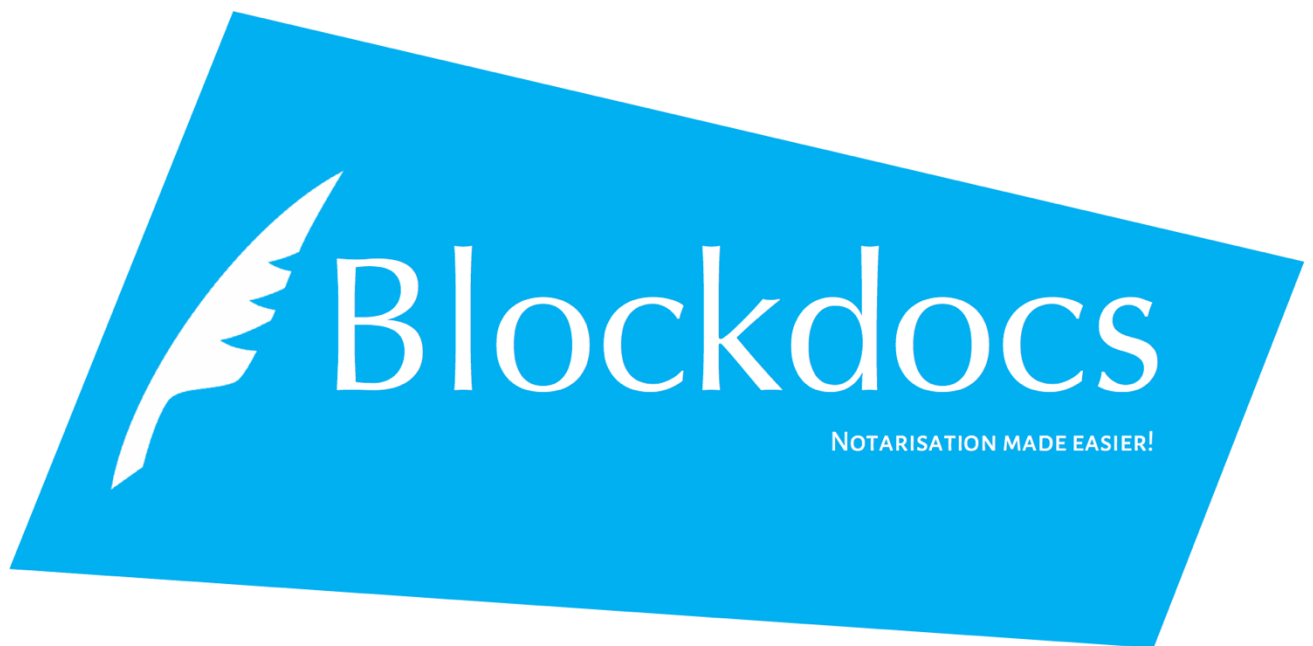




BLOCKDOCS SYSTEM MANUAL



Team 16: Sadir Abdul Hadi – Kristelle Feghali – Alexandru Chiriac

Client

Atos

I - Setting the project up

1. Components

To make the project functional, 3 components need to be installed:

a. Truffle

Installation instructions:

https://truffle.readthedocs.io/en/latest/getting_started/installation/

b. TestRPC

Installation instructions:

<https://github.com/ethereumjs/testrpc>

c. Meteor

Installation instructions:

<https://www.meteor.com/install>

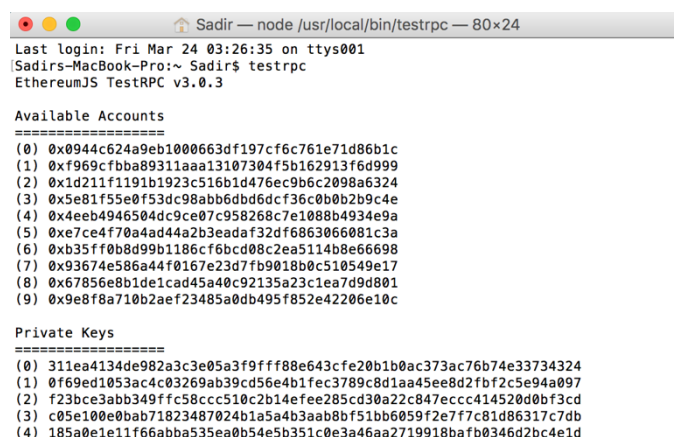
To give a small overview of what each component does:

- TestRPC is similar to an ethereum testnet, where transactions can take place and smart contracts can be deployed.
- Truffle compiles and deploys the smart contracts on testRPC
- Meteor runs the website server.

2. Running the project

a. Setting testRPC up

After installing testRPC, run the `testrpc` command on terminal.

A terminal window titled 'Sadir — node /usr/local/bin/testrpc — 80x24'. The terminal shows the command 'testrpc' being executed, which outputs 'EthereumJS TestRPC v3.0.3'. Below this, it lists 'Available Accounts' with 10 hexadecimal addresses and 'Private Keys' with 10 corresponding private keys.

```
Sadir — node /usr/local/bin/testrpc — 80x24
Last login: Fri Mar 24 03:26:35 on ttys001
Sadirs-MacBook-Pro:~ Sadir$ testrpc
EthereumJS TestRPC v3.0.3

Available Accounts
=====
(0) 0x0944c624a9eb1000663df197cf6c761e71d86b1c
(1) 0xf969cfbba89311aaa13107304f5b162913f6d999
(2) 0x1d211f1191b1923c516b1d476ec9b6c2098a6324
(3) 0x5e81f55e0f53dc98abb6dbd6dcf36c0b0b2b9c4e
(4) 0x4eeb4946504dc9ce07c958268c7e1088b4934e9a
(5) 0xe7ce4f70a4ad44a2b3eadaf32df6863066081c3a
(6) 0xb35ff0b8d99b1186cf6bcd08c2ea5114b8e66698
(7) 0x93674e586a44f0167e23d7fb9018b0c510549e17
(8) 0x67856e8b1de1cad45a40c92135a23c1ea7d9d801
(9) 0x9e8f8a710b2aef23485a0db495f852e42206e10c

Private Keys
=====
(0) 311ea4134de982a3c3e05a3f9fff88e643cfe20b1b0ac373ac76b74e33734324
(1) 0f69ed1053ac4c03269ab39cd56e4b1fec3789c8d1aa45ee8d2fbf2c5e94a097
(2) f23bce3abb349ffc58ccc510c2b14efee285cd30a22c847eccc414520d0bf3cd
(3) c05e100e0bab71823487024b1a5a4b3aab8bf51bb6059f2e7f7c81d86317c7db
(4) 185a0e1e11f66abba535ea0b54e5b351c0e3a46aa2719918bafb0346d2bc4e1d
```

You should get "Available Accounts" and "Private Keys printed"

b. Deploying the smart contracts

On another terminal, go to the "Truffle Part" directory.

