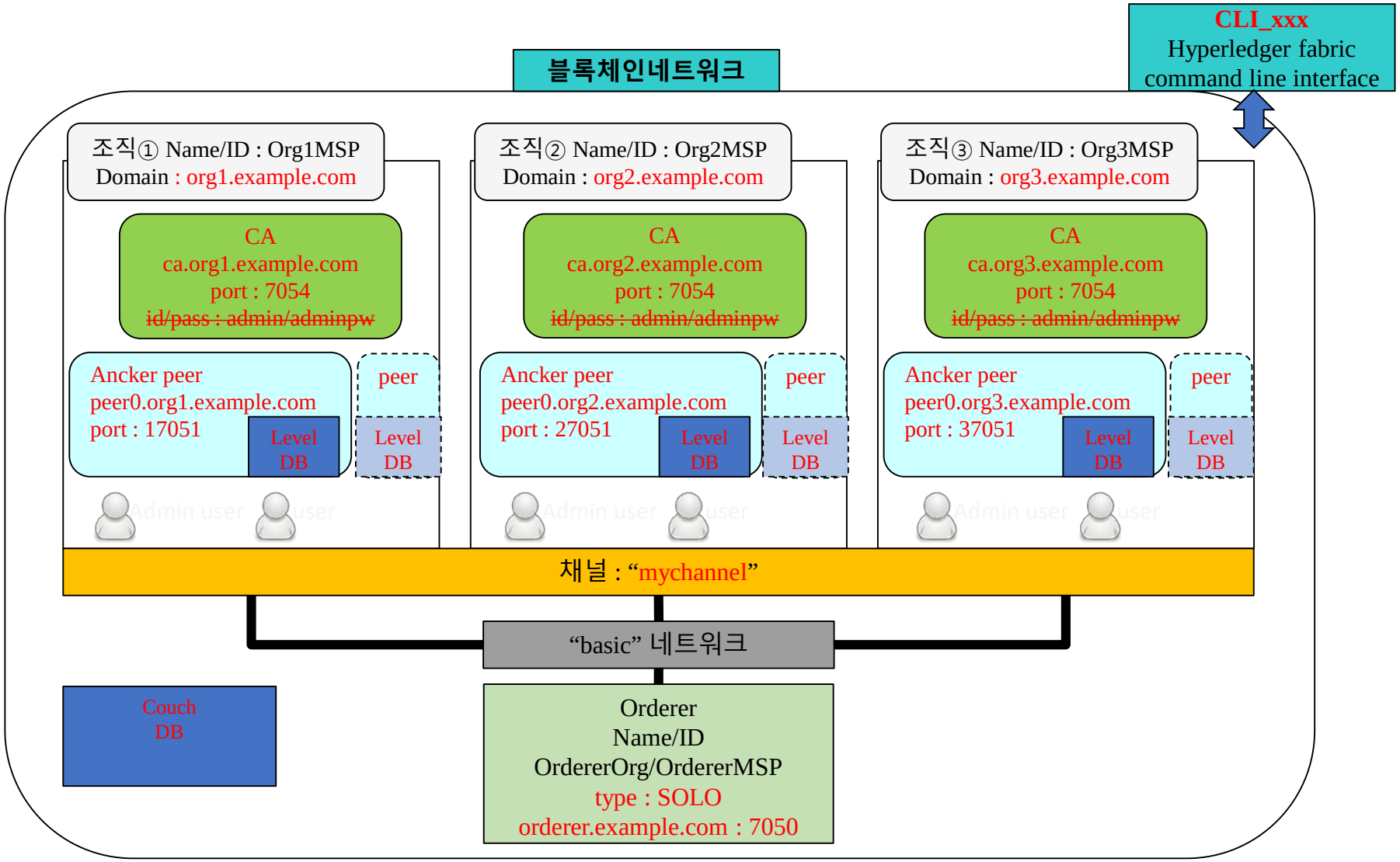
패브릭 네트워크 설계

- 구상한 주제에 맞는 네트워크에 대한 상세 정보 작성



패브릭 네트워크 구성 및 실행

- 전시간에 작성한 패브릭 네트워크 설계를 위한 상세 정보를 참고하여 패브릭 네트워크 구성 및 수정
  - configtx.yaml 수정
    - profile 수정
    - org 추가 / 삭제
  - crypto-config.yaml 수정
    - org 추가 / 삭제
    - peer 추가 / 삭제
  - generate.sh 수정
    - configtx.yaml파일의 profile ‘OneOrgOrdererGenesis’, ‘OneOrgChannel’ 명칭을 수정했을 시 수정

패브릭 네트워크 구성 및 실행

- 전시간에 작성한 패브릭 네트워크 설계를 위한 상세 정보를 참고하여 패브릭 네트워크 구성 및 수정
  - docker-compose.yml 수정
    - 구상한 네트워크 구성요소에 맞추어 작성
      - org, ca, peer 수 및 각각의 세부 설정 변경
      - TLS통신 여부 에 맞춘 설정 변경
      - 사용할 DB에 맞는 설정 변경(level DB or Couch DB)
  - start.sh 수정
    - 구상한 네트워크 구성요소에 맞게 docker-compose -f docker-compose.yml up -d xxx xxx xxx xxx 수정
    - 채널 생성 / 가입 부분 수정

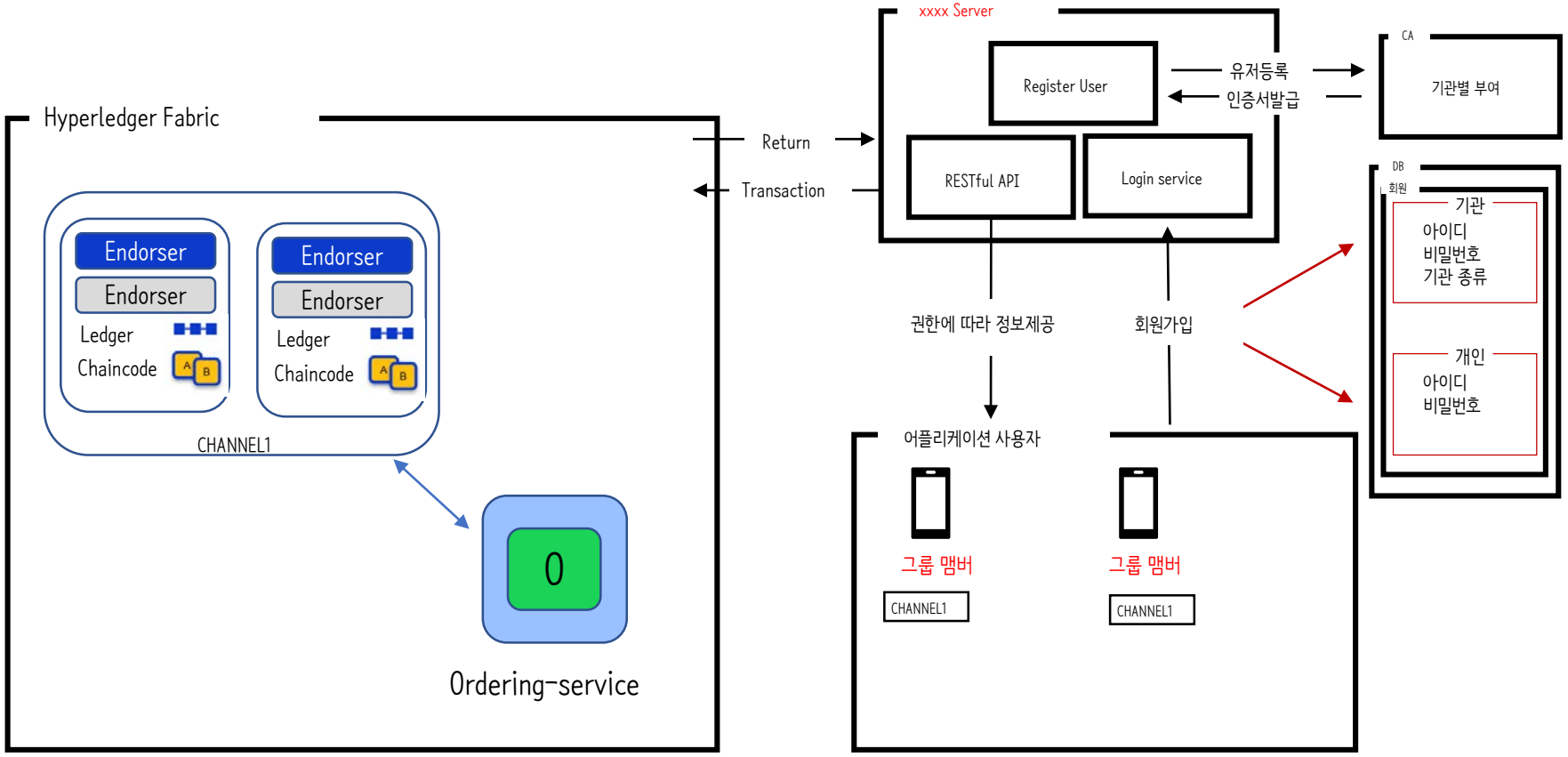
패브릭 네트워크 구성 및 실행

- 수정 완료 후 네트워크 동작 확인 및 디버깅

```
bstudent@bstudent-VirtualBox:~/fabricbook$ docker exec cli_org3 peer node status
2019-05-02 16:31:36.378 UTC [nodeCmd] status -> WARN 001 admin client failed to connect
 to peer0.org3.example.com:7051: failed to create new connection: context deadline exce
eded
Error: admin client failed to connect to peer0.org3.example.com:7051: failed to create
new connection: context deadline exceeded
status:UNKNOWN
```

```
bstudent@bstudent-VirtualBox:~/fabricbook$ docker exec peer0.org3.example.com peer chan
nel list
2019-05-02 16:27:22.064 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and ordere
r connections initialized
Channels peers has joined:
mychannel
```

전체 구조도작성 및 설계


Hyperledger Fabric
Endorser
Endorser
Ledger
Chaincode
Endorser
Endorser
Ledger
Chaincode
CHANNEL1
0
Ordering-service
Return
Transaction
xxxx Server
Register User
RESTful API
Login service
유저등록
인증서발급
CA
기관별 부여
DB
회원
기관
아이디
비밀번호
기관 종류
개인
아이디
비밀번호
권한에 따라 정보제공
회원가입
어플리케이션 사용자
그룹 맴버
CHANNEL1
그룹 맴버
CHANNEL1

