Blockchain Application based on Scala. Use Case

Web3scla Ethereum integration

accenture technology

David Urdiales Nieto
Juan Manuel García Navarro

Cádiz, 25th October 2018

Agenda

1.- Ethereum Overview

2.- Ethereum Components

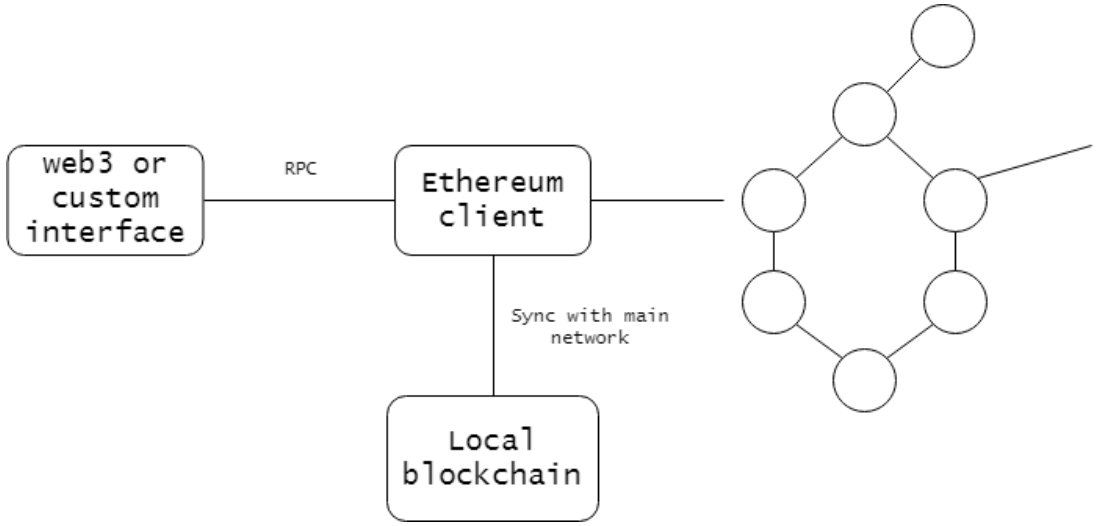
3.- Ethereum Concepts

4.- Demo Console

5.- Demo Scala-Ethereum

1.- Ethereum Overview

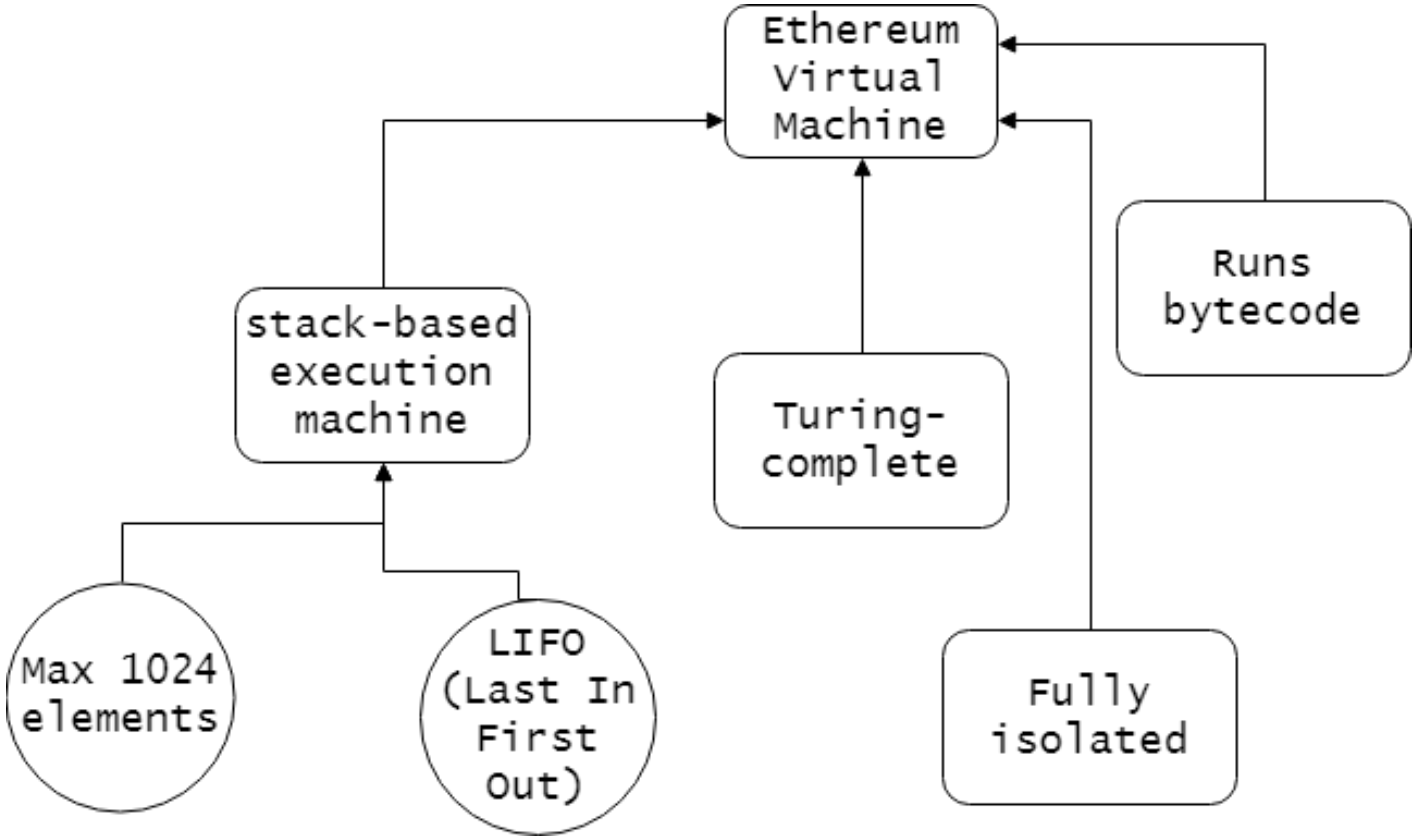
- Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology



