```

```
        })
    } else if (encodeURI(req.body.star.story).split(/%..|./).length - 1
        res.status(400).json({
            "status": 400,
            message: "Star story size must not exceed 500 bytes"
        })
    } else {
        //console.log(req.body.address);
        //console.log(validPool[0].address);
        let starIdx = validPool.findIndex(f => f.address === req.body.a
```

REQUIRED

❌ The user should only be able to register one star per validation. If you want to registe
the process each time.

```
        console.log('index of validated address in validPool: ' + starI
        if (starIdx >= 0) {
            req.body.star.story = new Buffer(req.body.star.story).toStr
            await blockchain.addBlock(new Block(req.body));
            const height = await blockchain.getBlockHeight();
            const response = await blockchain.getBlock(height);
            res.send(response);
        } else {

            res.status(400).json({
                "status": 400,
                message: "Public address not verifiedk"
            })
        }
    }
});









mempool = [];
```

SUGGESTION

⚠ I suggest that you store the validation information in the database to persist the data
use another machine to register the star.

```

app.post('/requestValidation', async (req, res) => {
    if (!req.body.address) {
        res.status(400).json({
            "status": 400,
            message: "Address must not be empty"
        })
        console.log('----------------------------');
        console.log('Empty address request made.');
    }
    else {
        let nowTime = new Date().getTime().toString().slice(0, -3);

        //is this redundant to repackage req.body into a new object via
        resp = new Response;
        resp.validationWindow = validationWindow;
        resp.address = req.body.address;
        resp.requestTimeStamp = nowTime;
        resp.message = resp.address + ':' + resp.requestTimeStamp + ':s
        console.log(resp.message);

        if (mempool.findIndex(f => f.address === req.body.address) ===
            console.log('----------------------------');
            //console.log(mempool.findIndex(f => f.address === req.body
            console.log('Address received: ' + (req.body.address));
            console.log('Request will only be valid for 5 minutes.');
            console.log('Message to sign/verify: ' + (resp.message));
            console.log('Please validate at */message-signature/validat
            console.log('Mempool length is: ' + mempool.length);
            console.log('');
            mempool.push(resp);
        } else if (mempool.findIndex(f => f.address === req.body.addres
            console.log('-----------------------------');
            let reqIdx = mempool.findIndex(f => f.address === req.body.
            let reqTimeStamp = mempool[reqIdx].requestTimeStamp;
            let timeDiff = nowTime - reqTimeStamp;
            console.log('Address received: ' + (req.body.address));
            console.log('current timestamp is: ' + nowTime);
            console.log('retrieved timestamp is: ' + reqTimeStamp);
            console.log('timeDiff is: ' + timeDiff);
            if (timeDiff <= validationWindow) {
                console.log('...........................');
                console.log('Request already exists...');
                console.log('Please validate at */message-signature/val
                console.log('');
            } else if (timeDiff > validationWindow) {
                console.log('...........................');
                console.log('Expired request exists, new request genera
                console.log('Address received: ' + (req.body.address));
                console.log('Request will only be valid for 5 minutes.'
                console.log('Please validate at */message-signature/val
                console.log('Mempool length is: ' + mempool.length);
                console.log('');
                mempool.splice(reqIdx); //remove expired entry before p
                mempool.push(resp);
            }
        }
        res.send(resp);
    }
});








///////////////////////////////////////////////////

//TEST message set from standard electrum wallet
// let addressElecStd = '1KwJmv6KqMNwqZMqd9ZdVYJH9VZ1vnctFt'
// let messageElecStd = '1KwJmv6KqMNwqZMqd9ZdVYJH9VZ1vnctFt:1532330740:
// let signatureElecStd = 'H0dFqcBJhBpRINpCHirDizr4eCfQiZyj63qC/g1kBQPL
// let statusTest = {
//     address: addressElecStd,
//     requestTimeStamp: '1539107147',
//     message: messageElecStd,
//     validationWindow: '50',
//     messageSignature: "valid"
// }

validPool = [];

```

```
//validPool.push(statusTest);
//console.log('validPool: ' + JSON.stringify(validPool[0]));

app.post('/message-signature/validate', async (req, res) => {
    console.log('---------------------------');
    //console.log('req body address: '+ req.body.address)
    if (!req.body.address || !req.body.signature) {
        res.status(400).json({
            "status": 400,
            message: "Address & signature data must not be empty"
        })
    } else if (mempool.findIndex(f => f.address === req.body.address) =
        console.log("A request for this address does not exist... submi
        res.status(400).json({
            "status": 400,
            message: "A request for this address does not exist... subm
        })
    } else if (mempool.findIndex(f => f.address === req.body.address) >
        //console.log(mempool.findIndex(f => f.address === req.body.add
        let reqIdx2 = mempool.findIndex(f => f.address === req.body.adc
        //console.log(reqIdx2);

