ESCREVENDO TESTES UNITÁRIOS DE SMART CONTRACTS EM GO

Júlio Campos

EU

- Servidor Público
- Engenheiro de Software
- Gestor Público

Jimmy Song (송재준)
@jimmysong

Blockchain is a specialized database.
Blockchain is slow and expensive.
Blockchain is difficult to maintain.
Blockchain is expensive.
Blockchain is hard to upgrade.

Blockchain is not a revolutionary technology.
Blockchain won't solve your problems.

♡ 627 2:12 PM - Jun 4, 2019

247 people are talking about this

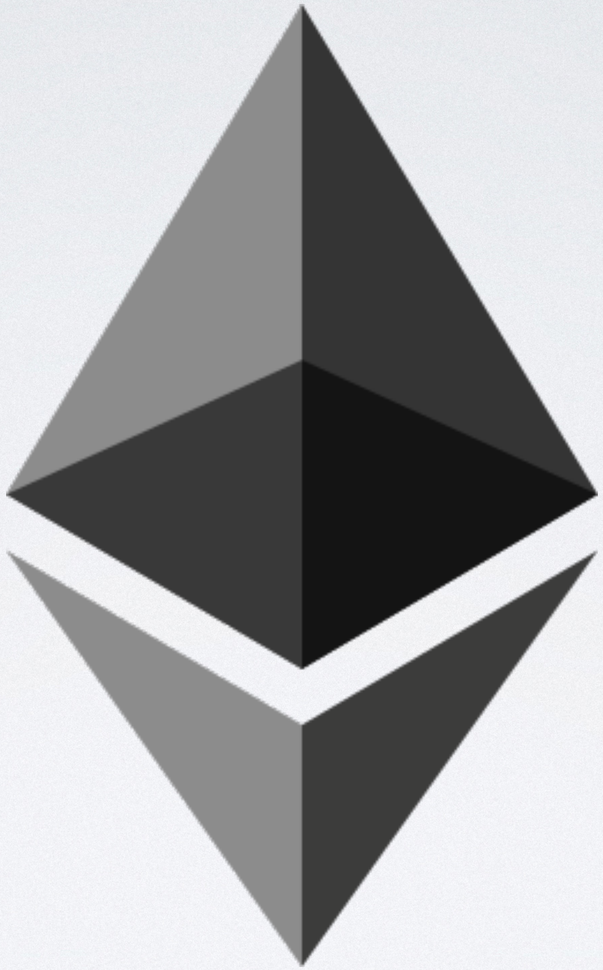
BLOCKCHAIN

BLOCKCHAIN

- Banco de dados
- Sistema distribuído
- Descentralizado

SMART CONTRACTS

- Nick Szabo
- Protocolo entre pessoas desconhecidas



Client	Language	Developers	Latest release
go-ethereum	Go	Ethereum Foundation	go-ethereum-v1.4.18
Parity	Rust	Ethcore	Parity-v1.4.0
cpp-ethereum	C++	Ethereum Foundation	cpp-ethereum-v1.3.0
pyethapp	Python	Ethereum Foundation	pyethapp-v1.5.0
ethereumjs-lib	Javascript	Ethereum Foundation	ethereumjs-lib-v3.0.0
Ethereum(J)	Java	<ether.camp>	ethereumJ-v1.3.1
ruby-ethereum	Ruby	Jan Xie	ruby-ethereum-v0.9.6
ethereumH	Haskell	BlockApps	no Homestead release yet

CLIENTES DE ETHEREUM

```
pragma solidity ^0.4.17;

/** A proposal contract with O(1) approvals. */
contract Proposal {
    mapping (address => bool) approvals;
    bytes32 public approvalMask;
    bytes32 public approver1;
    bytes32 public approver2;
    bytes32 public target;

    function Proposal() public {
        approver1 = 0x0000000000000000000000000000000000000123;
        approver2 = bytes32(msg.sender);
        target = approver1 | approver2;
    }

    function approve(address approver) public {
        approvalMask |= bytes32(approver);
        approvals[approver] = true;
    }

    function isApproved() public constant returns(bool) {
        return approvalMask == target;
    }
}
```

SOLIDITY

TESTES

- Deploy na rede principal
- Deploy em uma rede de testes
- Deploy em uma rede local

