

## Módulo 5 – Trabalhando com a biblioteca CC-Tools

### Configuração da VM

Nessa tarefa vamos utilizar a configuração da VM da tarefa anterior para usar o CC-Tools.

Apagar os containers antigos.

```
docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q) && docker volume prune && docker system prune
```

Subir uma rede usando o repositório **cc-tools-demo**

```
git clone https://github.com/goledgerdev/cc-tools-demo.git
```

Vendorar o chaincode

```
cd cc-tools-demo/chaincode  
go mod vendor  
cd ..
```

Vendorar o web service

```
cd rest-server  
docker network create cc-tools-demo-net  
./npmInstall.sh  
cd ..
```

Criar uma rede Hyperledger Fabric 2

```
./startDev2.sh
```

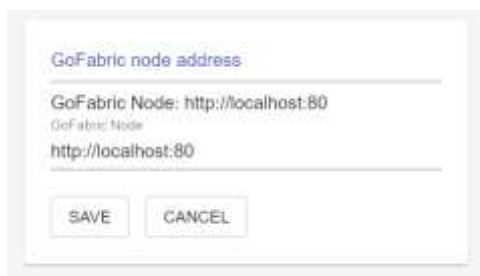
Abrir a aplicação Web.

```
./run-cc-web.sh
```

Abrir a aplicação acessando no browser.

<http://localhost:8080>

Configurar a aplicação e configurar para usar a org1.



# TRILHA BLOCKCHAIN HYPERLEDGER FABRIC

## Tarefa 1

Criar um novo asset com as seguintes configurações:

Asset: Car

Propriedades:

make: tipo string

model, tipo string

colour, tipo string

owner, tipo string

As chaves devem ser *make* e *model*

A propriedade *colour* deve ser obrigatória e *owner* deve ser opcional.

Reiniciar a rede com o comando:

```
./startDev2.sh
```

Criar um carro com o acesso na org3 através do swagger

<http://localhost:1080/api-docs>

Utilizar endpoint CreateAsset

## Tarefa 2

Criar um novo carro com a interface Web acessando a org3

Tirar printscreen da tela do goinitus.

## Tarefa 3

Dentro do asset *Car*, mudar a propriedade *owner* para tipo referência a *Person* (->*person*)

Atualizar o chaincode com o comando

```
./upgradeCC2.sh 0.2 2
```

Gravar a operação.

## Tarefa 4

Atualizar um carro com o acesso na org2 através da interface Web acessando a api no endereço <http://localhost:980> e cadastrar um owner para um carro.

Tirar um printscreen da tela de listagem de Car

## Tarefa 5

Criar um dataType custom para fazer a seguinte validação:

# TRILHA BLOCKCHAIN HYPERLEDGER FABRIC

tipo *number*

Valor deve ser maior ou igual a 1.0 e menor ou igual a 10.0

Acrescentar a propriedade *bookRating* do novo tipo dentro do asset *Book*

Atualizar o *chaincode* para a versão 0.3 sequencia 3

Gravar o resultado da operação de atualização do chaincode.

## Tarefa 5

Usando a interface gráfica atualizar um asset *Book* com um valor inválido (menor que 1.0 ou maior que 10.0)

Tirar print screen da mensagem de erro na tela.

Atualizar com um valor válido.

Tirar printscreen da listagem de *Book*

## Tarefa 6

Acrescentar o operador *Writers* no item *colour* do asset *Car* para permitir que essa propriedade seja modificada apenas pela client da org1 e da org2

**Writers:** `[]string{'org1MSP', 'org2MSP'},`

Atualizar o *chaincode* para a versão 0.3 sequencia 3

Acessar a interface gráfica na org3 (<http://localhost:1080>) e tentar atualizar a propriedade *colour*

Tirar printscreen da mensagem de erro da tela.

## Tarefa 7

Acessar o *goinitus* apontando para a API da org1 ou da org2.

Entrar na tela de atualizar o asset *Car* e preencher uma atualização de uma *colour* de um asset *Car*.

Selecionar o botão *CURL* e copiar o conteúdo.

Realizar a operação via linha de comando utilizando o *curl*.