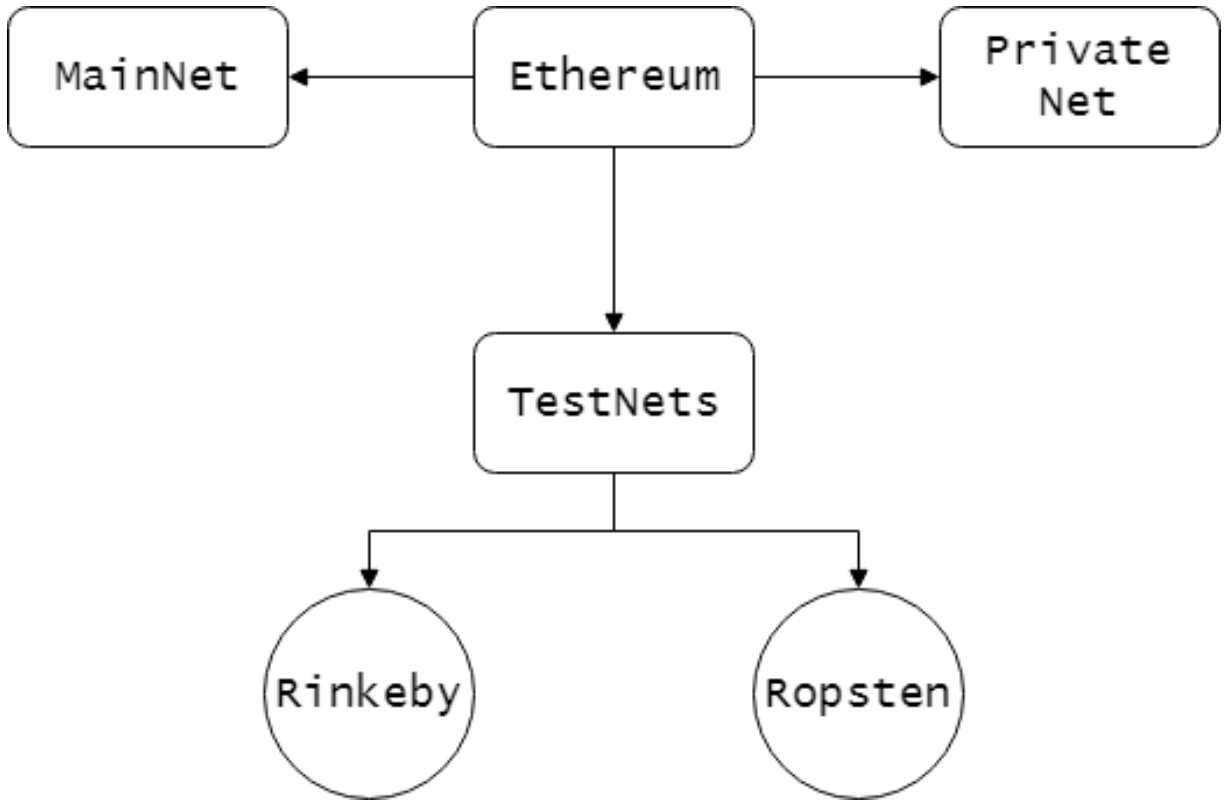
2.- Ethereum Components

- EVM (Ethereum Virtual Machine)
- Networks (MainNet,TestNets, private nets)
- Ether (Cryptocurrency)
- Clients

