

Smart contracts Event Radar



ETHEREUM ENS LOOKUP

Junaid
JUNAIDEV@HOTMAIL.COM

Contents

Introduction	2
Tools\APIs\Technology Stack.....	2
Technical Architecture	2
1) Restful Services endpoint:	2
2) Front-end:	3
Testing.....	4
How to run	5

Smart Contracts Event Radar

Introduction

This web app could be used to monitor events of specified contract. In version 1.0.0 the smart contract address and ABI is hardcoded in application, but in future version there will be option to add any contract dynamically. Currently app monitors events of ENS Ethereum Name Service smart contract located at address '0x6090A6e47849629b7245Dfa1Ca21D94cd15878Ef'.

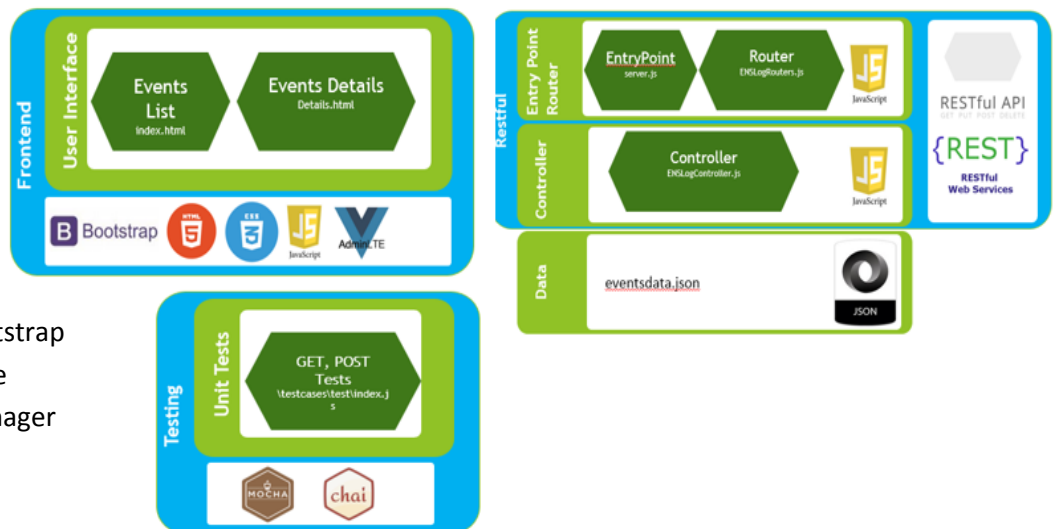
The app is divided into three different modules:

- Restful services endpoint,
- Client code and
- Testing code.

Tools\APIs\Technology Stack

Following APIs\Libs are used for application.

- Node.js based
- Java script
- Express
- JSON
- Mocha
- Chai
- Web3
- HTML \ CSS \ Bootstrap
- Visual Studio Code
- npm package manager



