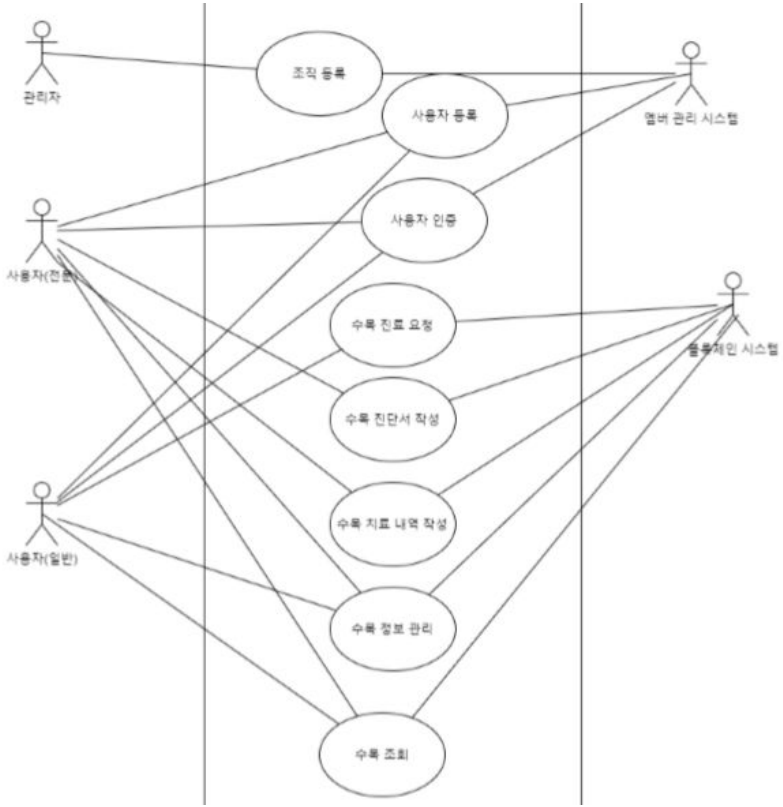
수목관리 시스템
(+BlockChain3.0)

[WeekdaysIdea] 박지은 이재호

블록체인 시스템 설계 & 구축

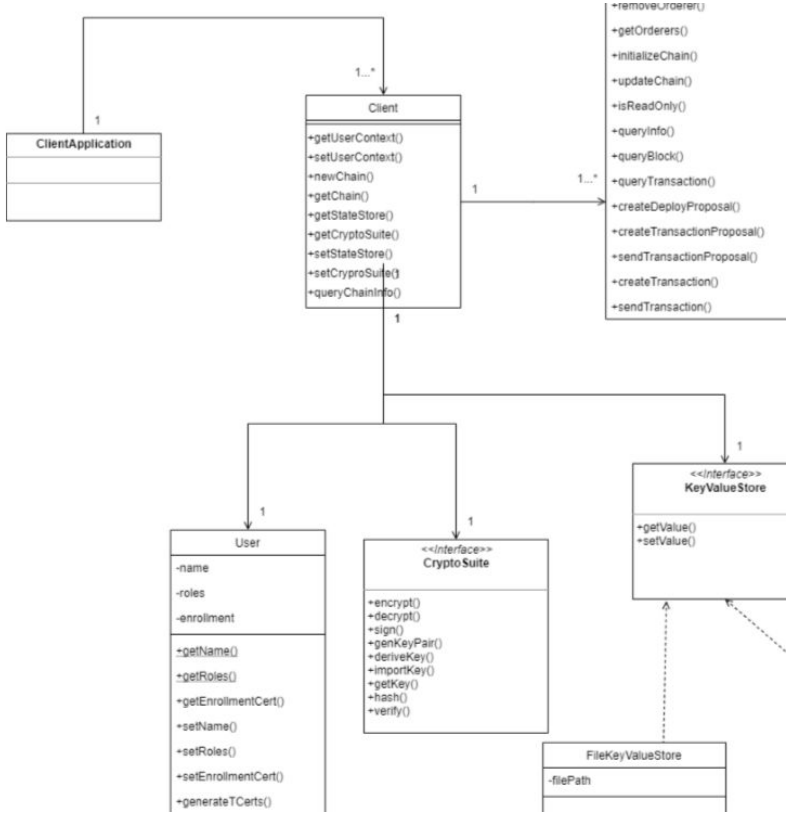
- 구성 요소
- 시스템 구조
- 상호작용(이벤트 흐름)
- 특성 및 고려사항
- 필요 기술 파악 및 역할 분담

