

ProjetEISI

Projet du cours élasticité et interopérabilité des systèmes d'information

Projet :

Pour ce projet, il nous a été demandé une application qui crée des *smart contract*. Les *smart contract* représentent des *baux ruraux*, un bail rural est composé du nom du propriétaire du terrain, la surface en ares ainsi que les points Gps qui délimitent 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 relevé 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 utilisé 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 :

Tout d'abord 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 le fichier *BailRural.sol* fourni avec ce TP (disponible aussi dans le répertoire [web3j-3.6.0](#)). Puis exécuter cette ligne de commande en remplaçant *out_dir* par le chemin complet du répertoire de destination :

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

Vous devriez alors avoir deux fichiers dans votre dossier de sortie *out_dir*. Un fichier *file_name.bin* et un autre fichier *file_name.abi*.

Enfin l'opération qui générera la classe *file_name* grâce au cli web3j. Ouvrir un terminal à partir du répertoire /bin de web3j, y déposer les fichiers *bin* et *abi* précédemment créés, et exécuter 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 ces opérations réalisées, votre application est prête à interagir avec Ganache.

Pour adapter le code à votre environnement vous aurez aussi besoin de faire les changements suivants au niveau de Java et Python.

Java : */nc/unc/smartContractBailRural/webservice/services/EtherumServices.java*

```
② 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 saisir la clé privée de votre wallet que vous aurez au préalable copiée depuis Ganache.

Python : */FlaskFolium/main.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"
```

Il faut indiquer ici le chemin complet de votre répertoire contenant le fichier *data.geojson*

Amélioration :

Par manque de temps développer, il n'a pas été possible de mettre en place un formulaire permettant de renseigner le nom du propriétaire et la surface du terrain.

Fonctionnement :

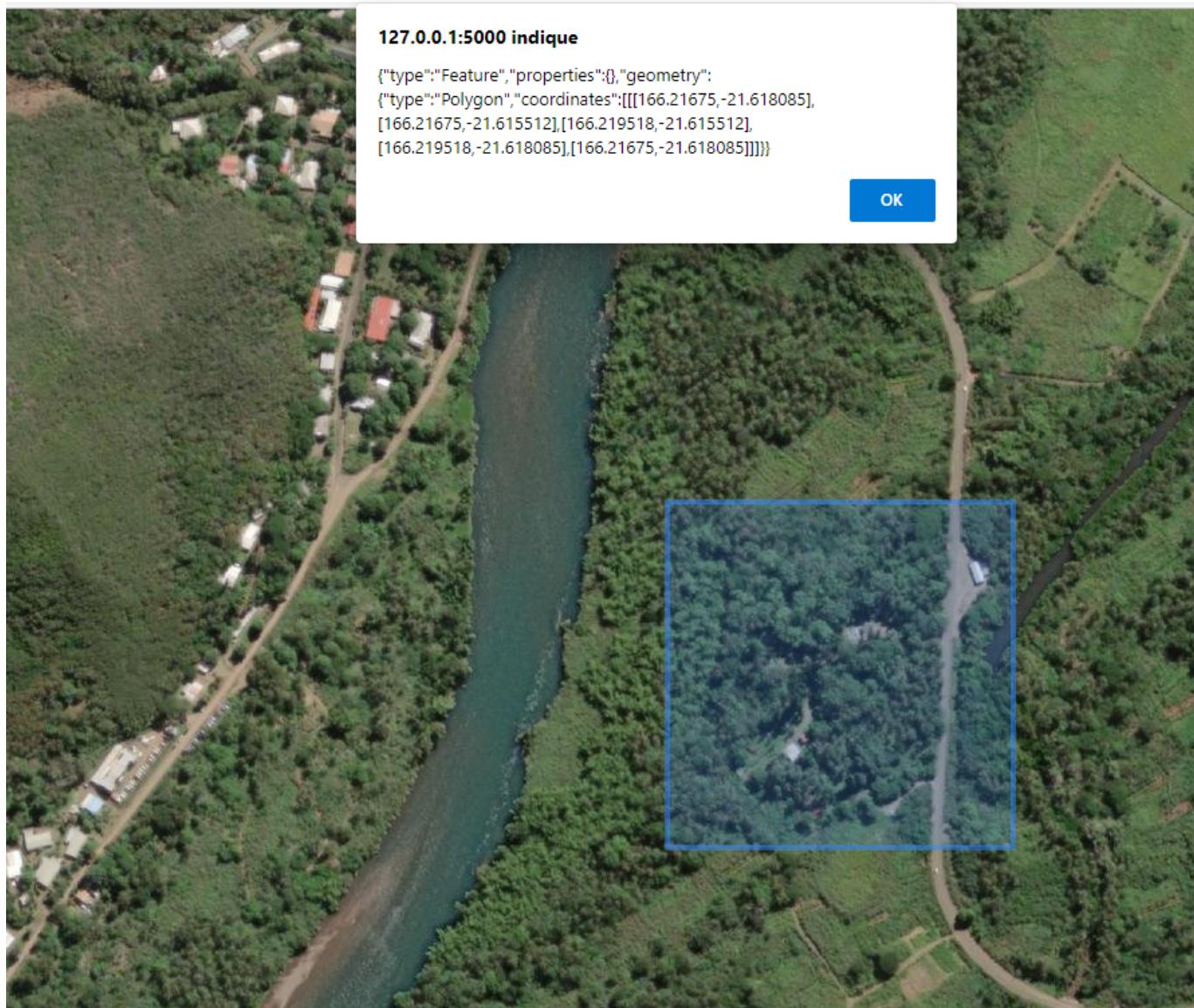
L'application python permet d'afficher dans une page web la carte de la Nouvelle-Calédonie et plus spécifiquement la commune de Thio.



