

# Suporte ao Desenvolvimento

## Setup

Pensando em integração com sistemas já existentes nas grandes corporações, foi escolhida a JVM como ambiente para execução, assim, facilitando a integração com os sistemas legados.

Para este lab é necessário utilizar:

- JDK - <https://bit.ly/1IO1FSV> - Baixar versão mais recente da JDK
- IntelliJ - <https://bit.ly/1IDJJ7h> - Baixar versão Community

Não há necessidade de customização na instalação de nenhuma das ferramentas.

Em uma pasta de projetos, realize os dois clones abaixo:

```
"git clone https://github.com/corda/cordapp-example"  
"git clone https://github.com/corda/cordapp-template-kotlin"
```

## Projeto 1 - Aula 1 - Deploy

Nesta aula vamos ver:

- Modelo de deploy dos nós de Corda
- Organização de pastas do Corda
- Como configurar um nó de Corda através da task "deployNodes"
- Como configurar um nó através do arquivo "node.conf"

### Modelo de Deploy

Vamos primeiro dar uma olhada em como fazer o deploy do projeto. Os projetos em Corda são conhecidos como corDapps (Corda Distributed Application), os corDapps são distribuídos com o seu gestor de dependências, então não é necessário a instalação de nenhuma outra ferramenta para a utilização deles.

O primeiro projeto que vamos dar uma olhada é o *corDapp-example*, este projeto implementa o conceito de IOU (I Own You), onde cada participante lança na rede que ele está devendo para a outra parte.

Utilizando o terminal de sua máquina acesse a sua pasta de workspace e execute o comando sem as aspas:

“git clone <https://github.com/corda/cordapp-example>”

```
PS C:\Users\Rodrigo\Projetos\workspace> git clone https://github.com/corda/cordapp-example
Cloning into 'cordapp-example'...
remote: Counting objects: 4140, done.
remote: Compressing objects: 100% (123/123), done.
remote: Total 4140 (delta 45), reused 100 (delta 15), pack-reused 3964
Receiving objects: 100% (4140/4140), 1.70 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (1449/1449), done.
PS C:\Users\Rodrigo\Projetos\workspace>
```

Ao finalizar o clone acesse a pasta “cordapp-example”:

“cd cordapp-example”

Na pasta “cordapp-example” execute o comando:

“git status”

```
PS C:\Users\Rodrigo\Projetos\workspace> git clone https://github.com/corda/cordapp-example
Cloning into 'cordapp-example'...
remote: Counting objects: 4140, done.
remote: Compressing objects: 100% (123/123), done.
remote: Total 4140 (delta 45), reused 100 (delta 15), pack-reused 3964
Receiving objects: 100% (4140/4140), 1.70 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (1449/1449), done.
PS C:\Users\Rodrigo\Projetos\workspace> cd .\cordapp-example\
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example> git status
On branch release-V3
Your branch is up-to-date with 'origin/release-V3'.
nothing to commit, working tree clean
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example>
```

Você deve visualizar que o projeto está “up-to-date”.

Agora vamos fazer o deploy do projeto, para isto, vamos utilizar o executável “gradlew” que é uma versão portátil de um gestor de dependência para Java. Execute o comando:

“./gradlew clean deployNodes”

```
> Task :kotlin-source:deployNodes
Bootstrapping local network in C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes
Node config files found in the root directory - generating node directories
Generating directory for Notary
Generating directory for PartyA
Generating directory for PartyB
Generating directory for PartyC
Nodes found in the following sub-directories: [Notary, PartyA, PartyB, PartyC]
Waiting for all nodes to generate their node-info files...
...still waiting. If this is taking longer than usual, check the node logs.
Distributing all node info-files to all nodes
Gathering notary identities
Notary identities to be used in network parameters: O=Notary, L=London, C=GB (non-validating)
No existing whitelist file found.
Calculating whitelist for current installed Cordapps..
Cordapp whitelist generated in C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\whitelist.txt
Updating whitelist
Bootstrapping complete!

BUILD SUCCESSFUL in 4m 24s
12 actionable tasks: 11 executed, 1 up-to-date
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example>
```

No final da execução você deve receber a mensagem “*Build Successful*”.