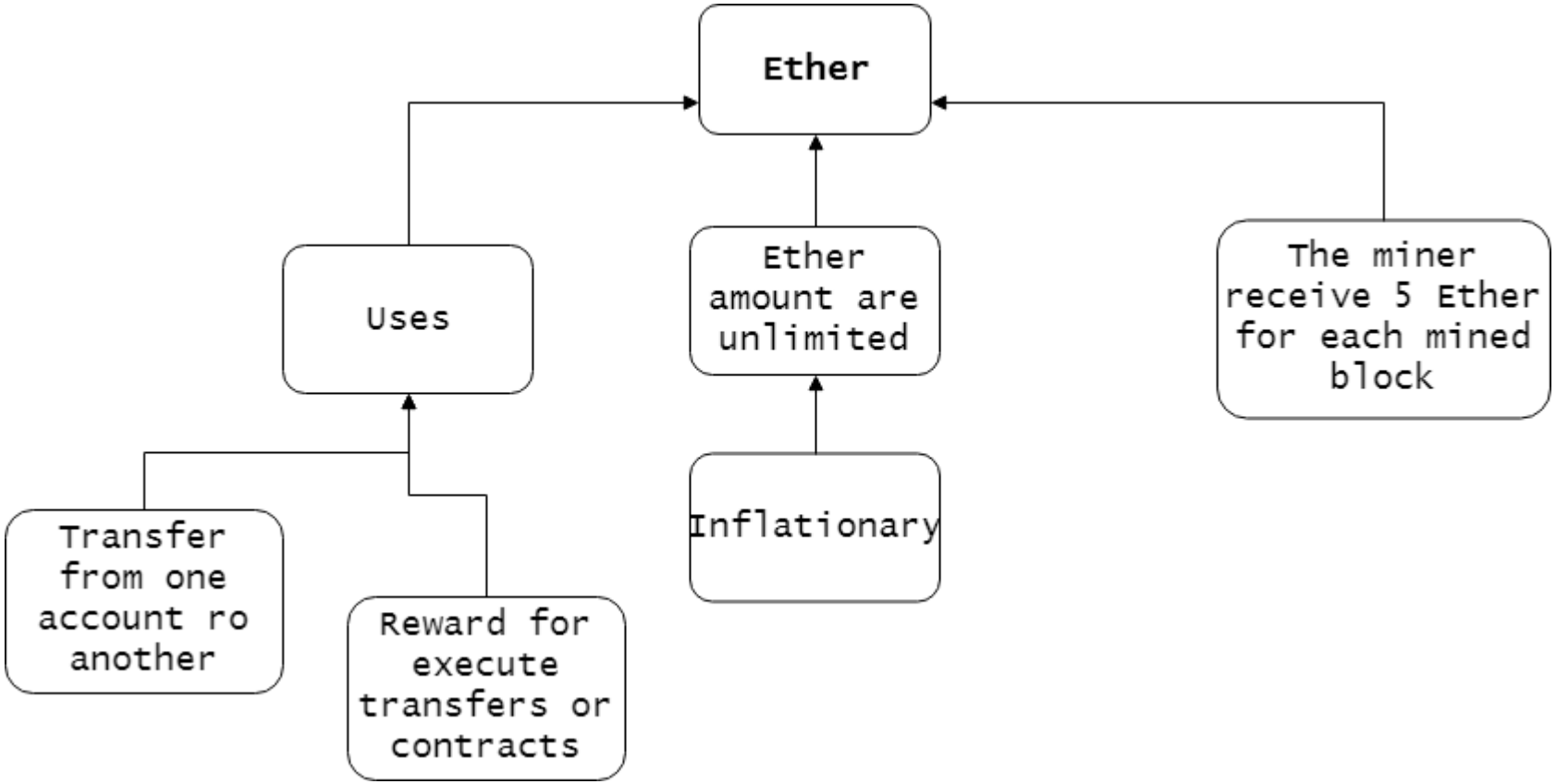
2.- Ethereum Components - EVM



2.- Ethereum Components - Networks



2.- Ethereum Components - Ether



2.- Ethereum Components - Ether

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

2.- Ethereum Components - Clients

- There are multiple Ethereum clients that must implement the **JSON-RPC** protocol. We can interact with blockchain using the methods offered by these clients.

- In this Project, we have used **Go-Ethreum** as a client
- The go-ethereum client is commonly referred to as **geth. Geth** is the command line interface for running a full ethereum node implemented in Go. By installing and running geth, you can take part in the ethereum frontier live network

3.- Ethereum Concepts

- Accounts
- Miners
- Gas
- Smart Contracts (Solidity)
- dApps (descentralized applications)

3.- Ethereum Concepts - Accounts

There are two types of accounts

Externally Owned Account (EOA)

Features:

- Address
- It is controlled public-private key
- Has an ether balance
- It is allowed to send and receive ether

Contract account(Wallet account)

Features:

- Address
- Hasn’t private key
- Has an ether balance
- Manages and executes code and storage

3.- Ethereum Concepts - Miners

- Mining nodes are those nodes that are in charge of verifying the blocks of transactions that are carried out in the network by means of the working test (PoW).
- Rewards:
  - 5 ether for each mined block for winner.
  - Cost of the gas expended within the block.
  - An extra reward for including Uncles as part of the block, in the form of an extra 1/32 per Uncle included.

