**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | 여기모영 |
| 팀 명 | 꾹꾹 |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.1 |
| **Date** | 2022-05-16 |

|  |  |
| --- | --- |
| **팀원** | 안 성열 (조장) |
| 김 상윤 |
| 민 태식 |
| **지도교수** |  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “여기모영”을 수행하는 팀 “꾹꾹”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “꾹꾹”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 결과보고서-여기모영.doc |
| **원안작성자** | 김상윤, 민태식, 안성열 |
| **수정작업자** | 김상윤, 민태식, 안성열 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2022-05-04 | 안성열 | 1.0 | 최초 작성 | 목차 및 1번 내용 작성 |
| 2022-05-16 | 안성열 | 1.1 | 초안 작성 | 2, 3번 내용 작성 |
| 2022-05-22 | 안성열 | 1.2 | 추가 작성 | 4번 내용 작성 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목차

[1 기존 프로젝트 4](#_Toc104131305)

[1.1 기존 프로젝트 폐기 이유 4](#_Toc104131306)

[2 프로젝트 개요 5](#_Toc104131307)

[2.1 프로젝트 추진 배경 및 필요성 5](#_Toc104131308)

[3 개발 내용 및 결과물 6](#_Toc104131309)

[3.1 목표 6](#_Toc104131310)

[3.2 개발 내용 및 결과물 7](#_Toc104131311)

[3.2.1 앱 어플리케이션(React-Native) 7](#_Toc104131312)

[3.2.2 웹 어플리케이션 서버(Node.js) 7](#_Toc104131313)

[3.2.3 AWS EC2 12](#_Toc104131314)

[3.2.4 데이터베이스 12](#_Toc104131315)

[3.2.5 블록체인 13](#_Toc104131316)

[4 자기평가 15](#_Toc104131317)

[5 부록 16](#_Toc104131318)

[5.1 사용자 메뉴얼 16](#_Toc104131319)

[5.1.1 블록체인 채굴 참여 16](#_Toc104131320)

[5.2 운영자 메뉴얼 17](#_Toc104131321)

[5.2.1 블록체인 네트워크 구성 17](#_Toc104131322)

# 기존 프로젝트

기존 프로젝트명은 ‘국민을국회로’ 이다. ‘국민을국회로’는 입법예고제도를 통해 공시된 발의 법안을 사용자에게 제공하고, 블록체인 기반의 투표를 통해 여론을 형성하는 모바일 어플리케이션 형태의 서비스이다.

입법예고제도란 발의된 법안이 심사를 통해 최종적으로 공포되는 과정 전까지 국민들의 의견을 수렴하기 위해 마련된 제도이다.

기존 프로젝트의 제작 배경은 다음과 같다.

* 새로 발의된 법안은 ‘국회 웹사이트’, ‘정부 웹사이트’ 두 가지의 방법으로 게시된다. 새로 발의된 법안은 하나의 웹 사이트에서 찾아볼 수 없다.
* 각 웹사이트에는 국민의 의견을 수렴하는 기능이 마련되어 있지만, 현재 찬성, 반대 등의 의견이 90% 이상을 차지한다. 기능의 본 목적과 사용자의 요구사항이 일치하지 않는다.

기존 프로젝트에서는 다음과 같은 해결책을 마련했다.

* 국회 홈페이지, 정부 홈페이지의 게시된 법안을 크롤링하여 모바일 어플리케이션의 형태로 사용자에게 제공한다.
* 의견 수렴 기능을 ‘찬반 투표 기능’과 ‘세부 의견 작성’으로 기능을 분리한다. 블록체인을 사용하여 찬반 투표 과정에서 투표 결과와 비밀 투표를 가능하게 했다.

## 기존 프로젝트 폐기 이유

* 현실적인 문제 : ‘법’ 이라는 주제의 진입장벽이 높기 때문에 사람들의 관심을 끌기 어렵고 타겟 사용자의 수가 극히 적다. 또한, 현재 국회나 정부에서 국민들의 의견을 필수적으로 반영해야 한다는 조건이 없어 ‘입법예고제도’의 실효성에 의문이 있다.
* 기술적인 문제 : 블록체인 기술을 이용하여 투표 조작을 막을 수는 있지만, 부 계정을 이용한 여론 조작을 막을 수는 없다.

# 프로젝트 개요

새로운 프로젝트명은 ‘모두모영’ 이다. ‘모두모영’은 모임의 회비 내역을 공유하는 모바일 어플리케이션이다. 모임의 총무는 결제 내역을 추가하고 영수증 실물을 업로드 한다. 모임의 모든 참가자들은 이를 조회할 수 있다. 또한, 사용자는 모임의 회비 내역을 저장하는 방식을 일반 DB와 블록체인으로 선택할 수 있다.

## 프로젝트 추진 배경 및 필요성

매년 모든 대학교의 학생회에서 학생 회비와 관련된 잡음이 지속되고 있다. 학생회의 목적에 맞지 않는 회비를 사용하거나, 회비 사용 내역에 조작이 의심되는 경우가 다수 발견된다.

