

OTP 인증방식을 활용한 블록체인 모니터링 시스템

2018. 11. 7

| | |
|--------|---------|
| 중부대학교 | 정보보호학과 |
| 담당 교수 | 양환석 교수님 |
| 3조 사이다 | 박진아 |
| | 강신선 |
| | 박지성 |
| | 이지현 |

목 차

- 조원 편성
- 주제 선정
- 구 상 도
- 추진 경과
- 개발 환경 및 시스템 개발
- 개발 시스템 운영
- 결론 및 기대효과

조원 편성

| 이름 | 역할 |
|-----|--|
| 박진아 | 웹 페이지 & DB 구축, 블록체인 모니터링 시스템 구현, 프로젝트 총괄 |
| 강신선 | 웹 페이지 & DB 구축, OTP 시스템 구현, 웹 서버 구축 |
| 박지성 | 웹 페이지 & DB 구축, OTP 시스템 구현, 웹 서버 구축 |
| 이지현 | 웹 페이지 & DB 구축, 블록체인 모니터링 시스템 구현, 웹 서버 구축 |

주제 선정

❑ 전자서명법 개정에 따라 공인인증서의 의무사용 규정 폐지 (2018.8.27)

❑ 이에 따라 대체 기술 개발이 필수적

➢ 공인인증서를 대체할 인증 기술로 블록체인 기술이 주목 ⇨ **블록체인 네트워크 시스템을 연구**

➢ OTP 인증을 통과한 관리자만이 **블록체인 모니터링 시스템에 접근**할 수 있도록 설정하여 보안성 강화

지긋지긋한 공인인증서 끝, 이젠 '블록체인'

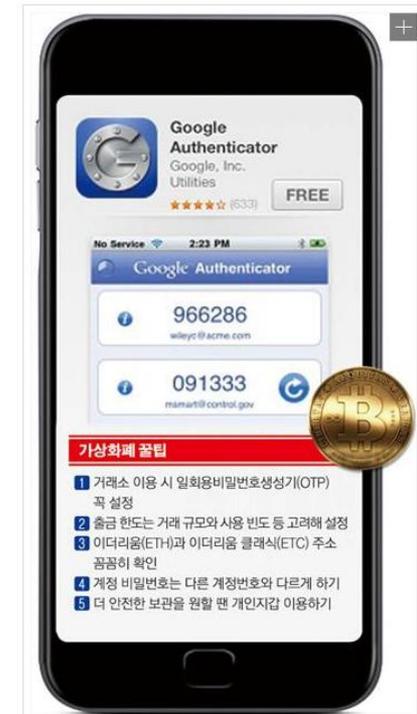
OTP 설정 필수..개인지갑에 보관하면 더 안전

이남의 기자 | 입력:

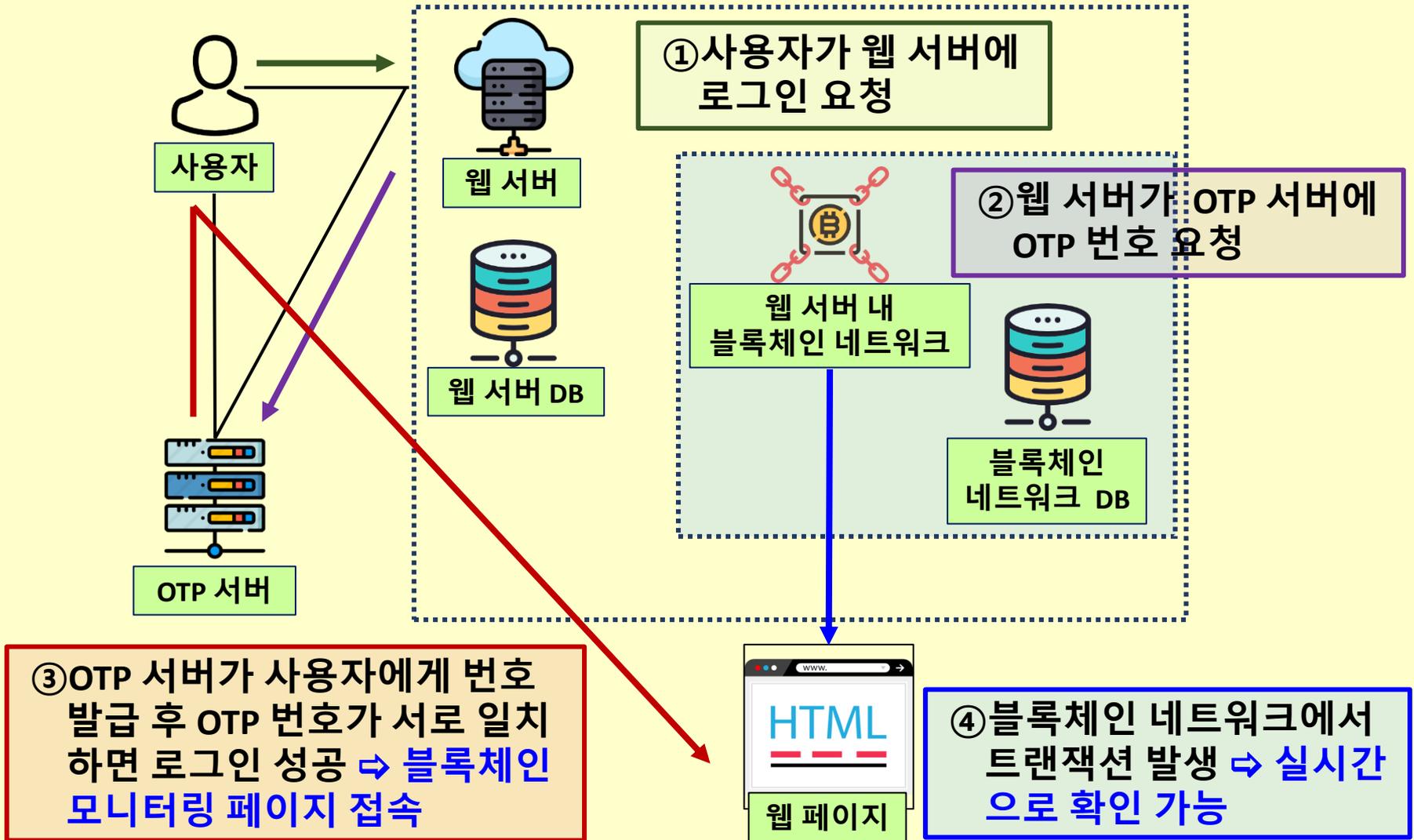
기사공유 f t

기사입력: 2017-06-07 10:45

비트코인 거래 전 이걸 꼭!



구상도



추진 경과

| 기간(2018년) 작업 | 3월 | 4월 | 5월 | 6월 | 7월 | 8월 | 9월 | 10월 |
|-----------------|----|----|----|----|----|----|----|-----|
| 주제선정 및 자료조사 | | | | | | | | |
| 웹 서버 구축 | | | | | | | | |
| 웹 페이지 설계 및 구현 | | | | | | | | |
| OTP 구현 | | | | | | | | |
| 블록체인 모니터링 구현 | | | | | | | | |
| 시스템 통합 및 성능 보강 | | | | | | | | |

개발 환경 및 시스템 개발(1/15)

개발 환경

서버

- Ubuntu 16.04 LTS

데이터베이스

- MySQL

개발언어

- JAVA, JSP, CSS, HTML,
JavaScript, Nodejs

개발 환경 및 시스템 개발(2/15)

블록체인 종류 및 특성

퍼블릭 블록체인

1. 누구나 참여 가능
2. 누구나 거래 검증 / 승인
3. 누구나 트랜잭션 생성
4. 예) **비트코인, 이더리움**

VS

프라이빗 블록체인

1. 허가된 기관, 사용자만 참여 가능
2. 허가된 기관, 사용자가 거래 검증 / 승인
3. 허가된 기관, 사용자만 트랜잭션 생성
4. 예) **Hyperledger 프로젝트**

개발 환경 및 시스템 개발(3/15)

Hyperledger 프로젝트

Hyperledger 프레임워크



Hyperledger 참여 기업

Hyperledger Fabric 구성 요소

Docker - Docker version 17.03.2-ce, build f5ec1e2

Docker-compose - docker-compose version 1.19.0, build 9e633ef

Node - v8.11.3

Npm - v5.6.0

Go - go1.9.2 linux/amd64

Git - v2.7.4

Python - v2.7.12

MySQL - Server version 5.7.23-0Ubuntu0.16.04.1

Swagger - v2.0

개발 환경 및 시스템 개발(4/15)

Hyperledger Fabric - docker

```
Docker 이미지 다운 -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0
```

```
Installing Hyperledger Fabric docker images
```

```
==> Pulling fabric Images
```

```
==> FABRIC IMAGE: peer
```

```
==> List out hyperledger docker images
```

