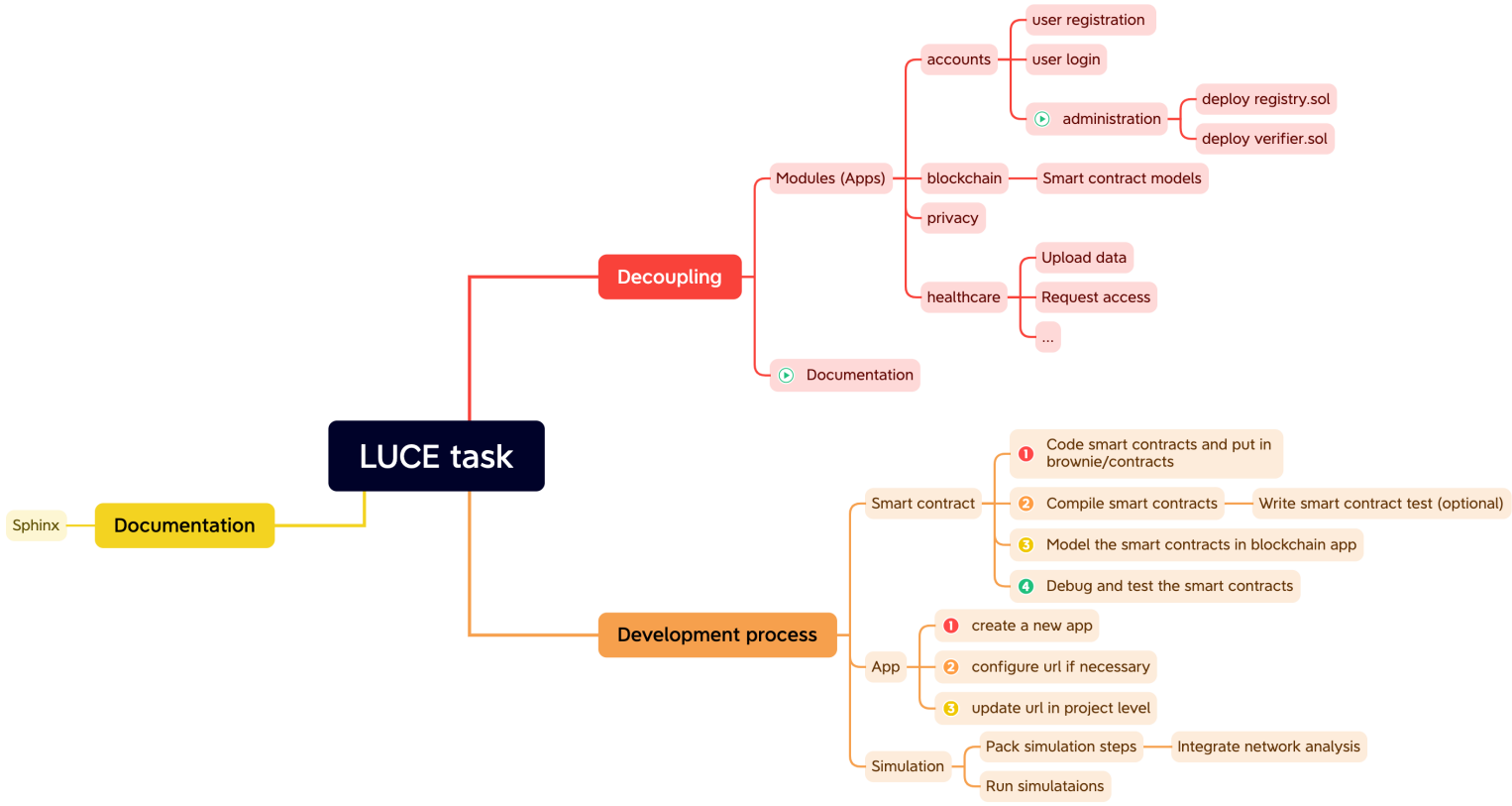
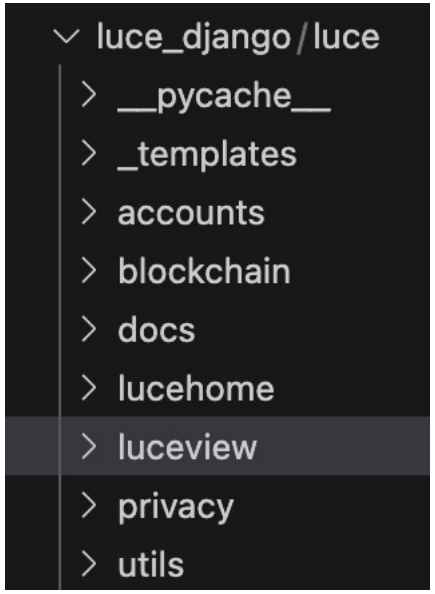