Os comandos que executamos, “clean” e “deployNodes”, tem como objetivo limpar os builds antigos que possam ter sido feitos e realizar o deploy de vários nodes de Corda de acordo com a especificação que está presente no arquivo “*gradle.build*”, vamos entrar em mais detalhes sobre o que tem neste arquivo a seguir.

## Organização

Vamos explorar o que temos nos arquivos que foram gerados com o comando “deployNodes”. Acesse a pasta “kotlin-src\build\nodes”:

“cd kotlin-src\build\nodes”

Ao listar o diretório você deve ter um resultado assim:

```
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes> ls

Directory: C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes

Mode                LastWriteTime         Length Name
----                -
d-----          02/07/2018         18:21         .cache
d-----          02/07/2018         18:22         Notary
d-----          02/07/2018         18:22         PartyA
d-----          02/07/2018         18:22         PartyB
d-----          02/07/2018         18:22         PartyC
-a-----          02/07/2018         18:21           320 runnodes
-a-----          02/07/2018         18:21          113 runnodes.bat
-a-----          02/07/2018         18:21       942211 runnodes.jar
-a-----          02/07/2018         18:22          558 whitelist.txt
```

O executável “*runnodes*” irá inicializar todos os nós presentes nesta pasta, para facilitar os nossos testes.

Dentro de cada uma das pastas presentes, vamos ter um deploy completo de um nó Corda. Acessa a pasta “*PartyA*” para darmos uma olhada nos arquivos presentes.

```
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes> cd .\PartyA\
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyA> ls

Directory: C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyA

Mode                LastWriteTime         Length Name
----                -
d-----          02/07/2018        18:22         additional-node-infos
d-----          02/07/2018        18:22         certificates
d-----          02/07/2018        18:22         cordapps
d-----          02/07/2018        18:22         drivers
d-----          02/07/2018        18:22         logs
-a-----          02/07/2018        18:22      41825797 corda-webserver.jar
-a-----          02/07/2018        18:21      53770433 corda.jar
-a-----          02/07/2018        18:22       4521 network-parameters
-a-----          02/07/2018        18:21       308 node.conf
-a-----          02/07/2018        18:22       4545 nodeInfo-E4477B559304AADFC0638772C0956A38FA2E2A7A5EB0E65D0D83E5884831879A
-a-----          02/07/2018        18:22      65536 persistence.mv.db
-a-----          02/07/2018        18:22         5 process-id

PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyA>
```

Temos várias pastas e vários arquivos, vamos passar para cada um deles para entender qual o seu objetivo.

Pastas:

- “**additional-node-infos**”: pasta que contém todas as informações de rede dos demais nós da rede. No Corda, a comunicação é feita diretamente entre os nós, ou seja, existe um “Full-Mesh” entre os nós de uma mesma rede, nesta pasta temos todas as informações sobre os certificados das outras partes e o host e a porta de destino para realizar uma comunicação.
- “**certificates**”: pasta que contém os seus próprios certificados, armazenados em keystores padrão java. É nela onde fica armazenada a sua privada.
- “**cordapps**”: pasta que armazena todos os corDapps que este nó pode executar. No Corda, é necessário fazer a distribuição do binário de cada corDapp que você quer executar, diferente do Ethereum, o código não é enviado junto com as transações. Apesar de não parecer seguro, todas as transações enviam o hash do corDapp que está sendo utilizado, desta forma, garantindo que as duas partes possuem o mesmo código.
- “**drivers**”: pasta que armazena extensões para o seu nó de Corda, como o “Jolokia” que cria um servidor HTTP para acessar as informações de JMX do seu nó.
- “**logs**”: pasta onde ficam armazenados os logs da aplicação.