| | | | Docker 이미지 확인 | |
|----|------------------------------|------------------|---------------|----------------------|
| x8 | hyperledger/fabric-ca | latest | 72617b4fa9b4 | |
| 1t | hyperledger/fabric-ca | x86_64-1.1.0 | 72617b4fa9b4 | 7 weeks ago 299MB |
| 6f | hyperledger/fabric-ca | latest | b7bfddf508bc | 7 weeks ago 1.46GB |
| C7 | hyperledger/fabric-tools | latest | b7bfddf508bc | 7 weeks ago 1.46GB |
| 4t | hyperledger/fabric-tools | x86_64-1.1.0 | b7bfddf508bc | 7 weeks ago 1.46GB |
| 06 | hyperledger/fabric-orderer | latest | ce0c810df36a | 7 weeks ago 180MB |
| 40 | hyperledger/fabric-orderer | x86_64-1.1.0 | ce0c810df36a | 7 weeks ago 180MB |
| 0t | hyperledger/fabric-peer | latest | b023f9be0771 | 7 weeks ago 187MB |
| 77 | hyperledger/fabric-peer | x86_64-1.1.0 | b023f9be0771 | 7 weeks ago 187MB |
| 19 | hyperledger/fabric-javaenv | latest | 82098abb1a17 | 7 weeks ago 1.52GB |
| 62 | hyperledger/fabric-javaenv | x86_64-1.1.0 | 82098abb1a17 | 7 weeks ago 1.52GB |
| Df | hyperledger/fabric-ccenv | latest | c8b4909d8d46 | 7 weeks ago 1.39GB |
| St | hyperledger/fabric-ccenv | x86_64-1.1.0 | c8b4909d8d46 | 7 weeks ago 1.39GB |
| == | hyperledger/fabric-zookeeper | latest | 92cbb952b6f8 | 2 months ago 1.39GB |
| == | hyperledger/fabric-zookeeper | x86_64-0.4.6 | 92cbb952b6f8 | 2 months ago 1.39GB |
| x8 | hyperledger/fabric-kafka | latest | 554c591b86a8 | 2 months ago 1.4GB |
| 1t | hyperledger/fabric-kafka | x86_64-0.4.6 | 554c591b86a8 | 2 months ago 1.4GB |
| 6f | hyperledger/fabric-couchdb | latest | 7e73c828fc5b | 2 months ago 1.56GB |
| 6f | hyperledger/fabric-couchdb | x86_64-0.4.6 | 7e73c828fc5b | 2 months ago 1.56GB |
| | hyperledger/fabric-tools | x86_64-1.0.0-rc1 | 85d6d3ca0a30 | 10 months ago 1.32GB |

Fabric 기반의 블록체인 개발환경 구축을 위해 필요한 Docker 이미지 다운 및 확인

개발 환경 및 시스템 개발(6/15)

Hyperledger Fabric – network(2/2)

네트워크 구축

```
workspace/fabric-samples/first-network# ./byfn.sh -m up
```

네트워크 시작

```
workspace/fabric-samples/first-network# docker-compose -f docker-compose-cli.yaml up
```

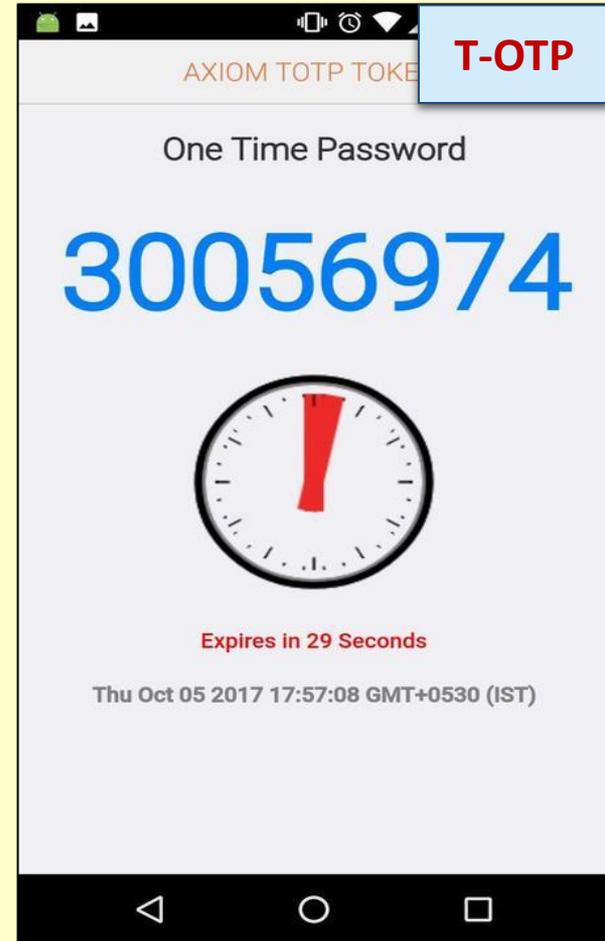
```
Creating peer1.org1.example.com ...
Creating peer0.org2.example.com ...
Creating orderer.example.com
2018-10-27 06:59:07.246 UTC [msp] getMspConfig -> INFO 001 Loading NodeOUs
General.LedgerType = "file"
General.ListenAddress = "0.0.0.0"
2018-10-27 06:59:07.329 UTC [nodeCmd] serve -> INFO 002 Starting peer:
Version: 1.1.0
General.ListenPort = 7050
Go version: go1.9.2
2018-10-27 06:59:04.492 UTC [msp] getMspConfig -> INFO 001 Loading NodeOUs
General.TLS.Enabled = true
2018-10-27 06:59:04.537 UTC [nodeCmd] serve -> INFO 002 Starting peer:
2018-10-27 06:59:05.088 UTC [msp] getMspConfig -> INFO 001 Loading NodeOUs
OS/Arch: linux/amd64
Version: 1.1.0
Go version: go1.9.2
2018-10-27 06:59:05.139 UTC [nodeCmd] serve -> INFO 002 Starting peer:
Experimental features: false
2018-10-27 06:59:04.105 UTC [msp] getMspConfig -> INFO 001 Loading NodeOUs
OS/Arch: linux/amd64
General.TLS.PrivateKey = "/var/hyperledger/orderer/tls/server.key"
Version: 1.1.0
Chaincode:
Base Image Version: 0.4.6
2018-10-27 06:59:04.151 UTC [nodeCmd] serve -> INFO 002 Starting peer:
General
Experimental
Base
Version
```