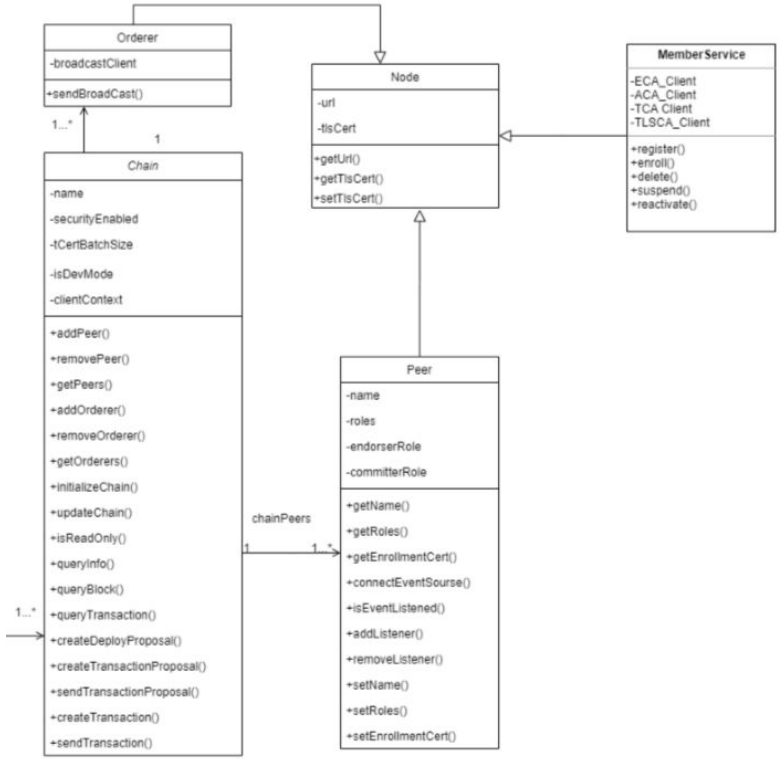
Usecase 다이어그램



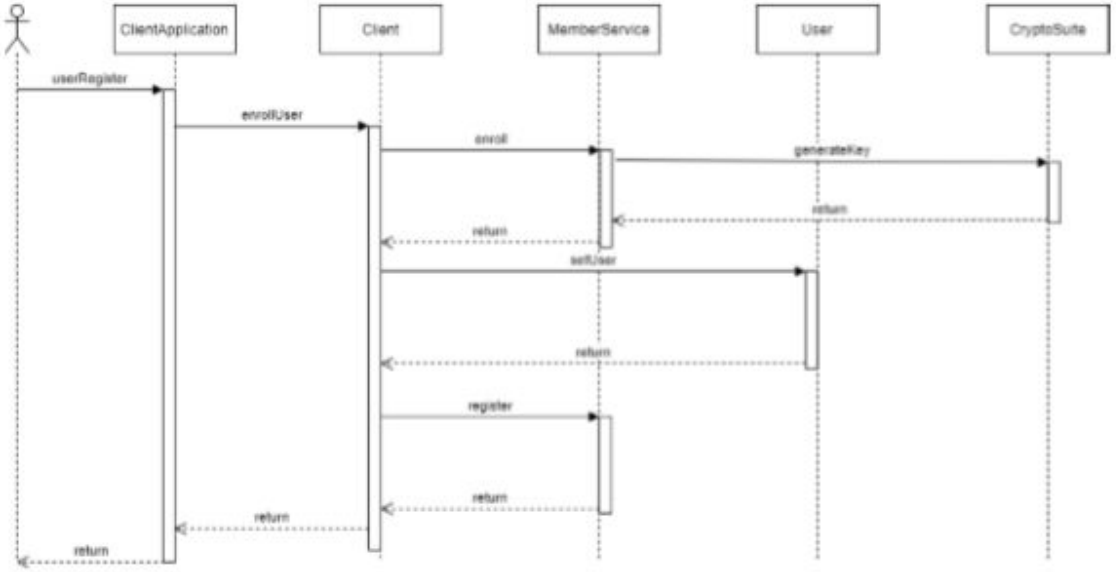

```
# 도메인
treeconnector.com

# 도메인 별 노드
orderer

# 조직
doctors / treats / condomngs / individuals

# 조직별 노드
ca / peer0
```