TESTES

- Truffle
- Mocha
- Chai
- Ou seja, JavaScript


JS

https://gitlab.com/go-truffle/enhanced-perigord

```go
func getMnemonic() string {
  viper.SetConfigFile("perigord.yaml")
  if err := viper.ReadInConfig(); err != nil {
    log.Fatal()
  }
  mnemonic := viper.GetStringMapString("networks.dev")["mnemonic"]
  return mnemonic
}
```

```go
func getNetworkAddress() string {
  viper.SetConfigFile("perigord.yaml")
  if err := viper.ReadInConfig(); err != nil {
     log.Fatal()
  }
  networkAddr := viper.GetStringMapString("networks.dev")["url"]
  return networkAddr
}
```

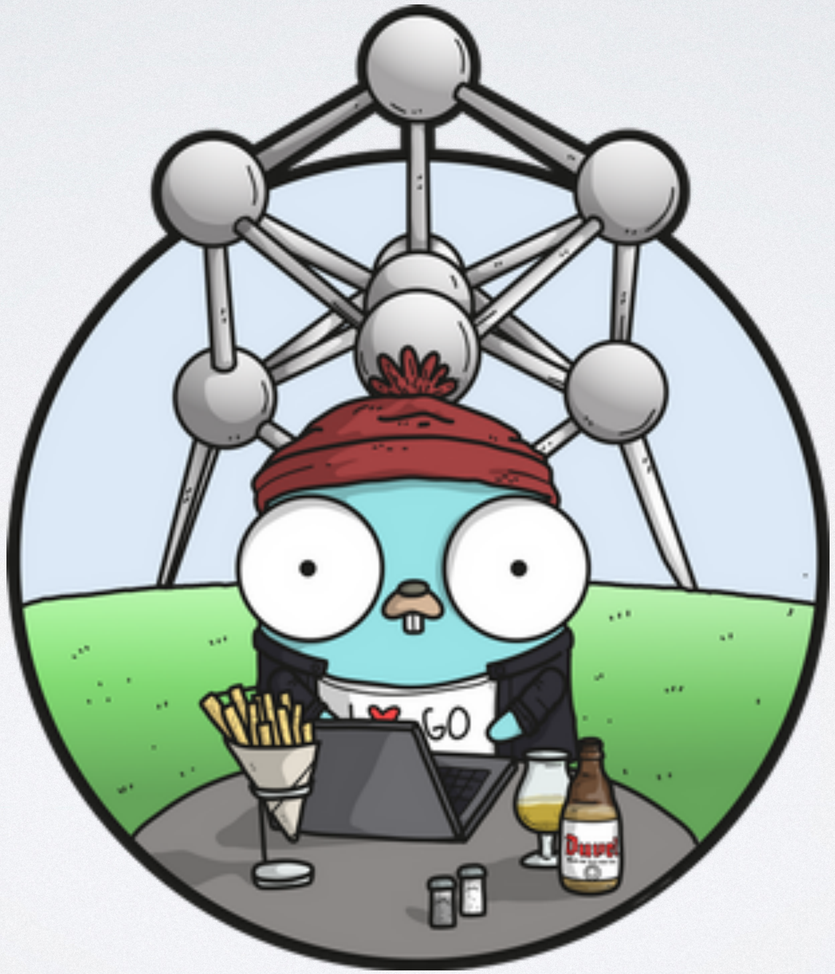
```go
func sendETH(s *MarketSuite, c *ethclient.Client, sender int, receiver common.Address, value *big.Int) {
}
```

Não está completa

```go
func ensureAuth(auth bind.TransactOpts) *bind.TransactOpts {
  return &bind.TransactOpts{
     auth.From,
     auth.Nonce,
     auth.Signer,
     auth.Value,
     auth.GasPrice,
     auth.GasLimit,
     auth.Context}
}

func changeAuth(s MarketSuite, account int) bind.TransactOpts {
  return *s.network.NewTransactor(s.network.Accounts()[account])
}
```