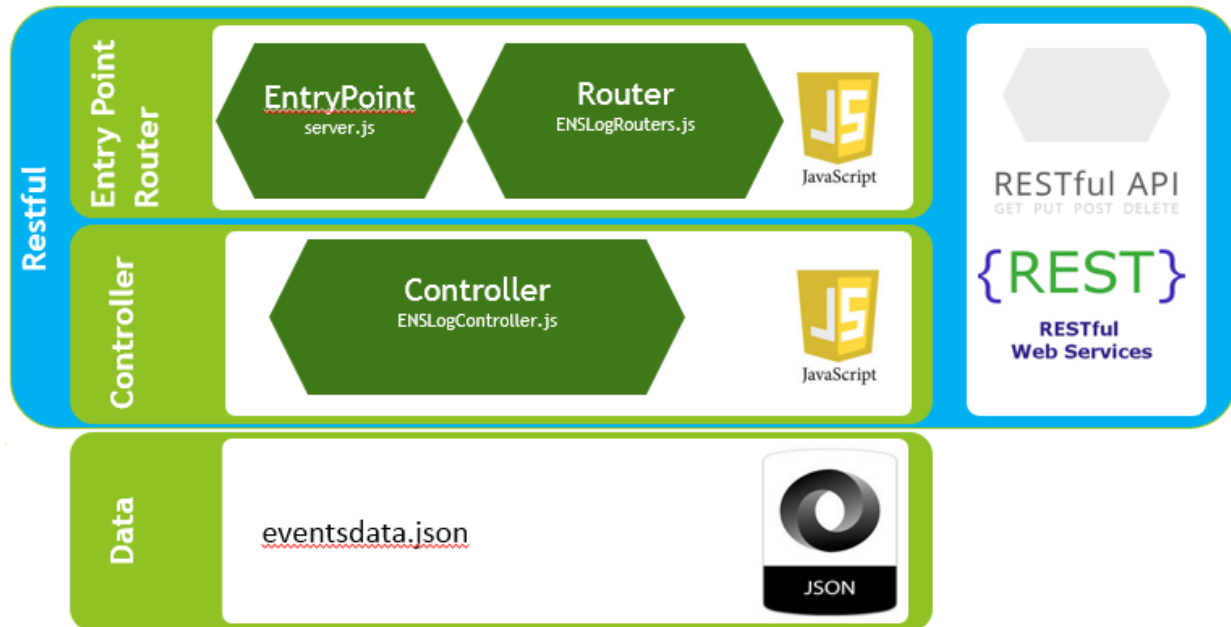
Technical Architecture

1) Restful Services endpoint:

The purpose of backend module is to receive events in json from client code and save in underlying data file. When required data by client code this will send back data in json format. The events occurring in smart contract are saved in json file.

This module is developed using node.js and express based restful services. The service entry point is located in server.js file. The endpoint routes are mapped by using file located in side \api\routes\ENSLogRoutes.js file and controllers are written in \api\controllers\ENSLogController.js. As

this is initial stage of project so data is saved in eventsdata.json data file.



2) Front-end:

The front end is based on bootstrap and Admin LTE. For now in version 1.0.0 the event lookup calls are sent to Ethereum node directly by front end. The calls are sent for looking events of ENS smart contract. When an event occurs front end shows notification and sends back event in json format to restful end point for persistence in server side Jason data file. Along with sending data to server the data table of events is also refreshed by querying data from restful end point. There are two files in front end one is index.html for listing all events that are recorded. The second file is details.html which renders details of selected event. In version 1.0.0 the data of event is not formatted like values for gas and timestamp inside event is not formatted.



Testing

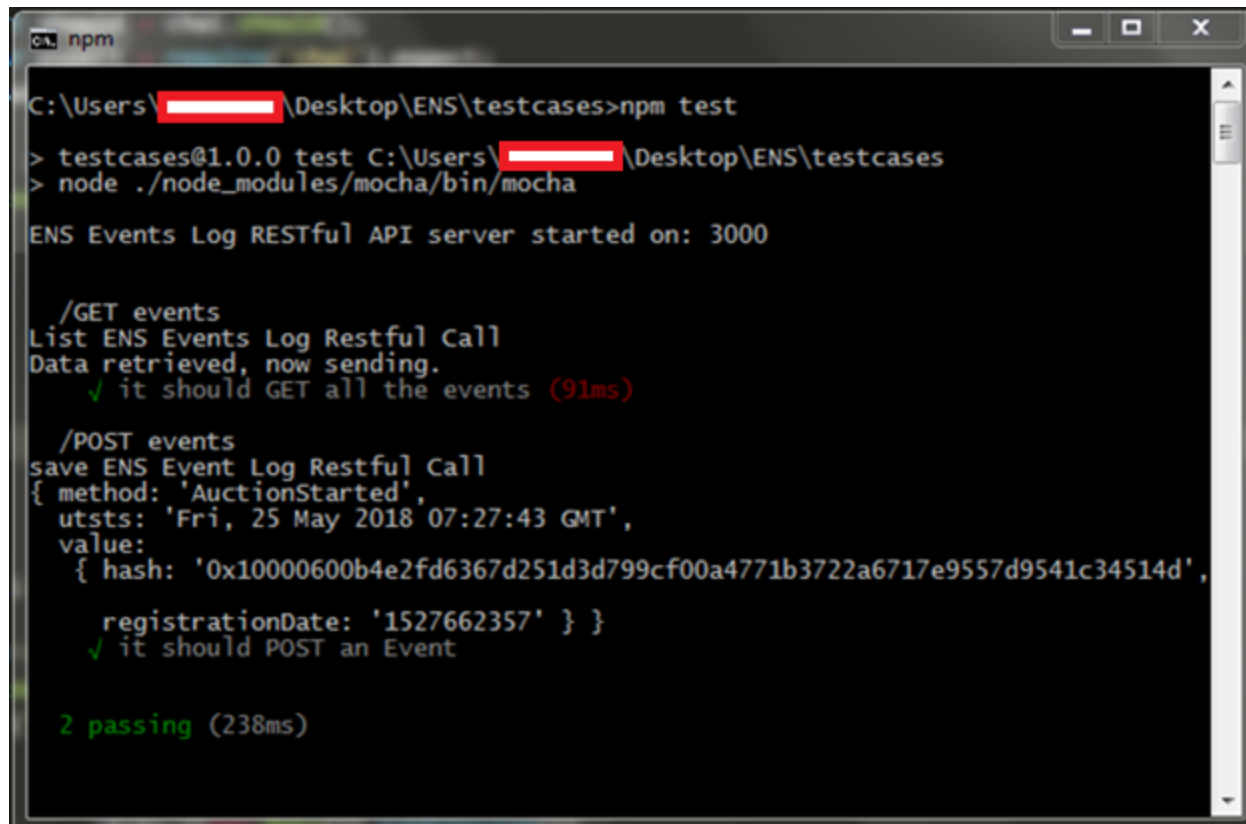
For demonstration of unit testing the unit tests are written by using chai and mocha java script libraries. These libs are used for TDD and automated unit testing.