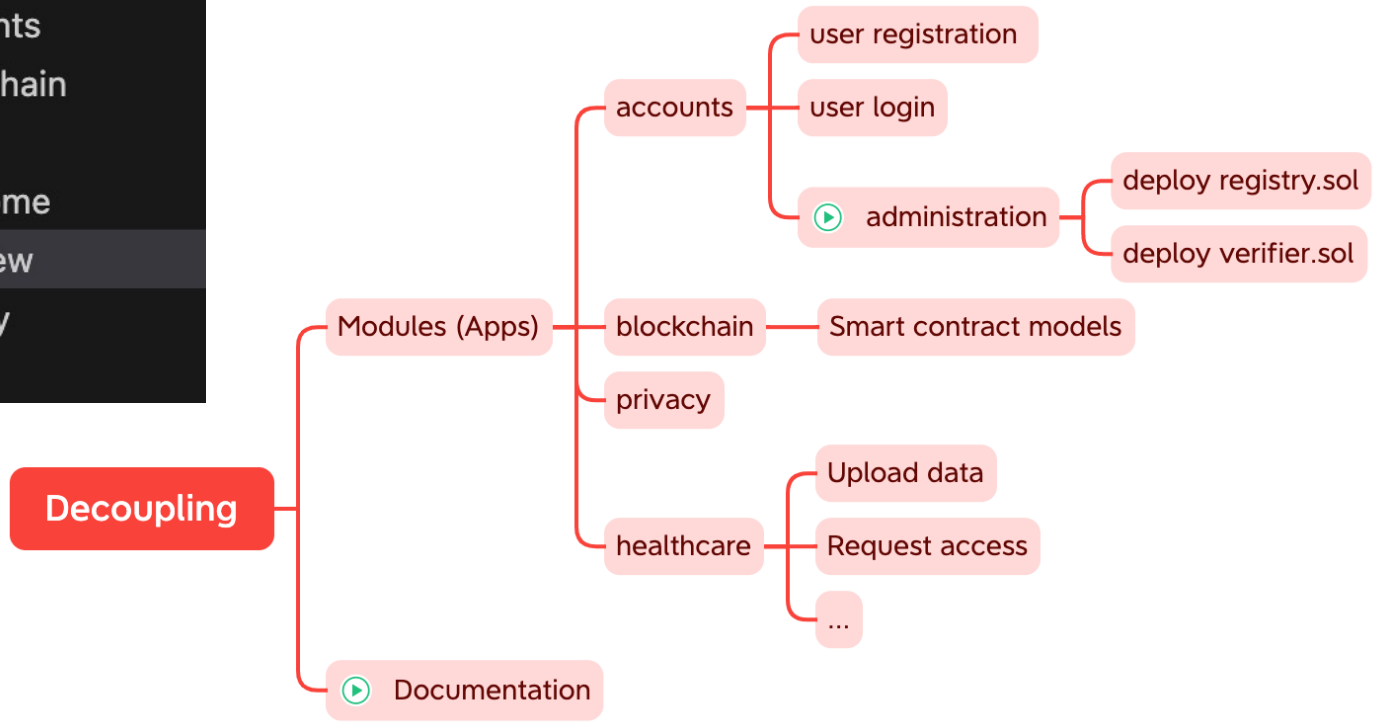
LUCE task

