THE DEVELOPER'S CONFERENCE

Pare de se perder! Crie rapidamente sua blockchain com o Convector Suite

Jefferson Bicca Charczuk
Software developer

Muito prazer em conhecê-l@s!

THE DEVELOPER'S CONFERENCE

Jefferson Bicca Charczuk

Desenvolvedor desde 2004
Ciência da Computação PUCRS, 2007
Trabalhando no Serpro desde 2010

#java #javascript #typescript #microservices #tdd #ddd #blockchain

Dúvidas para quem começa

- Como monto uma rede blockchain para desenvolvimento?
- Como estruturo o chaincode?
- Como testo o chaincode?

Precisa entender o básico

- Organization
- Channel
- Peers, CAs, Orderer
- Policies
- Collections config
- Install chaincode
- Instantiate chaincode

Não é sugerida uma estrutura básica

- Mistura entre modelo e controle
- Funciona bem para chaincodes pequenos e com complexidade "gerenciável":
  - Mas e se o chaincode crescer?
  - Mas e se o negócio for complexo?

Precisa subir uma rede

- Ciclo de instalação e instanciação de chaincode na rede
  - Isso leva tempo, consequentemente
- Precisa ser feito um *invoke* na rede. Se quiser automatizar:
  - Criar um script com os *invokes*
  - Criar API e automatizar as requisições por ela
- Testes são no nível de serviço

Convector Suite

THE DEVELOPER'S CONFERENCE

Convector

JavaScript fullstack smart contract systems

The way to develop world-class smart contract systems for Hyperledger Fabric.

Hurley

Streamline your development process

Quickly setup your Hyperledger Fabric development environment.

Facilita desenvolvimento chaincode

- Facilita na criação, desenvolvimento e testes
  - CLI e Runners
  - Testes de unidade com o Mocha
  - Validação de schema
- Sugere padronização de estruturação
  - TypeScript
  - Model/Controller

CLI

THE DEVELOPER'S CONFERENCE

```
npm i -g @worldsibu/convector-cli

# Create a new project with a default chaincode package
conv new myproject -c mychaincode

cd myproject
npm i

# Bring up a development blockchain in your computer
npm run env:restart

# Install the chaincode to the blockchain
npm run cc:start -- mychaincode

# Apply all the changes you want to the project.
# When you need to upgrade your chaincode in the blockchain, you can simply do
npm run cc:upgrade -- mychaincode 1.2
```

Testes de unidade com o Mocha

THE DEVELOPER'S CONFERENCE

```
it('deve criar Pessoa em collection privada palestrantesConfs, async() => {

   const pessoa = new Pessoa({
      id: uuid(),
      nome: 'FULANO TAL',
      cpf: '000.000.001-91',
      conf: 'TDC',
   });

   const collectionName = "palestrantesConfs";
   const txId = await pessoaCtrl.registrarComoPalestrante(pessoa, collectionName);
   expect(txId).to.exist;

   const pessoaBlockchain = await Pessoa.getOne(pessoa.id, Pessoa, {
privateCollection : collectionName });
   expect(pessoaBlockchain.id).to.exist;
});
```

Validação de schema

THE DEVELOPER'S CONFERENCE

```
@Required()
@Validate(yup.string())
public nome: string;

@Required()
@Validate(yup.array(Trilha.schema()))
public trilha: Array<FlatConvectorModel<Trilha>>;
```

Models

THE DEVELOPER'S CONFERENCE

```
export class Pessoa extends ConvectorModel<Pessoa> {

   @ReadOnly()
   @Required()
   public readonly type = 'br.com.confs.Pessoa;
   // métodos herdados da super classe
   static schema<T extends ConvectorModel<any>>
   static getOne<T extends ConvectorModel<any>>
   static query<T>
   static getAll<T extends ConvectorModel<any>>
   update(content: any)
   fetch(storageOptions?: any)
   history()
   save(storageOptions?: any)
   clone()
   toJSON(skipEmpty?: boolean)
   delete(storageOptions?: any)
}
```

Controllers

THE DEVELOPER'S CONFERENCE

```
@Controller('pessoa')
export class PessoaController extends ConvectorController<ChaincodeTx> {

 @Invokable()
 public async registrarComoPalestrante(@Param(Pessoa) pessoa: Pessoa,
   @Param(yup.string()) collections: string
 ) {
   const mspid = this.identity.getMSPID();
   this.valida(pessoa, mspid);

   const collectionsArray: Array<String> = collections.split(",");
   this.validaPresencaCollections(collectionsArray);

   await pessoa.save( { privateCollection : collection });

   return JSON.stringify({ txID: await this.getTxId() });
 }

}
```

Controllers - Native API


```
@Controller('pessoa')
export class PessoaController extends ConvectorController<ChaincodeTx> {

 @Invokable()
 public async consultarPalestrantes(@Param(yup.string()) collections: string,
   @Param(yup.string()) conferencia: string
 ) {
  let queryString = { selector: { conf: undefined } };
  queryString.selector.conf = conf;

  let results;
  try {
      results = await this.tx.stub.getStub().getPrivateDataQueryResult(collection,
JSON.stringify(queryString))
  } catch (e) {
      console.log(e); // dont blame me!
  }

}
```

Criar ambiente para desenvolvimento

- Gera uma rede com a quantidade de organizações, canais, usuários e políticas de acordo com parâmetros passados. Isto inclui:
  - Material criptográfico
  - Criação e *enroll* de usuários admin e user1..*
  - Políticas de acordo com arquivos org*.config.json
  - Arquivo docker-compose.yml

Hurley

THE DEVELOPER'S CONFERENCE

```
# Inicializa uma rede com 3 organizações (npm run env:restart)
hurl new --organizations 3

# Instala e instancia o chaincode (npm run cc:start -- tdcblockchain)
hurl install tdcblockchain node -P ./chaincode-tdcblockchain --collections-config
./collections_config.json;

# Invocando transação na blockchain
hurl invoke pessoa pessoa_registrarComoPalestrante "{\"id\":1, \"nome\":\"FULANO
TAL\",\"cpf\":\"00.000.001-91\",\"conf\":\"TDC\",\"trilha\":[{\"id\":1,\"status\":
\"A\",\"nome\":\"blockchain\"]}" "palestrantesConfs" -u user1 -o org1
```

Gera a API a partir do chaincode

Convector Rest Server

THE DEVELOPER'S CONFERENCE

```
npm install -g @worldsibu/conv-rest-api
conv-rest-api generate api -c tdcblockchain -f org1.tdcblockchain.config.json
npx lerna bootstrap
npx lerna run start --scope server --stream
```

Obrigado!

Jefferson Bicca Charczuk
Software developer

jefferson.rs@gmail.com
http://www.medium.com/@jeffbicca
@jeffbicca

THE DEVELOPER'S CONFERENCE