인터페이스 설계(체인코드), UI설계

Fabcar

전체리스트 가져오기

차 정보 가져오기

새로운 차 생성하기

차 소유자 변경하기

Fabcar

전체리스트

Car No	색상	메이커	모델명	소유자

차 정보 가져오기

새로운 차 생성하기

차 소유자 변경하기

Fabcar

새로운 차 생성하기

Car No

색상

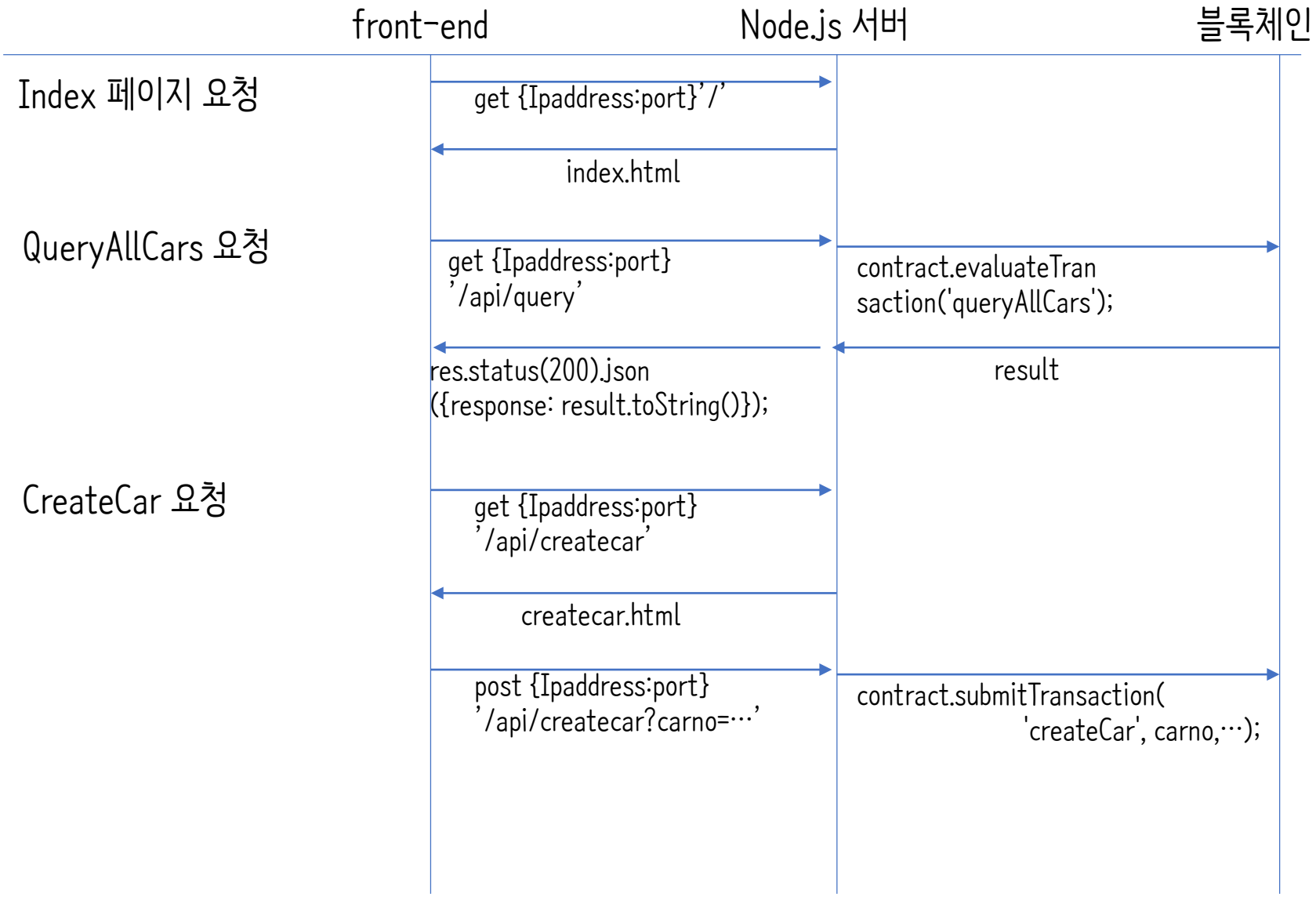
메이커

모델명

소유자

등록하기

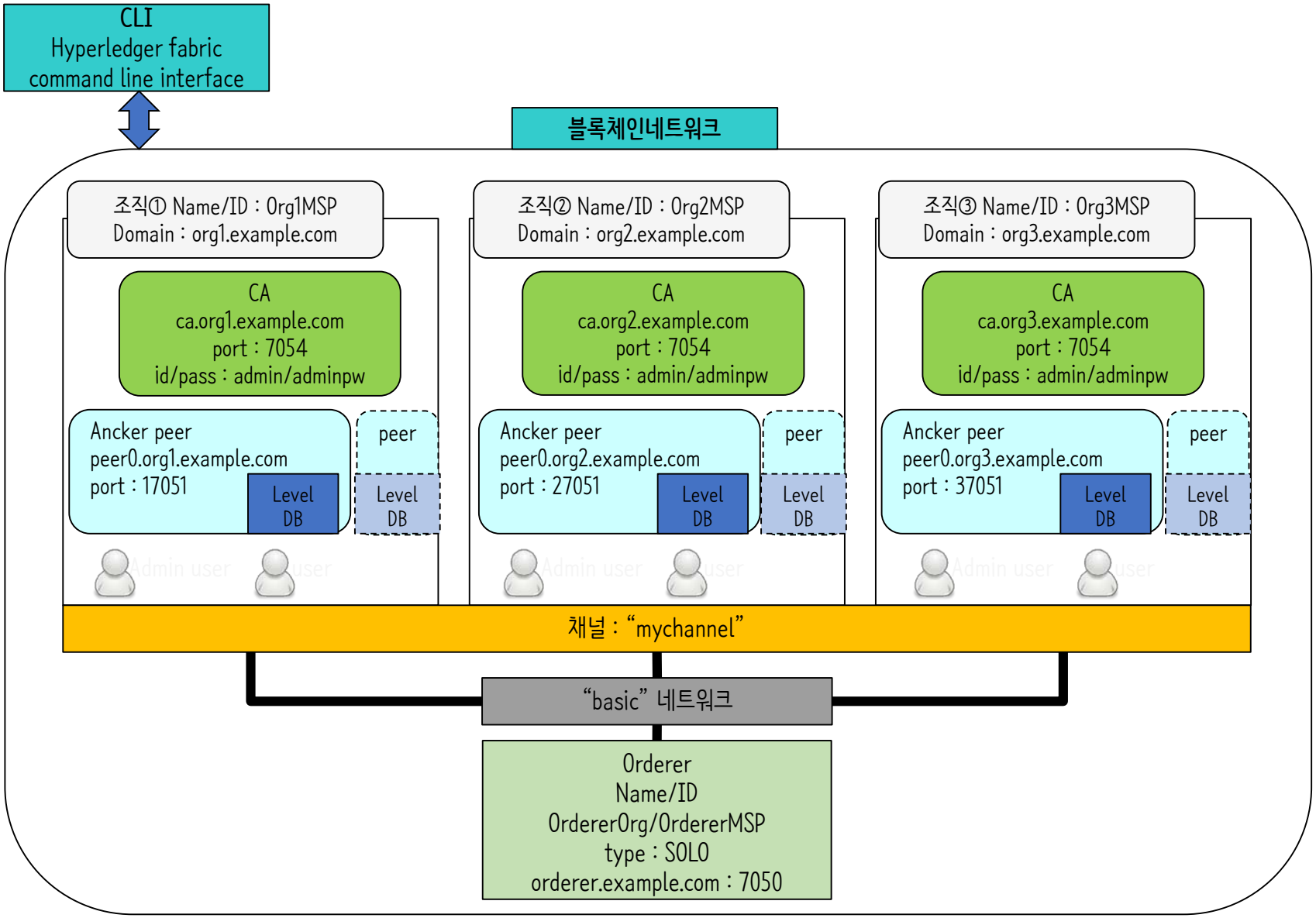
인터페이스 설계(체인코드), UI설계



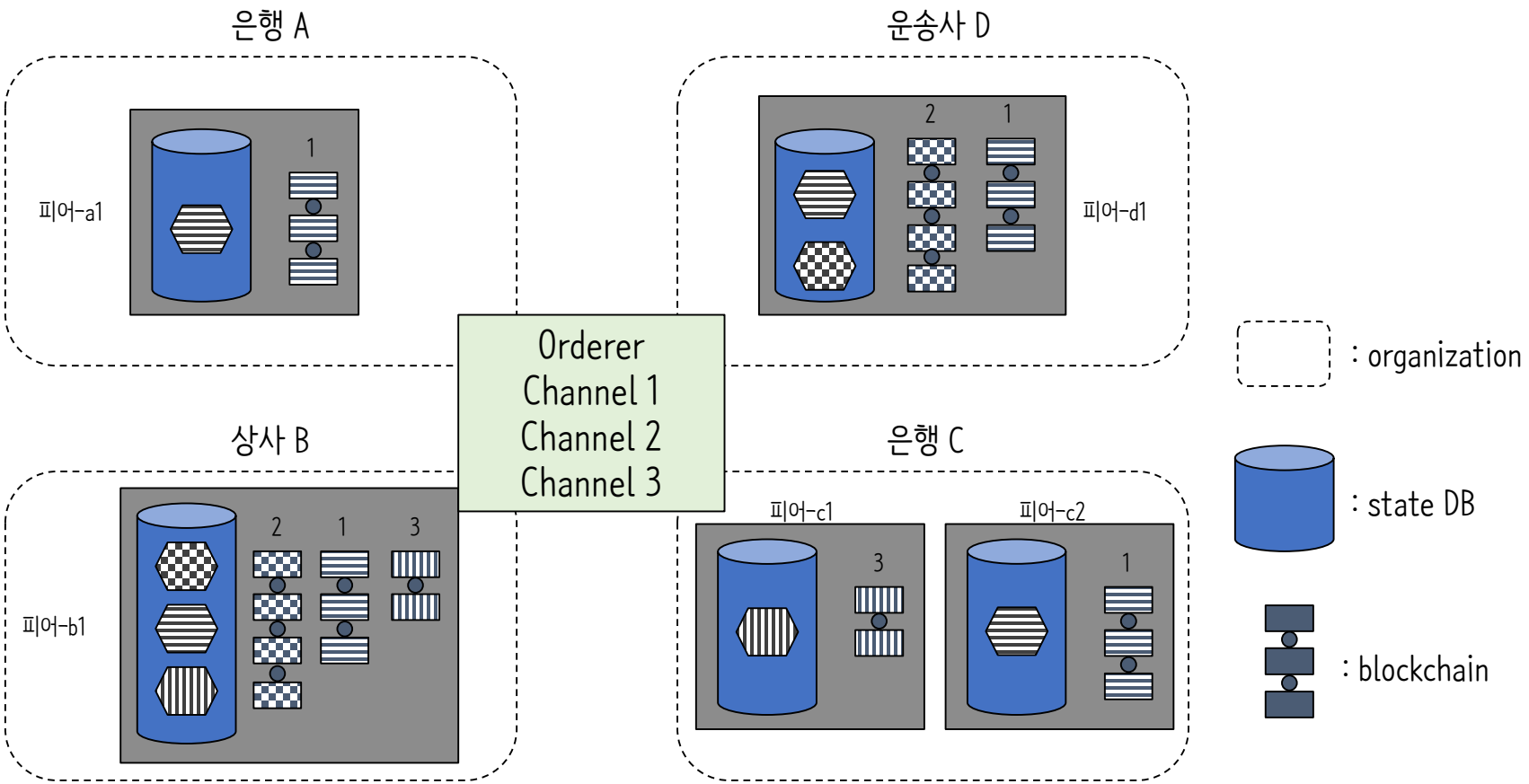
네트워크 설계

- Organization 수 :
- Channel
  - 채널 수 :
  - 채널 이름 :
- Orderer
  - Orderer 수 :
  - Concensus 방식 :
  - 주소 및 포트 :
- Peer
  - Organization 별 peer 수 :
  - 주소 및 포트 :

네트워크 설계


CLI
Hyperledger fabric
command line interface
블록체인네트워크
조직① Name/ID : Org1MSP
Domain : org1.example.com
조직② Name/ID : Org2MSP
Domain : org2.example.com
조직③ Name/ID : Org3MSP
Domain : org3.example.com
CA
ca.org1.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org2.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org3.example.com
port : 7054
id/pass : admin/adminpw
Ancker peer
peer0.org1.example.com
port : 17051
Level DB
peer
Level DB
Ancker peer
peer0.org2.example.com
port : 27051
Level DB
peer
Level DB
Ancker peer
peer0.org3.example.com
port : 37051
Level DB
peer
Level DB
Admin user
user
Admin user
user
Admin user
user
채널 : “mychannel”
“basic” 네트워크
Orderer
Name/ID
OrdererOrg/OrdererMSP
type : SOLO
orderer.example.com : 7050

네트워크 구성


은행 A
피어-a1
1
운송사 D
2
1
피어-d1
Orderer
Channel 1
Channel 2
Channel 3
상사 B
피어-b1
2
1
3
은행 C
피어-c1
3
피어-c2
1
: organization
: state DB
: blockchain

프로젝트 개요

프로젝트명

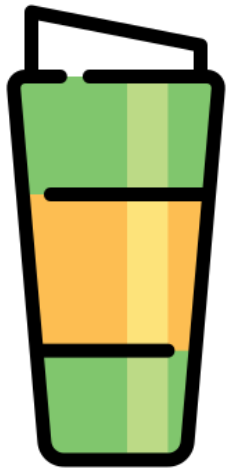
teamate (티메이트)

요약

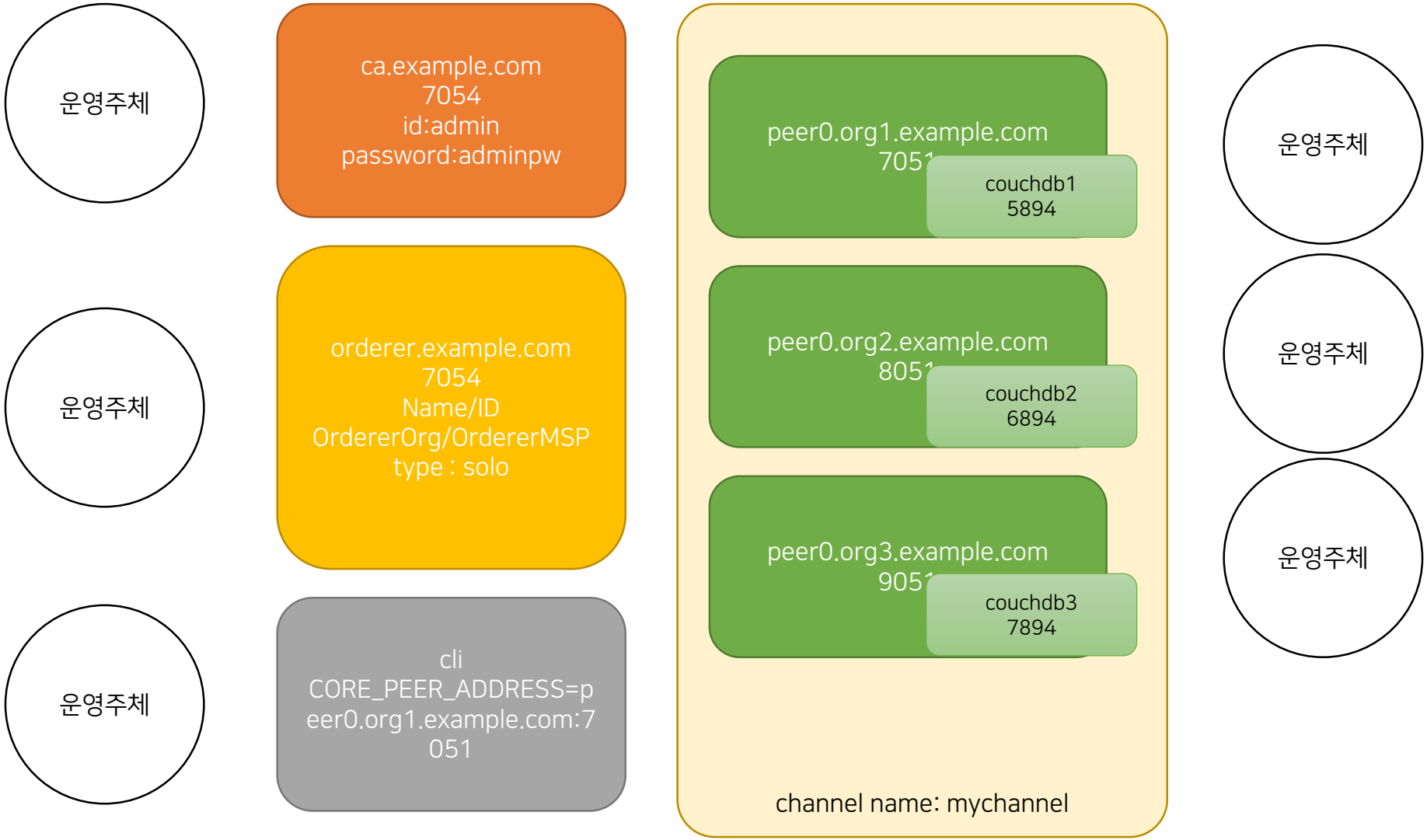