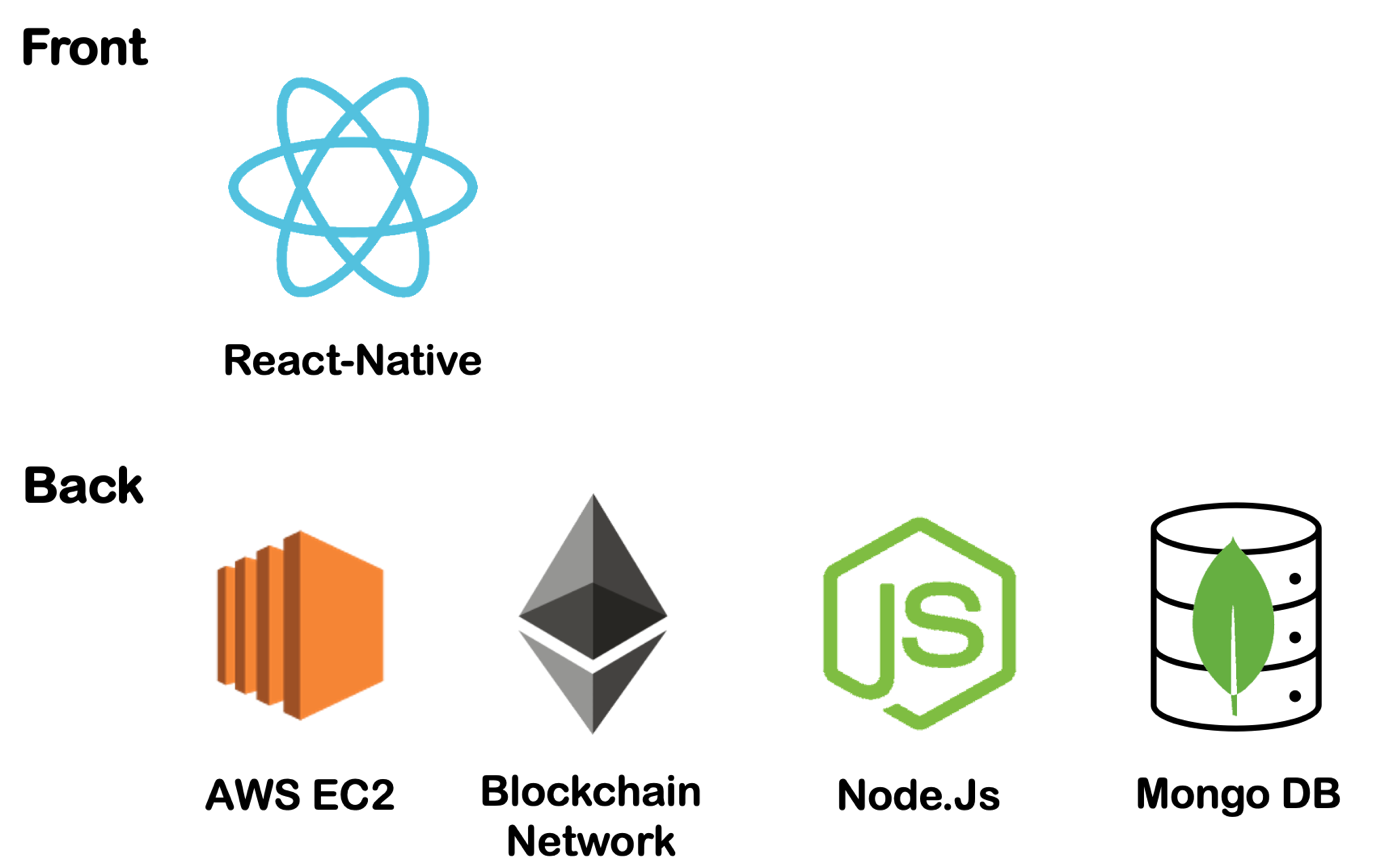
해당 문제점은 총무가 회비 사용 내역을 분실하거나, 내역 집계 중에 실수를 저지르거나, 회비 사용 내역을 곧바로 공개하지 않거나, 현실적으로 회비 사용 내역을 모두에게 공개하기 어렵기 때문에 발생한다. 본 프로젝트는 총무가 모임의 회비를 편하게 관리하고, 모임 참여자가 회비 내역을 모두 조회하여 서로 간의 갈등을 해결하기 위해 제작되었다.

# 개발 내용 및 결과물

## 목표

* 사용자에게 모바일 어플리케이션 형태의 서비스를 제공하기 위해 앱 어플리케이션 서버(React-Native)를 구성한다.
* 클라이언트의 요청을 처리하기 위해 웹 어플리케이션 서버(Node.js)를 구성한다.
* 고정된 IP, PORT 를 사용하기 위해 웹 어플리케이션 서버를 AWS EC2에서 구동한다.
* 사용자에게 모임의 정보 저장 방식을 2가지로 제공하기 위해 일반 DB와 블록체인 네트워크를 구축한다.
* 모임 사용 내역의 신뢰성을 부여하기 위해 실물 영수증의 데이터를 저장한다.

아래는 프로젝트에 사용된 도구들을 이미지로 나타낸 것이다.



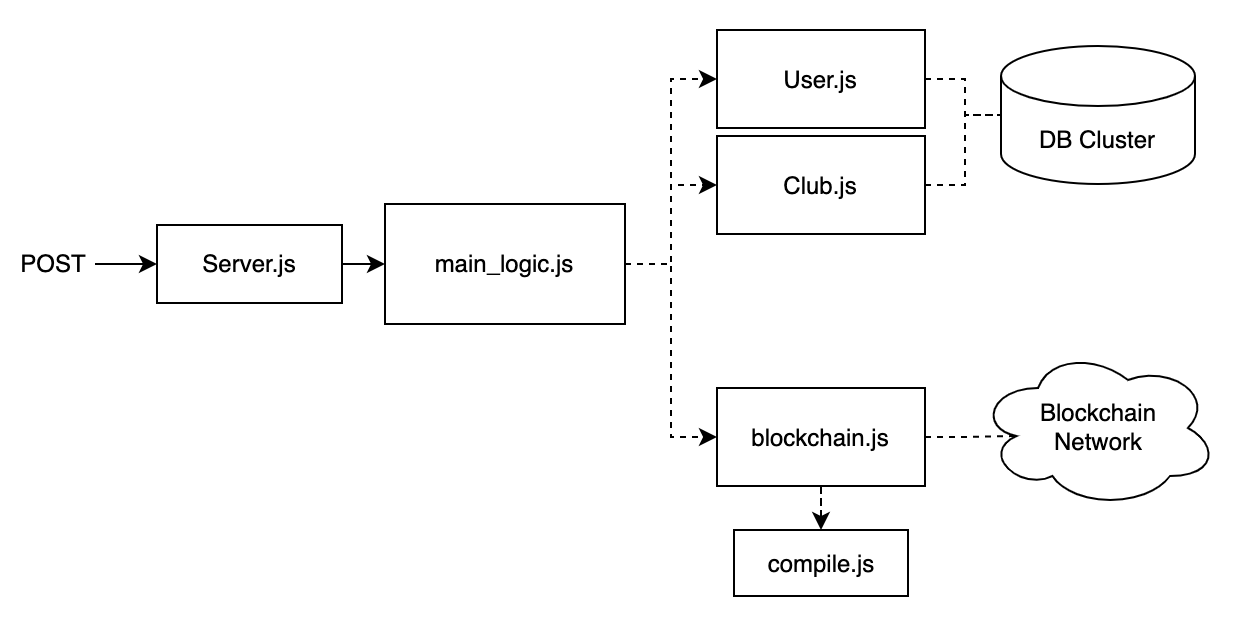
## 개발 내용 및 결과물

### 앱 어플리케이션(React-Native)

### 웹 어플리케이션 서버(Node.js)

Node.js는 web3 모듈을 이용하여 블록체인 네트워크(Geth)와 통신하기 위한 표준 플랫폼이다. 또한 본 프로젝트에서는 소규모의 서비스를 제공하기 때문에 Node.js 가 적합하다고 판단했다.