Além das pastas, temos também os arquivos:

- **“corda-webserver.jar”**: um webserver feito pela R3 que permite a exposição de APIs escritas no seu corDapp.
- **“corda.jar”**: executável do Corda. Este executável é o nó do Corda, no início de sua execução ele irá escanear a pasta *“cordapps”* e vai inicializar todos os corDapps presentes.
- **“network-parameters”**: arquivo que descreve o funcionamento da rede e quais os corDapps podem ser executados.
- **“node.conf”**: arquivo que contém as configurações do seu nó. Entraremos mais a fundo na sequência.
- **“nodeInfo-xxxxxx”**: arquivo que descreve unicamente o seu nó, este arquivo é distribuído na rede para que os demais nós consigam entrar em contato com o seu nó.
- **“persistence.mv.db”**: por default, o Corda utiliza o banco de dados H2, um banco de dados SQL in-memory, todas as informações são enviadas periodicamente para este arquivo.

## Configuração Nó

Vamos voltar para a pasta *“nodes”*, utilize o comando:

```
“cd ..”
```

Poderíamos agora executar o comando `runnodes` e verificar o Corda executando, porém, como temos 4 nós definidos, 1 notary e 3 parties, vamos ajustar o arquivo *“build.gradle”*. Volte para a pasta *“kotlin-source”*:

```
“cd ../../”
```

Abra o arquivo *“build.gradle”* com o seu editor de texto.

```
77 task deployNodes(type: net.corda.plugins.Cordform, dependsOn: ['jar']) {
78     directory "./build/nodes"
79     node {
80         name "O=Notary,L=London,C=GB"
81         notary = [validating : false]
82         p2pPort 10006
83         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
84     }
85     node {
86         name "O=PartyA,L=London,C=GB"
87         p2pPort 10007
88         rpcSettings {
89             address("localhost:10008")
90             adminAddress("localhost:10048")
91         }
92         webPort 10009
93         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
94         rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
95     }
96     node {
97         name "O=PartyB,L=New York,C=US"
98         p2pPort 10010
99         rpcSettings {
100             address("localhost:10011")
101             adminAddress("localhost:10051")
102         }
103         webPort 10012
104         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
105         rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
106     }
107     node {
108         name "O=PartyC,L=Paris,C=FR"
109         p2pPort 10013
110         rpcSettings {
111             address("localhost:10014")
112             adminAddress("localhost:10054")
113         }
114         webPort 10015
115         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
116         rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
117     }
118 }
```

No arquivo “*build.gradle*” temos a task “*deployNodes*”, que utilizamos anteriormente. Nela, temos, o diretório onde os nodes serão instalados, e vários “*nodes*” que representam cada um dos nós que vamos fazer o deploy.

O “node” possui as configurações:

- “**name**”: nome legal deste nó. No corda, todos os nós precisam ter um nome legal, no formato x500(<https://tools.ietf.org/html/rfc1779>), este nome será utilizado como um dos atributos do certificado digital do nó (certificado x509v3 - <https://www.ietf.org/rfc/rfc5280.txt>).
- “**notary**”: parâmetro opcional, ele irá descrever quais tipos de serviço de notary serão oferecidos por este nó. Os valores padrão são “*validating: true*”, onde todos os dados são validados pelo notary e “*validating: false*” onde apenas os hashes dos states são armazenados.
- “**p2pPort**”: porta que será enviada para os demais nós conseguirem se comunicar com este nó. Esta é a porta de entrada para o nó que precisa aceitar todos os requests dos demais nós.

- **“rpcSettings”**: contém as informações das portas para conexão RPC com o nó. Todas as comunicações de serviços seus com o seu nó devem ser através do endereço listado no “address”, e todo serviço de manutenção do nó deve ser comunicar com o endereço listado no “adminAddress”.
- **“webPort”**: a webPort é a porta que será exposta pelo “corda-webserver”.
- **“cordapps”**: contém a lista dos corDapps dos quais seu nó depende, caso eles estejam em um repositório maven, eles são baixados automaticamente e colocados na pasta “cordapps”.
- **“rpcUsers”**: contém a lista de usuários que podem fazer acesso via RPC e quais os serviços eles podem executar.