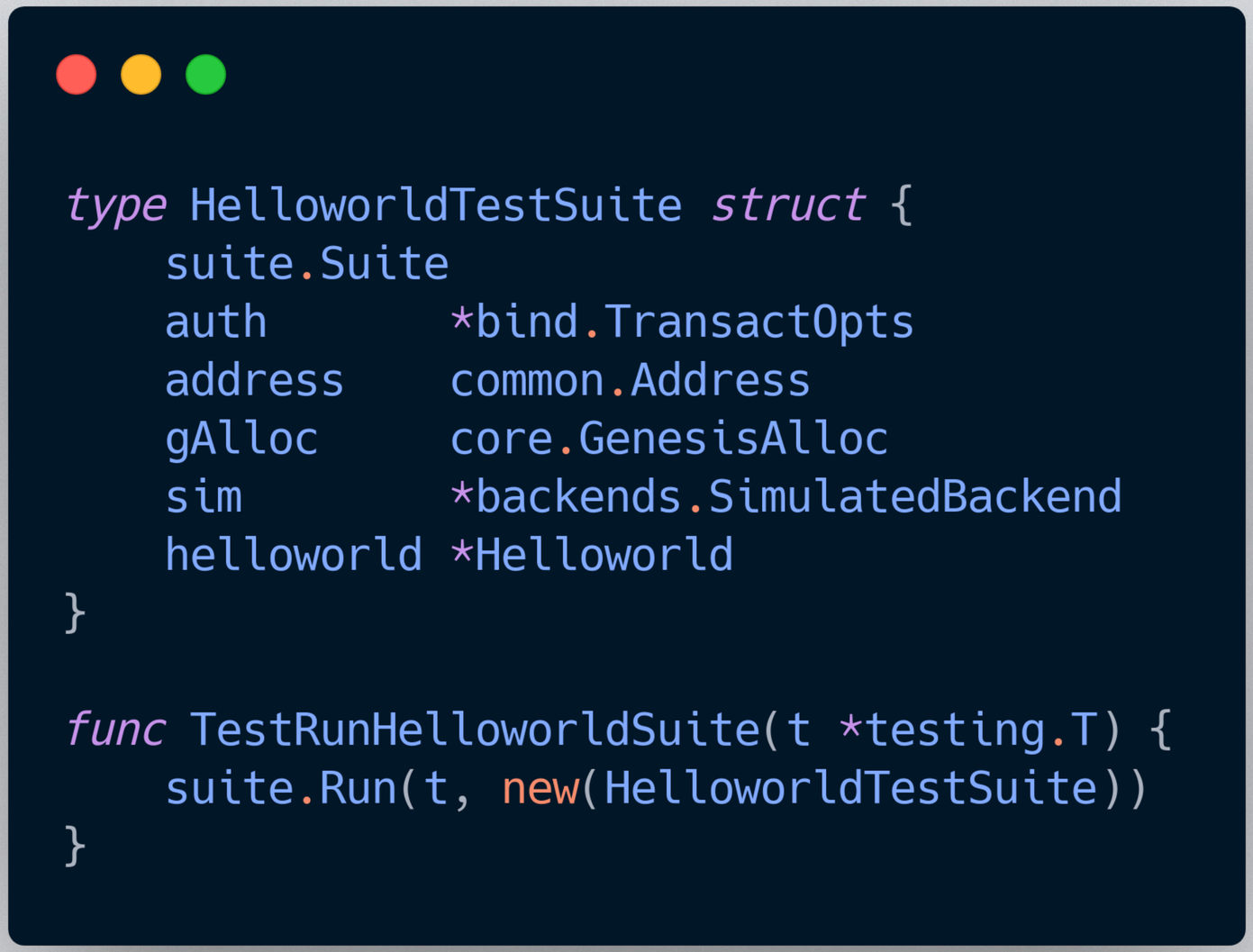
```go
auth = changeAuth(*s, sender) //Change auth fo senderAcc to make a deposit on behalf of the sender

client, _ := ethclient.Dial(getNetworkAddress())

//Let's check the current balance
balance, _ := client.BalanceAt(context.Background(), contract.AddressOf("Market"), nil)
c.Assert(balance.Int64(), Equals, big.NewInt(0).Int64()) //Balance should be 0

//Let's transfer 3 ETH to the contract on behalf of the sender
value := big.NewInt(3000000000000000000) // in wei (3 eth)
contractReceiver := contract.AddressOf("Market")
sendETH(s, client, sender, contractReceiver, value)

balance2, _ := client.BalanceAt(context.Background(), contract.AddressOf("Market"), nil)

c.Assert(balance2.Int64(), Equals, value.Int64()) //Balance should be 3 ETH
```

