

# Compte Rendu Final

\*Groupe BROUILLET Arthur, POIRIER Louis, FROEHLI Jean-Baptiste

## Introduction

Pour rappel, notre projet était de créer un système de vote en ligne basé sur une blockchain. Pour cela nous avons utilisé python et la POO pour la blockchain, et le framework flask pour la parti web-frontend controlable depuis python.

## Le projet

Structure :

Aroborescence du projet

```
├── README.md
├── project
│   ├── __init__.py
│   ├── admin
│   │   ├── __init__.py
│   │   └── routes.py
│   ├── main
│   │   ├── __init__.py
│   │   └── routes.py
│   ├── static
│   │   ├── css
│   │   │   ├── admin
│   │   │   │   └── style.css
│   │   │   ├── style_about.css
│   │   │   ├── style_login.css
│   │   │   ├── style_para.css
│   │   │   ├── style_vote.css
│   │   │   ├── style_voted.css
│   │   │   └── style.css
│   │   ├── img
│   │   ├── js
│   │   │   ├── script_login.js
│   │   │   ├── script_para.js
│   │   │   └── script.js
│   └── templates
│       ├── admin
│       │   └── index.html
│       ├── about.html
│       ├── index.html
│       ├── login.html
│       └── presentation.html
```

```

|   |   | vote.html
|   |   | voted.html
|   |   | _init__.py
|   |   | config.py
|
| blockch.py
| datas.py
| run.py

```

## Fonctionnement

# Algorithmes

```
# Initialisation, création du bloc genesis et méthode de création d'un bloc et d'ajout à la chaine

def __init__(self):
    # Création de la chaine de blocs
    self.chain = []
    # Création du bloc genesis
    self.create_block(proof=1, previous_hash='0', vote='Default')

    # Création d'un bloc avec ses différentes infos (position, horodatage, vote, hash précédent, etc...)
def create_block(self, proof, previous_hash, vote):
```

```

block = {'index': len(self.chain) + 1,
        'timestamp': str(datetime.datetime.now()),
        'proof': proof,
        'vote' : vote,
        'previous_hash': previous_hash
        }
self.chain.append(block)
return block

```

Le chiffrement se fait par hashage et la vérification de la signature se fait par vérification de la correspondance entre le hash du bloc et le hash du bloc signé. Pour ce qui est de la méthode de hashage, nous avons utilisé l'algo SHA256 qui est un des plus utilisés et qui est assez rapide.

Le SHA256 correspond à un hash de 256 bits (32 octets) hexadécimaux.

```

# Méthode de hashage d'un bloc avec hashlib et json
def hash(self, block):
    encoded_block = json.dumps(block, sort_keys=True).encode()
    return hashlib.sha256(encoded_block).hexdigest()

```

Lors du vote, la fonction `mine_block` est appelée et va créer un bloc avec les données du vote et le hash du bloc précédent. Elle va ensuite vérifier que le hash du bloc créé commence par un nombre de 0 correspondant à la difficulté de la blockchain. Une fois que c'est le cas, le bloc est ajouté à la blockchain et la fonction renvoie le bloc créé.

```

# Méthode de preuve de travail (proof of work) et de vérification de la difficulté
def proof_of_work(self, previous_proof):
    new_proof = 1
    check_proof = False

    while check_proof is False:
        hash_operation = hashlib.sha256(
            str(new_proof**2 - previous_proof**2).encode()).hexdigest()
        if hash_operation[:5] == '00000':
            check_proof = True
        else:
            new_proof += 1

    return new_proof

# Fonction qui vient miner un nouveau bloc contenant les infos

def mine_block(vote):
    previous_block = blockchain.print_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
    previous_hash = blockchain.hash(previous_block)

```

```

block = blockchain.create_block(proof, previous_hash,vote)

response = {'message': 'Minage nouveau bloc',
            'index': block['index'],
            'timestamp': block['timestamp'],
            'proof': block['proof'],
            'previous_hash': block['previous_hash'],
            'vote': vote
            }

# Renvoie le code 200 comme ack et le bloc créé
return 200, response

```

## Autres algorithmes

On retrouve ensuite tout le temps la même structure au niveau du projet avec un fichier `routes.py` qui va définir les différentes routes et fonctions relatives. On retrouve aussi des fonctions de calcul dans le fichier `datas.py` pour permettre de calculer certaines choses mais la n'été pas le coeur du projet.

```

# Exemple de structure d'une route dans le fichier routes.py

# Route et méthodes posts et get si besoin
@app.route('/vote', methods=['GET', 'POST'])
def vote():
    vote = []

    # Récupération des données
    vote = request.form.getlist('vote')

    # Récupération des candidats
    get_cand = get_candidats(candidats)

    # Condition pour éviter de renvoyer une erreur si le vote est vide
    if vote != [] :
        voted = True
        return redirect(url_for("main.vote_block",
has_voted=voted,vote=get_candidats(candidats)[int(vote[0])]["candidat"])))

"""Fonction permettant de compléter la liste des candidats si il y en a plus ou
moins de 6, nombre que nous avons prévu a titre démonstratif"""

def get_candidats(candidats):
    if len(candidats) > 6 :
        return candidats[:6]
    elif candidats == []:
        for i in range(6):
            candidats.append({
                'candidat': 'Candidat Inconnu',

```

```
        'parti': 'Inconnu',
        'vote': 0,
        'presence': 0,
        'image' : r"lien web image/chemin image"
    })
    return candidats
elif len(candidats) < 6 :
    for i in range(6-len(candidats)):
        candidats.append({
            'candidat': 'Candidat Inconnu',
            'parti': 'Inconnu',
            'vote': 0,
            'presence': 0,
            'image' : r"lien web image/chemin image"
        })
    return candidats
else :
    return candidats
```

## Répartition et méthodes

Nous avons beaucoup travaillé par discord et github avançant chacun sur certaines pages avant de les mettre en commun pour les rendre cohérentes entre elles. Pour les parties plus complexe comme l'algo relative à la blockchain, c'était surtout un travail en commun entre Louis et moi et nous expliquions ensuite le fonctionnement et l'intérêt de chaque partie à Arthur. Pour le reste, c'était variable car cela dépendait du besoin des différentes pages. La difficulté principale était quand à elle de relier les différentes parties front et back entre elles et de faire que lorsque on vote, cela vient s'ajouter chaîne qui est ensuite analysable depuis un autre endroit, etc...