팀프로젝트 맴버를 구하기 위한 페이지

기능

팀프로젝트 리스트
팀프로젝트 등록
사용자 등록
사용자 리스트
사용자 팀프로젝트 참여
팀프로젝트 참여사용자평가



네트워크 구조


운영주체
운영주체
운영주체
ca.example.com
7054
id:admin
password:adminpw
orderer.example.com
7054
Name/ID
OrdererOrg/OrdererMSP
type : solo
cli
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
peer0.org1.example.com
7051
couchdb1
5894
peer0.org2.example.com
8051
couchdb2
6894
peer0.org3.example.com
9051
couchdb3
7894
channel name: mychannel
운영주체
운영주체
운영주체

teamate 체인코드

- 이름
  - teamate
- 기능

addUser

User정보 생성하기
key: email - args[0]
averate: 0
project list: []

addRating

User의 프로젝트점수 추가하기
key: email – args[0]
project name – args[1]
project rate – args[2]

readRating

User의 프로젝트점수 읽어오기
key: email –args[0]

```
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {

    function, args := APIstub.GetFunctionAndParameters()

    if function == "addUser" {
        return s.addUser(APIstub, args)
    } else if function == "addRating" {
        return s.addRating(APIstub, args)
    } else if function == "readRating" {
        return s.readRating(APIstub, args)
    }
    return shim.Error("Invalid Smart Contract function name.")
}
```

체인코드 데이터

email address

Key

ARRAY
[project name : score]

Project List

float64

Average Rate Score


```
23  type UserRating struct{
24      User string `json:"user"`
25      Average float64 `json:"average"`
26      Rates []Rate `json:"rates"`
27  }
28  type Rate struct{
29      ProjectTitle string  `json:"projecttitle"`
30      Score float64 `json:"score"`
31  }
```