3.- Ethereum Concepts - Gas

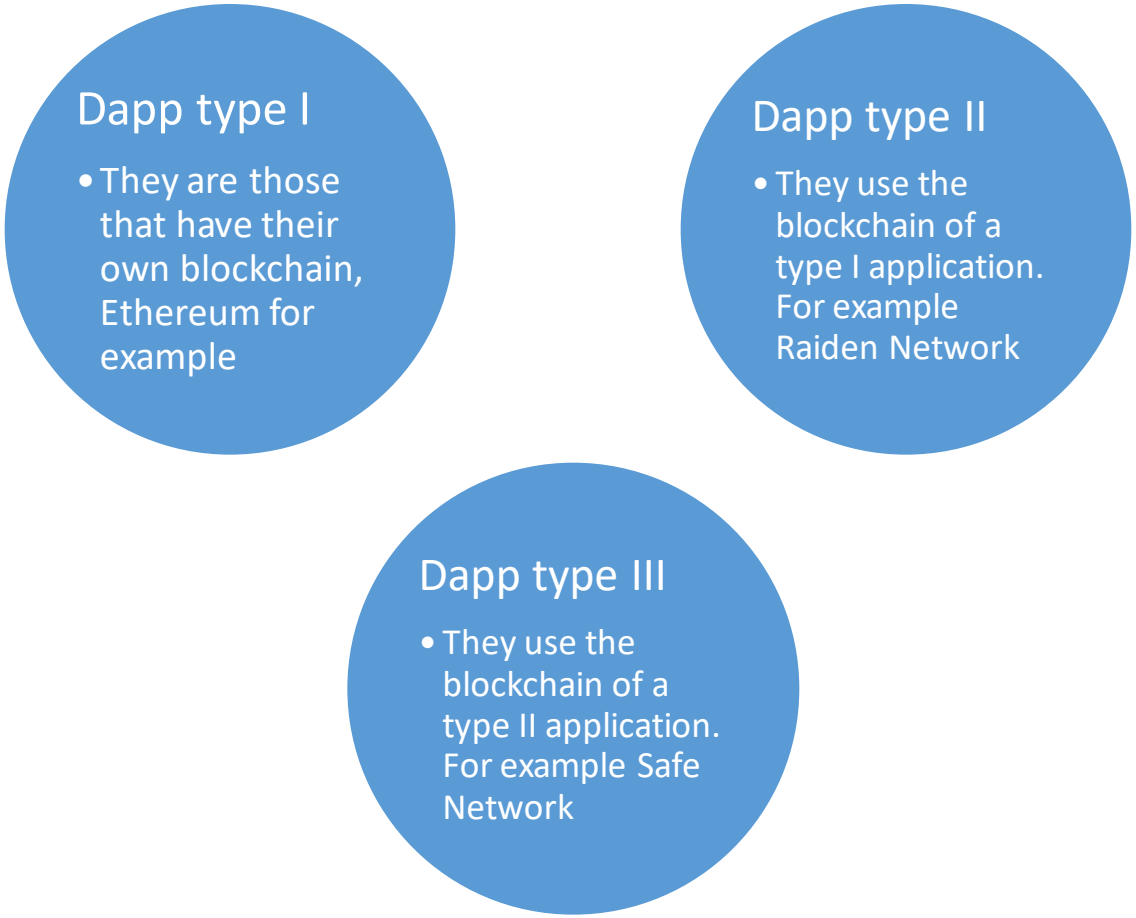
- We can think of gas as the cost of carrying out an operation on the Ethereum network.
- Features:
  - It is proportional to the calculus complexity.
  - Optimized contracts. The better we write contracts the less gas they will consume.
  - The gas ensures use correct of the ethereum network.

3.- Ethereum Concepts – Smart Contracts

- These are programs that are executed when certain conditions are met and execute particular clauses.
- Features:
  - They are programmed in the Solidity language.
  - They are "turing-complete", so it is possible to implement any program.
  - The execution of a smart contract consume gas (ether).

3.- Ethereum Concepts – Dapp

- Decentralized applications (Dapp)
- Features:
  - Decentralized. The application must operate with the support on the users, not on a centralized organization
  - Open source
  - BlockChain technology



Demo console

4.- Demo console

1. Open command line console
2. Create a folder on your computer (lambdaEther), move into
3. Create a subfolder (ethnet) in lambdaEther.
4. Create an account to be used as the node's primary account.

   geth account new –datadir ethnet

```
C:\Users\j.garcia.navarro\ethereum\redesPrivadas\lambdaPoC>geth account new --datadir ethnet
INFO [10-17|19:23:49.635] Maximum peer count                       ETH=25 LES=0 total=25
Your new account is locked with a password. Please give a password. Do not forget this password.
Passphrase:
Repeat passphrase:
Address: {1ba9eaf66bb3c9861d8f478fd7c62c38e5e10e74}
```