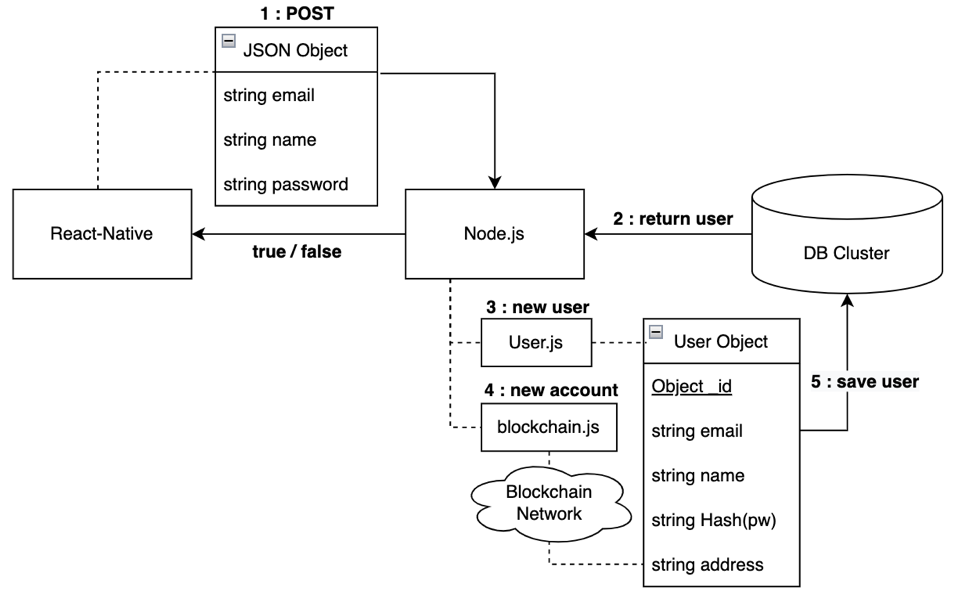
웹 어플리케이션 서버는 총 6개의 컴포넌트로 구성되어 있다.



* server.js : 앱 서버에서 요청한 기능을 비즈니스 로직(main\_logic)으로 연결하는 역할을 수행한다.
* main\_logic.js : 정의된 기능에 맞는 데이터 처리 역할을 담당한다. 모임 정보 저장 방식에 따라 두가지 방식으로 분기된다. 다음 페이지에서 자세히 설명한다.
* User.js : DB에 저장할 User의 데이터 구조가 정의되어 있다. 3.2.4에서 자세하게 설명한다.
* Club.js : DB에 저장할 Club의 데이터 구조가 정의되어 있다. 3.2.4에서 자세하게 설명한다.
* blockchain.js : 블록체인 네트워크와 RPC(Remote Procedure Call) 통신을 수행한다. 3.2.5에서 자세하게 설명한다.
* compile.js : Club 스마트 컨트랙트를 컴파일하고 bytecode, ABI(Application Binary Interface)를 반환하는 역할을 수행한다.

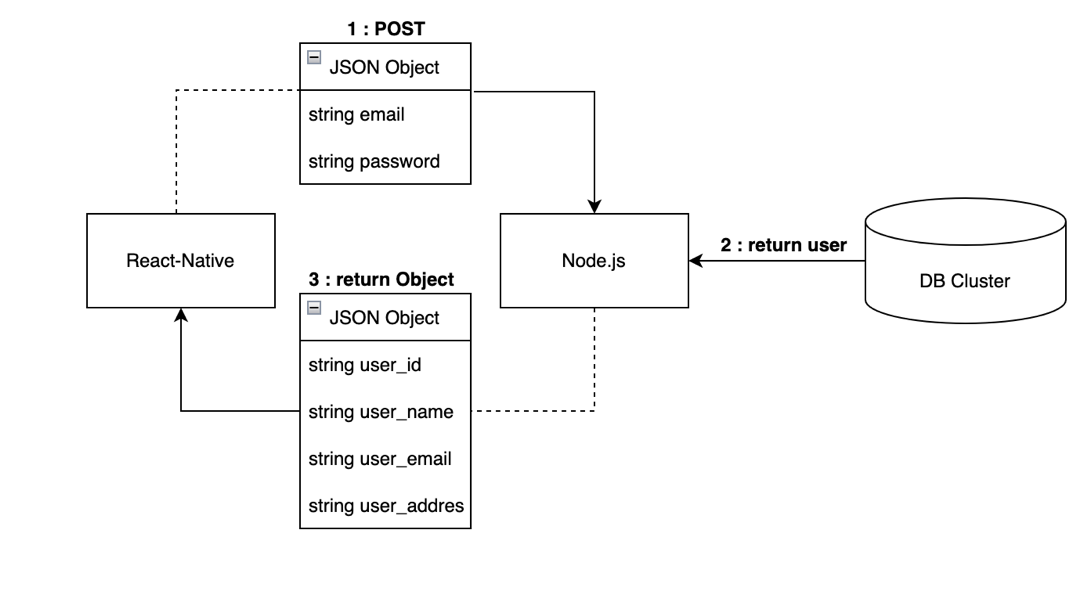
main\_logic 컴포넌트에서는 사용자가 요청한 기능에 따라 총 7개의 역할을 수행한다.