어플리케이션 - 사용자

사용자 등록

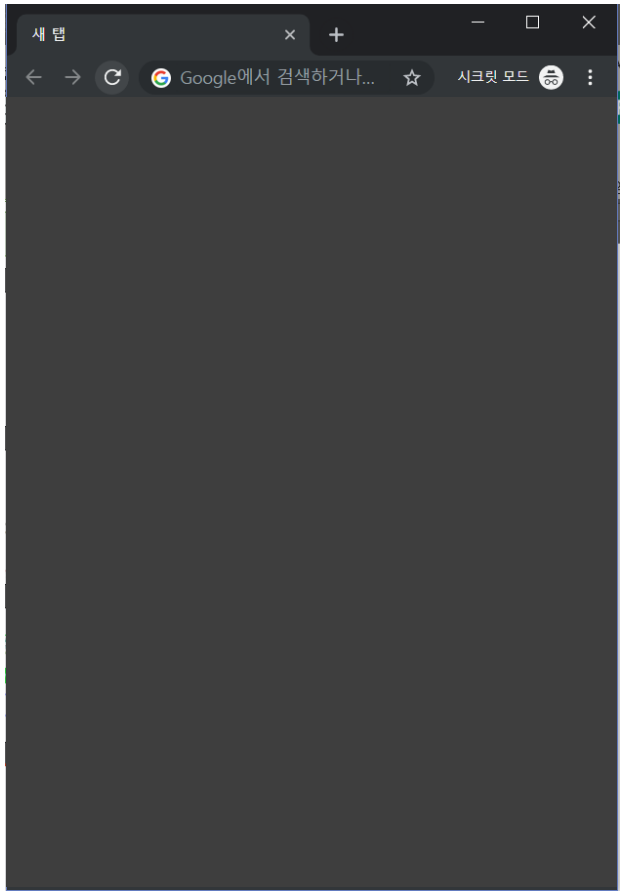
사용자 목록

어플리케이션 - 프로젝트

프로젝트 등록

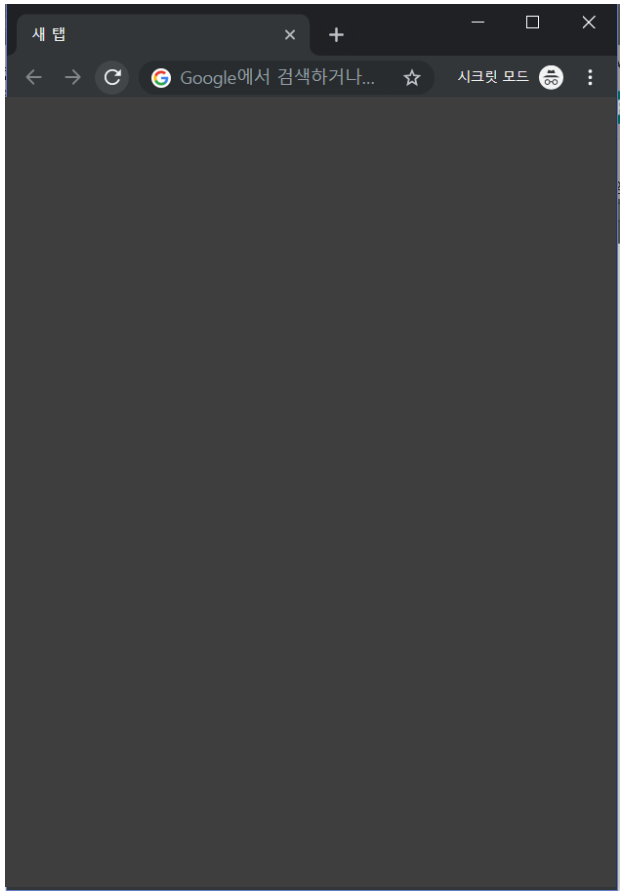
프로젝트 목록

어플리케이션 – 프로젝트참여, 평가



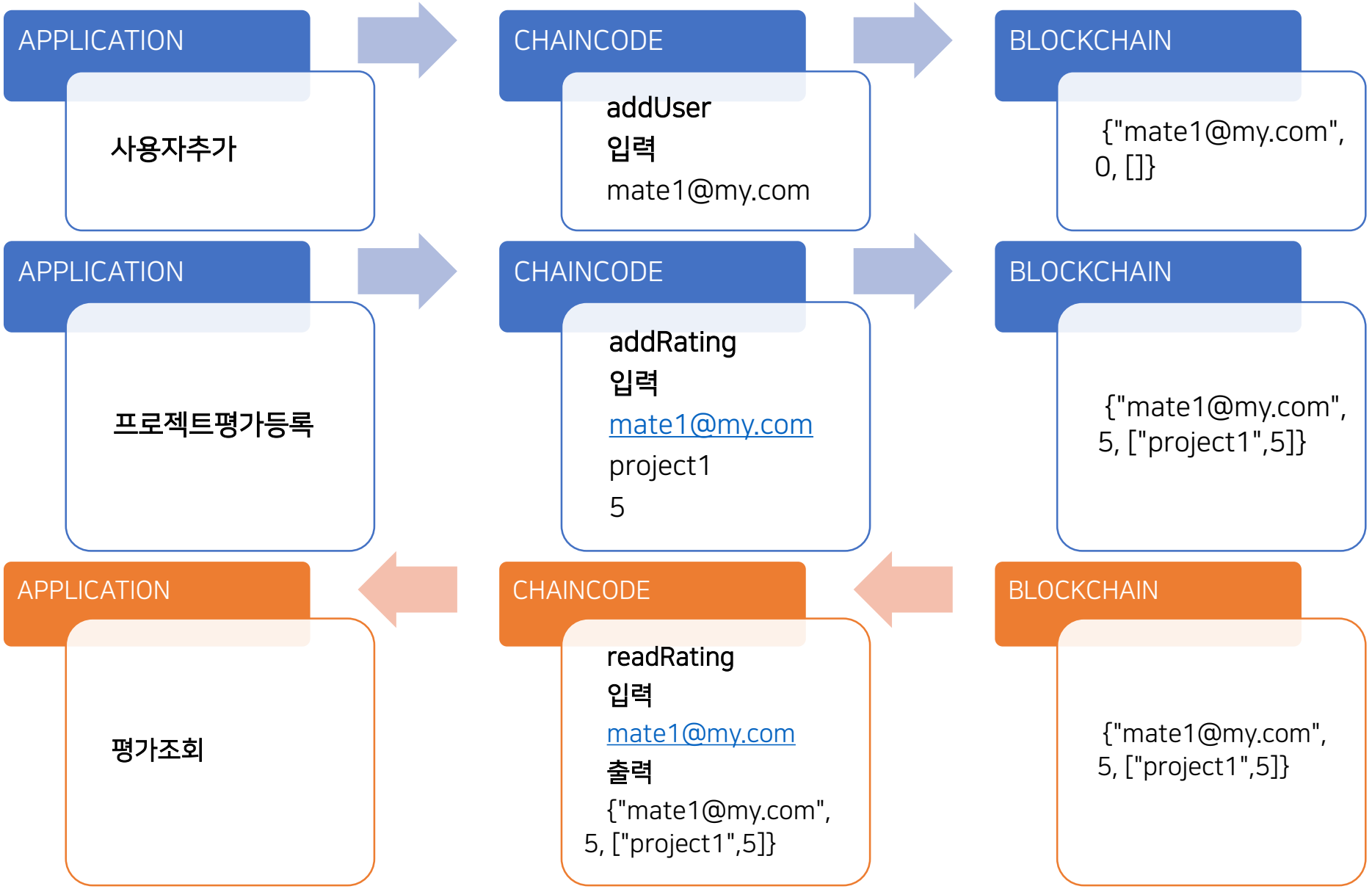
프로젝트 참여

참여 프로젝트 정보
(참여사용자클릭)

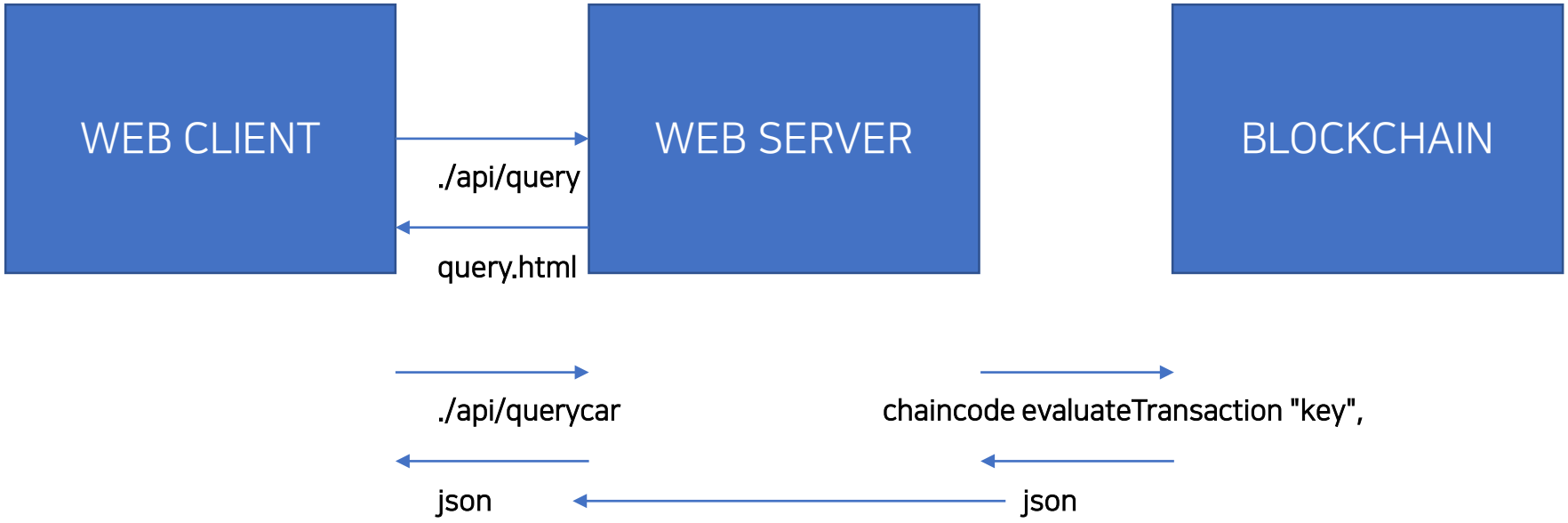


참여 사용자 평가

기능 플로우


APPLICATION
사용자추가
CHAINCODE
addUser
입력
mate1@my.com
BLOCKCHAIN
{"mate1@my.com",
0, []}
APPLICATION
프로젝트평가등록
CHAINCODE
addRating
입력
mate1@my.com
project1
5
BLOCKCHAIN
{"mate1@my.com",
5, ["project1",5]}
APPLICATION
평가조회
CHAINCODE
readRating
입력
mate1@my.com
출력
{"mate1@my.com",
5, ["project1",5]}
BLOCKCHAIN
{"mate1@my.com",
5, ["project1",5]}

기능상세


WEB CLIENT
WEB SERVER
BLOCKCHAIN
./api/query
query.html
./api/querycar
chaincode evaluateTransaction "key",
json
json

18일 화요일	dApp 프로토타입개발	최광훈	dApp Front End 예제뜯어보기	fabcar/basicnetwork/front-end
				node.js 간단한 사용법
				front-end 자세히보기
				네트워크구성 확장
		신운섭	dApp 프로토타입 기획	체인코드 이름, 함수프로토타입정의
				front-end와 패브릭 연동계획
			dApp UI I UX 디자인	node.js 를 이용한 front-end학습
19일 수요일	dApp 개발	최광훈		front-end 페이지 구성
			dApp 프로토타입 개발	체인코드 인터페이스 작성
			dApp 프로토타입 배포	패브릭 네트워크확장하기
		신운섭		전체 설계및구조 수정및 보안
				패브릭 네트워크 설계
				패브릭 네트워크 구성
				UI구현/front-end 페이지 구현
9일차 목요일	dApp 배포	최광훈	dApp 개발 / 배포	체인코드 구현
				체인코드배포를 포함한 스크립트만들기
			dApp 개발	node.js를 이용한 웹서버 구현
		신운섭		시스템 가동/연동테스트
			dApp 개발 / 배포 (리허설)	문서화(발표자료, 소스코드정리)
				데모준비

