

ProjetEISI

Projet du cours élasticité et interopérabilité des système d'information

Projet :

Pour ce projet, il nous a été demandé un application qui crée des *smart contract*. Les *smart contract* représente des *baux ruraux*, un bail rural est composé du nom du propriétaire du terrain, la surface en ares ainsi que les points gps qui limite le terrain.

Applicatif :

l'application se divise en un client python avec le framework Flask et la librairie folium pour l'affichage de la carte et le relever des points gps. Le back est une application SpringBoot qui se connecte au server Ganache pour générer un *smart contract*. La dernière partie est le *smart contract* pour la blockchain Ethereum.

Pour réaliser cette application nous avons utiliser la librairie *web3j* et son CLI (*web3j_cli*) ainsi que le compilateur *sol_c* que vous trouverez dans le dépôt GIT sous le dossier

```
web3j-3.6.0
```

Installation :

1. Application python :

Vous aurez premièrement besoin du framework *Flask* et des librairies suivantes : folium, pandas, requests.

Installation du framwork Flask :

```
pip install Flask
```

Installation de la librairie folium :

```
pip install folium
```

Installation de la librairie pandas :

```
pip install pandas
```

Installation de la librairie requests :

```
pip install request
```

2. Application Java :

premièrement il faut générer la classe java qui va correspondre au smart contract défini dans le fichier.sol. Pour réaliser cette opération il vous faudra ouvrir votre terminal à partir du dossier où vous avez mis votre fichier solc-windows.exe et entrer cette ligne de commande en l'adaptant :

```
solc-windows.exe file_name.sol --bin --abi --optimize -o out_dir
```

Une fois cette opération finie vous devriez avoir deux fichier dans votre dossier de sortie out_dir. Un fichier file_name.bin et un autre fichier file_name.abi

Finalement l'opération qui vous générera la classe file_name grâce au cli web3j. Ouvrir un terminal à partir du fichier /bin et entrer la commande suivante :

```
web3j generate solidity -a ./path/filename.abi -b ./path/filename.bin -o  
/path/to/yourSrc/main/java -p package.name Une fois toutes les opération précédente réalisé  
votre application est prête à interagir avec Ganache
```

Pour adapter mon code à votre environnement vous aurez aussi besoin de faire des changements au niveau du code Java et Python.

Au niveau de Java :

```
④ EtherumServices.java ×  
12  
13 import java.math.BigInteger;  
14 import java.util.List;  
15 import java.util.stream.Collectors;  
16  
17 2 usages  
18 @Service  
19 public class EtherumServices {  
20  
21     1 usage  
22     private static final String RPC_SERVER_URL = "http://localhost:7545";  
23  
24     1 usage  
25     private static final String PRIVATE_KEY = ""; //Entrer ici votre clé privée de votre smart contract que vous trouverez sur ganache
```

Vous devrez placer la clé privée que vous trouverez sous Ganache.

Au niveau du Python :

```
py main.py ×  
1 import os  
2 from shutil import copy  
3 import pandas as pd  
4 import requests  
5 import json as gson  
6  
7 from flask import Flask  
8 import folium  
9 from folium.plugins import Draw  
10  
11 app = Flask(__name__)  
12  
13 directory_temp = "path_vers_votre_dossier_Téléchargement/data.geojson"
```

Amélioration :

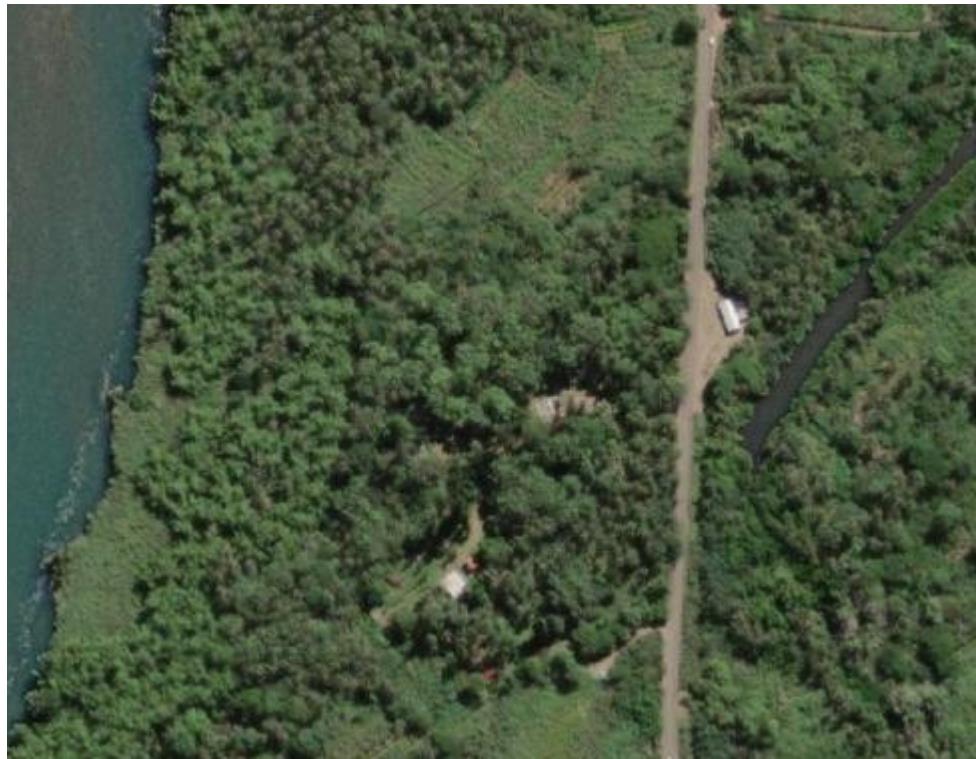
Nous n'avons pas eu le temps développer un petit formulaire pour permettre à l'utilisateur de renseigner le nom du propriétaire et la surface du terrain.