GO

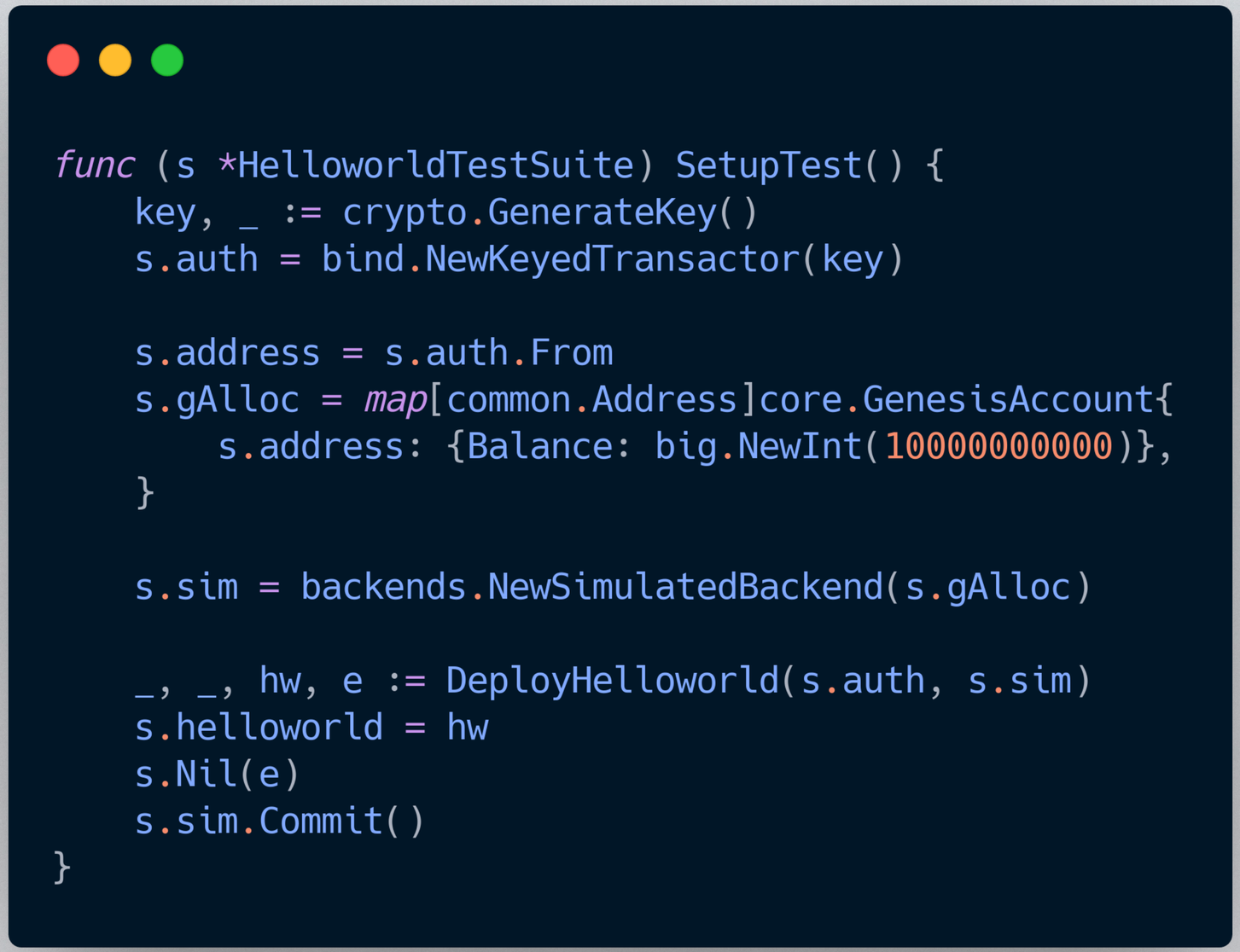


```
pragma solidity ^0.4.23;
contract helloworld {
    function say() public pure returns (string) {
        return 'hello etherworld';
    }
}
```