체인코드 구현

- 작성된 체인코드 프로토타입에 세부 구현사항 구현

```
85 func (s *SmartContract) queryCar(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
86
87   if len(args) != 1 {
88     return shim.Error("Incorrect number of arguments. Expecting 1")
89   }
90
91   carAsBytes, _ := APIstub.GetState(args[0])
92   return shim.Success(carAsBytes)
93 }
```

기능별 파라미터 확인

기능별 World state 접근 구현 (GetState는 Key를 파라미터로 호출)

```go
func (s *SmartContract) initLedger(APIstub shim.ChaincodeStubInterface) sc.Response {
  cars := []Car{
    Car{Make: "Toyota", Model: "Prius", Colour: "blue", Owner: "Tomoko"},
    Car{Make: "Ford", Model: "Mustang", Colour: "red", Owner: "Brad"},
    Car{Make: "Hyundai", Model: "Tucson", Colour: "green", Owner: "Jin Soo"},
    Car{Make: "Volkswagen", Model: "Passat", Colour: "yellow", Owner: "Max"},
    Car{Make: "Tesla", Model: "S", Colour: "black", Owner: "Adriana"},
    Car{Make: "Peugeot", Model: "205", Colour: "purple", Owner: "Michel"},
    Car{Make: "Chery", Model: "S22L", Colour: "white", Owner: "Aarav"},
    Car{Make: "Fiat", Model: "Punto", Colour: "violet", Owner: "Pari"},
    Car{Make: "Tata", Model: "Nano", Colour: "indigo", Owner: "Valeria"},
    Car{Make: "Holden", Model: "Barina", Colour: "brown", Owner: "Shotaro"},
  }

  i := 0
  for i < len(cars) {
    fmt.Println("i is ", i)
    carAsBytes, _ := json.Marshal(cars[i])
    APIstub.PutState("CAR"+strconv.Itoa(i), carAsBytes)
    fmt.Println("Added", cars[i])
    i = i + 1
  }

  return shim.Success(nil)
}
```

Putstate는 Key별로 Byte로 마샬된 데이터를 파라미터로 사용하여 호출

```go
func (s *SmartContract) createCar(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

  if len(args) != 5 {
    return shim.Error("Incorrect number of arguments. Expecting 5")
  }

  var car = Car{Make: args[1], Model: args[2], Colour: args[3], Owner: args[4]}

  carAsBytes, _ := json.Marshal(car)
  APIstub.PutState(args[0], carAsBytes)

  return shim.Success(nil)
}
```

```go
func (s *SmartContract) queryAllCars(APIstub shim.ChaincodeStubInterface) sc.Response {

  startKey := "CAR0"
  endKey := "CAR999"

  resultsIterator, err := APIstub.GetStateByRange(startKey, endKey)
  if err != nil {
    return shim.Error(err.Error())
  }
  defer resultsIterator.Close()

  // buffer is a JSON array containing QueryResults
  var buffer bytes.Buffer
  buffer.WriteString("[")

  bArrayMemberAlreadyWritten := false
  for resultsIterator.HasNext() {
    queryResponse, err := resultsIterator.Next()
    if err != nil {
      return shim.Error(err.Error())
    }
    // Add a comma before array members, suppress it for the first array member
    if bArrayMemberAlreadyWritten == true {
      buffer.WriteString(",")
    }
    buffer.WriteString("{\"Key\":")
    buffer.WriteString("\"")
    buffer.WriteString(queryResponse.Key)
    buffer.WriteString("\"")

    buffer.WriteString(", \"Record\":")
    // Record is a JSON object, so we write as-is
    buffer.WriteString(string(queryResponse.Value))
    buffer.WriteString("}")
    bArrayMemberAlreadyWritten = true
  }
  buffer.WriteString("]")

  fmt.Printf("- queryAllCars:\n%s\n", buffer.String())

  return shim.Success(buffer.Bytes())
}
```

Key의 범위를 사용하여 많은 양의 데이터를 한꺼번에 받음

이터레이터구조를 사용하여 JSON형태로 변환

데이터의 수정

```
178  func (s *SmartContract) changeCarOwner(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {
179
180    if len(args) != 2 {
181      return shim.Error("Incorrect number of arguments. Expecting 2")
182    }
183
184    carAsBytes, _ := APIstub.GetState(args[0])
185    car := Car{}
186
187    json.Unmarshal(carAsBytes, &car)
188    car.Owner = args[1]
189
190    carAsBytes, _ = json.Marshal(car)
191    APIstub.PutState(args[0], carAsBytes)
192
193    return shim.Success(nil)
194  }
195
```

데이터를 가져오기

가져온 바이트 데이터 언마샬링 후 수중

수정된 데이터를 다시 마샬링한 뒤 데이터 저장하기

CHAINCODE GO SDK

https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim

https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim#ChaincodeStubInterface

SplitCompositeKey documentation can be found in interfaces.go

type ChaincodeStubInterface

```
type ChaincodeStubInterface interface {
    // GetArgs returns the arguments intended for the chaincode Init and Invoke
    // as an array of byte arrays.
    GetArgs() [][]byte

    // GetStringArgs returns the arguments intended for the chaincode Init and
    // Invoke as a string array. Only use GetStringArgs if the client passes
    // arguments intended to be used as strings.
    GetStringArgs() []string

    // GetFunctionAndParameters returns the first argument as the function
    // name and the rest of the arguments as parameters in a string array.
    // Only use GetFunctionAndParameters if the client passes arguments intended
    // to be used as strings.
    GetFunctionAndParameters() (string, []string)
```

체인코드의 예

- World State를 활용
  - fabcar.go
  - sacc.go
  - marbles.go

- Private Data를 활용
  - marbles_private.go (collection_config.json)

실습

- 배포 가능한 체인코드를 구현
- 체인코드가 문법적오류가 없는지 빌드 및 디버깅
  - GOPATH 확인
  - 체인코드 인터페이스 모듈 다운로드 확인
  - GOBUILD
- 작성된 체인코드를 설계 구현한 네트워크에 적용하여 쉘스크립트 업데이트 하기
  - docker volume 에 리눅스 체인코드 디렉토리를 /opt/gopath/src/github.com/ 에 연결하였는지 확인
  - install, instantiate, query, invoke실행 해보기
  - cc.sh 쉘스크립트 만들기 (설치, 배포, invoke/query)

dApp 개발

- node.js를 이용한 웹서버 구현
- 웹페이지 인터페이스와 Node.js연동
- node.js와 패브릭 연동

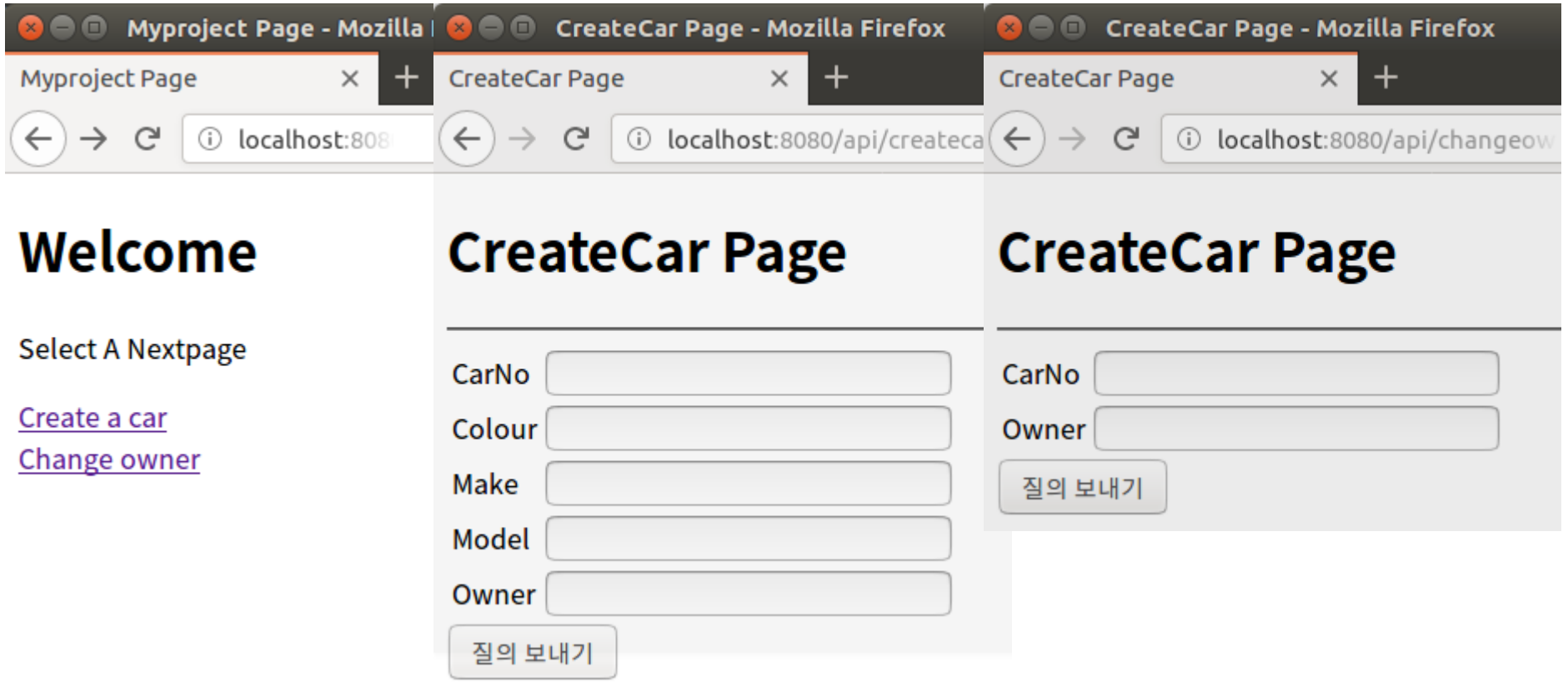
nodes.js 구조 작성

- // 서버의존모듈
- // 인터페이스 연동
- // 컨트렉트 API
- // 폼 인터페이스 처리
- // 서버 가동

```
19   // 서버의존모듈
20   const express = require('express');
21
22   // 인터페이스 연동
23   app.get('/', function (req, res) {…
28   });
29   app.get('/api/query', async function (req, res) {…
64   });
65   app.get('/api/querycar/', async function (req, res) {…
82   });
83   app.get('/api/createcar', function (req, res) {…
88   });
89   app.get('/api/changeowner', function (req, res) {…
94   });
95   // 폼 인터페이스 처리
96   app.post('/api/createcar/', async function (req, res) {…
142  });
143  app.post('/api/changeowner/', async function (req, res) {…
186  });
187  // 서버 가동
188  app.listen(PORT, HOST);
189  console.log(`Running on http://localhost:${PORT}`);
190  // 컨트렉트 API
191
```