* 회원가입 (register)



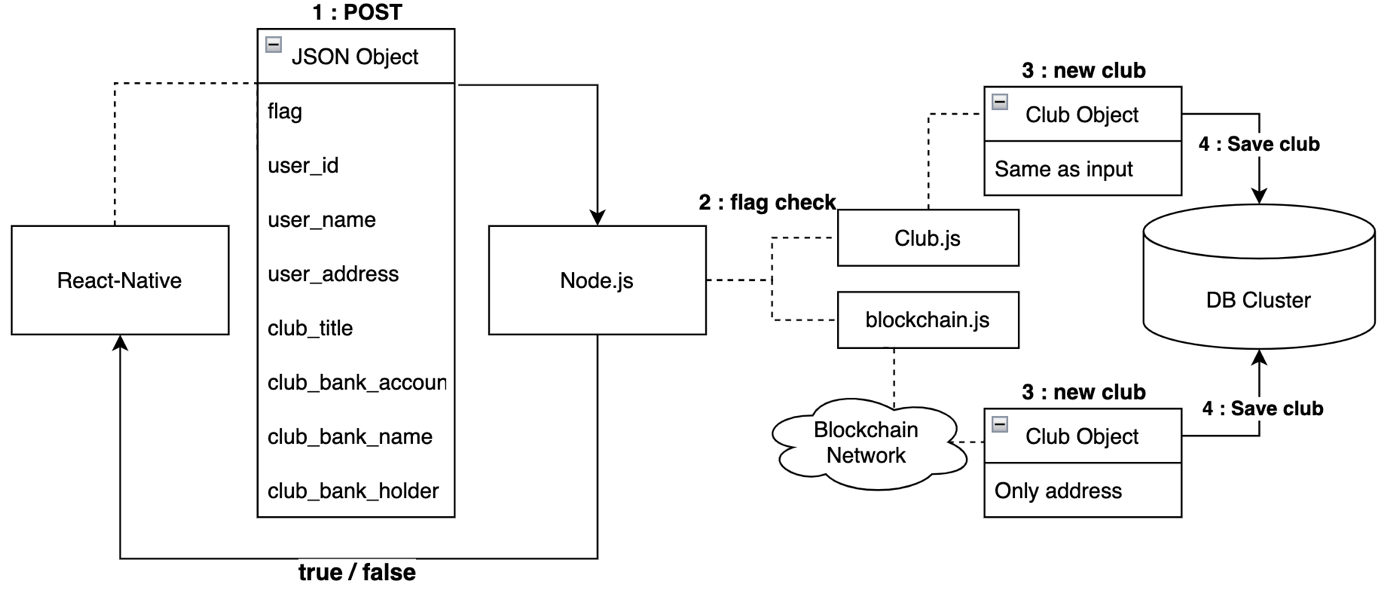
사용자가 입력한 데이터를 받는다. DB을 조회하여 이메일 중복 검사를 진행한다. 새로운 User를 만들고 블록체인 네트워크에서 계정을 생성한다. DB에 User를 저장한다.

* 로그인 (login)



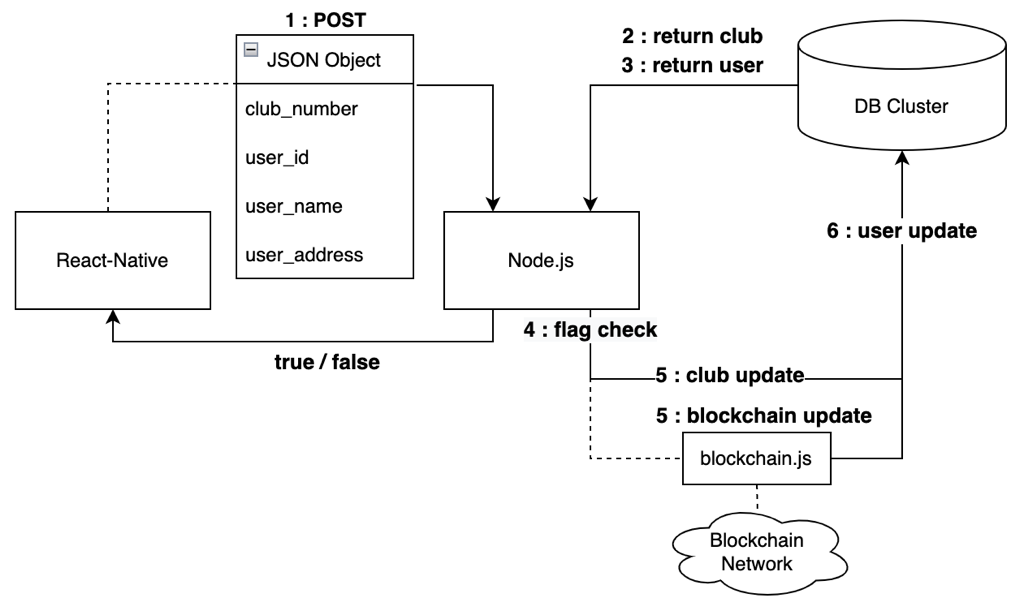
사용자가 입력한 데이터를 받는다. DB를 조회하여 비밀번호 일치 여부를 진행한다. User의 데이터 일부분을 가공하여 응답한다.

* 모임 생성 (createClub)