Vamos remover o Node da PartyC. Você deve obter um resultado igual ao abaixo:

```
77 task deployNodes(type: net.corda.plugins.Cordform, dependsOn: ['jar']) {
78     directory "./build/nodes"
79     node {
80         name "O=Notary,L=London,C=GB"
81         notary = [validating : false]
82         p2pPort 10006
83         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
84     }
85     node {
86         name "O=PartyA,L=London,C=GB"
87         p2pPort 10007
88         rpcSettings {
89             address("localhost:10008")
90             adminAddress("localhost:10048")
91         }
92         webPort 10009
93         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
94         rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
95     }
96     node {
97         name "O=PartyB,L=New York,C=US"
98         p2pPort 10010
99         rpcSettings {
100             address("localhost:10011")
101             adminAddress("localhost:10051")
102         }
103         webPort 10012
104         cordapps = ["$corda_release_group:corda-finance:$corda_release_version"]
105         rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
106     }
107 }
```

Volte para a pasta raiz do projeto:

“cd ..”

Vamos executar novamente o comando para realizar o deploy dos nós:

“./gradlew clean deployNodes”

Voltando para a pasta onde estão os nodes, agora devemos ver apenas os nodes “Notary”, “PartyA” e “PartyB”.

```
“cd kotlin-source/build/nodes”
```

Já sabemos como configurar um nó com a task “*deployNodes*”, vamos olhar agora como configurar um nó após o deploy.

Acesse a pasta do nó “PartyA”.

```
“cd PartyA”
```

Abra o arquivo “node.conf”.

```
PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyA> cat .\node.conf
myLegalName="O=PartyA,L=London,C=GB"
p2pAddress="localhost:10007"
rpcSettings {
    address="localhost:10008"
    adminAddress="localhost:10048"
}
rpcUsers=[
    {
        password=test
        permissions=[
            ALL
        ]
        user=user1
    }
]
webAddress="localhost:10009"

PS C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyA>
```

As configurações que foram feitas na task “*deployNodes*” é transportada para este arquivo, toda mudança que for feita neste arquivo irá refletir na forma como o nó irá se comportar. Altere o valor de “webAddress” para “localhost:11009”.

Para mais informação em relação à configuração dos nós, acesse:

<https://docs.corda.net/corda-configuration-file.html>

Agora ao executar o comando “*runnodes*” vamos visualizar a inicialização de 5 processos Java.