Nous avons choisi arbitrairement cette propriété :



Nous réalisons un tracer :



Après avoir appuyer sur le bouton export la transaction est enregistrée :

```
Run SmartContractBailRuralApplication x

[ 2023-05-08 15:03:27.722 INFO 56796 --- [ restartedMain] percy.lee.springboot.autoconfigure.GetMapping : Mapped GET PATH [/v2/api/docs] onto method com.percy.lee.springboot.autoconfigure.GetMapping
[ 2023-05-08 15:03:27.835 INFO 56796 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
[ 2023-05-08 15:03:28.014 INFO 56796 --- [ 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 56796 --- [ restartedMain] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
[ 2023-05-08 15:03:28.035 INFO 56796 --- [ restartedMain] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
[ 2023-05-08 15:03:28.066 INFO 56796 --- [ restartedMain] d.s.d.w.s.ApiListingReferenceScanner : Scanning for api listing references
[ 2023-05-08 15:03:28.203 INFO 56796 --- [ restartedMain] n.u.s.SmartContractBailRuralApplication : Started SmartContractBailRuralApplication in 4.77 seconds (JVM running for 5.328)
[ 2023-05-08 15:03:50.497 INFO 56796 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
[ 2023-05-08 15:03:50.497 INFO 56796 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
[ 2023-05-08 15:03:50.498 INFO 56796 --- [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 :

ADDRESS	BALANCE	TX COUNT	INDEX
0xca51B12eEe35BAF6AF8339BEddb8A10315D6F4BE	99.89 ETH	38	0

Du côté de Ganache, nous retrouvons cette enregistrement :

MNEMONIC: stem crack hour credit ability immense theme foot coyote act worth happy
HD PATH: m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x...ca51B12eEe35BAF6AF8339BEddb8A10315D6F4BE	99.89 ETH	38	0

au niveau des Blocks :

BLOCK	MINED ON	GAS USED	TRANSACTIONS
38	2023-05-08 15:03:50	322646	1 TRANSACTION
37	2023-05-08 15:03:50	402414	1 TRANSACTION

l'enregistrement au niveau des transactions :

TX HASH	FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE	CONTRACT CALL
0x8d9aa00db2dbc93e448fe0224425f34537b23d17b1796d4344f5b4302f55c7d2	0x...ca51B12eEe35BAF6AF8339BEddb8A10315D6F4BE	0x74221633497528cf7d25b6c430c084acec5bfafa	322646	0	
TX HASH	FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE	CONTRACT CREATION
0xc7030bd3d65eb96ff76dc6bf0afa501a28f4dbf3147c0138aad0de412e85ebf8	0x...ca51B12eEe35BAF6AF8339BEddb8A10315D6F4BE	0x74221633497528cf7d25b6c430c084acec5bfafa	402414	0	

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]
```