웹인터페이스와 node.js연동

```
24  app.get('/', function (req, res) {
25    fs.readFile('index.html', function (error, data) {
                res.send(data.toString());
27    });
28  });
```



contract와 node.js 연동 작성

- query와 invoke 부분 작성

```
8   // Hyperledger Bridge
9   const { FileSystemWallet, Gateway } = require('fabric-network');
10  const fs = require('fs');
11  const path = require('path');
12
13  const ccpPath = path.resolve(__dirname, '..', '..', 'basic-network', 'connection.json');
14  const ccpJSON = fs.readFileSync(ccpPath, 'utf8');
15  const ccp = JSON.parse(ccpJSON);
```

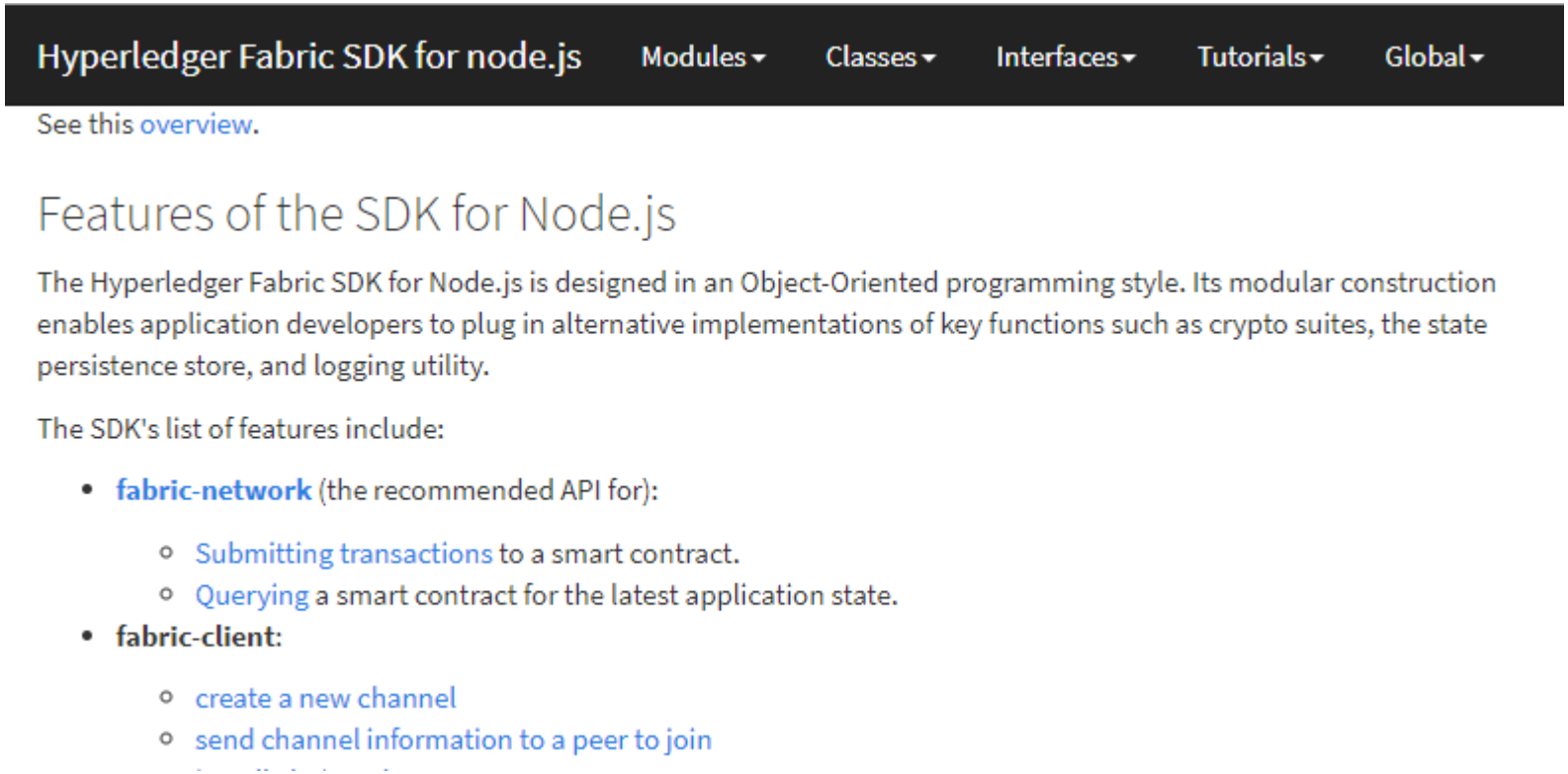
```
62  // 192.168.1.56:8080/api/querycar?carno=CAR5
63  app.get('/api/querycar/', async function (req, res) {
64                  // create the key value store as defined in the fabric-client/config/default.json 'key-va
65      try {
66      var carno = req.query.carno;
67      console.log(carno);
68
69          // Create a new file system based wallet for managing identities.
70          const walletPath = path.join(process.cwd(), 'wallet');
71          const wallet = new FileSystemWallet(walletPath);
72          console.log(`Wallet path: ${walletPath}`);
73
74          // Check to see if we've already enrolled the user.
75          const userExists = await wallet.exists('user1');
76          if (!userExists) {
77              console.log('An identity for the user "user1" does not exist in the wallet');
78              console.log('Run the registerUser.js application before retrying');
79              return;
80          }
81
82          // Create a new gateway for connecting to our peer node.
83          const gateway = new Gateway();
84          await gateway.connect(ccp, { wallet, identity: 'user1', discovery: { enabled: false } });
85
86          // Get the network (channel) our contract is deployed to.
87          const network = await gateway.getNetwork('mychannel');
88
89          // Get the contract from the network.
90          const contract = network.getContract('fabcar');
91
92          // Evaluate the specified transaction.
93          // queryCar transaction - requires 1 argument, ex: ('queryCar', 'CAR4')
94          // queryAllCars transaction - requires no arguments, ex: ('queryAllCars')
95          const result = await contract.evaluateTransaction('queryCar', carno);
96
97          console.log(`Transaction has been evaluated, result is: ${result.toString()}`);
98          res.status(200).json({response: result.toString()});
99      } catch (error) {
100         console.error(`Failed to evaluate transaction: ${error}`);
101         process.exit(1);
102     }
103 });
```

```
app.post('/api/createcar/', async function (req, res) {
    try {
    var carno = req.body.carno;
    var colour = req.body.colour;
    var make = req.body.make;
    var model = req.body.model;
    var owner = req.body.owner;
        // Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);
        // Check to see if we've already enrolled the user.
        const userExists = await wallet.exists('user1');
        if (!userExists) {
            console.log('An identity for the user "user1" does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            return;
        }
        // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: 'user1', discovery: { enabled: false } });
        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');
        // Get the contract from the network.
        const contract = network.getContract('fabcar');
        // Submit the specified transaction.
        // createCar transaction - requires 5 argument, ex: ('createCar', 'CAR12', 'Honda', 'Accord', '
        // changeCarOwner transaction - requires 2 args , ex: ('changeCarOwner', 'CAR10', 'Dave')
//          await contract.submitTransaction('createCar', 'CAR11', 'Hnda', 'Aord', 'Bla', 'Tom');
        await contract.submitTransaction('createCar', carno, make, model, colour, owner);
        console.log('Transaction has been submitted');
        // Disconnect from the gateway.
        await gateway.disconnect();
    } catch (error) {
        console.error(`Failed to submit transaction: ${error}`);
        process.exit(1);
    }
});
```

APPLICATION SDK

https://fabric-sdk-node.github.io/release-1.4/index.html

https://fabric-sdk-node.github.io/release-1.4/module-fabric-network.Transaction.html



체인코드 연동 예

- sacc : fabricbook
  - ORG1, ORG2, ORG3 네트워크 (no TLS, couchdb, solo orderer)
  - 체인코드: set, get, getAllKeys
  - 웹서버: nodejs
    - 처리 라우팅

    METHOD:GET **PATH**:/ **SEND**: index.html
    METHOD:GET **PATH**:/api/querykey/ **SEND**: querykey.html
    METHOD:GET **PATH**:/api/createkey **SEND**: createkey.html

    METHOD:GET **PATH**:/api/query **SEND**: JSON (all data from BC or error)
    METHOD:GET **PATH**:/api/querykey/:id **SEND**: JSON (one data from BC or error)
    METHOD:POST **PATH**:/api/createkey **SEND**: JSON (result)

  - 웹클라이언트
    - HTML+JQUERY(AJAX)

체인코드 연동 예

- fabcar : fabric-front-end
  - ORG1, ORG2, ORG3 네트워크 (no TLS, couchdb, solo orderer)
  - 체인코드: queryCar, createCar, queryAllCars, changeCarOwner..
  - 웹서버: nodejs
    - 처리 라우팅

    app.get('/api/queryallcars', function (req, res) {
    app.get('/api/query', async function (req, res) {
    app.get('/api/querycars/', async function (req, res) {
    app.get('/api/createcar', function (req, res) {
    app.post('/api/createcar/', async function (req, res) {
    app.get('/api/changeowner', function (req, res) {
    app.post('/api/changeowner/', async function (req, res) {