클래스 다이어그램

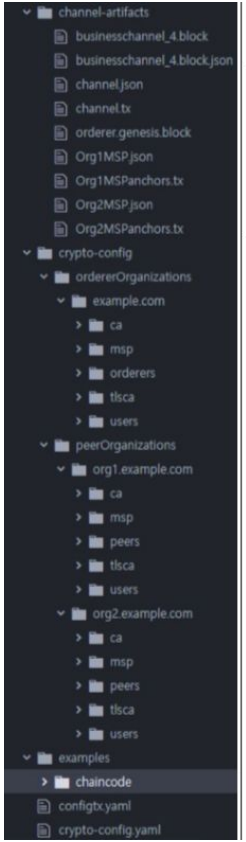

ClientApplication
Client
+getUserContext()
+setUserContext()
+newChain()
+getChain()
+getStateStore()
+getCryptoSuite()
+setStateStore()
+setCryproSuite()
+queryChainInfo()
+removeOrderer()
+getOrderers()
+initializeChain()
+updateChain()
+isReadOnly()
+queryInfo()
+queryBlock()
+queryTransaction()
+createDeployProposal()
+createTransactionProposal()
+sendTransactionProposal()
+createTransaction()
+sendTransaction()
User
-name
-roles
-enrollment
+getName()
+getRoles()
+getEnrollmentCert()
+setName()
+setRoles()
+setEnrollmentCert()
+generateTCerts()
<<Interface>>
CryptoSuite
+encrypt()
+decrypt()
+sign()
+genKeyPair()
+deriveKey()
+importKey()
+getKey()
+hash()
+verify()
<<Interface>>
KeyValueStore
+getValue()
+setValue()
FileKeyValueStore
-filePath


Orderer
-broadcastClient
+sendBroadCast()
Node
-url
-tlsCert
+getUrl()
+getTlsCert()
+setTlsCert()
MemberService
-ECA_Client
-ACA_Client
-TCA Client
-TLSCA_Client
+register()
+enroll()
+delete()
+suspend()
+reactivate()
Chain
-name
-securityEnabled
-tCertBatchSize
-isDevMode
-clientContext
+addPeer()
+removePeer()
+getPeers()
+addOrderer()
+removeOrderer()
+getOrderers()
+initializeChain()
+updateChain()
+isReadOnly()
+queryInfo()
+queryBlock()
+queryTransaction()
+createDeployProposal()
+createTransactionProposal()
+sendTransactionProposal()
+createTransaction()
+sendTransaction()
chainPeers
Peer
-name
-roles
-endorserRole
-committerRole
+getName()
+getRoles()
+getEnrollmentCert()
+connectEventSourse()
+isEventListened()
+addListener()
+removeListener()
+setName()
+setRoles()
+setEnrollmentCert()