        //if request isn't made, this will bomb the app
        let reqTimeStamp2 = mempool[reqIdx2].requestTimeStamp;
        //console.log(mempool[reqIdx2].requestTimeStamp);
        //console.log(reqTimeStamp2);

        ///////////////////////////  testing, switch later
        // since timestamp is part of message, need to hotwire in the m

        let message2 = mempool[reqIdx2].message;
        //let message2 = '142BDCeSGbXjWKaAnYXbMpZ6sbrSAo3DpZ:153233074C
        ///////////////////////////  testing, switch later
        ///////////////////////////  testing, switch later

        let nowTime2 = new Date().getTime().toString().slice(0, -3)
        console.log('Timestamp of signature receipt: ' + nowTime2);
        let timeDiff2 = nowTime2 - reqTimeStamp2;

        let status = {
            address: req.body.address,
            requestTimeStamp: reqTimeStamp2,
            message: mempool[reqIdx2].message,
            validationWindow: timeDiff2,
            messageSignature: "invalid"
        }

        let sigValidity = bitcoinMessage.verify(message2, req.body.addr
        if (!sigValidity) {
            console.log('Invalid signature');
        } else if (sigValidity) {
            if (timeDiff2 <= validationWindow) {
                console.log("Ownership of blockchain address is verifie
                console.log("Please proceed to */block to complete star
                status.messageSignature = 'valid'
                validPool.push(status);

                console.log('display status object: ' + JSON.stringify(
            } else {
                console.log("Time limit exceeded, request expired, plea
            }
        }
        let resp2 = {
            registerStar: true,
            status: status
        }
        res.send(resp2);
    }
});







app.get('/stars/:address', async (req, res) => {
    // LANDMINES
    //1  address entering api has prefix to chop off
    //2  old blocks in chain cause findIndex() method to fail - replace
    //3  genesis block does not have all body properties for findIndex(
    //4  does js shallow copy by default?

    // TODO
    //

    console.log('----------------------------');
     //let lookupAddress = req.params.address.slice(8); //removing addr

    console.log('Received request: ' + req.params.address);
    let lookup  = req.params.address.split(':');
    console.log('lookup prefix: ' + lookup[0]);
    console.log('lookup value: ' + lookup[1]);

    const blockPool = await blockchain.getAllBlocks();

    blockPool.shift();
    // let blockPoolShift = Object.assign({}, blockPool);
    //console.log(Object.getOwnPropertyNames(blockPoolShift[height-1]))

    if (lookup[0] === 'address') {
        const adrFinds = blockPool.filter(f => f.body.address === looku

        adrFinds.forEach(function(obj) {
            obj.body.star.storyDecoded = (new Buffer(obj.body.star.stor
        });
        console.log('adrFinds: ' + JSON.stringify(adrFinds));

        if (adrFinds.length > 0) {
            res.send(adrFinds) // server response
        } else {
            res.status(400).send("Public address not found")
        }
    } else if (lookup[0] === 'hash') {
        const hashFinds = blockPool.filter(f => f.hash === lookup[1]);

        hashFinds.forEach(function(obj) {
            obj.body.star.storyDecoded = (new Buffer(obj.body.star.stor
        });
        console.log('hashFinds: ' + JSON.stringify(hashFinds));

        if (hashFinds.length > 0) {
            res.send(hashFinds) // server response
        } else {
            res.status(400).send("Public address not found in blockchai
        }
    } else {
        res.status(400).send("Request not found in blockchain")
    }
});














// redirect to home - needs to be final redirect
// app.get('*', function (req, res) {
//     res.redirect('/');
// });

app.listen(port,
    () => console.log(`app listening on port ${port}!`));

```

▸ scratch.js

▸ responseClass.js

▸ privateBlockchain.js

▸ levelFunctions.js

▸ home_files/header.html

▸ home_files/filelist.xml

▸ home_files/colorschememapping.xml

▸ home.html

▸ first_time_setup_notes.md

▸ blockchainClass.js

▸ blockClass.js

▸ README.md

RETURN TO PATH

Rate this review