Blockchain Application based on Scala. Use Case

Scorex: A Scala blockchain framework

accenture technology

David Urdiales Nieto
Juan Manuel García Navarro

Cádiz, 25th October 2018

Agenda

1.- Scorex Overview

2.- Scorex Components

3.- Scorex Concepts

- Proof and Proposition,
- Box, NodeViewModifier,
- Block, History, Vault, NodeViewHolder

4.- Workshop Demo

1.- Scorex Overview

- Scorex Framework
- Language
- Features

Scorex Framework

- Framework for building Blockchain systems
- Modular Architecture. Choose and combine cosensus protocols, transactional structures and networking infrastructure
- Still under active development by developers and researchers in the field.

Language

- Core of framework written in Scala
- Functional Programming Code Style.

Features

- Asynchronous network layer on top of TCP
- JSON API
- Command line client for the JSON API
- Only 4K lines of code

2.- Scorex Components

- Scala
- SBT
- Circe
- Akka
- Scala logging and logback

3.- Scorex Concepts

- Proof and Proposition
- Box
- NodeViewModifier
- Block
- History
- Vault
- NodeViewHolder

Proof

- **A proof** is the more general abstraction that can be provided to open a box or to modify an account
- A proof is non-interactive and thus serializable
- A proof may be ***a sign*** generated by a *private key*, that can be validated using its *public key*.

```
trait Proof[P <: Proposition] extends BytesSerializable {
  def isValid(proposition: P, message: Array[Byte]): Boolean
}
```

Package: scorex.core.transaction.proof

Box

- **A Box** is a ***state element*** locked by ***some proposition***

```
trait Box[P <: Proposition] extends BytesSerializable {
  val value: Box.Amount
  val proposition: P

  val id: ADKey
}

object Box {
  type Amount = Long
}
```

Package: scorex.core.transaction.box

NodeViewModifier

- **A NodeViewModifier** is the abstract representation of a **Transaction**
- Something that ***can change*** *the* ***state of a node***

```
sealed trait NodeViewModifier extends BytesSerializable with ScorexEncoding {
  self =>

  val modifierTypeId: ModifierTypeId

  //todo: check statically or dynamically output size
  def id: ModifierId

  def encodedId: String = encoder.encode(id)

  override def equals(obj: scala.Any): Boolean = obj match {
    case that: NodeViewModifier => (that.id sameElements id) && (that.modifierTypeId == modifierTypeId)
    case _ => false
  }
}
```

Package: scorex.core

Block

- **A block** is an atomic piece of data network participants.
- A block ***has a sequence of transactions***, ***consensus data***, a ***signature*** and ***aditional data***

```
trait Block[TX <: Transaction]
  extends TransactionsCarryingPersistentNodeViewModifier[TX] {

  def version: Version

  def timestamp: Timestamp
}
```

```
trait PersistentNodeViewModifier extends NodeViewModifier {
  def parentId: ModifierId
}
```

```
trait TransactionsCarryingPersistentNodeViewModifier[TX <: Transaction]
  extends PersistentNodeViewModifier {

  def transactions: Seq[TX]
}
```

Package: scorex.core.block

History

- **History** is the ***blocktree*** where longest chain (simplification) is being considered as canonical one

```
trait History[PM <: PersistentNodeViewModifier, SI <: SyncInfo, HT <: History[PM, SI, HT]]
  extends HistoryReader[PM, SI] {
```

Package: scorex.core.consensus

Vault

- Abstract interface for Wallet
- ***A storage*** *for node-specific information*

```
trait Vault[TX <: Transaction,
            PMOD <: PersistentNodeViewModifier, V <: Vault[TX, PMOD, V]] extends NodeViewComponent {
  self: V =>

  def scanOffchain(tx: TX): V

  def scanOffchain(txs: Seq[TX]): V

  def scanPersistent(modifier: PMOD): V

  def scanPersistent(modifiers: Option[PMOD]): V = modifiers.foldLeft(this) { case (v, mod) =>
    v.scanPersistent(mod)
  }

  def rollback(to: VersionTag): Try[V]
}
```