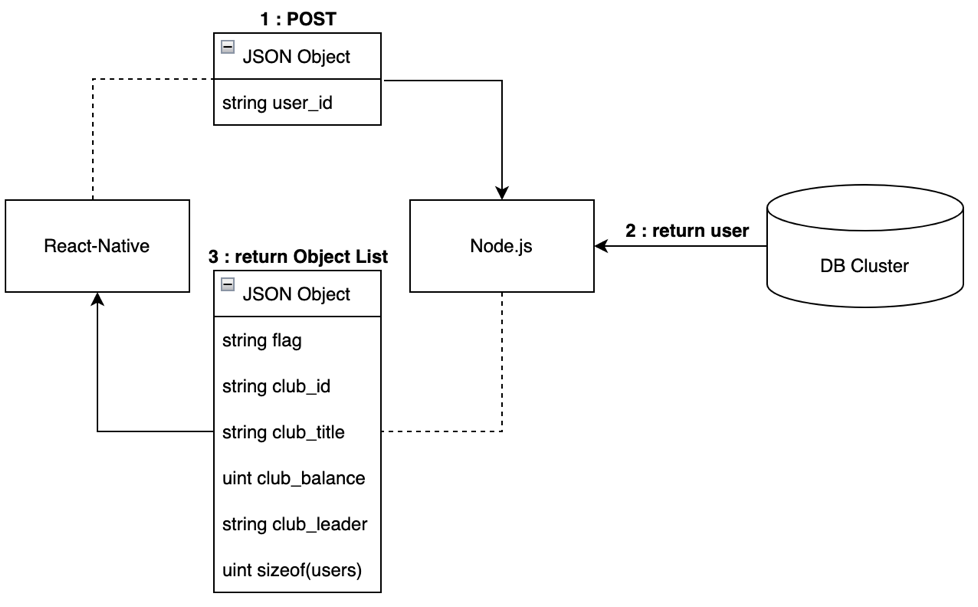
사용자가 입력한 데이터를 받는다. 모임의 저장 방식(flag)에 따라 분기한다. 일반 DB 방식의 경우, 새로운 Club을 만들고 입력한 데이터를 대치한다. 블록체인 방식의 경우, 입력한 데이터로 스마트 컨트랙트를 발급하고 주소를 반환한다. 새로운 Club에 주소를 대치한다. DB에 Club을 저장한다.

* 모임 참가 (joinClub)



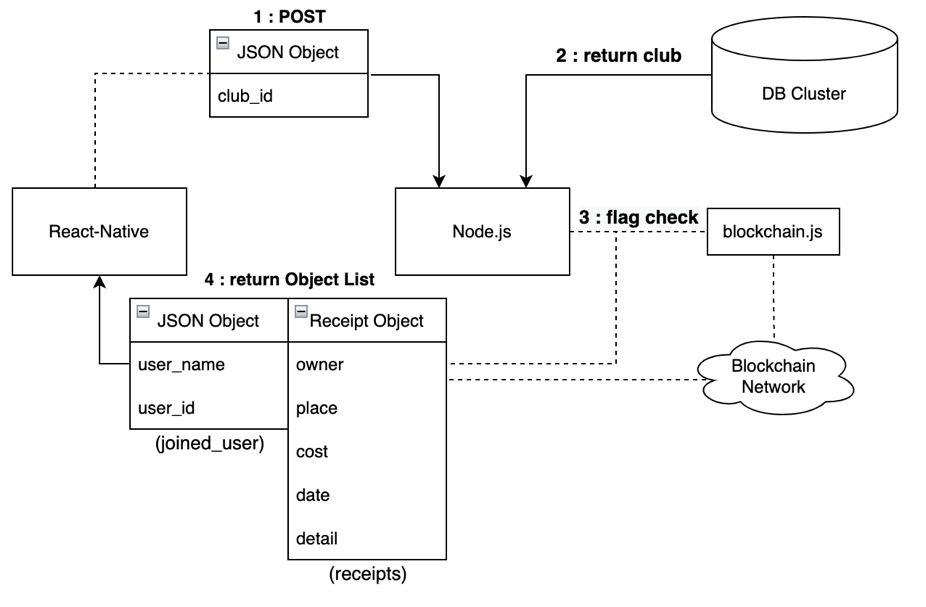
사용자가 입력한 데이터를 받는다. DB에서 일치하는 club, user를 얻는다. Club의 저장 방식에 따라 분기한다. 일반 DB 방식의 경우, club의 참여자에 user를 추가한다. 블록체인 방식의 경우, 스마트 컨트랙트 변수에 user를 추가한다. 마지막으로 user의 참여모임에 club을 추가한다.

* 사용자 모임 조회 (userClubInfo)



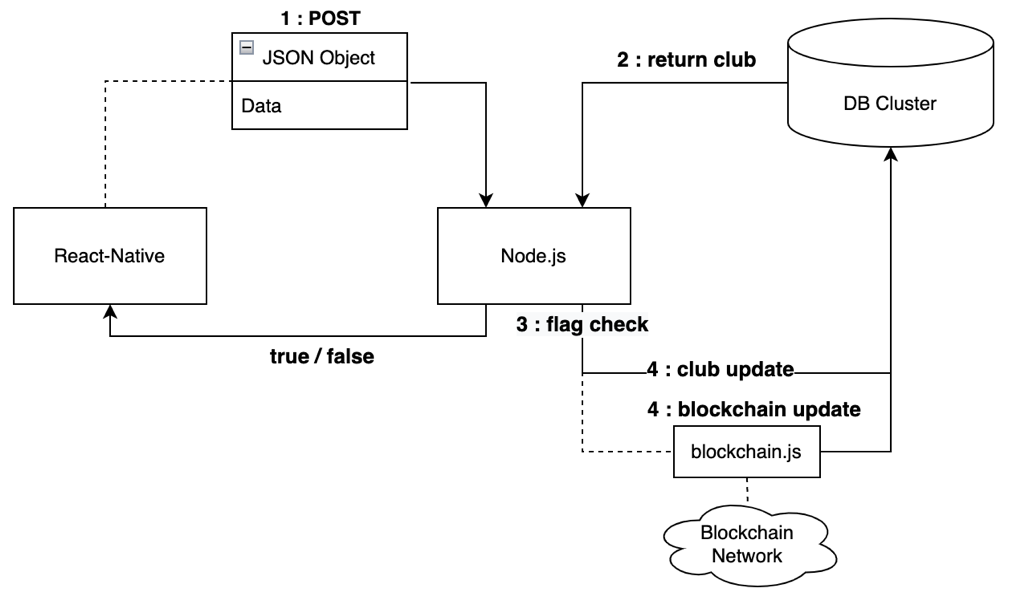
사용자가 입력한 데이터를 받는다. DB에서 user를 조회한다. User의 참여 모임 정보의 일부분을 가공하여 응답한다.