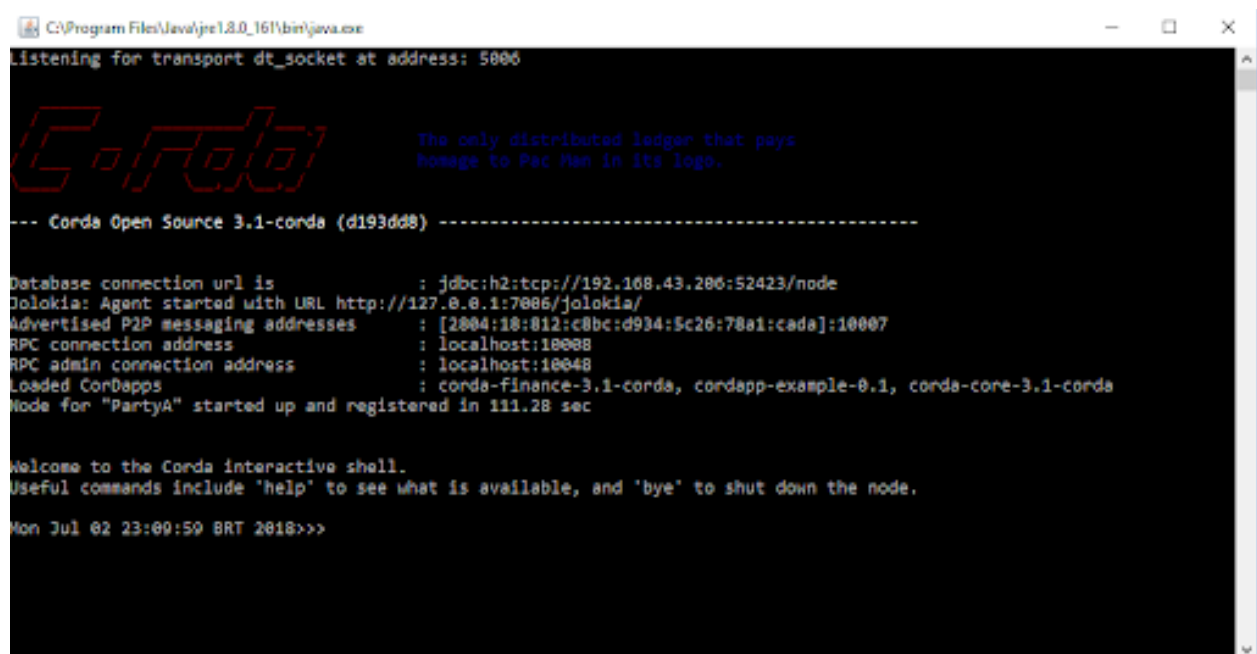
```
“./runnodes”
```



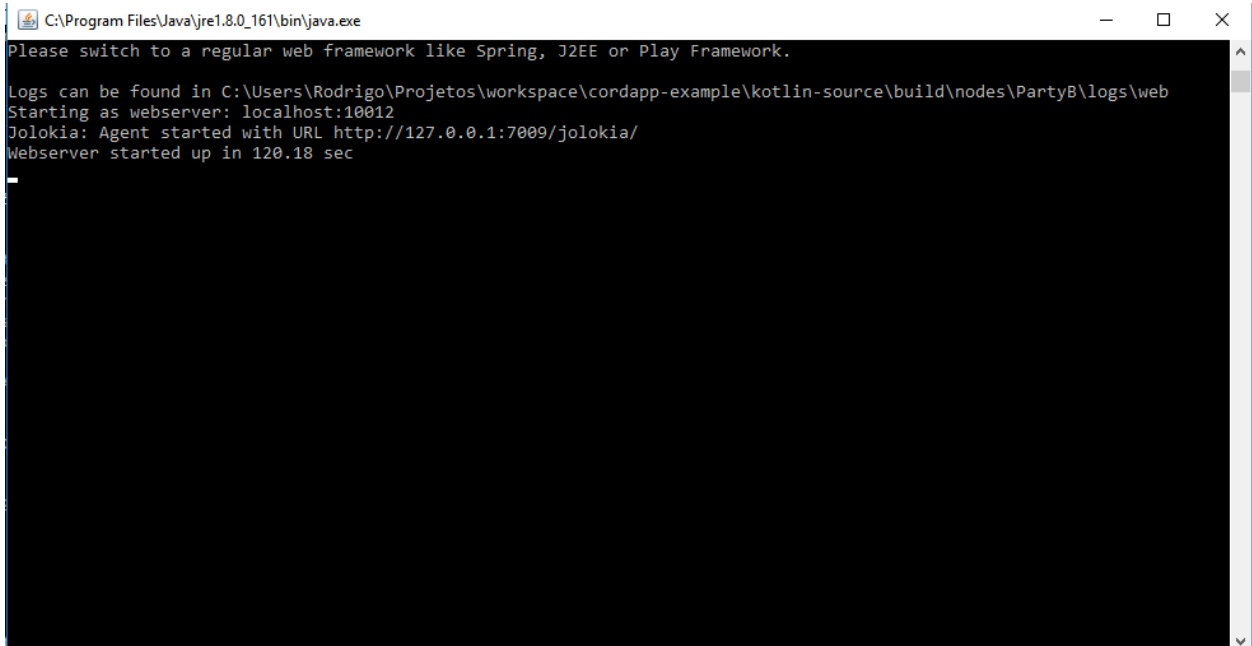


Vamos ter dois tipos de executáveis sendo rodados:

- Executáveis de Corda, que irão apresentar a tela a seguir quando finalizarem a inicialização:

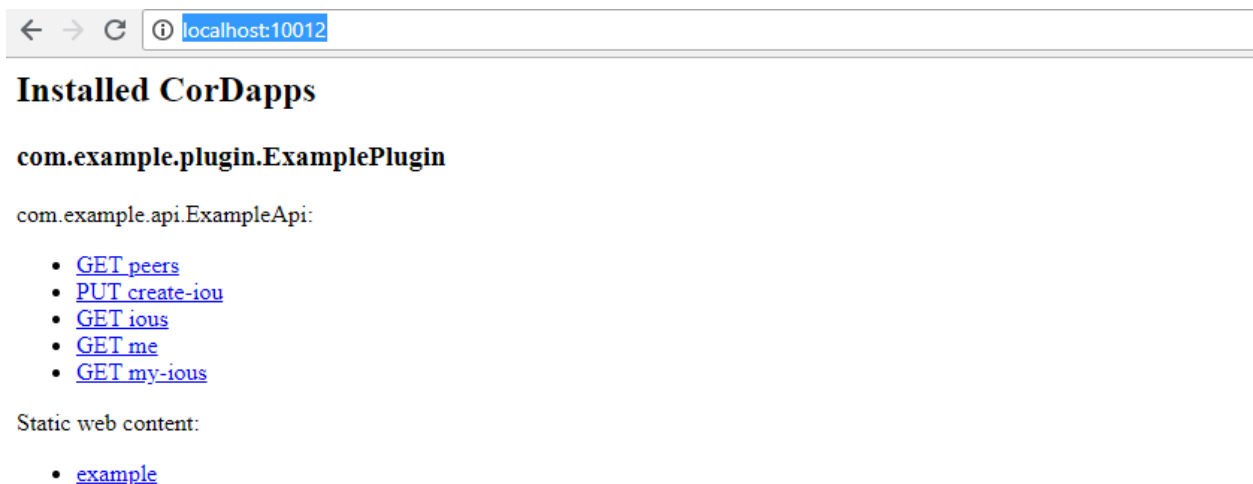


- Executáveis Corda-Webserver, que irão apresentar a tela a seguir:



```
C:\Program Files\Java\jre1.8.0_161\bin\java.exe
Please switch to a regular web framework like Spring, J2EE or Play Framework.
Logs can be found in C:\Users\Rodrigo\Projetos\workspace\cordapp-example\kotlin-source\build\nodes\PartyB\logs\web
Starting as webserver: localhost:10012
Jolokia: Agent started with URL http://127.0.0.1:7009/jolokia/
Webserver started up in 120.18 sec
```

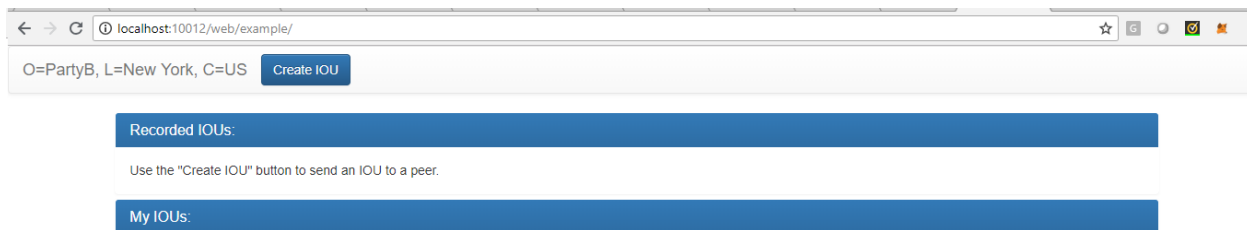
Agora que está tudo inicializado, acesse o endereço: <http://localhost:10012/>



Na página que foi carregada, vamos ter as seguintes funções:

- **GET peers**: que irá retornar todos os demais nós da rede que este nó conhece.
- **PUT create-iou**: que irá criar uma nova dívida na rede.
- **GET ious**: que irá retornar todas as dívidas que eu conheço.
- **GET me**: que irá retornar o nome x500 do nó que está sendo acessado.
- **GET my-ious**: que irá retornar todas as dívidas que estão no nome do nó que está sendo acessado.

Além disso, temos um link para um conteúdo estático, que é uma página onde podemos fazer requisições aos métodos que foram criados.



Sinta-se à vontade para brincar com os métodos e verificar os resultados das chamadas.

Para acessar as informações do outro nó, acesse pelo endereço: <http://localhost:11009>