4.- Demo console

5. Write the genesis.json file and save it in the first folder (ethnget account new –datadir lambdaEtheret). In the <account> field put the account generated before and the fileld <id> put a Integer

```
{
    "coinbase": "0x0000000000000000000000000000000000000001",
    "difficulty": "0x20",
    "extraData": "",
    "gasLimit": "0x8000000",
    "nonce": "0x0000000000000113",
    "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "timestamp": "0x00",
    "alloc": {
        "<account>": { "balance" : "5000" }
    },
    "config": {
        "chainId": <id>,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    }
}
```

4.- Demo console

6. Init network. Exectute **geth --datadir <folderName> init <genesis.json>**

```
C:\Users\j.garcia.navarro\ethereum\redesPrivadas\lambdaPoC>geth --datadir net init genesis-lambda.json
INFO [10-06|13:39:06.801] Maximum peer count                       ETH=25 LES=0 total=25
INFO [10-06|13:39:06.828] Allocated cache and file handles         database=C:\\Users\\j.garcia.navarro
\\ethereum\\redesPrivadas\\lambdaPoC\\net\\geth\\chaindata cache=16 handles=16
INFO [10-06|13:39:06.876] Writing custom genesis block
INFO [10-06|13:39:06.882] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0
 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-06|13:39:06.898] Successfully wrote genesis state         database=chaindata
                                                                 hash=b47b8a…cd72f4
INFO [10-06|13:39:06.913] Allocated cache and file handles         database=C:\\Users\\j.garcia.navarro
\\ethereum\\redesPrivadas\\lambdaPoC\\net\\geth\\lightchaindata cache=16 handles=16
INFO [10-06|13:39:06.951] Writing custom genesis block
INFO [10-06|13:39:06.955] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0
 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-06|13:39:06.971] Successfully wrote genesis state         database=lightchaindata
                                                                 hash=b47b8a…cd72f4
```

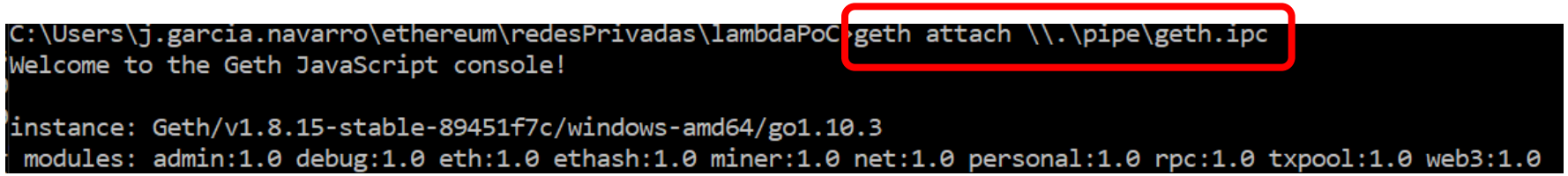
4.- Demo console

7. Execute **geth --datadir <folderName> --rpc –networkid <id> --mine --unlock “<account>”**. Insert password

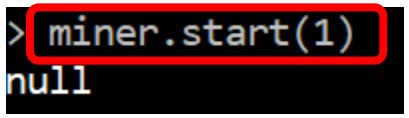
```
C:\Users\j.garcia.navarro\ethereum\redesPrivadas\lambdaPoC>geth --datadir net --rpc --networkid 2018 --mine
INFO [10-06|13:50:19.125] Maximum peer count                       ETH=25 LES=0 total=25
INFO [10-06|13:50:19.156] Starting peer-to-peer node               instance=Geth/v1.8.15-stable-89451f7c/windows-amd
64/go1.10.3
INFO [10-06|13:50:19.171] Allocated cache and file handles         database=C:\\Users\\j.garcia.navarro\\ethereum\\r
edesPrivadas\\lambdaPoC\\net\\geth\\chaindata cache=768 handles=1024
INFO [10-06|13:50:19.238] Initialised chain configuration          config="{ChainID: 2018 Homestead: 0 DAO: <nil> DA
OSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil> Engine: unknown}"
INFO [10-06|13:50:19.262] Disk storage enabled for ethash caches   dir=C:\\Users\\j.garcia.navarro\\ethereum\\redesP
rivadas\\lambdaPoC\\net\\geth\\ethash count=3
INFO [10-06|13:50:19.287] Disk storage enabled for ethash DAGs     dir=C:\\Users\\j.garcia.navarro\\AppData\\Ethash
                                   count=2
INFO [10-06|13:50:19.307] Initialising Ethereum protocol           versions="[63 62]" network=2018
INFO [10-06|13:50:19.323] Loaded most recent local header          number=0 hash=b47b8a…cd72f4 td=32
INFO [10-06|13:50:19.336] Loaded most recent local full block      number=0 hash=b47b8a…cd72f4 td=32
INFO [10-06|13:50:19.350] Loaded most recent local fast block      number=0 hash=b47b8a…cd72f4 td=32
INFO [10-06|13:50:19.361] Loaded local transaction journal         transactions=0 dropped=0
INFO [10-06|13:50:19.385] Regenerated local transaction journal    transactions=0 accounts=0
INFO [10-06|13:50:19.394] Starting P2P networking
INFO [10-06|13:50:21.556] UDP listener up                          self=enode://1cfbab9aedc22971a480b451fbc9fcc84721
dd35edcc1bfd43de18b620f5af3c5437cd7e3b85a59f28332e5019baf703be8ccff48f3fc50a97d86dc569bc14e7@[::]:30303
INFO [10-06|13:50:21.582] RLPx listener up                         self=enode://1cfbab9aedc22971a480b451fbc9fcc84721
dd35edcc1bfd43de18b620f5af3c5437cd7e3b85a59f28332e5019baf703be8ccff48f3fc50a97d86dc569bc14e7@[::]:30303
INFO [10-06|13:50:21.587] IPC endpoint opened                      url=\\\\.\\pipe\\geth.ipc
INFO [10-06|13:50:21.624] HTTP endpoint opened                     url=http://127.0.0.1:8545 cors= vhosts=localhost
INFO [10-06|13:50:21.641] Transaction pool price threshold updated price=1000000000
INFO [10-06|13:50:21.650] Updated mining threads                   threads=0
INFO [10-06|13:50:21.659] Transaction pool price threshold updated price=1000000000
INFO [10-06|13:50:21.668] Etherbase automatically configured       address=0x5ee2f5012Fdf6F5F73508De833C2325bf8326C1
5
INFO [10-06|13:50:21.683] Commit new mining work                   number=1 sealhash=033f5a…4f3489 uncles=0 txs=0 ga
s=0 fees=0 elapsed=0s
```