Docker-compose-cli.yaml 스크립트를 활용하여 네트워크를 시작, 생성된 채널 정보 확인 가능

개발 환경 및 시스템 개발(8/15)

T-OTP(Time-based One Time Password)

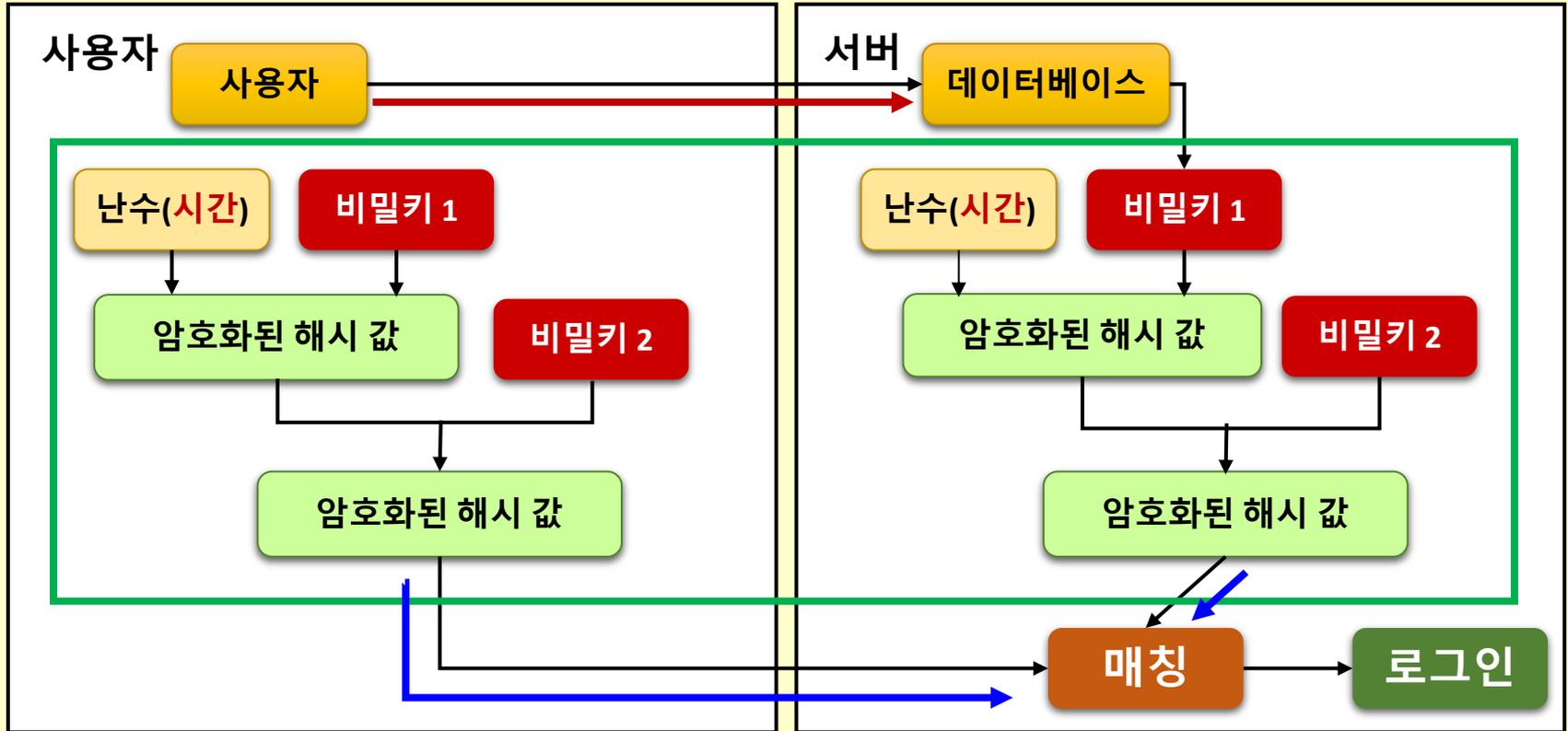
- ◆ 기존의 보안카드의 고정된 패스워드의 취약점을 보완하기 위하여 개발
- ◆ 고정 패스워드가 아닌 설정된 시간(초)마다 다른 패스워드를 사용하는 일회용 비밀번호
 - 비밀키와 현재 시간을 결합하여 암호 생성
 - 시간에 따라 지속적으로 암호 값이 변화 (기본 30초)
 - HMAC-SHA256 알고리즘을 이용하여 암호 생성
- ◆ OTP 종류는 시간 동기화방식(T-OTP), 이벤트 방식(H-OTP) 등으로 분류