시퀀스 다이어그램

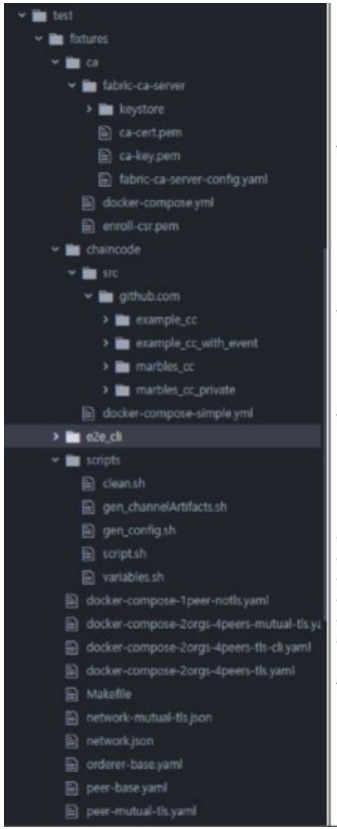

ClientApplication
Client
MemberService
User
CryptoSuite
userRegister
enrollUser
enroll
generateKey
return
return
setUser
return
register
return
return
return

프로젝트 구조



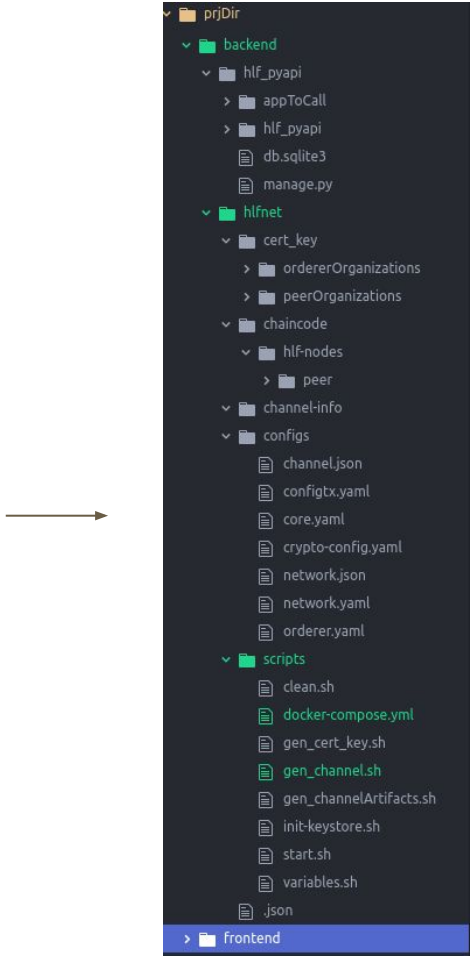
+ channel-artifacts
  - 채널.json
  - 채널.tx
  - 채널별블록.block
  - 채널별블록설정.json
  - 조직별MSP.json
  - 조직별MSPanchers.tx

+ crypto-config
  (설정.yaml파일에 따라 조직 별로 생성)
  + 조직명
    + 도메인명
      + ca
        - private key 파일
        - RSA 인증서(.pem) 파일
      + tlsca
        - private key 파일
        - RSA 인증서(.pem) 파일
      + msp
        + admincerts
        + cacerts
        + tlscacerts
      + 노드들(orderers/peers)
        + 도메인명
        (orderer는 서브도메인명.도메인)
        (peer는 계정명@도메인)
          + msp
            + admincerts
            + cacerts
            + keystore
            + signcerts
            + tlscerts
          + tls



+ ca(인증서버)
  - 도커 컴포즈 파일(컨테이너 구동설정)
  - RSA 키 파일(.pem) : CSR 등록
  + fabric-ca-server(컨테이너)
    - 컨테이너 설정.yaml 파일(포트,TLS)
    - RSA 키 파일(.pem) : cert, key 2개
    + keystore
      - private key 파일들
+ chaincode(체인코드)
  + src
    + 코드디렉터리
      + 디렉터리1
      + 디렉터리2
+ scripts(블록체인 구동)
  - gen_channelArtifacts.sh(채널)
  - gen_config.sh(인증서)
  - script.sh(블록체인 net 구동)
  - variables.sh(환경변수 설정)
  - clean.sh(제거)
- docker-compose-조직피어구동명.yaml
- network.json(SDK로드에서 참조용)
- orderer-base.yaml(오더러 설정)
- peer-base.yaml(피어 설정)
- network-mutual-tls.json(TLS설정)
- peer-mutual-tls.json(TLS설정)

