ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

CÔNG NGHỆ PHẦN MỀM (MỞ RỘNG)

Báo cáo

Ứng dụng Blockchain xây dựng từ điển bất động sản

GVHD: Quản Thành Thơ
SV: Nguyễn Khoa Gia Cát 1912749
Huỳnh Tấn Luân 1914054
Trịnh Nguyên Bảo Tuấn 1912371

TP. HỒ CHÍ MINH, THÁNG 11/2021

Mục lục

1 Hiện thực các hàm

Trong 2 tuần qua, nhóm đã viết các hàm để tương tác với server blockchain dựa trên data được cung cấp (`rv_listing_dataset.json`. Trong đó:

- Hàm add() thêm 1 asset là object trong file json.
- Hàm get_asset () lấy dữ liệu của một object dựa vào id.
- Hàm update() cập nhật một object đã thêm.

Để thuận tiện cho việc hiện thực thì nhóm đã chuyển đổi file `.jsonl` thành sang định dạng `.json`.

1.1 add(dict_values)

Hàm này thêm một asset mới vào hệ thống.
Input: dict_values là 1 object dạng json miêu tả asset cần thêm vào.
Output: return id của transaction tương ứng.
Hiện thực:

```
## dict_values:  a line in file json
def add(dict_values):
    addAsset = {}
    addAsset['data']=dict_values

    prepare_creation = bdb.transactions.prepare(
        operation='CREATE',
        signers=user.public_key,
        asset=addAsset,
    )

    fulfilled_creation = bdb.transactions.fulfill(
        prepare_creation, user.private_key
    )

    # Send the transaction to BigchainDB
    sent_creation = bdb.transactions.send_commit(fulfilled_creation)
    return fulfilled_creation["id"]
```

1.2 set_asset_fields(asset, fields)

Hàm này dùng để phụ trợ cho hàm get_asset(asset_id).
Input: asset là một object dạng json, fields cũng là một object dạng json có các key là con của asset.
Sau khi thực hiện thì các trường trong asset sẽ thay đổi tương ứng theo fields.
Hiện thực:

```
def set_asset_fields(asset, fields):
  if fields == {}:
    return
  for field in fields.keys():
```

```
    value = fields[field]
    if type(value) is dict:
      set_asset_fields(asset[field], value)
    else:
      asset[field] = value
```

1.3 get_asset(asset_id)

Hàm này dùng để lấy thông tin của một asset ở thời điểm hiện tại.
Input: asset_id là id của asset cần lấy.
Output: return thông tin asset tương ứng
Hiện thực:

```
def get_asset(asset_id):
  try:
    transactions = bdb.transactions.get(asset_id=asset_id)
  except:
    print("Unable to get transactions!")
    return
  if transactions == [] or transactions is None:
    return {}
  asset = {}
  for transaction in transactions:
    if asset == {}:
      asset = transaction["asset"]["data"]
    else:
      set_asset_fields(asset, transaction["metadata"]["data"])
  return asset
```

1.4 update(asset_id, input_data_changed)

Hàm này thay đổi thông tin của asset cần thêm, bằng cách ghi lại những thay đổi trong trường metadata của transaction mới.
Input: asset_id là id của asset cần update, input_data_changed miêu tả sự thay đổi thông tin của asset (có dạng json chứa các trường cần thay đổi tương ứng với asset ban đầu).
Output: return id của transaction tương ứng.
Hiện thực:

```
def update(asset_id, input_data_changed):
    transfer_asset = {
        'id' : asset_id
    }
    data_changed = {}
    data_changed['data']=input_data_changed

    output_index = 0

    transactions = bdb.transactions.get(asset_id=asset_id)
    last_transaction = transactions[-1]
```

```
output = last_transaction['outputs'][output_index]

transfer_input = {
    'fulfillment': output['condition']['details'],
    'fulfills': {
    'output_index': output_index,
    'transaction_id': last_transaction['id'],
    },
    'owners_before': output['public_keys'],
}

prepare_creation = bdb.transactions.prepare(
    operation='TRANSFER',
    asset = transfer_asset,
    inputs = transfer_input,
    metadata= data_changed,
    recipients=user.public_key
)

last_transaction = bdb.transactions.fulfill(
    prepare_creation,
    private_keys=user.private_key
)

sent_creation = bdb.transactions.send_commit(last_transaction)
return last_transaction["id"]
```

2 Sử dụng trong thực tế

Việc áp dụng hợp lý các hàm đã hiện thực ở trên sẽ xây dựng được ứng dụng từ điển bất động sản. Dưới đây là một ví dụ về việc áp dụng các hàm trên:

- Trước tên, để thay đổi một đối tượng (ở đây là thông tin của căn hộ), cần có một UI (User Interface) cho phép người dùng nhập vào các thông tin cần tìm kiếm của căn hộ. Sử dụng các hàm cần thiết để lấy được thông tin của các căn hộ và id tương ứng của asset. Sau đó dựa vào thông tin của các căn hộ để hiển thị cho người dùng.

```
data_searched = bdb.assets.get(search="D'Lusso")

if data_searched == []:
  print("not found")
else:
  for item in data_searched:
    print(json.dumps({"data" : get_asset(item["id"]), "id" : item["id"]},
        ensure_ascii=False, indent=4))
```

- Sau đó, sẽ có một UI cho phép người dùng chọn căn hộ mong muốn. Dựa vào đối tượng mà người dùng chọn, suy ra được asset id tương ứng. Khi đã có asset id thì có thể gọi các hàm đã hiện thực ở trên tùy theo nhu cầu sử dụng.

```
asset_chosen_id = data_searched[-1]["id"]

update(asset_chosen_id, {"furniture_status" : "Full"})
```

3 Nhận xét

- Việc quản lý dữ liệu đảm bảo tính minh bạch.
- Việc lấy thông tin của asset ở thời điểm hiện tại tốn nhiều thời gian nếu có nhiều sự thay đổi trên asset đó.
- Phần hiện thực ở trên chưa bao gồm phân quyền truy cập cho các user khác nhau.

Tài liệu