There are two tests written in `testcases\test\index.js`. One testcase is for POST call and saving event JSON. Second test case is for GET call, for getting list of already persisted events. The test cases could be invoked by using following command:

```
npm test
```

Before test cases invocation make sure that required libraries are installed by using npm and server is not running at port 3000. The test cases code will run server for itself.



```

C:\Users\ [redacted] \Desktop\ENS\testcases>npm test

> testcases@1.0.0 test C:\Users\ [redacted] \Desktop\ENS\testcases
> node ./node_modules/mocha/bin/mocha

ENS Events Log RESTful API server started on: 3000

  /GET events
  List ENS Events Log Restful Call
  Data retrieved, now sending.
    ✓ it should GET all the events (91ms)

  /POST events
  save ENS Event Log Restful Call
  { method: 'AuctionStarted',
    utsts: 'Fri, 25 May 2018 07:27:43 GMT',
    value: { hash: '0x10000600b4e2fd6367d251d3d799cf00a4771b3722a6717e9557d9541c34514d',
      registrationDate: '1527662357' } }
    ✓ it should POST an Event

  2 passing (238ms)

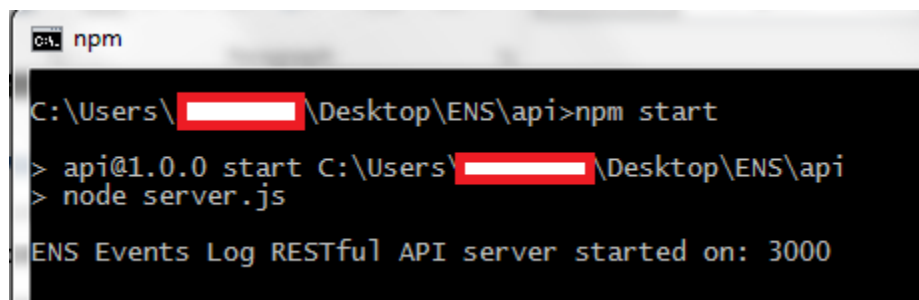
```

How to run

For running application first make sure that all libs are installed by using npm. After that First browse to api folder and issue following command:

```
api\ npm start
```

If everything is properly setup following screen short will be shown:



```

C:\Users\ [redacted] \Desktop\ENS\api>npm start

> api@1.0.0 start C:\Users\ [redacted] \Desktop\ENS\api
> node server.js

ENS Events Log RESTful API server started on: 3000

```

This indicates that server is running at port 3000. The Restful calls could also be verified by using post man. Following is post call demonstration for saving an event in server json.