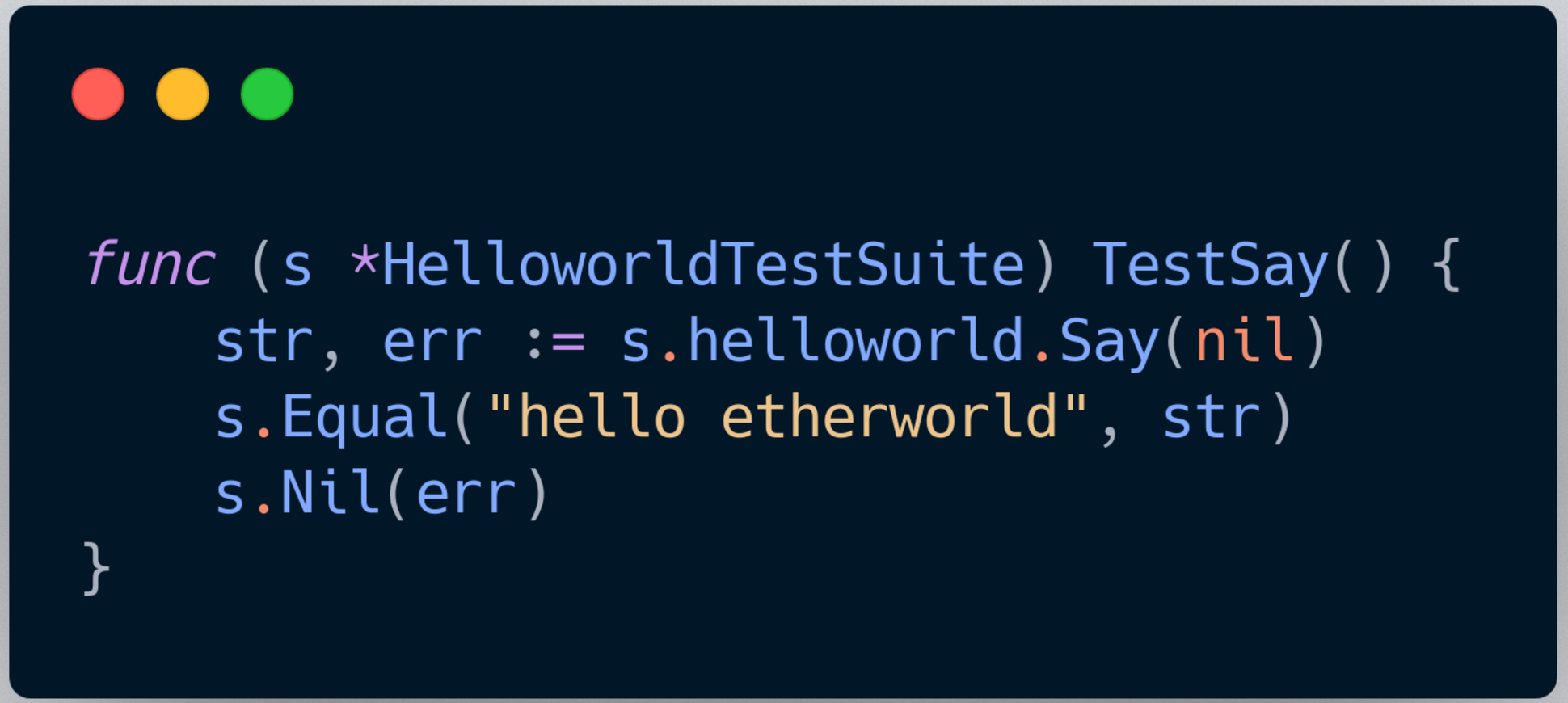
```
import (
    "math/big"
    "testing"

    "github.com/ethereum/go-ethereum/accounts/abi/bind"
    "github.com/ethereum/go-ethereum/accounts/abi/bind/backends"
    "github.com/ethereum/go-ethereum/common"
    "github.com/ethereum/go-ethereum/core"
    "github.com/ethereum/go-ethereum/crypto"
    "github.com/stretchr/testify/suite"
)
```