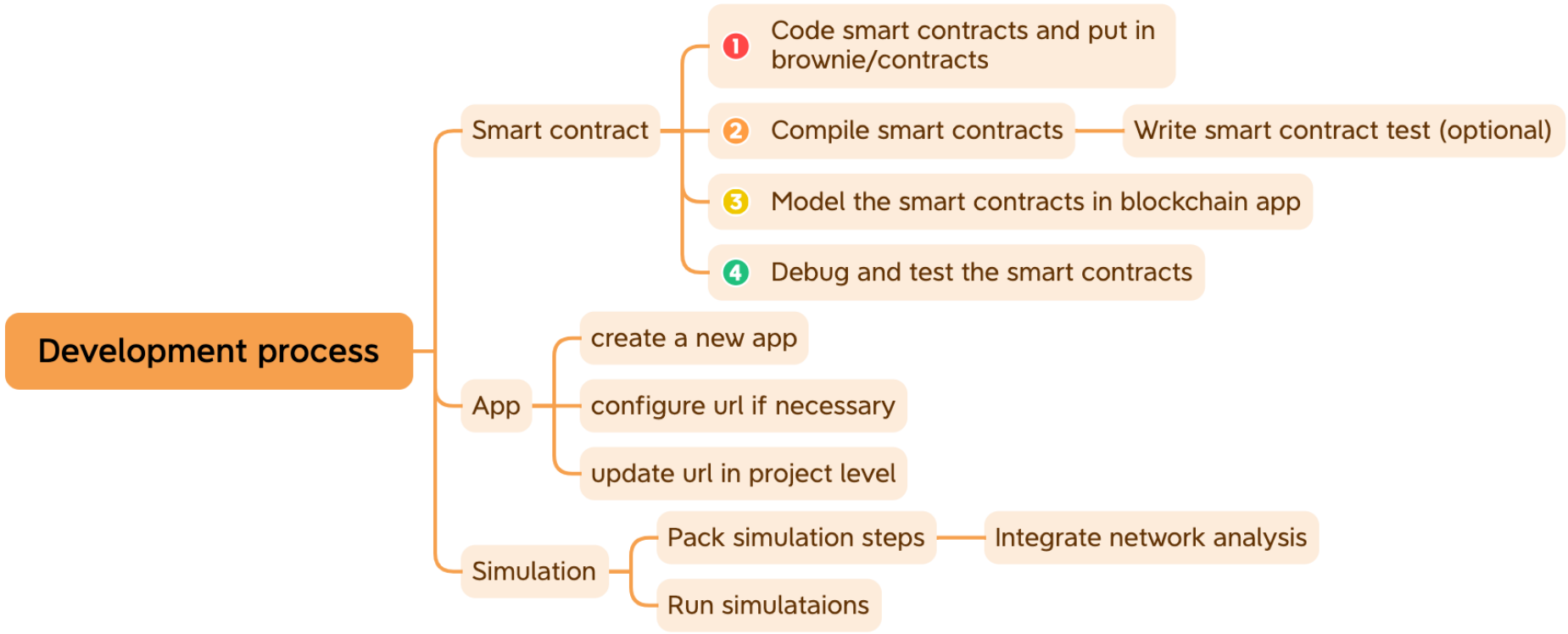

LUCE task
Documentation
Sphinx
Decoupling
Modules (Apps)
accounts
user registration
user login
administration
deploy registry.sol
deploy verifier.sol
blockchain
Smart contract models
privacy
healthcare
Upload data
Request access
...
Documentation
Development process
Smart contract
1 Code smart contracts and put in brownie/contracts
2 Compile smart contracts
Write smart contract test (optional)
3 Model the smart contracts in blockchain app
4 Debug and test the smart contracts
App
1 create a new app
2 configure url if necessary
3 update url in project level
Simulation
Pack simulation steps
Integrate network analysis
Run simulataions

Decoupling

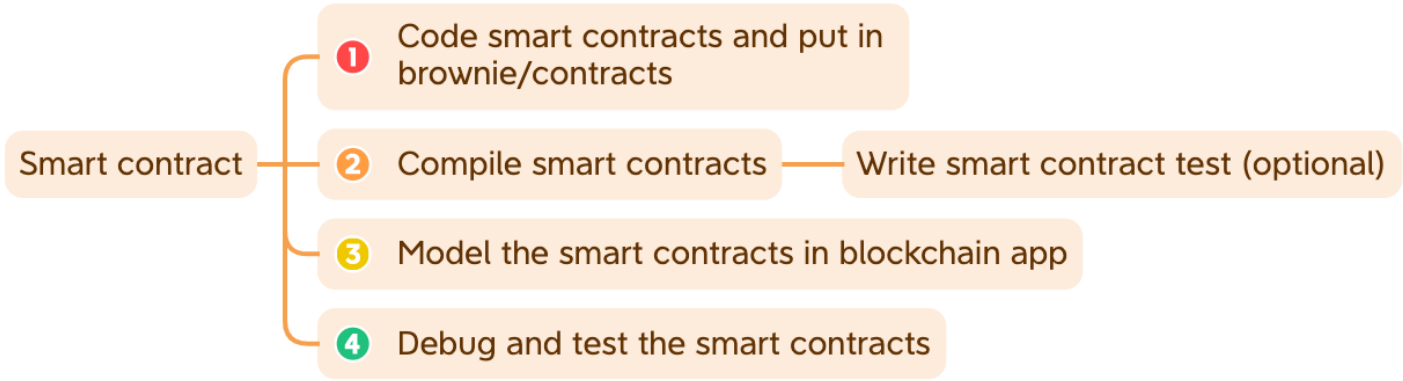

luce_django / luce
__pycache__
_templates
accounts
blockchain
docs
lucehome
luceview
privacy
utils


Decoupling
Modules (Apps)
accounts
user registration
user login
administration
deploy registry.sol
deploy verifier.sol
blockchain
Smart contract models
privacy
healthcare
Upload data
Request access
...
Documentation