개발 환경 및 시스템 개발(9/15)

T-OTP 알고리즘

비밀키 1=id, 비밀키 2=pw,
시간을 받아서 난수를 생성



① 사용자가 웹 서버 로그인 요청

② SHA256 알고리즘을 사용, 해시 값 생성

③ 사용자와 서버의 해시 값 동일 \Rightarrow 로그인 성공

개발 환경 및 시스템 개발(10/15)

시스템 개발 (1/6)

| | | | | |
|--|---|--|-------------------|---------|
| <pre>CREATE TABLE `number` `previous` `data_hash` PRIMARY CREATE TABLE `tx_id` `type` `timestamp` `chaincode` `channel` `number` PRIMARY CREATE TABLE `channel_name` `height` `current_block` `previous_block` PRIMARY KEY) ENGINE=InnoDB</pre> | 0 | <div style="border: 1px solid yellow; padding: 2px;">[Redacted]</div> | 저장된 블록 정보 | 블록 저장 |
| | | <div style="border: 2px solid red; padding: 2px;">BnF1NK9oPG402c8iK/2mLM0+pR3nvKX1g4n0BIjVuo8=</div> | | 트랜잭션 저장 |
| | 1 | MsJuqTy4Cf+yGATDPZisry4+ve2LLyFfENeyiWwa5xk= uuyf4pb0wg+oGF0YItXfaUYybhckL+QHC2WJbcSrVxY= | | |
| | 2 | h4kC0ZFMRWZVNRB8EKy/Bu3csrL/C0HnxAhT3Z9X2Z8= IFECTpRVu8zEUhvf | 이전해시와 데이터해시 확인 가능 | 채널 저장 |

블록, 트랜잭션, 채널의 정보 등이 블록체인 DB에 저장

개발 환경 및 시스템 개발(11/15)

시스템 개발 (2/6)

블록체인 막대그래프

블록 테이블

트랜잭션 테이블

```
Block = function(a) {
  this.elem = a;
  this.data = function() {
    var op = {title:{text:'블록높이'},legend:{data:['원래높이','성장높이']},dataZoom:[
{type:'slider',start:0,end:100},{type:'inside',}],toolbox:{feature:{restore:{},saveAsImage:{}}
},tooltip:{trigger:'axis'}
    };
    return op;
  };
  this.render = function() {
    $("#" + this.elem).empty();
    var chart = echarts.init($("#" + this.elem));
    chart.setOption(this.data());
  };
};

Details = function(a, b) {
  this.blocktable = a;
  this.transtable = b;
  this.render = function() {
    //block table
    blocktable = this.blocktable
    $.get("http://127.0.0.1:10010/blocks", function(data) {
      var datat = data.split(',')
      for (j = 0; j < datat.length; j++) {
        datat[j] = datat[j].split(',')
      }
      co
      $( "#" + blocktable ).DataTable({
        data: datat,
        columns: [
          { title: "거래 번호" },
          { title: "거래 유형" },
          { title: "거래 시간" },
          { title: "스마트 계약 이름" },
          { title: "채널 이름" },
          { title: "블록 주소" }
        ]
      });
    });
  };
};

//transaction table
transtable = this.transtable
$.get("http://127.0.0.1:10010/transactions", function(data) {
  var datat = data.split(',')
  for (j = 0; j < datat.length; j++) {
    datat[j] = datat[j].split(',')
  }
  $( "#" + transtable ).DataTable({
    data: datat,
    columns: [
      { title: "거래 번호" },
      { title: "거래 유형" },
      { title: "거래 시간" },
      { title: "스마트 계약 이름" },
      { title: "채널 이름" },
      { title: "블록 주소" }
    ]
  });
});
```