* 모임 페이지 이동 (gotoClub)

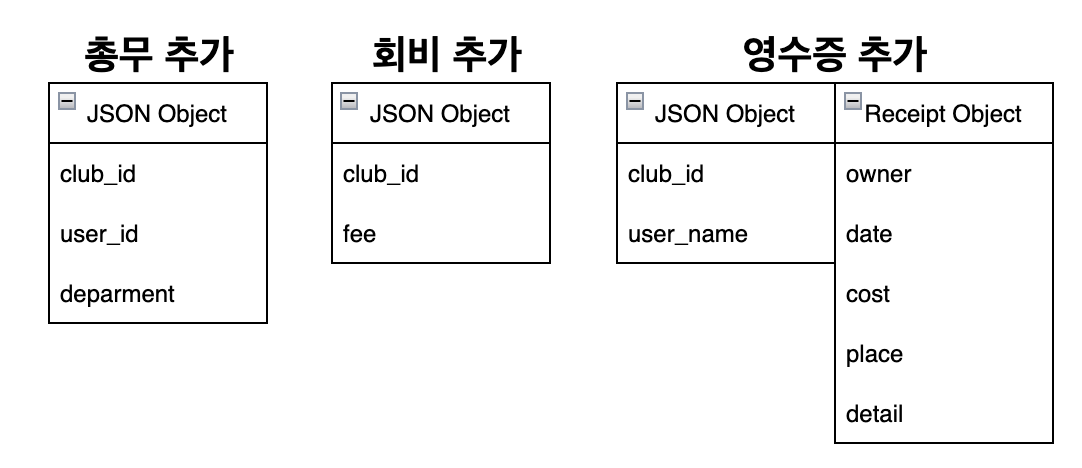


사용자가 입력한 데이터를 받는다. DB에서 club을 조회한다. Club의 저장 방식에 따라 분기한다. 일반 DB 방식의 경우, club의 참여자 정보와 club의 영수증 정보 일부분을 가공하여 응답한다. 블록체인 방식의 경우, 스마트 컨트랙트의 변수 중 일부분을 가공하여 응답한다.

* 총무 추가 (addClubMember), 회비 추가 (addClubFee), 영수증 추가 (addClubReceipt)



사용자가 입력한 데이터를 받는다. DB에서 club을 조회한다. Club의 저장 방식에 따라 분기한다. 일반 DB 방식의 경우, club의 데이터를 변경한다. 블록체인 방식의 경우, 스마트 컨트랙트의 변수를 변경한다. 세 기능 모두 동일한 흐름이나, 아래 그림과 같이 입력되는 데이터에 차이가 있다.



### AWS EC2

AWS EC2는 서버를 개설하는 가장 일반적인 방법이다. EC2 인스턴스에서 Node.js와 Geth를 구동한다. Geth에서 블록을 채굴할 수 있는 최소 사양 조건과 지원받을 수 있는 금액 조건에 맞추어 t2.medium을 선택했다. React-Native의 연결 주소를 고정하기 위해 Elastic IP를 배정받았고 Port를 3001로 고정했다.

### 데이터베이스

어떤 데이터베이스를 선택했고 그 이유?

데이터베이스의 데이터 저장 구조

* 일반 DB 선택 시
* 블록체인 선택 : 블록체인에 배포된 스마트 컨트랙트의 주소인 CA(Contract Address)가 저장된다.

### 블록체인

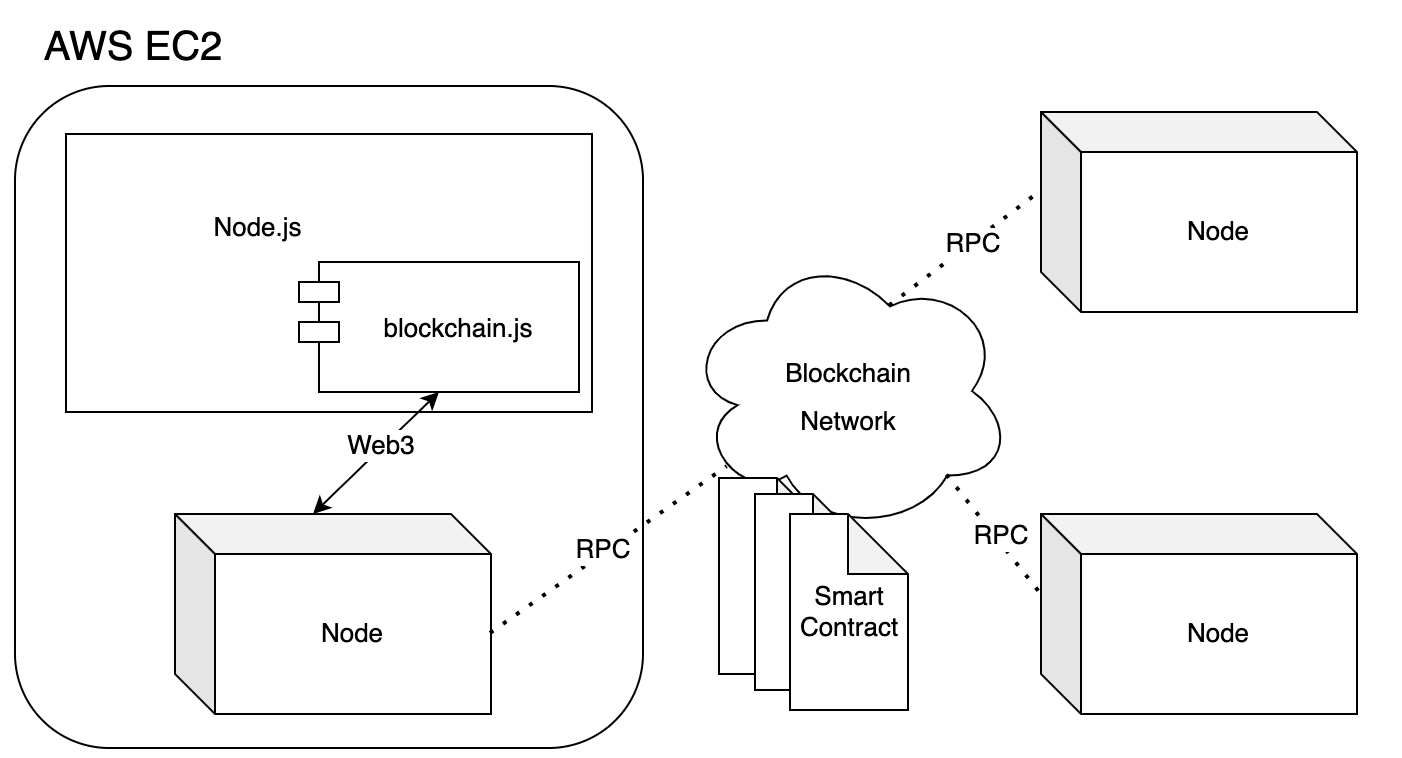
블록체인은 P2P 기반의 데이터 저장 방식이다. 데이터는 블록에 기록되고 해당 블록의 해쉬값이 다음 블록에 저장된다. 누군가 임의적으로 블록의 데이터를 변경하면 이어지는 블록들의 해쉬값이 변경된다. 블록체인 네트워크의 참가자들은 모두 동일한 데이터를 가지고 있기 때문에, 이를 승인하지 않는다. 결과적으로, 블록체인에 저장된 데이터를 조작할 수 없다. 본 프로젝트에서는 총무가 회비를 데이터를 고의적으로 조작하는 것을 방지하기 위해 블록체인을 도입했다.