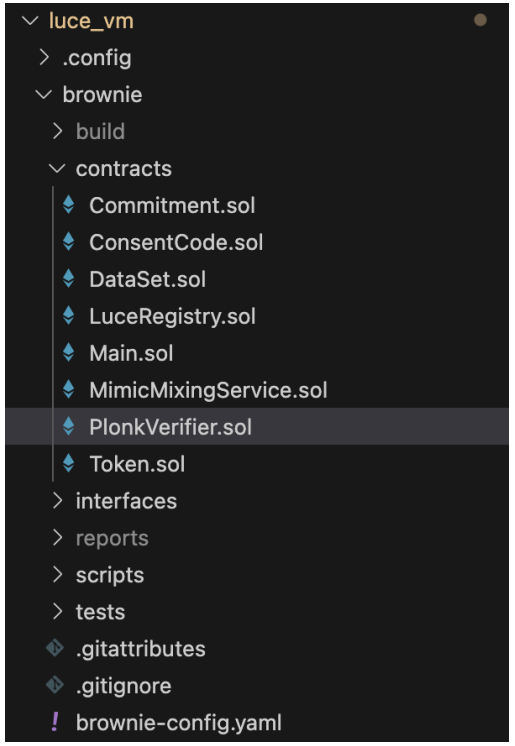
Development process



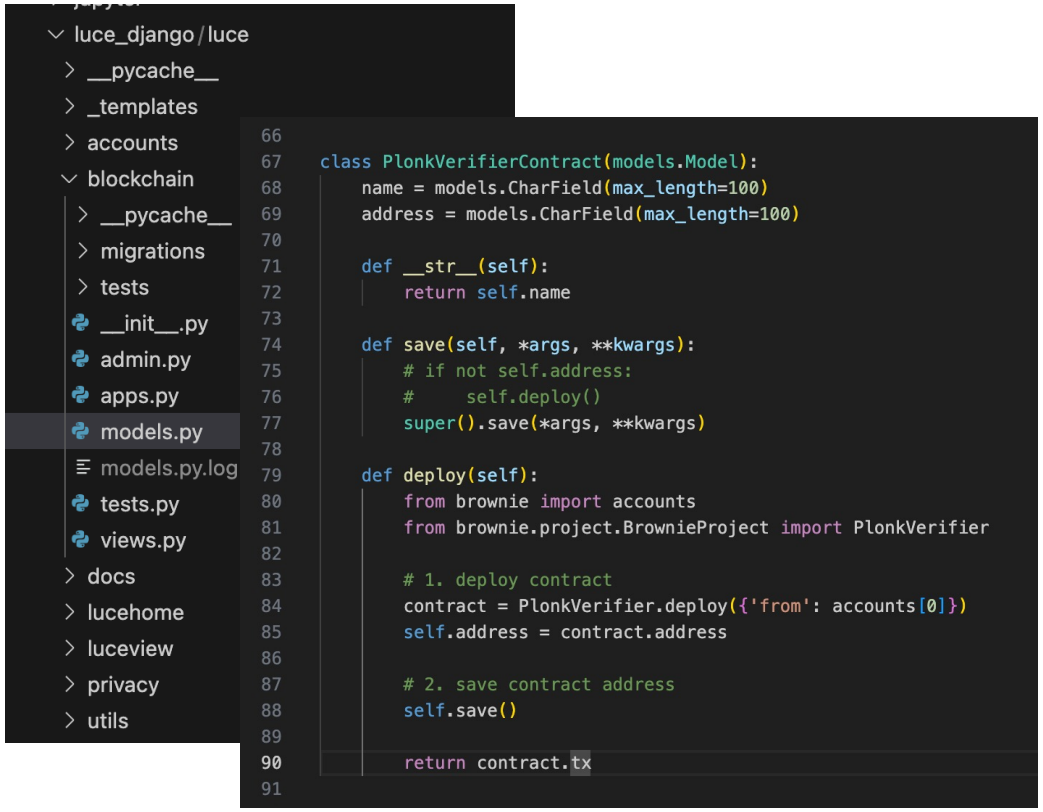
Development process – Smart contract



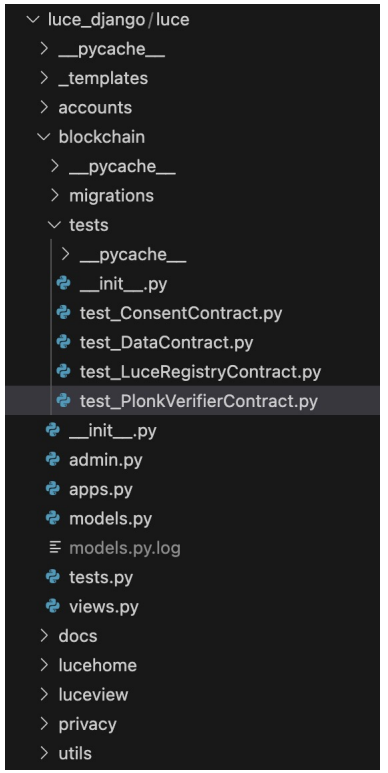
Development process – Smart contract



1. Code smart contracts and put in brownie/contracts
2. Compile smart contracts with brownie

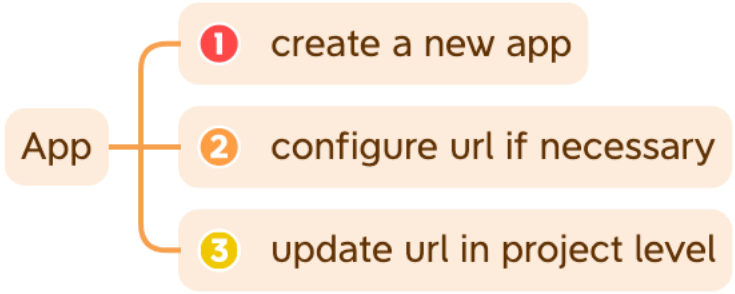


3. Model the smart contracts in blockchain app

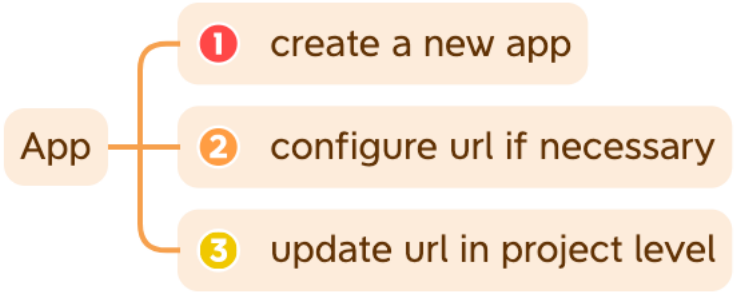


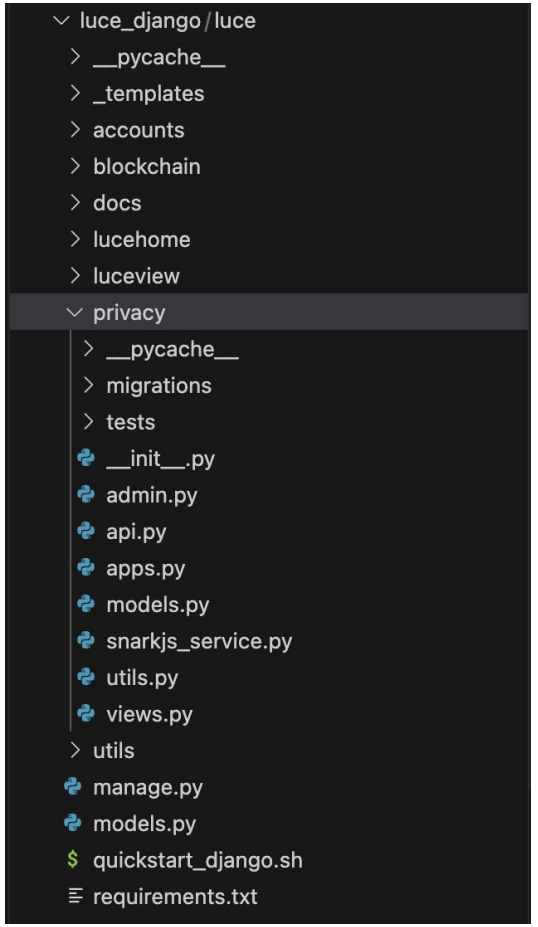
4. Debug and test the smart contracts

Development process – New app



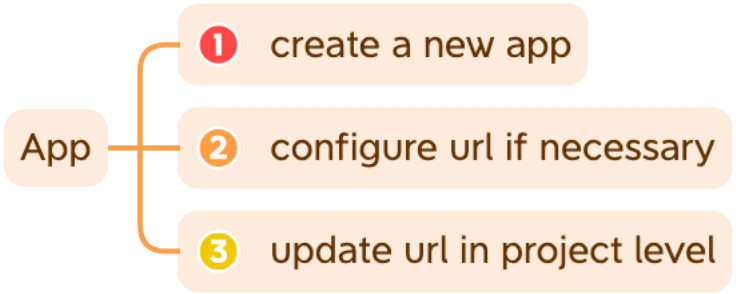
Development process – New app