블록체인 네트워크에서 발생하는 트랜잭션을 모니터링 시스템에 막대그래프 형식으로 표현, 블록과 트랜잭션의 상세 정보 확인 가능

개발 환경 및 시스템 개발(12/15)

시스템 개발 (3/6)

사용자 계정 유효성 검증

```
<script>
function login(){
    var queryString = $("form[id=loginForm]").serialize();

    $.ajax({
        type:"POST",
        url:"<%=request.getContextPath()%>/jdbctest/login.do",
        data: queryString,
        success : function(data) {
            if(data == "로그인 성공") {
                alert("로그인 성공");
            } else {
                alert("로그인 실패");
            }
        },error : function() {
            alert("오류 발생");
        }
    });
}
</script>
```

```
<%@ page import = "java.sql.*" %>
<%@ page import = "java.util.Map" %>
<%@ page import = "java.util.HashMap" %>
<%@ include file="/jdbctest/jdbcDB.jsp" %>
<%
    request.setCharacterEncoding("UTF-8");
    String email = request.getParameter("email");
    String password = request.getParameter("password");
    Map<String, String> User = new HashMap<String, String>();
    out.clear();
    try {
        stmt = conn.createStatement();
        String sql = String.format("SELECT * from user WHERE email = '%s' AND password = '%s'", email, password);
        ResultSet rs = stmt.executeQuery(sql);
        int count = 0;
        while (rs.next()) {
            count++;
            User.put("name", rs.getString("name"));
            User.put("year", rs.getString("year"));
            User.put("month", rs.getString("month"));
            User.put("day", rs.getString("day"));
            User.put("birthDay", rs.getString("birthDay"));
            User.put("gender", rs.getString("gender"));
            User.put("idCardNumber", rs.getString("idCardNumber"));
            User.put("email", rs.getString("email"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
%>
```

사용자 계정 정보 확인 쿼리

사용자가 입력한 값과 DB에 저장된 값을 비교하여 동일한 경우 로그인

개발 환경 및 시스템 개발(13/15)

시스템 개발 (4/6)

hmac 암호화 알고리즘

```
private byte[] hmac_sha(String crypto, byte[] keyBytes, byte[] text) {
    try {
        Mac hmac;
        hmac = Mac.getInstance(crypto);
        SecretKeySpec macKey = new SecretKeySpec(keyBytes, "RAW");
        hmac.init(macKey);
        return hmac.doFinal(text);
    } catch (GeneralSecurityException gse) {
        throw new Un
    }
}
```

HEX 문자열 Byte 변환

```
private byte[] hexStr2Bytes(String hex) {
    byte[] bArray = new BigInteger("10" + hex, 16).toByteArray();

    byte[] ret = new byte[bArray.length - 1];
    for (int i = 0; i < ret.length; i++)
        ret[i] = bArray[i + 1];
    return ret;
}
```

HMAC (Hash-based Message Authentication Code)
암호 해시 알고리즘을 매개변수로 사용하여 T-OTP 값 생성

개발 환경 및 시스템 개발(14/15)

시스템 개발 (5/6)

```
private final int[] DIGITS_POWER
// 1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000, 1000000000 };

public String generateTOTP(String key, String time, String returnDigits) {
    int codeDigits = Integer.decode(returnDigits).intValue();
    String result = null;
    while (time.length() < 16)
        time = "0" + time;

    byte[] msg = hexStr2Bytes(time);
    byte[] k = hexStr2Bytes(key);
    byte[] hash = hmac_sha(crypto, k, msg);

    int offset = hash[hash.length - 1] & 0xf;
    int binary = ((hash[offset] & 0x7f) << 28 | ((hash[offset + 1] & 0xff) << 24) & 0xffff) >> 16;

    int otp = binary % DIGITS_POWER[codeDigits];

    result = Integer.toString(otp);
    while (result.length() < codeDigits) {
        result = "0" + result;
    }
    return result;
}

Session session = Session.getDefaultInstance(props);
protected PasswordAuthentication getPasswordAuthentication() {
    return new PasswordAuthentication(user, password.toCharArray());
}; // SMTP 사용자정보를 이용해 naver 메일과 연동

try {
    MimeMessage message = new MimeMessage(session);
    message.setFrom(new InternetAddress(user));
    message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

    message.setSubject("OTP Number Send"); // 메일 제목

    String otpnum=getId(); // otp 값 받아오기
    message.setText(otpnum); // otp 값 입력
} catch (Exception e) {
    e.printStackTrace();
}
```

T-OTP 값 생성

T-OTP 값 생성

T-OTP 값 메일로 전송

Key : Hex encode time : 시간을 반영하는 값
returnDigits : 반환 할 자릿 수 crypto : 사용할 암호화 함수

개발 환경 및 시스템 개발(15/15)

시스템 개발 (6/6)