Ethereum Smart Contracts Event Radar

The screenshot shows a REST client interface with a POST request to `localhost:3000/ensevents/`. The request body is a JSON object:

```
1 {"method": "BidRevealed", "utsts": "Fri, 25 May 2018 05:14:48 GMT", "value": {"hash": "0x4d62bfff3573493cac6c40c25d94cfeff14877994568c505274705711b4eaf512", "owner": "0x64372db6405879214a0a76a7f1e9c013fd2fd84b", "value": "10000000000000000", "status": "2"}}}
```

Following is restful get call demonstration for getting specified event from server json.

The screenshot shows a REST client interface with a GET request to `localhost:3000/ensevents/0xe69c6bd8d17764ca6350a3ccffd8099d115396e56a15fc8cf1877ee862f7b3fb`. The response is a JSON object:

```
1 {
2   "method": "NewBid",
3   "value": {
4     "hash": "0xe69c6bd8d17764ca6350a3ccffd8099d115396e56a15fc8cf1877ee862f7b3fb",
5     "bidder": "0x60c4b1ca9efe96ab7c5b8130a1dc1e70f2477131",
6     "deposit": "10000000000000000"
7   }
8 }
```

The response status is 200 OK and the time taken is 95 ms.

Following is restful get call demonstration for getting list of all events from server json.

Ethereum Smart Contracts Event Radar

The screenshot shows a REST client interface with a GET request to `localhost:3000/ensevents/`. The response is a JSON array of three event objects. Each object contains a `method` (all are "AuctionStarted"), a `utsts` (all are "Fri, 25 May 2018 06:39:52 GMT"), and a `value` object with a `hash` and a `registrationDate` (all are "1527662357").

```
[{"method": "AuctionStarted", "utsts": "Fri, 25 May 2018 06:39:52 GMT", "value": {"hash": "0x293d4600b4e2fd6367d251d3d799cf00a4771b3722a6717e9557d9541c34514d", "registrationDate": "1527662357"}}, {"method": "AuctionStarted", "utsts": "Fri, 25 May 2018 06:39:52 GMT", "value": {"hash": "0x1aff38e975d8afdb7cdf61273f0a0693730e7aa171bc7098dc201ccbf2722409", "registrationDate": "1527662357"}}, {"method": "AuctionStarted", "utsts": "Fri, 25 May 2018 06:39:52 GMT", "value": {"hash": "0x643bae31976e2007eed844f0dad2ec77c0c0c33891e41d179b649b90176bfc", "registrationDate": "1527662357"}}]
```