+ client용 프로젝트 폴더(SDK코드)



도커, 네트워크 구축

```
# 포트 규칙
kafka brokers 노드 = 22222
네트워크별 orderer 노드 = 33333, 33334, 33335 ... : 33333
couchdb 노드 = 44444, 44445, 44446, ... : 44444
조직별 CA 노드 = 51999, 52999, 53999 ... : 50999
조직별 peer 노드 = 51xx1, 51xx2, 51xx3 ... : 50000~50009
```

```
- "50000"  # Rest
- "50001"  # Grpc        --> 앵커 피어(조직 간 가십 프로토콜)
- "50002"  # Peer CLI
- "50003"  # Peer Event
- "50004"  # eCAP
- "50005"  # eCAA
- "50006"  # tCAP
- "50007"  # eCAA
- "50008"  # tlsCAP
- "50009"  # tlsCAA
```

```
# 모든 컨테이너의 초기 유저네임:패스워드
admin:passwd

# 워킹 디렉토리 환경변수 설정
```

```
sudo vi /etc/environment
        PRJ_DIR="/워킹디렉토리/경로"
esc+w+q
source /etc/environment
echo $PRJ_DIR
cd $PRJ_DIR
```

```
# 하이퍼레저 초기 네트워크 컨테이너 생성
```

```
cd $PRJ_DIR/scripts
sh clear.sh
sh start.sh
docker ps -a
```

config 파일, 환경변수

```
# 컨테이너 디렉터리
/etc/hyperledger/channel-info : 채널 설정정보 --> orderer는 etc가 아니라 var였는데 내가 바꿈
/etc/hyperledger/msp/<노드명> : 다른 노드들(peer)의 MSP 정보
/etc/hyperledger/<노드명> : 내 노드의 cert_key 정보
/opt/chaincode/hlf-nodes/<노드명> : 컨테이너들의 워킹 디렉토리 for 체인코드

# MSP 디렉터리
호스트-노드 : /cert_key/<조직그룹명>/<조직도메인명>/<노드종류>/<노드명.조직도메인명>
호스트-유저 : /cert_key/<조직그룹명>/<조직도메인명>/users
컨테이너-노드 : /etc/hyperledger/msp/<노드명>
컨테이너-유저 : /etc/hyperledger/msp/users

# TLS 디렉터리-orderer(=노드명)
호스트 : /cert_key/<조직그룹명>/<조직도메인명>/<노드종류>/<노드명.조직도메인명>/tls
오더러 : /etc/hyperledger/msp/<노드명>/tls
키 : ORDERER_GENERAL_TLS_PRIVATEKEY
        =/etc/hyperledger/msp/<노드명>/tls/server.key

# TLS 디렉터리-peer(노드명)
호스트 : /cert_key/<조직그룹명>/<조직도메인명>/<노드종류>/<노드명.조직도메인명>/tls
피어 : /etc/hyperledger/tls
키 : CORE_PEER_TLS_KEY_FILE
        =/etc/hyperledger/tls/server.key
```

```
# 장고 API 컨테이너 접속
```
docker exec -it hlf-py-api

echo $FABRIC_CFG_PATH
/etc/hyperledger/fabric

/etc/hyperledger# ls
cert_key channel-info fabric

/etc/hyperledger/fabric# ls
configtx.yaml core.yaml msp orderer.yaml

/etc/hyperledger/fabric/msp# ls
admincerts cacerts config.yaml keystore

/opt/chaincode/hlf-nodes/peer/cert_key# ls
ordererOrganizations peerOrganizations
```
```