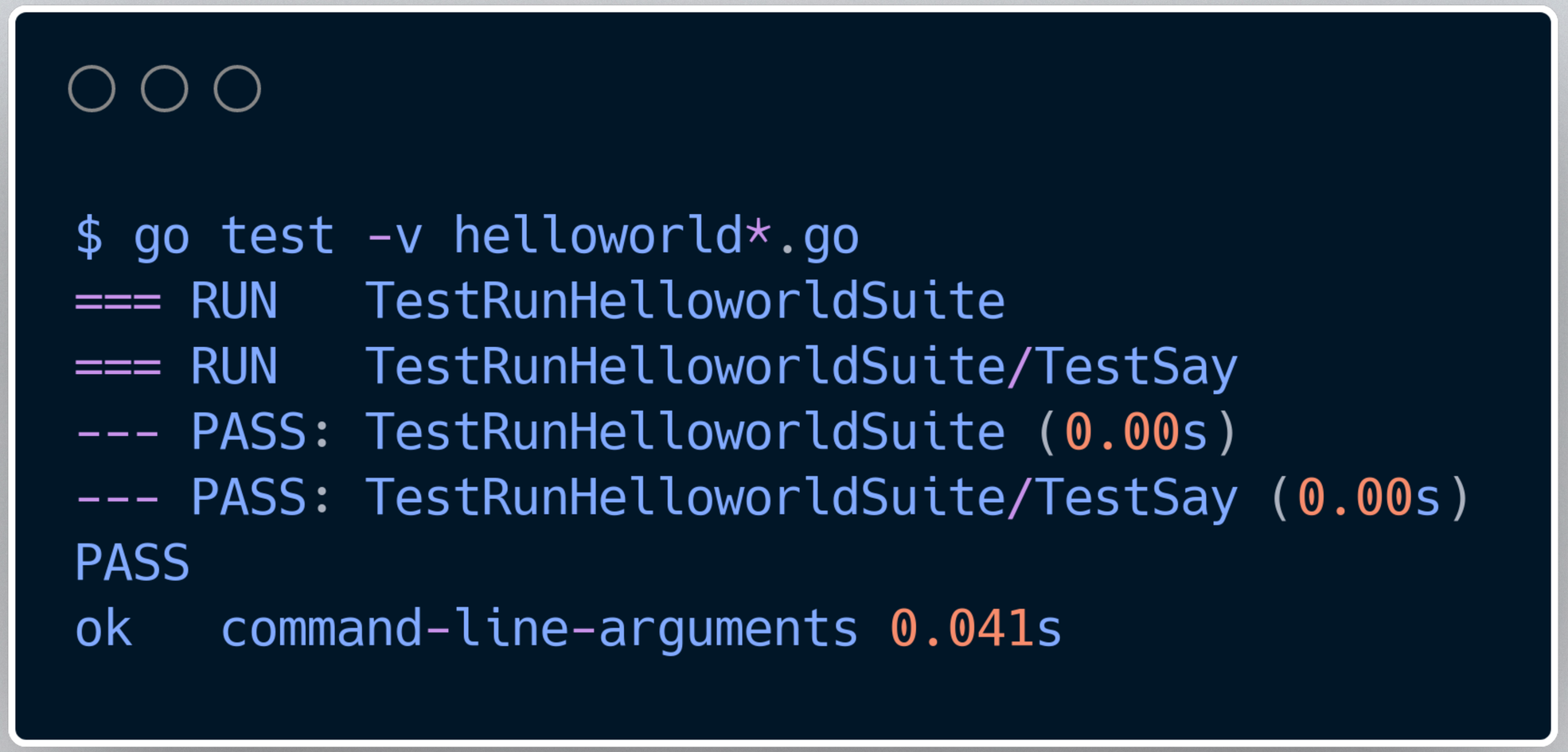
```go
type HelloworldTestSuite struct {
    suite.Suite
    auth       *bind.TransactOpts
    address    common.Address
    gAlloc     core.GenesisAlloc
    sim        *backends.SimulatedBackend
    helloworld *Helloworld
}

func TestRunHelloworldSuite(t *testing.T) {
    suite.Run(t, new(HelloworldTestSuite))
}
```



```go
func (s *HelloworldTestSuite) SetupTest() {
    key, _ := crypto.GenerateKey()
    s.auth = bind.NewKeyedTransactor(key)

    s.address = s.auth.From
    s.gAlloc = map[common.Address]core.GenesisAccount{
        s.address: {Balance: big.NewInt(10000000000)},
    }

    s.sim = backends.NewSimulatedBackend(s.gAlloc)

    _, _, hw, e := DeployHelloworld(s.auth, s.sim)
    s.helloworld = hw
    s.Nil(e)
    s.sim.Commit()
}
```



```go
func (s *HelloworldTestSuite) TestSay() {
    str, err := s.helloworld.Say(nil)
    s.Equal("hello etherworld", str)
    s.Nil(err)
}
```


$ go test -v helloworld*.go
=== RUN   TestRunHelloworldSuite
=== RUN   TestRunHelloworldSuite/TestSay
--- PASS: TestRunHelloworldSuite (0.00s)
--- PASS: TestRunHelloworldSuite/TestSay (0.00s)
PASS
ok   command-line-arguments 0.041s

APRENDA BLOCKCHAIN

OBRIGADO!

oi@juliocampos.com.br
Twitter: @jcserracampos