블록체인 네트워크는 이더리움 방식 중 하나인 Go-ethereum(이하 ‘Geth’) 클라이언트를 사용했다. Geth는 다른 도구(HyperLedger, EOS…)보다 많은 정보가 있어 비교적 학습이 빠르다는 이유로 채택했다.

현재 블록체인은 3개의 노드(작성자의 개인 머신 2개, AWS EC2 1개)로 구성되어 있다. 서비스의 특징에 맞게 블록체인의 환경설정을 변경했다.

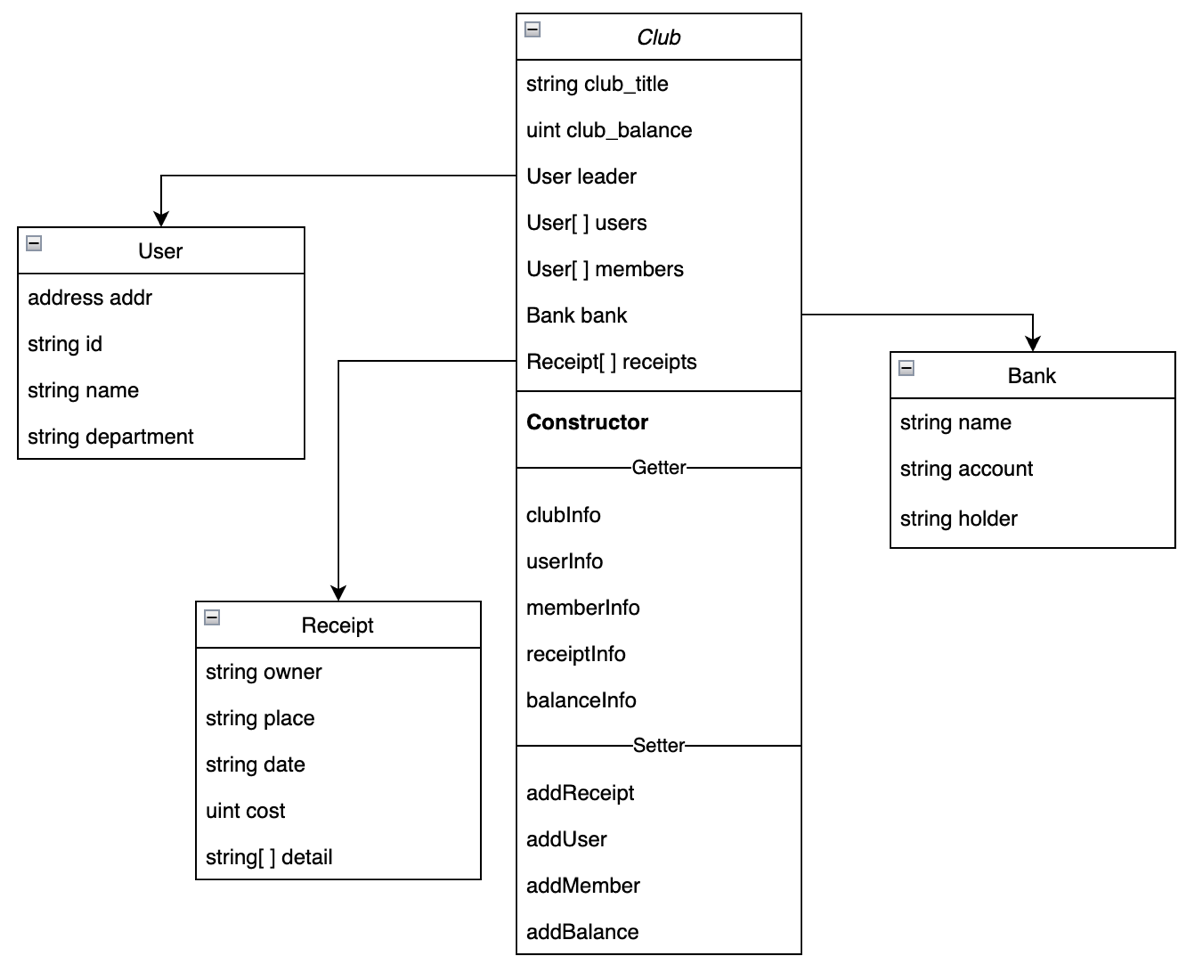
* Base64 데이터를 저장하기 위해 개별 트랜잭션 처리량을 128KB에서 약 10MB로 증가시켰다.
* Base64 데이터를 저장하는 함수를 실행하는 경우 트랜잭션 처리 비용인 gas를 약 300배 증가시켰다.
* 트랜잭션 처리 비용이 증가함에 따라, 최대 gas 사용량인 gasLimit을 300배 증가시켰다.
* 채굴에 사용되는 컴퓨팅 비용을 줄이기 위해 트랜잭션이 생성될 때 3개의 블록을 채굴하도록 스크립트를 작성했다.

이더리움 기반의 블록체인은 스마트 컨트랙트를 미리 작성하고, 서버에서 함수를 실행 요청할 수 있다. 함수를 실행하는 주체(서버)는 채굴자에게 지급되는 비용인 gas를 지불한다. 이때, 사용자는 채굴을 하지 않기 때문에 gas가 없다. 이를 해결하기 위해 사용자가 계정을 생성하는 경우, 10 Ether(, 약 300회 트랜잭션 처리 가능)를 보내는 방법을 사용했다.