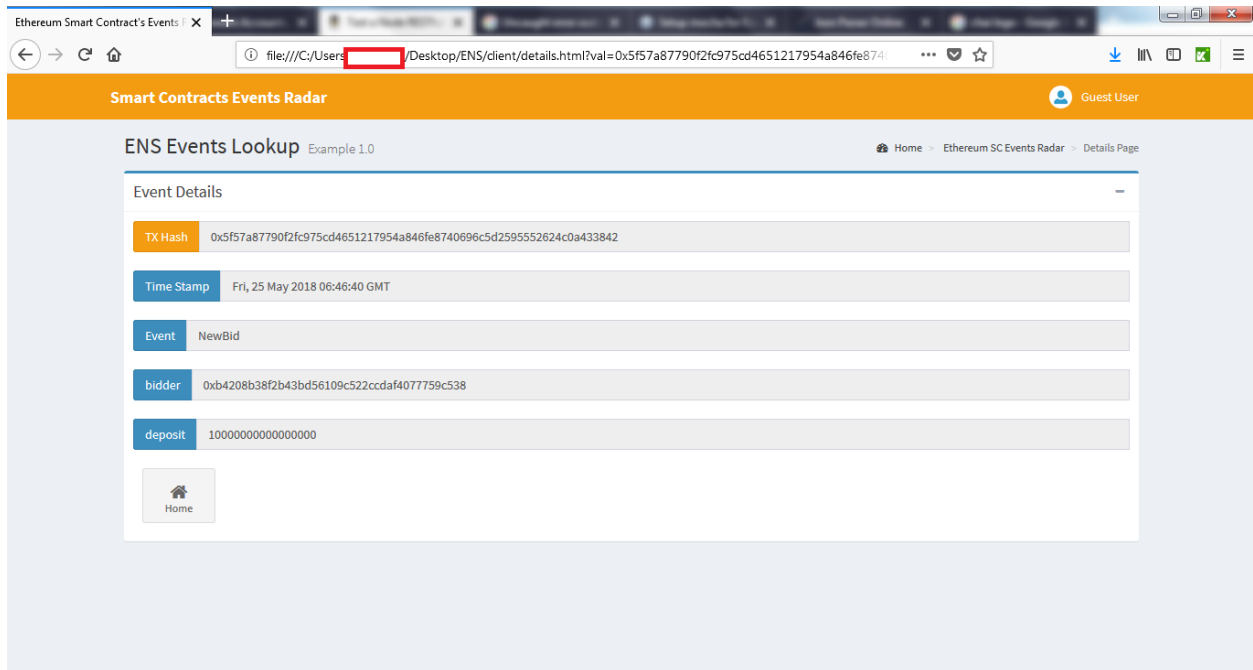
Once server is started and restful services are up and running after that client could be launched in any web browser. It could be hosted on any webserver or could be launched directly in web browser. The client uses web3.js for lookup events, if any event occurs it sends data to server endpoint and also re renders its data table.

The screenshot shows the "Ethereum Smart Contracts Events Radar" web application. The page title is "ENS Events Lookup Example 1.0". It features a "Data Table With Full Features" with a search bar and a table of events. The table has columns for Tx Hash, Time Stamp, Event, and Event Log. The events listed are AuctionStarted and NewBid.

Tx Hash	Time Stamp	Event	Event Log
0x1aff38e975d8af...	Fri, 25 May 2018 06:39:52 GMT	AuctionStarted	hash: 0x1aff38e975d8afdb7cdf61273f0a0693730e7aa171bc7098dc201ccbf2722409 registrationDate: 1527662357
0x293d4600b4e2fd...	Fri, 25 May 2018 06:39:52 GMT	AuctionStarted	hash: 0x293d4600b4e2fd6367d251d3d799cf00a4771b3722a6717e9557d9541c34514d registrationDate: 1527662357
0x2be0e9c90d5596...	Fri, 25 May 2018 06:50:26 GMT	AuctionStarted	hash: 0x2be0e9c90d55969a76530494111bf59625f2c5ae6a256fe026456a30d6195084 registrationDate: 1527662999
0x5f57a87790f2fc...	Fri, 25 May 2018 06:46:40 GMT	NewBid	hash: 0x5f57a87790f2fc975cd4651217954a846fe8740696c5d2595552624c0a433842 bidder: 0xb4208b38f2b43bd56109c522ccdaf4077759c538 deposit: 10000000000000000
0x5f57a87790f2fc...	Fri, 25 May 2018 06:46:41 GMT	NewBid	hash: 0x5f57a87790f2fc975cd4651217954a846fe8740696c5d2595552624c0a433842 bidder: 0xb4208b38f2b43bd56109c522ccdaf4077759c538 deposit: 10000000000000000
0x643bae31976e20...	Fri, 25 May 2018 06:39:52 GMT	AuctionStarted	hash: 0x643bae31976e2007eed844f0dad2ec77c0c0c33891e41d179b649b90176bfc registrationDate: 1527662357
0x7286bf299e45e0...	Fri, 25 May 2018 06:52:07 GMT	AuctionStarted	hash: 0x7286bf299e45e021d2a7805c65556b3e5491a7c4475bcc209c46a2d8cdc1c062

Ethereum Smart Contracts Event Radar

The details screen:



Possible Improvements

- Creating a standalone library for events lookup and moving that code to server side so no event is missed.
- Host client and server code on same root path so there is no need of CORS (Cross Origin Resource Sharing) permission in server.js
- Instead of using val in query sting param use data inside post