Write the following on terminal:

```
truffle compile
truffle build
truffle migrate
truffle console
x = BasicSign.deployed()
JSON.stringify(x.abi) (Copy and paste the value somewhere. It's the smart contract's abi)
x.address (Copy and paste the value somewhere. It's the smart contract's address)
```

```
[truffle(default)> JSON.stringify(x.abi)
' [{"constant":false,"inputs":[{"name":"nonce","type":"string"}],"name":"generate
Id","outputs":[{"name":"","type":"bytes32"}],"payable":false,"type":"function"},
{"constant":true,"inputs":[{"name":"","type":"bytes32"}],"name":"documents","out
puts":[{"name":"organizer","type":"address"}],"payable":false,"type":"function"}
,{"constant":false,"inputs":[],"name":"SimpleSign","outputs":[],"payable":false,
"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}
,{"name":"signId","type":"uint8"}],"name":"getSignDetails","outputs":[{"name":"","
"type":"address"}],"payable":false,"type":"function"}, {"constant":false,"inputs
": [{"name":"docId","type":"bytes32"}],"name":"addSignature","outputs":[],"payabl
e":true,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"b
ytes32"}],"name":"getDocumentDetails","outputs":[{"name":"organizer","type":"add
ress"}, {"name":"count","type":"uint256"}],"payable":false,"type":"function"}, {"c
onstant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"getDocumentO
rganizer","outputs":[{"name":"organizer","type":"address"}, {"name":"count","type
":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"n
ame":"docId","type":"bytes32"}, {"name":"index","type":"uint256"}],"name":"getDoc
umentSignature","outputs":[{"name":"value","type":"address"}],"payable":false,"t
ype":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],
"name":"removeDocument","outputs":[],"payable":false,"type":"function"}, {"consta
nt":false,"inputs":[{"name":"dochash","type":"bytes32"}],"name":"createDocument"
,"outputs":[{"name":"docId","type":"bytes32"}],"payable":true,"type":"function"}
,{"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"getSigns
Count","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function
"}, {"payable":false,"type":"fallback"}, {"anonymous":false,"inputs":[{"indexed":t
rue,"name":"from","type":"address"}, {"indexed":false,"name":"id","type":"bytes32
"}],"name":"Created","type":"event"}, {"anonymous":false,"inputs":[{"indexed":tru
e,"name":"from","type":"address"}, {"indexed":false,"name":"docId","type":"bytes3
2"}],"name":"Signed","type":"event"} ]'
[truffle(default)> x.address
'0x7aedd8f80b9f45e541cbf5085b86f4ab7e20246b'
```

c. Running the app

Open the code saved in BlockdocsApp directory, and modify all the occurrences of PCabi and PCaddress to the abid and the address you got earlier. (The occurrences are in the files located at BlockdocsApp/imports/ui/pages)

The lines that should be modified look like:

```
var PCabi = [{"constant":false,"inputs":[{"name":"nonce","type":"string"}],"name":"generateId","outputs":[{"name":"","type":"bytes32"}],"payable":false,"type":"function"}, {"constant":true,"inputs":[{"name":"","type":"bytes32"}],"name":"documents","outputs":[{"name":"organizer","type":"address"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[],"name":"SimpleSign","outputs":[],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"signId","type":"uint8"}],"name":"getSignDetails","outputs":[{"name":"","type":"address"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"addSignature","outputs":[],"payable":true,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"getDocumentDetails","outputs":[{"name":"organizer","type":"address"}, {"name":"count","type":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"getDocumentOrganizer","outputs":[{"name":"organizer","type":"address"}, {"name":"count","type":"uint256"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}, {"name":"index","type":"uint256"}],"name":"getDocumentSignature","outputs":[{"name":"value","type":"address"}],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"removeDocument","outputs":[],"payable":false,"type":"function"}, {"constant":false,"inputs":[{"name":"dochash","type":"bytes32"}],"name":"createDocument","outputs":[{"name":"docId","type":"bytes32"}],"payable":true,"type":"function"}, {"constant":false,"inputs":[{"name":"docId","type":"bytes32"}],"name":"getSignsCount","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"}, {"payable":false,"type":"fallback"}, {"anonymous":false,"inputs":[{"indexed":true,"name":"from","type":"address"}, {"indexed":false,"name":"id","type":"bytes32"}],"name":"Created","type":"event"}, {"anonymous":false,"inputs":[{"indexed":true,"name":"from","type":"address"}, {"indexed":false,"name":"docId","type":"bytes32"}],"name":"Signed","type":"event"}];
var PCaddress = '0x7aedd8bf80b9f45e541cbf5085b86f4ab7e20246b';
```

Now, on a new terminal, go to the BlockdocsApp directory, and write Meteor

The app should now be up and running on localhost:3000

II- Deploying the app

To deploy the app on a virtual machine, make use of the following library:
<https://github.com/zodern/meteor-up>

However, there is a little value in doing this before moving to a true ethereum node, and doing this would require installing payment services on the app. This requires the help of an expert.