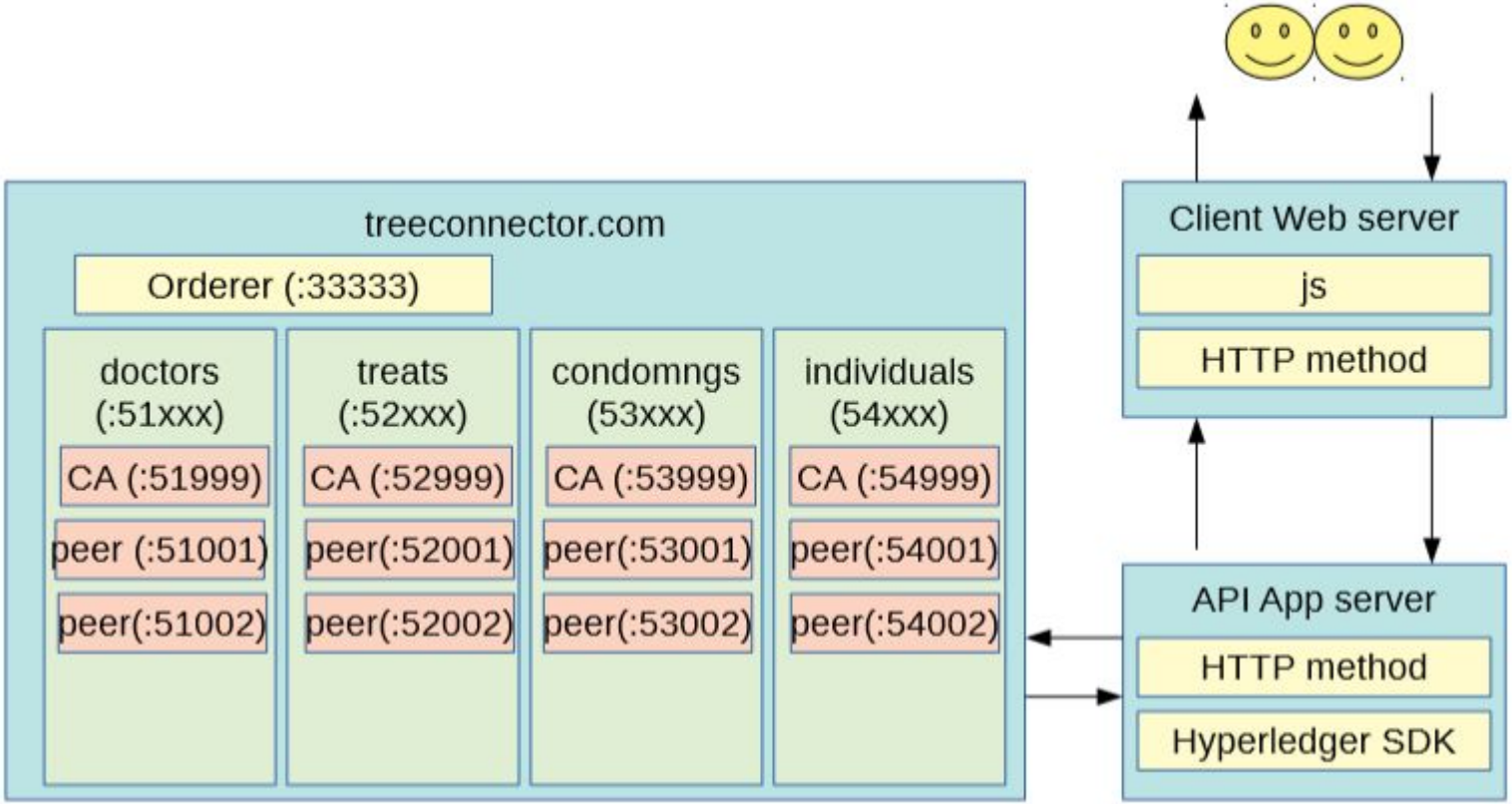
API 정의, 체인코드 구조

```
#개인 나무 관리
'''
나무에 소유자를 등록해서 이력 관리
사용자의 나무들을 불러오는 API
req {
        userId
}

res tree {
        id(String),
        ownerId(String),
        birthday(Date),
        locatoin(String),
        kind(String)

        diagnosis {
                date(Date),
                doctorId(String),
                Contents(String)
        }[]

        treatment {
                date(Date),
                therapistId(String),
                Contents(String)
        }[]
```

```go
type TreeConnector interface {
    AddUser(shim.ChaincodeStubInterface, *User) error
    GetUser(shim.ChaincodeStubInterface, string) (*User, error)
    UpdateUser(shim.ChaincodeStubInterface, *User) error
    ListUsers(shim.ChaincodeStubInterface) ([]*User, error)

    AddTree(shim.ChaincodeStubInterface, *Tree) error
    GetTree(shim.ChaincodeStubInterface, string) (*Tree, error)
    UpdateTree(shim.ChaincodeStubInterface, *Tree) error
    ListTrees(shim.ChaincodeStubInterface) ([]*Tree, error)

    AddDiagnosis(shim.ChaincodeStubInterface, *Diagnosis) error
    GetDiagnosis(shim.ChaincodeStubInterface, string) (*Diagnosis, error)
    UpdateDiagnosis(shim.ChaincodeStubInterface, *Diagnosis) error
    ListDiagnoses(shim.ChaincodeStubInterface) ([]*Diagnosis, error)

    AddTreatment(shim.ChaincodeStubInterface, *Treatment) error
    GetTreatment(shim.ChaincodeStubInterface, string) (*Treatment, error)
    UpdateTreatment(shim.ChaincodeStubInterface, *Treatment) error
    ListTreatments(shim.ChaincodeStubInterface) ([]*Treatment, error)
}
```

```go
type User struct {
    Id   string
    Name string
    Role string
}

type Tree struct {
    Id      string
    OwnerId string
    Kind    string
}

type Diagnosis struct {
    Id        string
    TreeId    string
    DoctorId  string
    Content   string
    Timestamp time.Time
}

type Treatment struct {
    Id          string
    TreeId      string
    TherapistId string
    Content     string
    Timestamp   time.Time
```