시간값 가져오기

T-OTP 해시값

```
public boolean isOK() {
    boolean ok = false;
    String seed32 = "secret";
    long nowtime =
    long T0 = 0;
    long X = 60;
    long testTime[]
    String steps =
    DateFormat df =
    df.setTimeZone(
        try {
            for (int i = 0; i < testTime.length; i++) {
                long T = (testTime[i] - T0) / X;
                steps = Long.toHexString(T).toUpperCase();
                while (steps.length() < 16)
                    steps = "0" + steps;
                if (id.equals(generateTOTP(seed32, steps, "8", "HmacSHA256")))
                    ok = true;
            }
        } catch (final Exception e) {
            System.out.println("Error : " + e);
        }
        String dab = generateTOTP(seed32, steps, "8", "HmacSHA256");
        if (dab.equals(id))
            ok = true;
        return ok;
    }
```

암호화 된 OTP 값이 일치할 경우 true, 일치하지 않을 경우 false

개발 시스템 운영(1/5)

시스템 운영 (1/5)

메인 페이지

커피킹 매장소개 작품소개 메뉴

로그인 회원가입

로그인 버튼 클릭

로그인 화면

커피킹 매장소개 작품소개 메뉴

로그인 회원가입

로그인 ▶ OTP비밀번호 OTP발송 OTP번호입력 OTP검증

로그인

kangsinsun2

....

로그인

아이디, 패스워드 입력 후 로그인 버튼 클릭

개발 시스템 운영(2/5)

시스템 운영 (2/5)

로그인 **OTP비밀번호** > OTP발송 OTP번호입력 OTP검증 **2차 비밀번호 입력**

로그인 OTP비밀번호 **OTP발송** > OTP번호입력 OTP검증 **OTP 번호 발송**

로그인 OTP비밀번호 OTP발송 **OTP번호입력** > OTP검증 **OTP 번호 확인**

입력한 네이버 메일에서 받은 OTP번호를 입력해주세요.

N
sp 위 이미지 클릭 시 네이버 페이지 열림

79681957

확인

확인

발급된 OTP 번호 입력

개발 시스템 운영(3/5)

시스템 운영 (3/5)

로그인 OTP비밀번호 OTP발송 OTP번호확인 **OTP인증성공**

환영합니다.
강신선님

OTP인증에 성공하였습니다.
(확인을 눌러주세요.)