4.- Demo console

9. Open another console
10. Execute **geth attach \\.\pipe\geth.ipc** to connect with geth



11. start mining:



4.- Demo console

12. Execute personal.newAccount(<“password”>) to create an account

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x5ee2f5012fdf6f5f73508de833c2325bf8326c15"
```

13. Get the list of accounts eth.accounts or personal.listAccounts

```
> personal.listAccounts
["0x5ee2f5012fdf6f5f73508de833c2325bf8326c15", "0x6ac147e6a2a706fe4ed43cba00ca2564524737b9"]
> eth.accounts
["0x5ee2f5012fdf6f5f73508de833c2325bf8326c15", "0x6ac147e6a2a706fe4ed43cba00ca2564524737b9"]
```

4.- Demo console

14. Send transaction between accounts

personal.sendTransaction({from:"<sourceAccount>",to:"<DestinationAccount>",value:"<wei amount>"},"<password>") if the password is empty write “”

```
> personal.sendTransaction({from:eth.accounts[0],to:eth.accounts[1],value:"50000"},"")
"0xdd846d959d940c9934c0bc192e4743607316ff3462645274e3ba11f4a6336eb5"
```

```
> eth.getBalance(eth.accounts[0])
1550000000000000000000
```

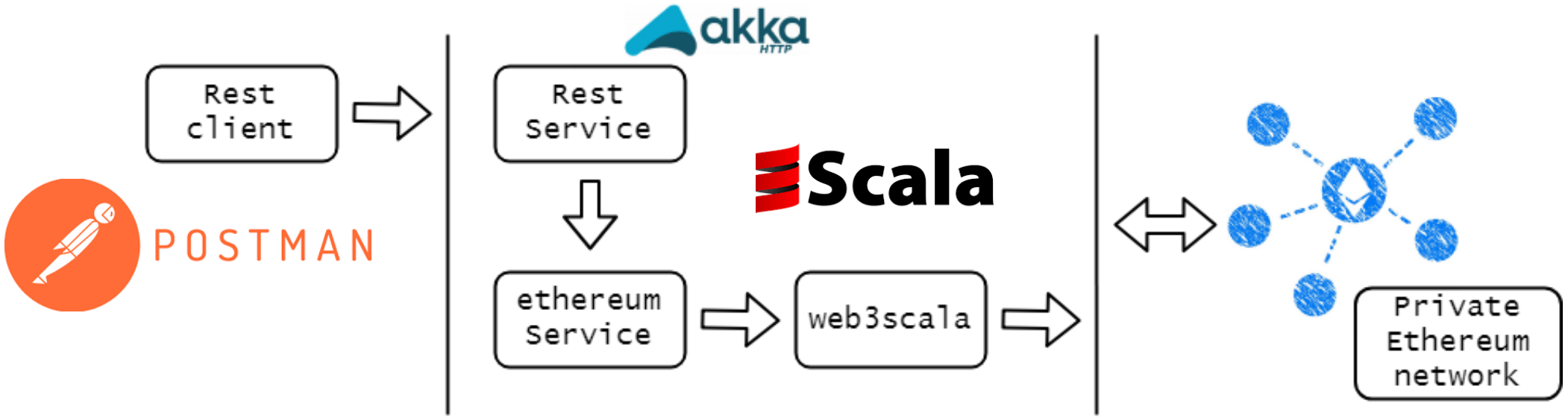
```
> eth.getBalance(eth.accounts[1])
50000
```

Scala demo

5.- Workshop Web3Scala-Ethereum-integration

- Web3Scala-Ethereum-integration
  - Overview
  - Components
    - web3scala
    - Rest service implemented with akka 
    - ADT modeling response
- Project Structure
- Running Example

5.- Workshop web3Scala-Ethereum-Integration overview



Web3scala Ethereum integration

5.- web3scala Concepts

- web3scala is a library that allows integration with Ethereum using the scala language
- Features
  - Complete implementation of JSON-RPC Ethereum client API over HTTP
- Dependencies
  - Dispatch Reboot for asynchronous HTTP interaction
  - Json4s-Jackson for JSON parsing/generation
  - jackson-module-scala to support Scala-specific datatypes

5.- Workshop web3Scala-Ethereum-Integration Components

- Rest service implements with Akka http