전체 시스템 구조 단순화


treeconnector.com
Orderer (:33333)
doctors (:51xxx)
CA (:51999)
peer (:51001)
peer(:51002)
treats (:52xxx)
CA (:52999)
peer(:52001)
peer(:52002)
condomngs (53xxx)
CA (:53999)
peer(:53001)
peer(:53002)
individuals (54xxx)
CA (:54999)
peer(:54001)
peer(:54002)
Client Web server
js
HTTP method
API App server
HTTP method
Hyperledger SDK

D-App 개발

1) 유즈케이스 시나리오에 따른 API 벡엔드 서버 개발

2) UI 설계에 따른 client 웹앱 개발

UI 설계 - 사용자 인증


(주)TreeConnector
Home 화면으로
내 정보 관리
로그아웃
[내 정보 관리]
나의 인증서 목록
(인증서 1) 민간 수목협회 - 개인
(인증서 2) 국공립 기관 - 수목원
추가
삭제
내 계정 정보 수정
New password :
New password check:
Email :
변경

UI 설계 - 주 메뉴


(주)TreeConnector
Home 화면으로
내 정보 관리
로그아웃
내 나무 관리
나무 거래
나무 정보 조회
나무 진료

UI 설계 - 나무 관리


(주)TreeConnector
Home 화면으로
내 정보 관리
로그아웃
[내 나무 관리]
나무 ID :
품종 :
지역 :
생일 :
특징 :
나무 사진
업로드
이력 정보
이력 정보
저장
취소

UI 설계 - 진료


(주)TreeConnector
Home 화면으로
내 정보 관리
로그아웃
[나무 진단서]
나무 사진
업로드
나무 ID :
품종 :
지역 :
병명 :
처방 :
진료 소견 및 기타 진료 정보
제출
취소

향후 일정

1) 조직 추가, intermediate CA
2) 지역별 채널 형성
3) 거래 usecase 추가
4) 네트워크, 벡엔드 트러블 슈팅
5) UI 개선
6) 부가기능(통계 시각화) 고려