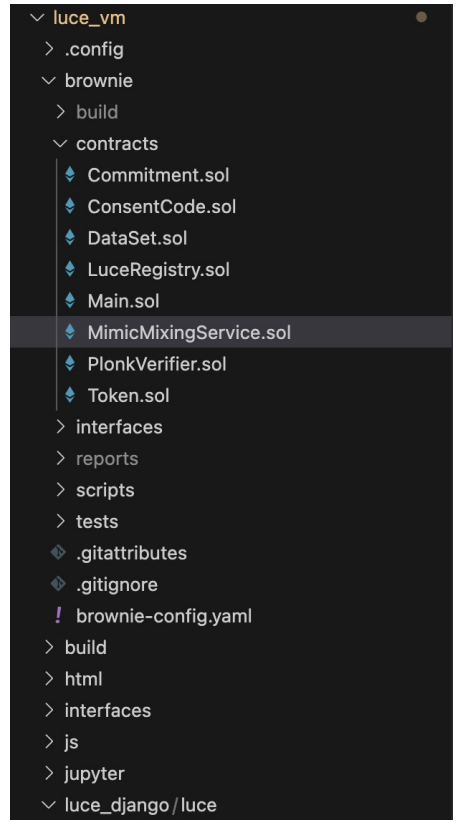




1. Create a new app - create

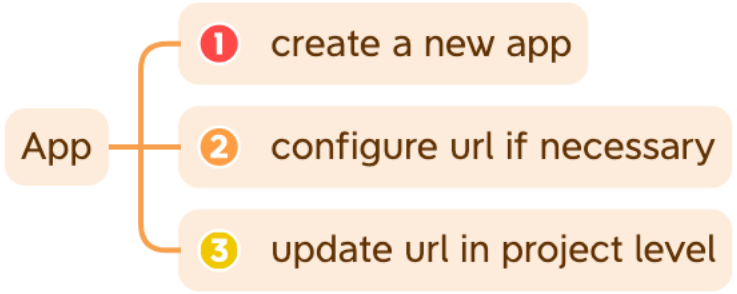
Development process – New app

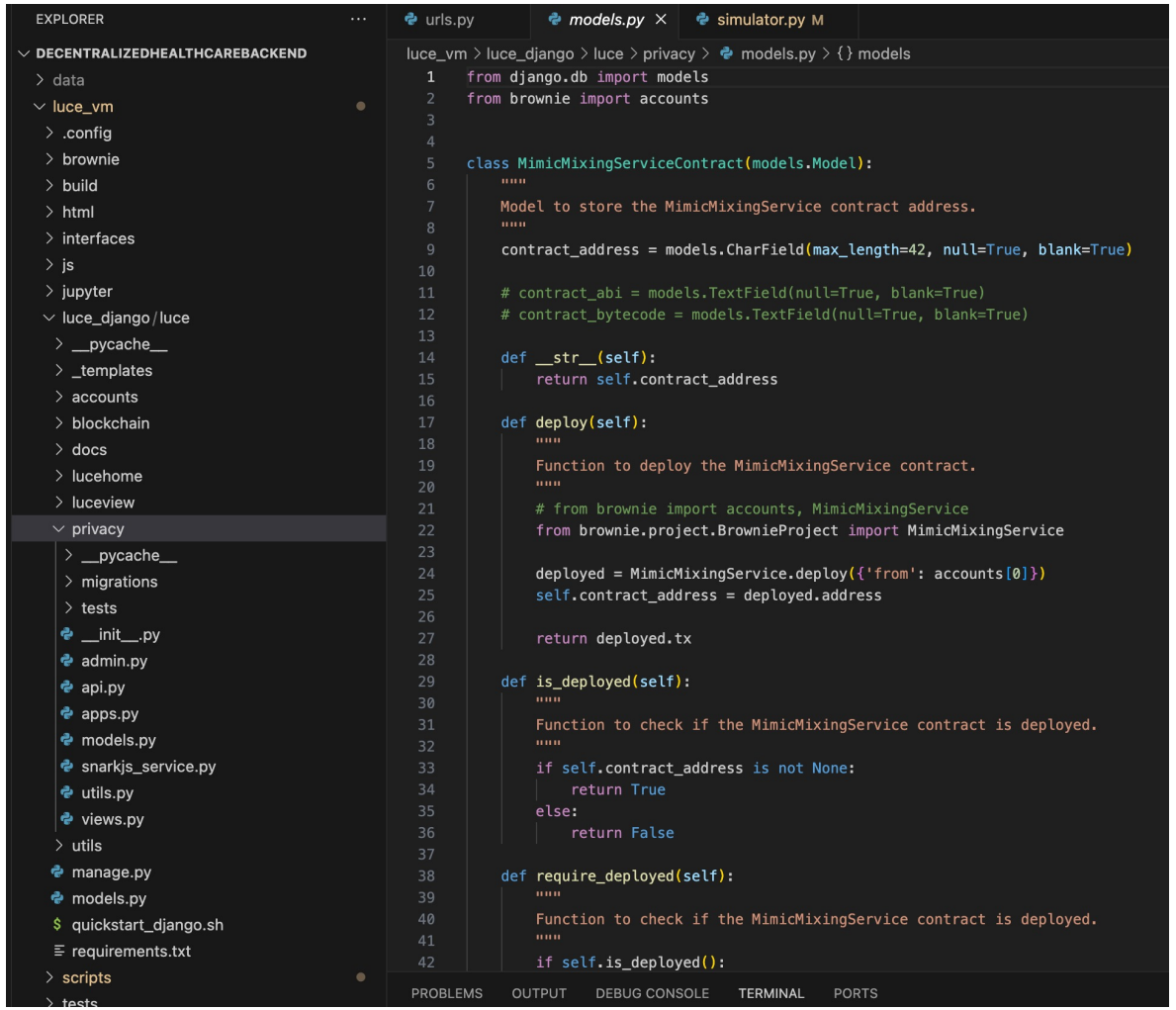




1. Create a new app – create the smart contract

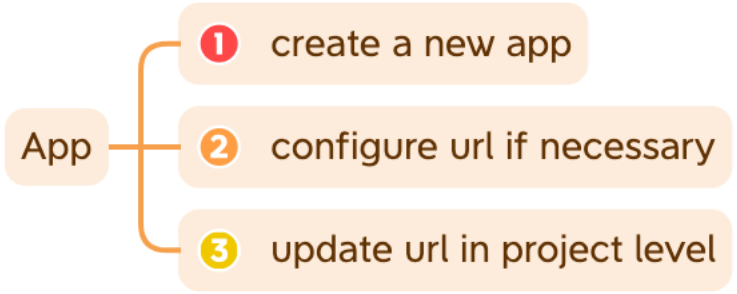
Development process – New app

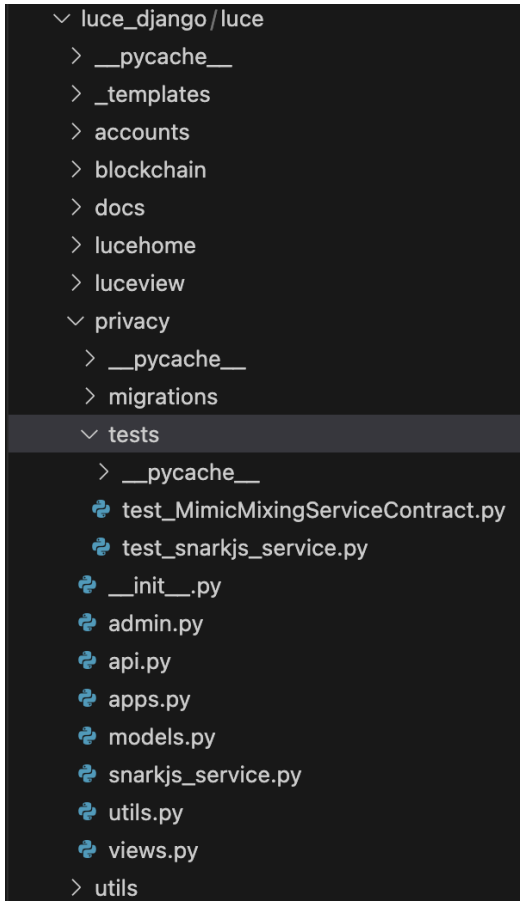




1. Create a new app – Model the smart contract

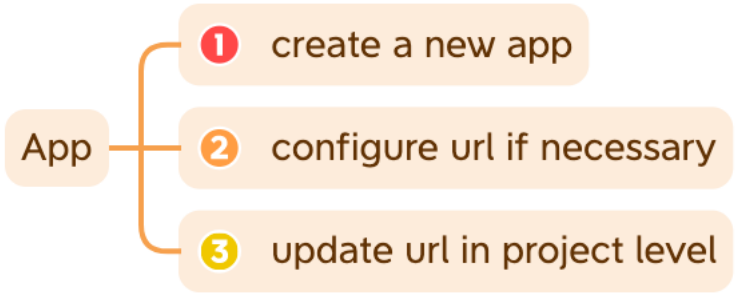
Development process – New app





1. Create a new app - Test

Development process – New app




```
4
5   urlpatterns = [
6       path('register/', UserRegistration.as_view(), name='register'),
7       path('<int:id>/', PublicUserInfoView.as_view()),