사용자가 모임을 블록체인으로 생성한 경우, AWS EC2에서 참여한 노드를 통해 Club 스마트 컨트랙트가 블록체인 네트워크에 배포된다. 된다. 해당 컨트랙트에서 데이터 변경이 일어날때마다 트랜잭션이 발생되고, 이는 모두 블록체인에 기록된다. 스마트 컨트랙트의 데이터 구조는 일반 DB에 저장되는 데이터 구조와 가능한 동일하게 구성했다.

아래는 스마트 컨트랙트의 데이터 구조이다.



스마트 컨트랙트에는 modifier를 설정하여 Getter / Setter 함수를 실행하는 주체를 Leader, Member, User로 제한했다. 일반 사용자(User)는 내역 조회만 가능하고 모임장(Leader), 총무(Member)는 변수 변경이 가능하도록 구성했다.

# 자기평가

## 안성열

Geth 클라이언트를 이용한 블록체인 네트워크 구성, 스마트 컨트랙트 코드 개발, Node.js 서버 개발, Node.js 서버와 블록체인 네트워크 통신, AWS EC2 개설 작업 및 코드 이식을 담당했다. 3팀의 팀장으로서 프로젝트 추진 및 진행상황을 관리했고, 팀원 간 적극적인 소통을 위해 노력했다.

# 부록

## 사용자 메뉴얼

### 블록체인 채굴 참여

본 프로젝트에서 개설한 네트워크에 참여하는 방법을 설명한다.

* Geth 클라이언트 다운로드

<https://geth.ethereum.org/downloads/> 에 접속하여 본인의 환경에 맞는 버전을 설치한다.

* Genesis.json 파일 다운로드

<https://github.com/kookmin-sw/capstone-2022-03/blob/master/genesis.json> 에 접속하여 파일을 다운로드 받는다.

* 블록체인 네트워크 실행

$ geth init --datadir <file\_name> genesis.json : 네트워크 개설을 위한 초기화 작업이다.

$ geth –datadir <file\_name> --networkid 2022 console : 네트워크를 개설한다.

* 블록체인 네트워크 노드 참가

> admin.addPeer("enode://c0a7e4fa2e1e9f23fdcf45e96eb432e251c31725e8756b05578fb9e3d96338e34241f46c32bc335a7aba61d59f57122fc31baef9eef5bd6f973890dc8538ab3b@118.33.68.135:30303?discport=0”)

> admin.peers

* 채굴 시작

> personal.newAccount(“<password>”) : 계정을 생성한다. 처음 실행하는 경우 자동으로 채굴비용을 지급하는 계정으로 설정된다.

> miner.start(<thread\_count>) : 채굴을 시작한다. Thread\_count 만큼 Thread를 사용한다. Thread\_count가 높을 수록 채굴이 빨라진다.

> miner.stop() : 채굴을 중지한다.

> eth.getBalance(eth.coinbase) : 현재까지의 채굴 보상을 확인한다.

## 운영자 메뉴얼

### 블록체인 네트워크 구성

본 프로젝트에서 개설한 네트워크에 참여하지 않고, 다른 사설 네트워크를 개설하는 방법을 설명한다.

* Geth 클라이언트 다운로드

<https://geth.ethereum.org/downloads/> 에 접속하여 본인의 환경에 맞는 버전을 설치한다.

* genesis.json 파일 다운로드

<https://github.com/kookmin-sw/capstone-2022-03/blob/master/genesis.json> 에 접속하여 파일을 다운로드 받는다. 이때, chainId, difficulty, gasLimit의 값을 변경했다.

chainId : 사설 블록체인 네트워크의 고유 id 값. 참가하기 위해 동일해야 한다.

difficulty : 블록을 채굴하는 난이도이다. 값이 높을수록 블록을 채굴하는 평균 시간이 증가한다.

gasLimit : 트랜잭션을 일으키는 최대 비용이다. 큰 데이터를 사용하면 비용이 증가하고, 이에 따라 gasLimit을 증가시켜야 한다.

* Automine.json 파일 다운로드

<https://github.com/kookmin-sw/capstone-2022-03/blob/master/automine.js> 에 접속하여 파일을 다운로드 받는다. 트랜잭션이 발생하면 3개의 블록을 채굴하도록 조정되었다.

* 사설 네트워크 개설

$ geth init --datadir <file\_name> genesis.json : 네트워크 개설을 위한 초기화 작업이다.

$ geth --datadir <file\_name> --networkid <chainId> --nodiscover –nat extip:<ip\_address> --http –http.addr “<ip\_address>” –http.api “web3,eth,personal” –allow-insecure-unlock –miner.gasprice 0 –preload “<automine.js pwd> console : 네트워크가 개설된다.

* + --nodiscover : 수동으로 노드를 추가한다.
  + --nat extip : 데이터 통신 시 지정 IP로 고정한다.
  + --http : 블록체인 네트워크 연결 시 http 방식을 사용한다.
  + --http.addr : http 방식에서 사용되는 endpoint를 고정한다.
  + --http.api : RPC 통신을 통해 외부에서 실행할 수 있는 모듈을 정의한다.
  + --allow-insecure-unlock : RPC 통신을 통해 외부에서 계정 해제를 가능하게 한다.
  + --miner.gasprice : 채굴자가 트랜잭션을 처리하는 가스 비용을 설정한다.
  + --preload : geth 클라이언트에서 동작하는 스크립트를 설정한다.
* 채굴 시작

> personal.newAccount(“<password>”) : 계정을 생성한다. 처음 실행하는 경우 자동으로 채굴비용을 지급하는 계정으로 설정된다.

> miner.start(<thread\_count>) : 채굴을 시작한다. Thread\_count 만큼 Thread를 사용한다. Thread\_count가 높을 수록 채굴이 빨라진다. (자동채굴 스크립트를 추가한 경우, 해당 명령어를 사용하지 않아도 된다.)

> miner.stop() : 채굴을 중지한다.

> eth.getBalance(eth.coinbase) : 현재까지의 채굴 보상을 확인한다.