  - 웹클라이언트
    - HTML+ejs+부트스트랩+폰트어썸

구현 및 테스트

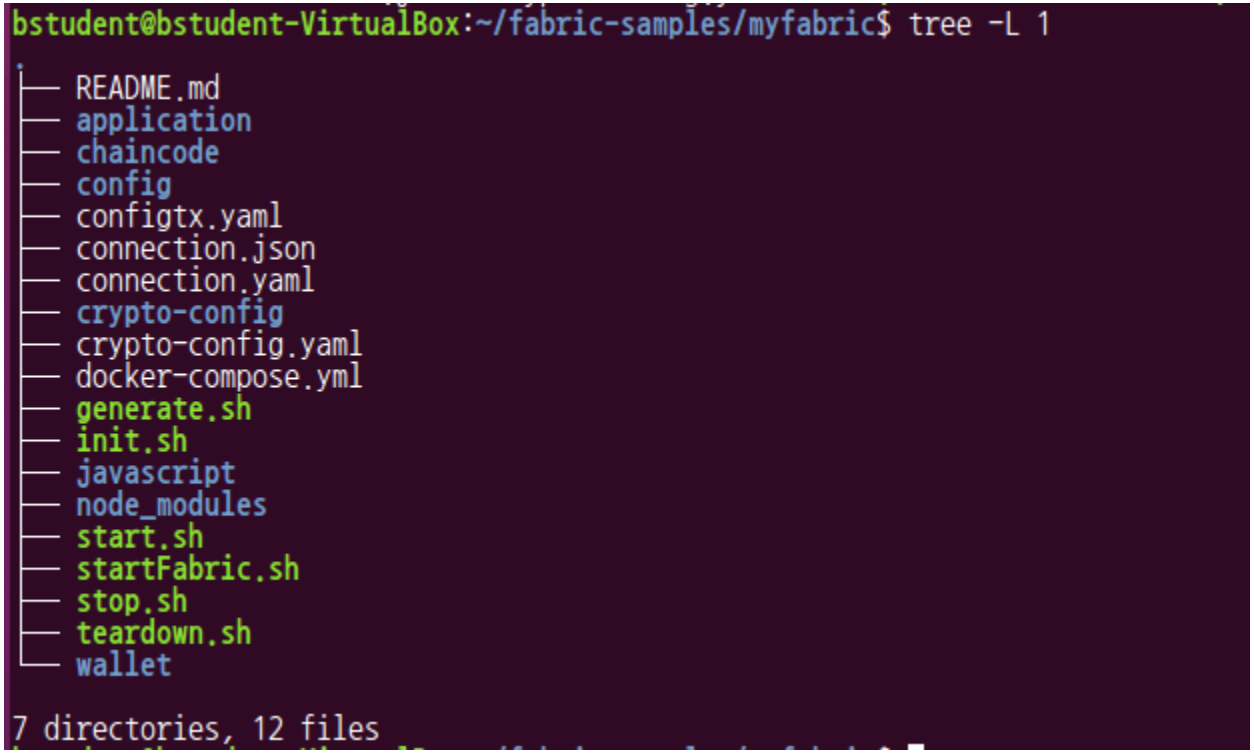
- package.json 작성
- node-module 설치
  - "express": "~4.13.1"
  - 서버 소스코드 작성 (의존모듈 임포트, 서버시작코드 등..)
  - 웹인터페이스 작성
  - 웹인터페이스 node.js연동 작성
  - node.js contract 연동 API 작성
  - node.js 폼메세지 처리 구현

- 의존 모듈 설치
  - npm install
- 시스템 가동
  - 각 디렉토리의 상대주소를 고려하여 node 실행
- 웹브라우저를 사용하여 접근
  - 예 localhost:8080

시스템 가동

- 준비사항
- 구성디렉토리
  - application : front-end 파일들
  - wallet : application을 위한 인증서
  - Chaincode : 개발된 체인코드
  - configtx설정파일, crypto-config설정파일
  - connection.json파일
  - docker-compose설정파일
  - 네트워크 구성, 체인코드 설치, query-invoke-query를 포함한 쉘 스크립트

쉘스크립트 수행 및 디버깅

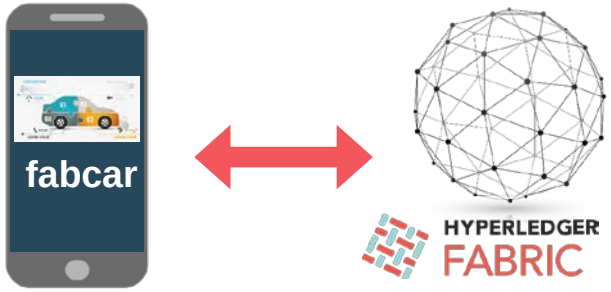


dApp 개발 / 배포 (리허설)

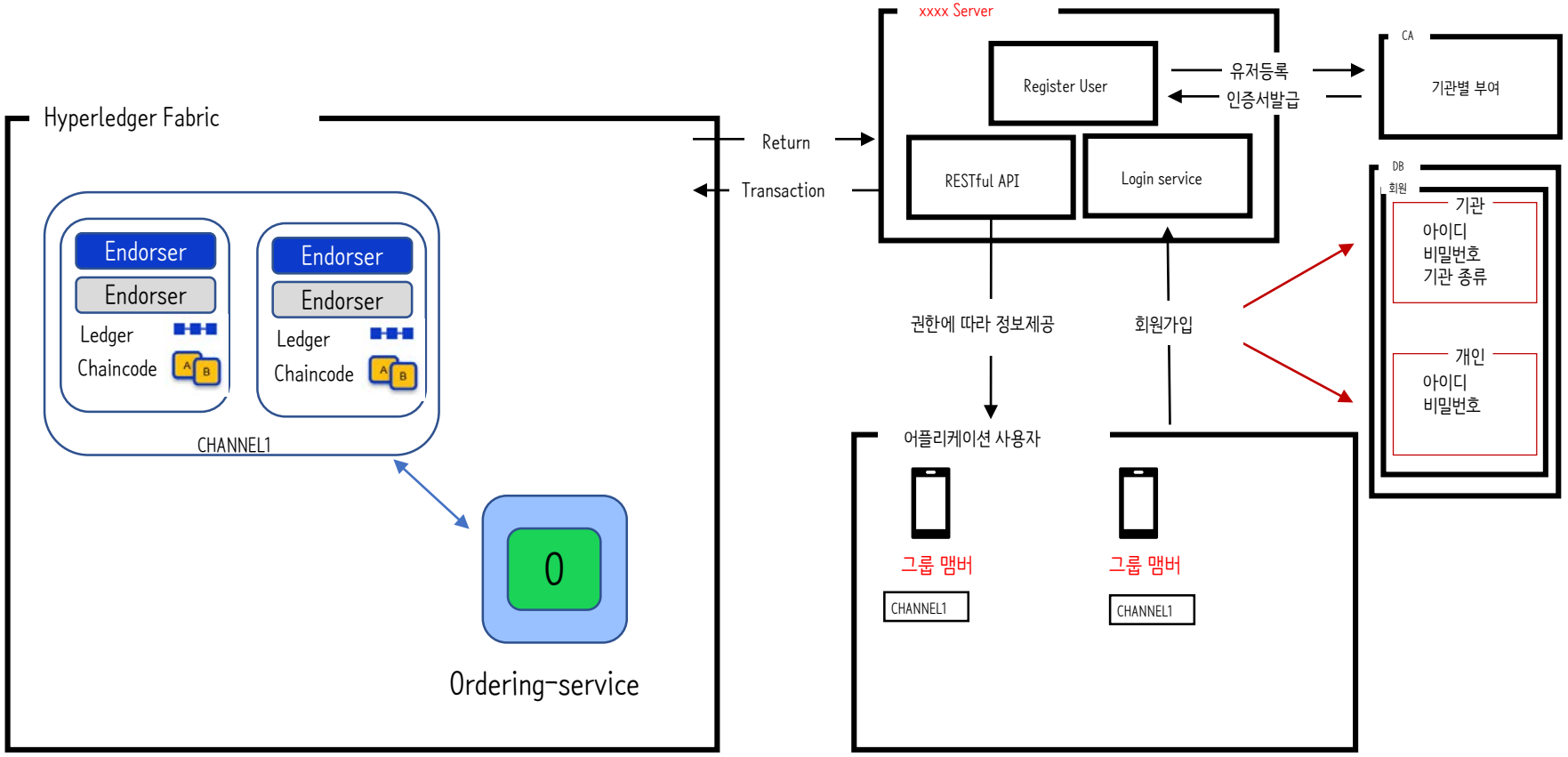
- 문서화(발표자료, 소스코드정리)
- 포함사항
- 발표순서정하기
- 발표내용
  - 비즈니스 모델 개요
  - 시스템 구조도
  - 체인코드(데이터, 기능리스트)
  - Front-end(웹인터페이스, 웹서버)
- 리허설

자동차정보앱 블록체인 (예시)

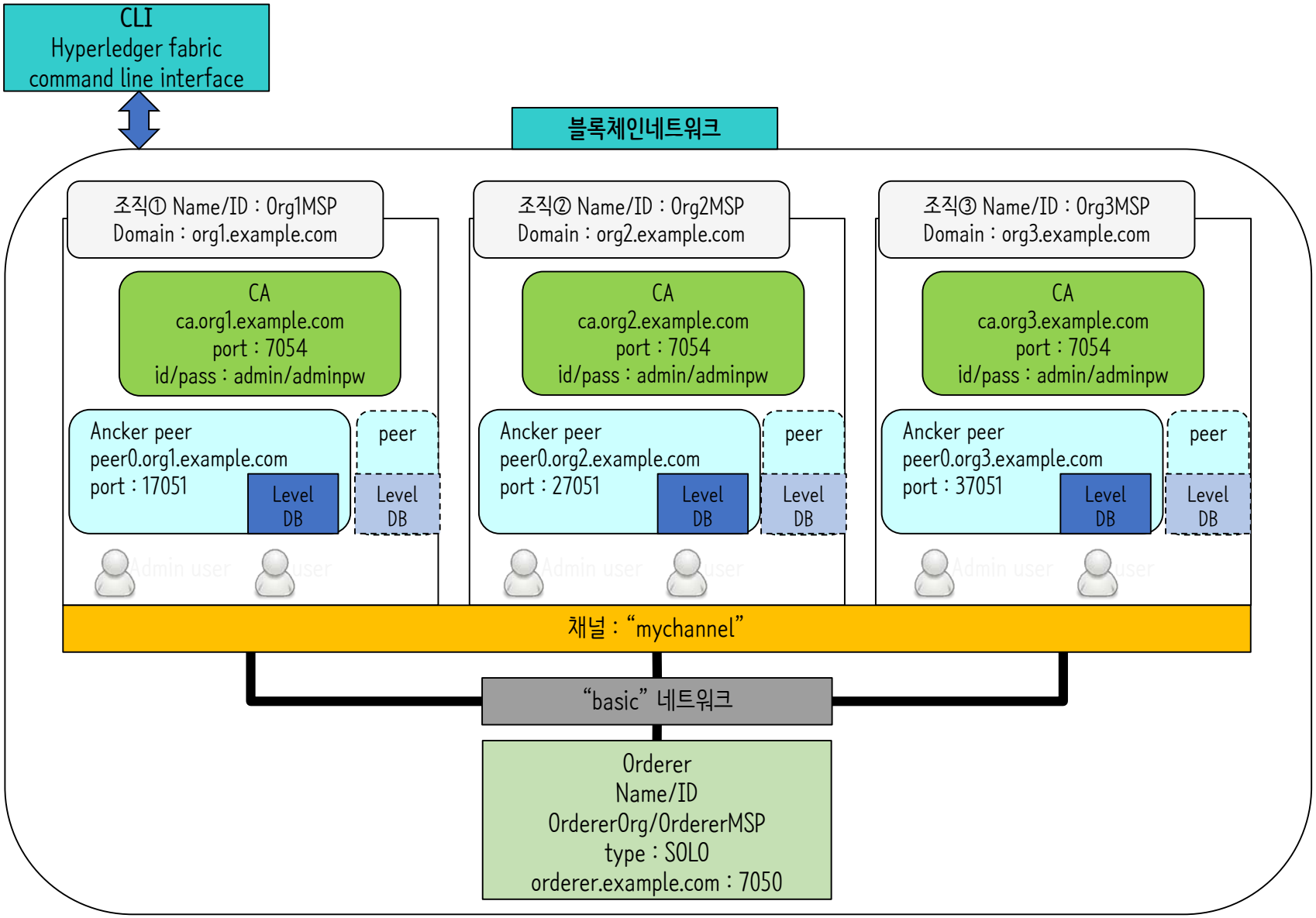
- 자동차정보앱 개요
  - 자동차모델에 대한 정보를 저장하는 블록체인을 구축하고
  - 생산자와 소비자사이를 이어줄 수 있는 편리한 Dapp을 구성
- 자동차정보앱 블록체인의 장점
  - 플랫폼과 데이터의 품질 향상 (투명성, 신뢰성, 가용성, 안정성)
  - 다른 기술과 융합(확장) 수월 (IoT, 인공지능, 빅데이터 등)
- 자동차정보앱 블록체인의 목적
  - 분산화 (Decentralization)
  - 보안성 강화 (Security)
  - 성능 향상과 투명성 (Performance and Transparency)
- 자동차정보앱 블록체인 개발을 통한 의미
  1. web3 기술 획득 데모 (Demonstration of procurement of web3 technology)
  2. 광범위한 블록체인과 데이터베이스로 확장 가능