Fonctionnement :

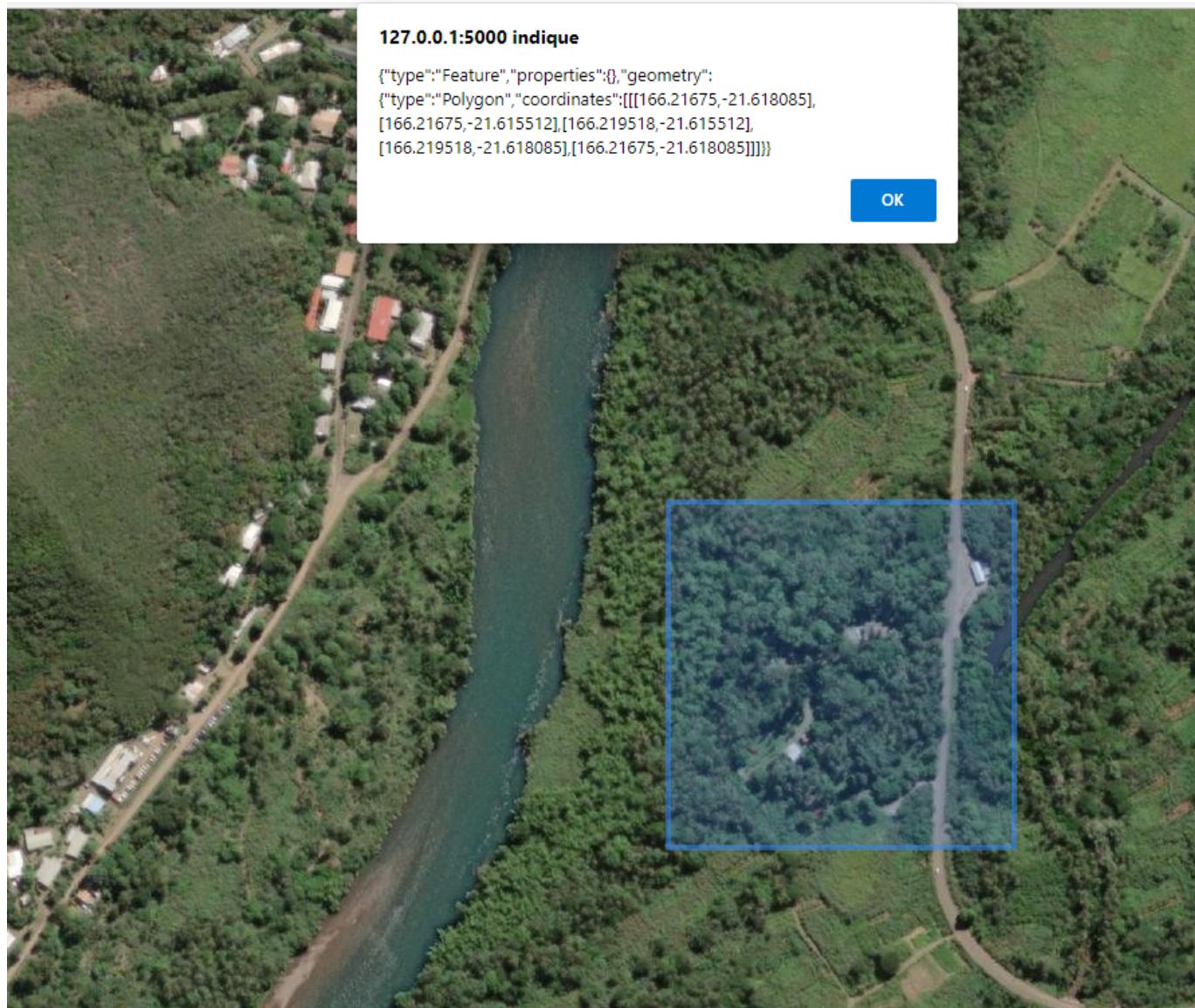
L'application python nous permet d'avoir une page web qui est centrée sur la Nouvelle-Calédonie et plus particulièrement sur la commune de Thio.