8       path('authenticated/', PrivateUserInfoView.as_view()),
9       path('authenticated/update/', UserUpdateView.as_view()),
10      path('all/', UserListView.as_view()),
11      path('login/', ObtainAuthToken.as_view()),
12  ]
```

2. Configure urls in app

```
from django.conf import settings

urlpatterns = [
    path('docs/', include('sphinxdoc.urls')),
    path('user/', include('accounts.urls')),
    path('contract/', include('luceview.urls')),
    path('admin/', include('luceview.urls')),

    # path('user/register/', UserRegistration.as_view()),
    # path('user/<int:id>/', PublicUserInfoView.as_view()),
    # path('user/authenticated/', PrivateUserInfoView.as_view()),
    # path('user/authenticated/update/', UserUpdateView.as_view()),
    # path('user/all/', UserListView.as_view()),
    # path('user/login/', ObtainAuthToken.as_view()),
    path('admin/deployRegistry/', LuceRegistryView.as_view()),
    # path('contract/all/', ContractsListView.as_view()),
    # path('contract/dataUpload/', UploadDataView.as_view()),
    # path('contract/requestAccess/', RequestDatasetView.as_view()),
    # path('contract/getLink/', GetLink.as_view()),
    # path('contract/<int:id>/', RetrieveContractByUserIDView.as_view()),
    # path('contract/search/', SearchContract.as_view()),
]
```