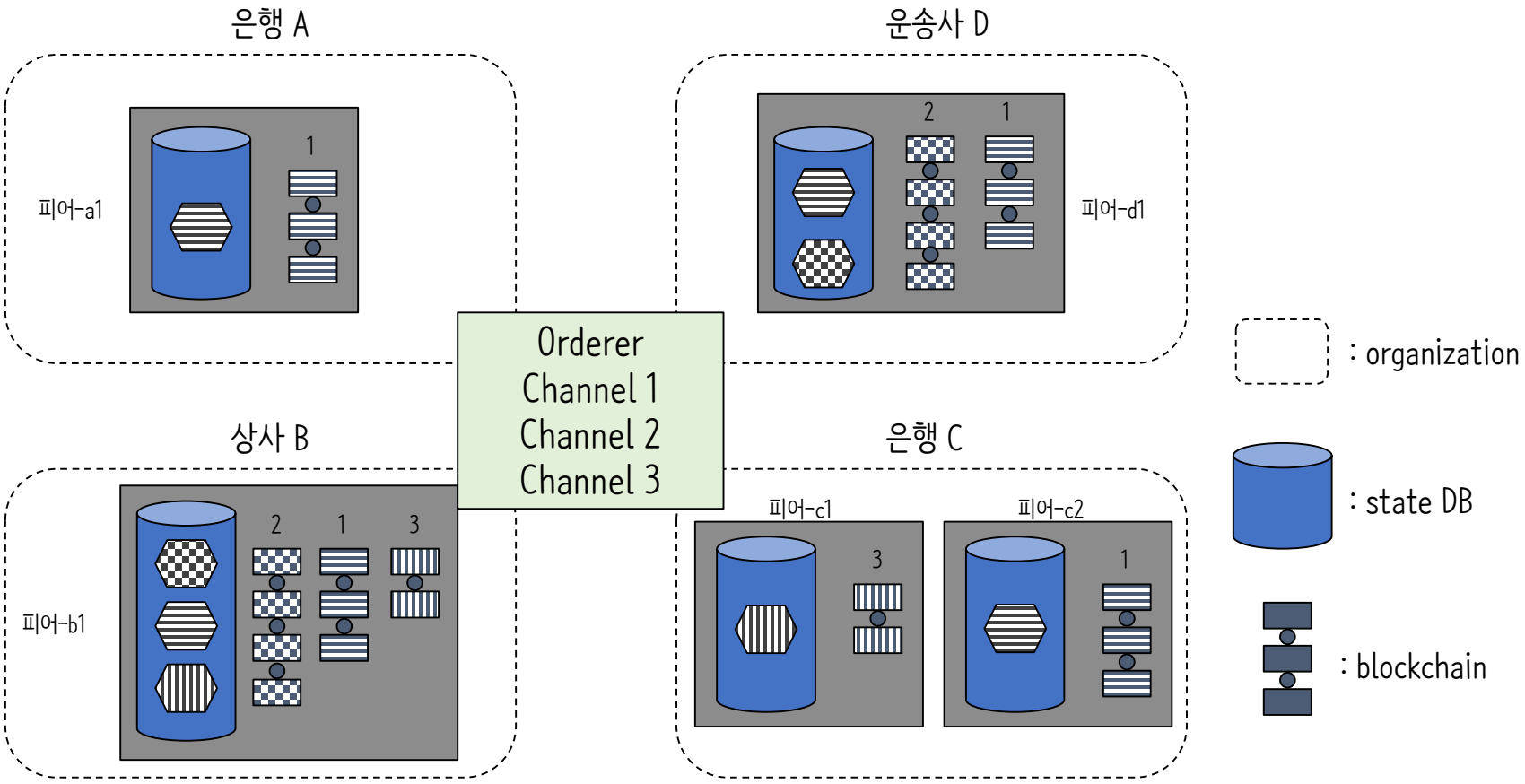
전체 구조도작성 및 설계


Hyperledger Fabric
Endorser
Endorser
Ledger
Chaincode
Endorser
Endorser
Ledger
Chaincode
CHANNEL1
0
Ordering-service
Return
Transaction
xxxx Server
Register User
RESTful API
Login service
유저등록
인증서발급
CA
기관별 부여
DB
회원
기관
아이디
비밀번호
기관 종류
개인
아이디
비밀번호
권한에 따라 정보제공
회원가입
어플리케이션 사용자
그룹 맴버
CHANNEL1
그룹 맴버
CHANNEL1

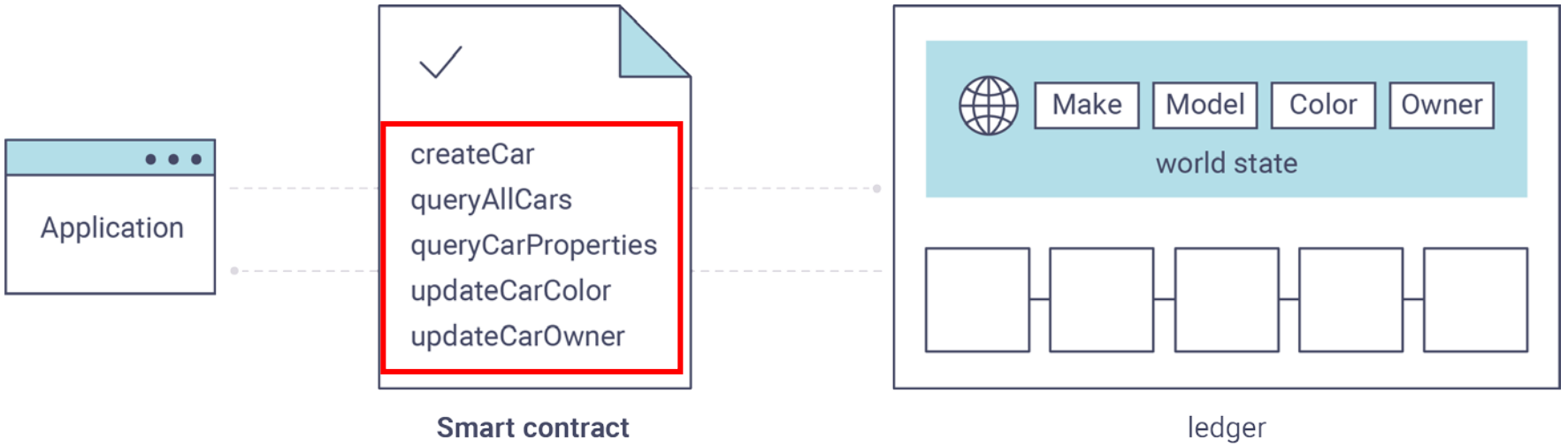
네트워크 설계


CLI
Hyperledger fabric
command line interface
블록체인네트워크
조직① Name/ID : Org1MSP
Domain : org1.example.com
조직② Name/ID : Org2MSP
Domain : org2.example.com
조직③ Name/ID : Org3MSP
Domain : org3.example.com
CA
ca.org1.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org2.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org3.example.com
port : 7054
id/pass : admin/adminpw
Ancker peer
peer0.org1.example.com
port : 17051
Level DB
peer
Level DB
Ancker peer
peer0.org2.example.com
port : 27051
Level DB
peer
Level DB
Ancker peer
peer0.org3.example.com
port : 37051
Level DB
peer
Level DB
Admin user
user
Admin user
user
Admin user
user
채널 : “mychannel”
“basic” 네트워크
Orderer
Name/ID
OrdererOrg/OrdererMSP
type : SOLO
orderer.example.com : 7050

네트워크 구성


은행 A
피어-a1
1
운송사 D
2
1
피어-d1
Orderer
Channel 1
Channel 2
Channel 3
상사 B
피어-b1
2
1
3
은행 C
피어-c1
3
피어-c2
1
: organization
: state DB
: blockchain

데이터, 기능리스트


Application
createCar
queryAllCars
queryCarProperties
updateCarColor
updateCarOwner
Smart contract
Make
Model
Color
Owner
world state
ledger

웹인터페이스

Fabcar

전체리스트 가져오기

차 정보 가져오기

새로운 차 생성하기

차 소유자 변경하기

Fabcar

전체리스트

Car No	색상	메이커	모델명	소유자

차 정보 가져오기

새로운 차 생성하기

차 소유자 변경하기

Fabcar

새로운 차 생성하기

Car No

색상

메이커

모델명

소유자

등록하기

데모준비

- 쉘스크립트를 수행하여 네트워크가동
- 필요시 체인코드 Invoke, Query수행
- 웹서버가동
- 웹서버 접속 후 기능사용