Nous avons choisi arbitrairement cette propriété :



Nous réalisons un tracer :



Après avoir appuyer sur le bouton export nous enregistrons la transaction :

```
Run SmartContractBailRuralApplication ×

2023-05-08 15:03:27.722 INFO 56790 --- [ restartedMain] percyourtechniquesmappinginternalmapping : mapped URL path [/v2/api/docs] onto method [GET]/docs/documentation/swagger2/web/swagger.json
↑ 2023-05-08 15:03:27.835 INFO 56790 --- [ restartedMain] o.s.b.d.a.OptionalaliveReloadServer : LiveReload server is running on port 35729
↓ 2023-05-08 15:03:28.014 INFO 56790 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
⇨ 2023-05-08 15:03:28.015 INFO 56790 --- [ restartedMain] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
⇨ 2023-05-08 15:03:28.035 INFO 56790 --- [ restartedMain] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
⇨ 2023-05-08 15:03:28.066 INFO 56790 --- [ restartedMain] s.d.s.w.s.ApiListingReferenceScanner : Scanning for api listing references
⇨ 2023-05-08 15:03:28.203 INFO 56790 --- [ restartedMain] n.u.s.SmartContractBailRuralApplication : Started SmartContractBailRuralApplication in 4.77 seconds (JVM running for 5.328)
刪 2023-05-08 15:03:50.497 INFO 56790 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-08 15:03:50.497 INFO 56790 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-05-08 15:03:50.499 INFO 56790 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Contract deployed at: 0x74221633497528cf7d25b0c430c084acec5bfafa
```

Du côté de Ganache, nous retrouvons cette enregistrement :

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS UPDATE AVAILABLE SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 38 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MERGE NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING STATUS AUTOMINING WORKSPACE BAILRURAL SWITCH

MNEMONIC stem crack hour credit ability immense theme foot coyote act worth happy

ADDRESS 0xca51B12eEe35BAF6AF8339BEddb8A10315D6F4BE BALANCE 99.89 ETH TX COUNT 38 INDEX 0 HD PATH m44'60'0'account_index

au niveau des Blocks :

CURRENT BLOCK 38	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING		WORKSPACE BAILRURAL	SWITCH	⚙️
BLOCK 38	MINED ON 2023-05-08 15:03:50						GAS USED 322646	1 TRANSACTION		
BLOCK 37	MINED ON 2023-05-08 15:03:50						GAS USED 402414	1 TRANSACTION		

l'enregistrement au niveau des transactions :

TX HASH 0x8d9aa00db2dbc93e448fe0224425f34537b23d17b1796d4344f5b4302f55c7d2	FROM ADDRESS 0x...ca51b12eEe35BAF6AF8339BEddb8A10315D6F4BE	TO CONTRACT ADDRESS 0x74221633497528cf7d25b6c430c084acec5bfafa	GAS USED 322646	VALUE 0	CONTRACT CALL
TX HASH 0xc7030bd3d65eb96ff76dc6bf0afa501a28f4dbf3147c0138aad0de412e85ebf8	FROM ADDRESS 0x...ca51b12eEe35BAF6AF8339BEddb8A10315D6F4BE	CREATED CONTRACT ADDRESS 0x74221633497528cf7d25b6c430c084acec5bfafa	GAS USED 402414	VALUE 0	CONTRACT CREATION

Et en dernier les logs sur la transaction :

```
[15:03:50]
[15:03:50] Transaction: 0xc7030bd3d65eb96ff76dc6bf0afa501a28f4dbf3147c0138aad0de412e85ebf8
[15:03:50] Contract created: 0x74221633497528cf7d25b6c430c084acec5bfafa
[15:03:50] Gas usage: 402414
[15:03:50] Block number: 37
[15:03:50] Block time: Mon May 08 2023 15:03:50 GMT+1100 (heure des îles Salomon)
[15:03:50]

[15:03:50] eth_getTransactionCount
[15:03:50]
[15:03:50] Transaction: 0x8d9aa00db2dbc93e448fe0224425f34537b23d17b1796d4344f5b4302f55c7d2
[15:03:50] Gas usage: 322646
[15:03:50] Block number: 38
[15:03:50] Block time: Mon May 08 2023 15:03:50 GMT+1100 (heure des îles Salomon)
[15:03:50]
[15:03:50] eth_getTransactionReceipt
[15:03:50] Transaction: 0x8d9aa00db2dbc93e448fe0224425f34537b23d17b1796d4344f5b4302f55c7d2
[15:03:50] Gas usage: 322646
[15:03:50] Block number: 38
[15:03:50] Block time: Mon May 08 2023 15:03:50 GMT+1100 (heure des îles Salomon)
```