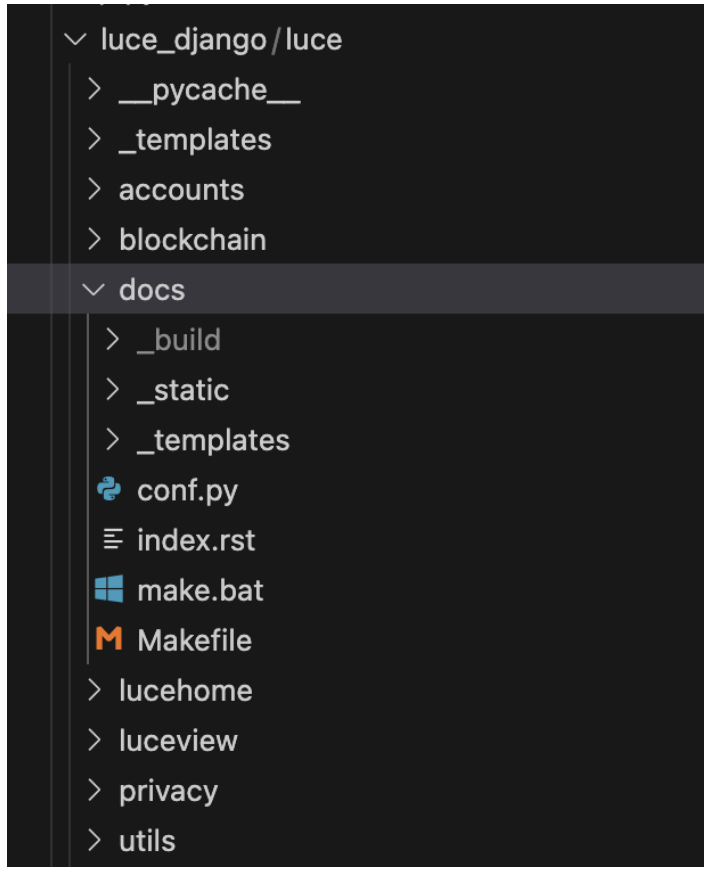
3. Updata urls in project level

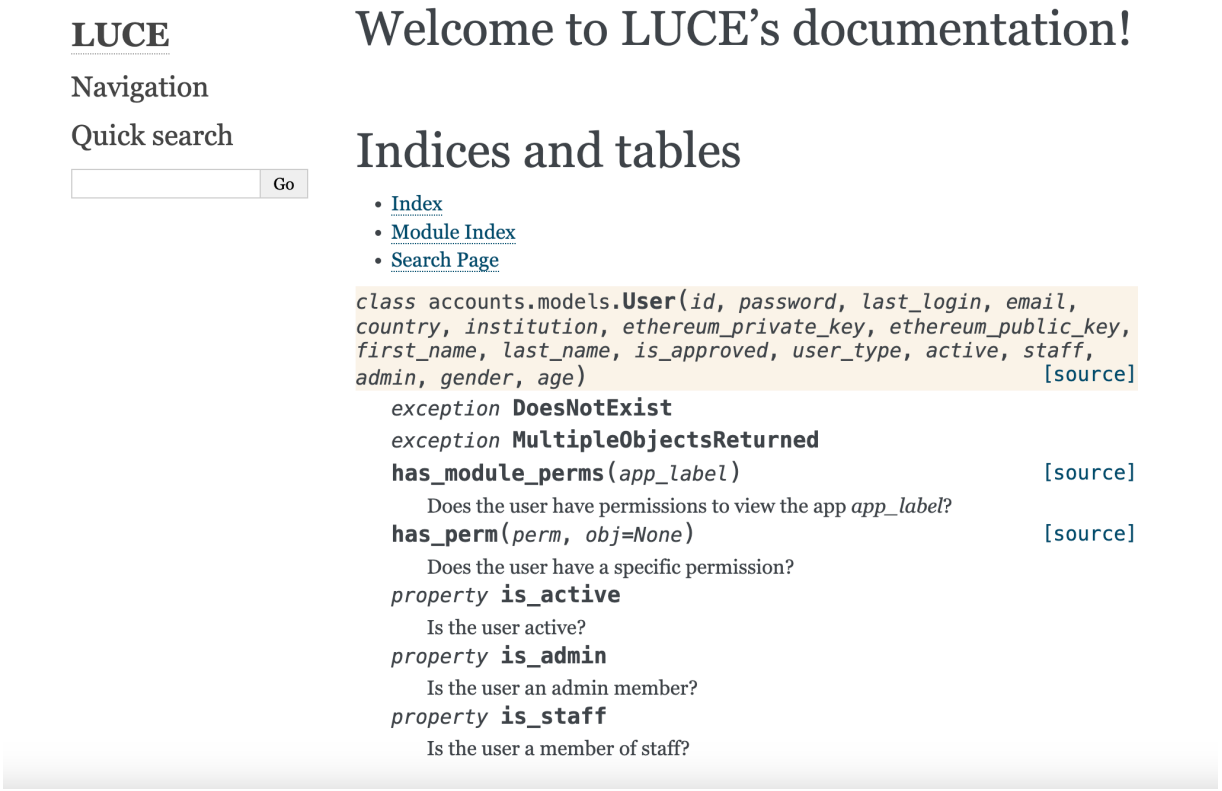
Development process – Simulation

```
21
22  class Simulator:
23      def __init__(self, num_of_users):
24          self.http = urllib3.PoolManager()
25          self.urls = {
26              "login": "http://127.0.0.1:8000/user/login/",
27              "register": "http://localhost:8000/user/register/",
28              "upload_data": "http://localhost:8000/contract/dataUpload/",
29              "deploy_registry": "http://localhost:8000/admin/deployRegistry/"
30          }
31
32          self.user = generate_users(num_of_users)
33          self.directed_graph = nx.DiGraph()
34
35          self.switcher = {1: self._senario_1}
36
37      def run(self, senario_num):
38          self.switcher.get(senario_num, lambda: "Invalid senario")()
39
40      def _login(self, url, user): ...
49
50      def _register(self, registration_url, registration_data): ...
60
61      def _upload_data(self, upload_url, data, token): ...
74
75      def _deploy_registry(self, url, admin_token): ...
84
85      def _address_to_label(self, address): ...
87
88      def _senario_1(self): ...
139
140
141 s = Simulator(3)
142 s.run(1)
```

1. Pack the simulation steps
2. Add senario into switcher
3. Run simulation

Documentation





LUCE

Navigation

Quick search

Go

Welcome to LUCE's documentation!

Indices and tables

- Index
- Module Index
- Search Page

class accounts.models.**User**(*id*, *password*, *last_login*, *email*, *country*, *institution*, *ethereum_private_key*, *ethereum_public_key*, *first_name*, *last_name*, *is_approved*, *user_type*, *active*, *staff*, *admin*, *gender*, *age*) [source]

exception **DoesNotExist**

exception **MultipleObjectsReturned**

has_module_perms(*app_label*) [source]

Does the user have permissions to view the app *app_label*?

has_perm(*perm*, *obj=None*) [source]

Does the user have a specific permission?

property **is_active**

Is the user active?

property **is_admin**

Is the user an admin member?

property **is_staff**

Is the user a member of staff?