Package: scorex.core.transaction.wallet

NodeViewHolder

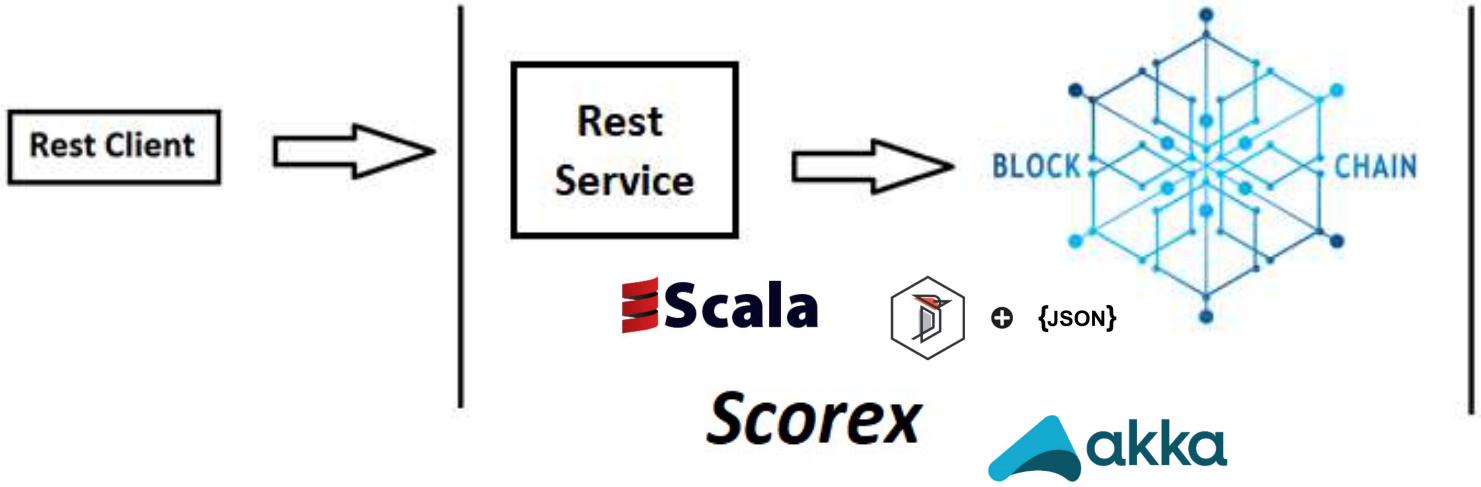
- *Represent the **node state***
- **Composed of**:
  - History, minimal state, memory pool, vault

Package: scorex.core

4.- Workshop Demo

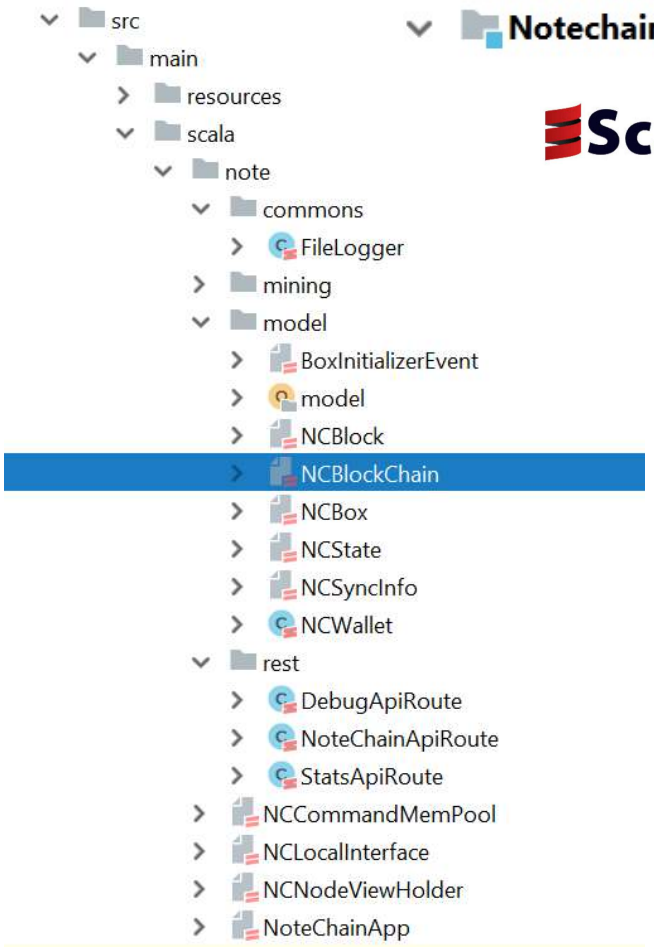
- Scorex Example
- Project Structure
- Running Example

Scorex Example



GitHub: https://github.com/durdiales/Notechain.git

Project Structure



Running Example

- Custom compile Scorex from commit
- Install localy

  `sbt "run ./src/main/resources/settings.conf"` on our repo

References

- https://github.com/ScorexFoundation/Scorex
- https://github.com/ScorexFoundation/ScorexTutorial
- https://underscore.io/blog/posts/2017/12/14/scorex.html
- https://github.com/mgosk/todo-blockchain

*Thanks to **César Antonio Enriquez Ramirez**.*
He worked with us for this workshop.

Q& A

Thanks!

We are Hiring!

for contact:

David Urdiales-Nieto

david.urdiales.nieto@accenture.com