kangsinsun2

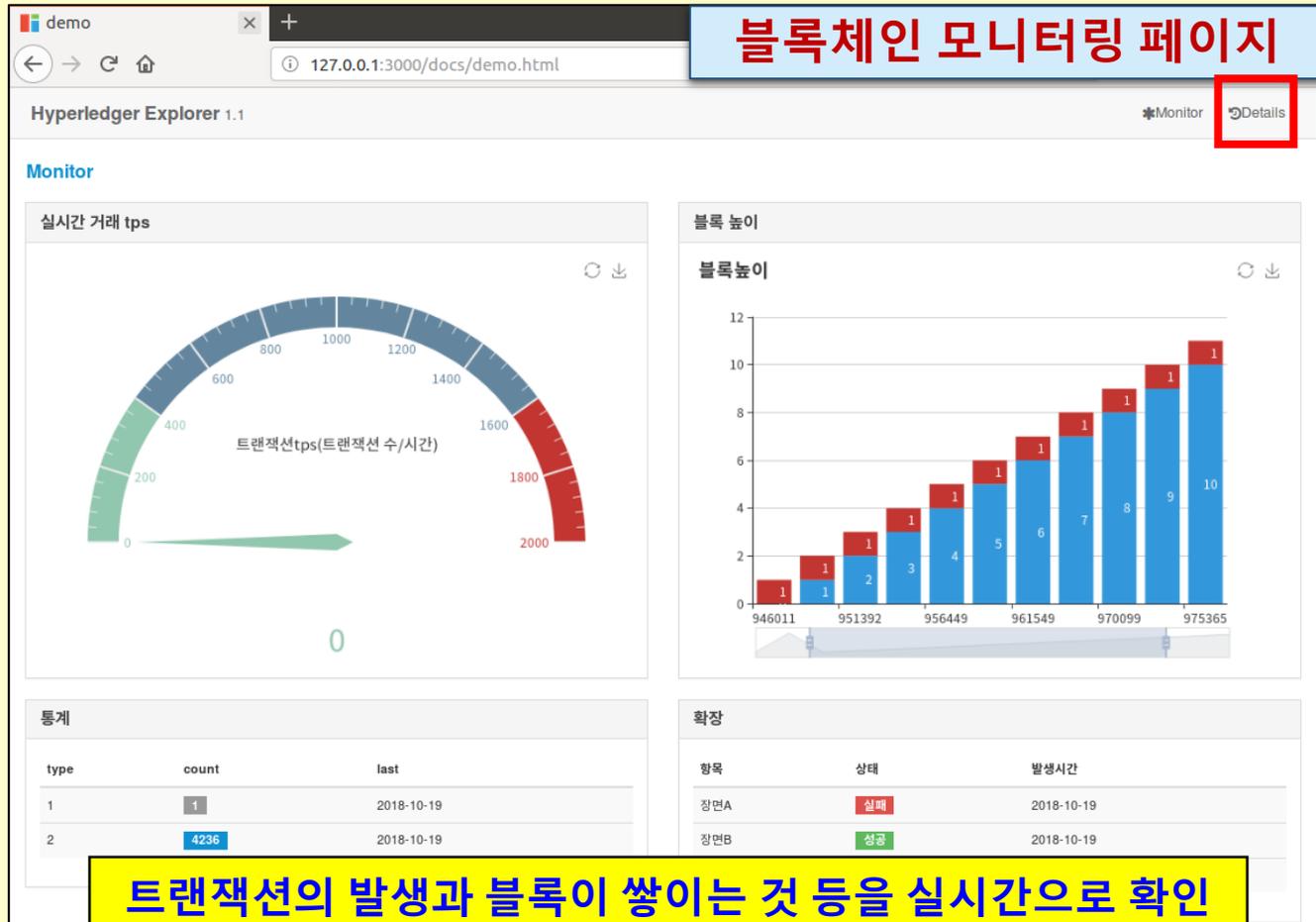
확인

검증 성공

정상 로그인 후 확인 버튼 클릭시 블록체인 모니터링 시스템으로 이동

개발 시스템 운영(4/5)

시스템 운영 (4/5)



개발 시스템 운영(5/5)

시스템 운영 (5/5)

트랜잭션에 의한 블록 값 변화

트랜잭션 상세 정보

Search:

| 거래 번호 | 거래 유형 | 거래 시간 | 스마트 계약 이름 | 채널 이름 | 블록 소속 |
|--|-------|--------|-----------|-----------|-------|
| 5200c358a9ce933366f505aeb474a296b032a33de3fe1cdba24825c23ca933a2 | 3 | 951392 | joongbu | mychannel | 2 |
| 54edb3f49d3d9b1f5fed54ed32260b5ea89466adc0d385c7934a4cbc404b0654 | 3 | 991617 | joongbu | mychannel | 15 |
| 60c4360c45ba460384ad7c6a3a8f43c5ab96119f56ad34ae4e047de760c8edd9 | 3 | 994686 | joongbu | mychannel | 16 |
| 67cbc3d49187a1b8eab1160bc85f34c947ca313673a0503f293df61752dc2e5d | 3 | 953846 | joongbu | mychannel | 3 |
| 77f02c6ab87435fdb18109cd1315ebf53b641cc767b12a345020236f6cff5b1e | 3 | 988439 | joongbu | mychannel | 14 |
| 81af4011678051f9c17fd624b8d427d60286c15ec02499269e86213e04283377 | 3 | 966731 | joongbu | mychannel | 7 |
| 8c990bcd4af6600ab230eadbb47fad3253be09e3187ab0cc2957da42aa2510c9 | 3 | 956449 | joongbu | mychannel | 4 |
| 92ce401c219574c937da4b7e345b319ae9a92bea30a9c48f3379412cb9355997 | 3 | 980571 | joongbu | mychannel | 12 |
| a6f20a4d67d2bca8a94e8ffeb26a6f11a82b81fff5558f7559953d4a60a89dc2 | 3 | 000457 | joongbu | mychannel | 18 |
| d4fae0412df875e70c75ae2f24ab168839d79c50a5532b33fa3416d3a19b05a0 | 3 | 997709 | joongbu | mychannel | 17 |

Showing 11 to 20 of 26 entries

Previous 1 2 3 Next

Showing 1 to 10 of 26 entries

블록에 대한 상세 정보 확인 및 특정 거래 검색 가능

결론 및 기대효과

◆ OTP 인증 방식의 블록체인 모니터링 시스템 개발 성과

- ✓ OTP 인증을 통해 인가된 사용자만이 블록체인 모니터링 시스템에 접근
- ✓ 모니터링 시스템을 통해 사용자들의 트랜잭션과 그에 따른 블록의 변화를 실시간으로 확인 함으로써 시스템의 성능을 검증

◆ 기대효과 (소감)

- ✓ 4차 산업혁명의 핵심기술 중 하나인 블록체인 기술을 연구하면서 많은 시행착오를 겪고 성장하는 계기
- ✓ 특히 조원들이 상호 의지하며 포기하지 않고 노력한 결과가 조그만 성과로 나타난 것이 보람찬 결실

Q&A

감사합니다