```scala
object Resource extends App {

  implicit val actorSystem = ActorSystem("system")
  implicit val actorMaterializer = ActorMaterializer()

  lazy val route =
    AccountResource.route ~ BlockResource.route

  val port = 9010
  Http().bindAndHandle(route, "localhost", port)

  println(s"server started at $port")
}
```

```scala
object AccountResource extends App {

  implicit val formats: DefaultFormats.type = DefaultFormats

  lazy val route =
    pathPrefix("api") {
      path("accounts") {
        get {
          complete {
            ToJson.toJson(AccountService.getAccounts)
          }
        }
      } ~
```

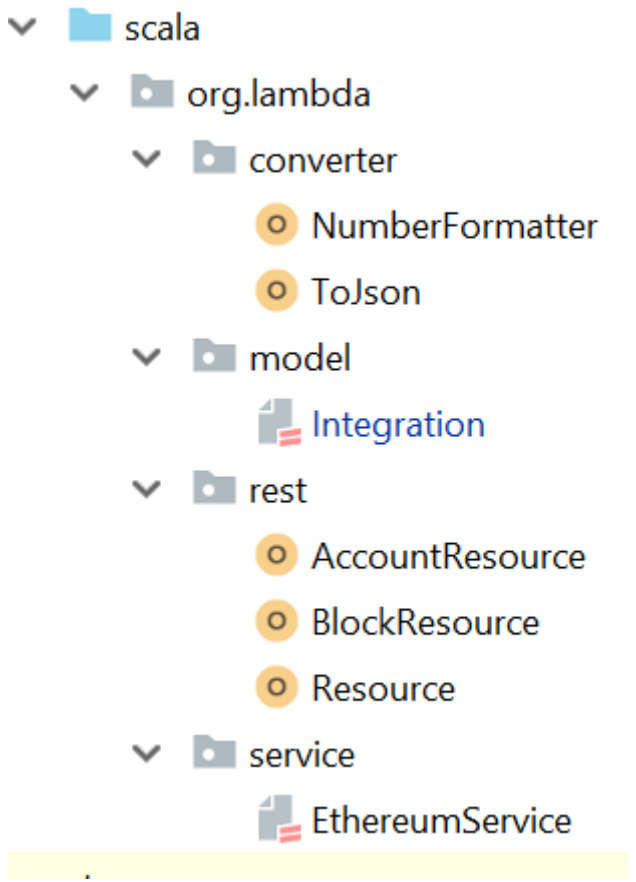
5.- Workshop web3Scala-Ethereum-Integration Components

- Model has implemented with ADT (Request/response)

```
trait IntegrationResponse
case class IntegrationAccounts(accounts:List[String]) extends IntegrationResponse
case class IntegrationBlockNumber(blockNumber:Long) extends IntegrationResponse
case class IntegrationBalance(balances:List[(String, String)]) extends IntegrationResponse
case class IntegrationMinig(mining:Boolean) extends IntegrationResponse
case class IntegrationTransactionCount(transactions:List[(String, String)]) extends IntegrationResponse

trait IntegrationRequest
case class Transaction(from:String,
                       to:String,
                       amount:String) extends IntegrationRequest
```

5.- Workshop Web3Scala-Ethereum-integration Project Structure



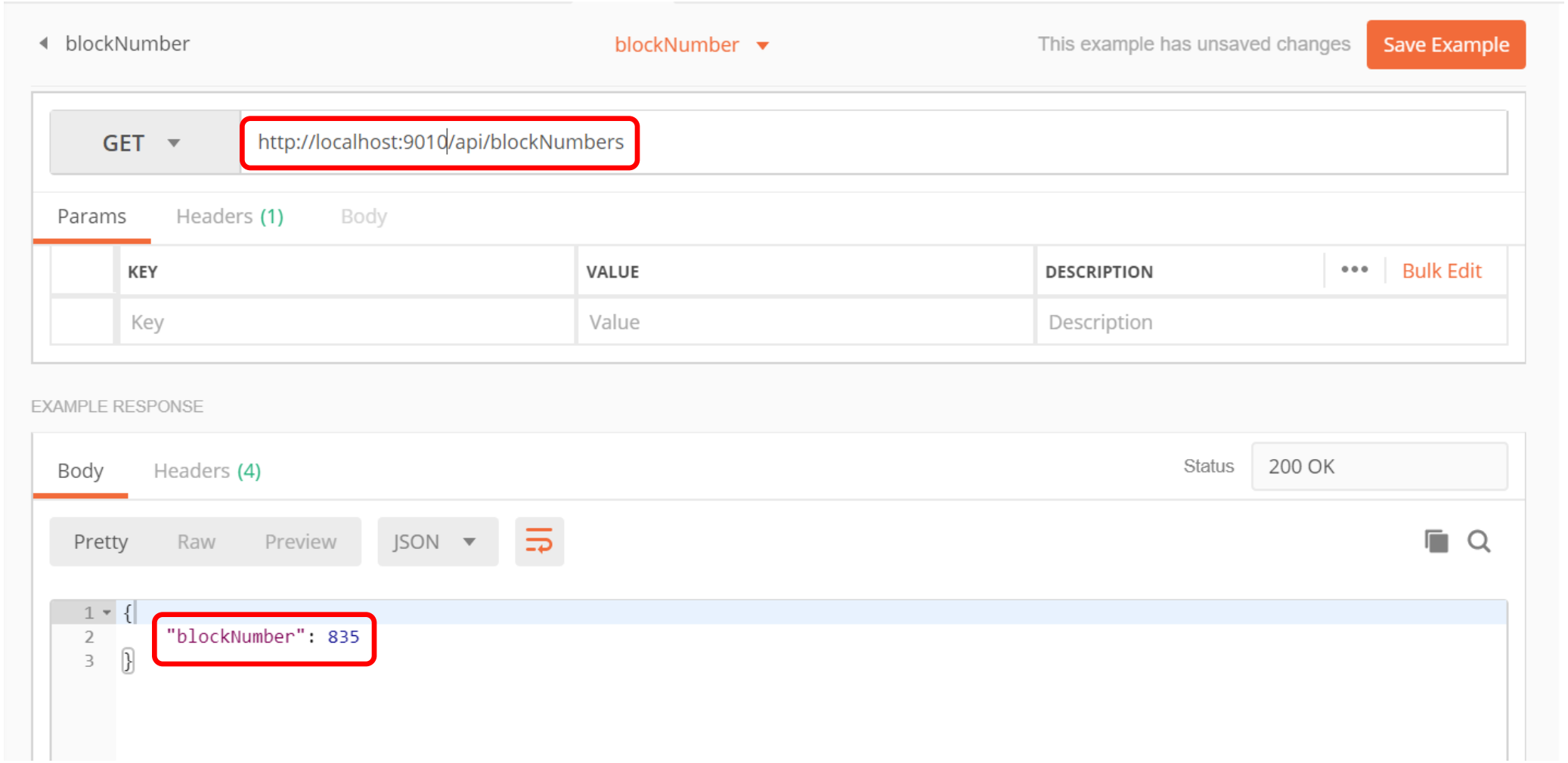
Running Example

- In the project folder, open a console and launch

> sbt run

```
C:\Users\j.garcia.navarro\workspaces\lambda\web3scala-ethereum-integration>sbt clean run
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=256m; support was removed in 8.0
[info] Loading settings for project global-plugins from idea.sbt ...
[info] Loading global plugins from C:\Users\j.garcia.navarro\.sbt\1.0\plugins
[info] Loading project definition from C:\Users\j.garcia.navarro\workspaces\lambda\web3scala-ethereum-integration\projec
t
[info] Loading settings for project web3scala-ethereum-integration from build.sbt ...
[info] Set current project to web3scala-ethereum-integration (in build file:/C:/Users/j.garcia.navarro/workspaces/lambda
/web3scala-ethereum-integration/)
[success] Total time: 0 s, completed 15-oct-2018 19:27:22
[info] Updating ...
[info] Done updating.
[warn] There may be incompatibilities among your library dependencies.
[warn] Run 'evicted' to see detailed eviction warnings
[info] Compiling 8 Scala sources to C:\Users\j.garcia.navarro\workspaces\lambda\web3scala-ethereum-integration\target\sc
ala-2.12\classes ...
[info] Done compiling.
[warn] Multiple main classes detected.  Run 'show discoveredMainClasses' to see the list
[info] Packaging C:\Users\j.garcia.navarro\workspaces\lambda\web3scala-ethereum-integration\target\scala-2.12\web3scala-
ethereum-integration_2.12-0.1.jar ...
[info] Done packaging.
[info] Running org.lambda.rest.Resource
server started at 9010
```

Running Example - Postman



References

- http://ethdocs.org/en/latest/index.html
- https://github.com/web3scala/web3scala
- https://github.com/ethereum/wiki/wiki/JSON-RPC
- https://github.com/ethereum/go-ethereum/wiki/geth

Q& A

Thanks!

We are Hiring!

for contact:

Juan Manuel García Navarro

j.garcia.navarro@accenture.com