

# 웹 / 앱 게시판 프로젝트

기간

2017.08 ~ 2017.10

이유란



<https://github.com/leeyoulan/project>

# 목차

## 1. 웹 게시판

- 1) 화면구성 ( 회원가입 )
- 2) 화면구성 ( 게시글 상세 )
- 3) 경로 정의

## 2. 앱 게시판

- 1) 화면구성 ( 로그인 )
- 2) 화면구성 ( 게시글 목록 )
- 3) API 정의 , Status Code 정의
- 4) BODY ( IN\_JSON / OUT\_JSON ) 정의

## 3. DB 구성 ( 웹/앱 동일)

# 웹 게시판 (개인 프로젝트)

- 개발툴 : Node.js (Express 프레임워크) ( v. 6. 11. 2 ) / NPM ( v. 3. 10. 10 )  
/ MySQL ( v. 5. 6. 36 )  
/ HTML(ejs 템플릿 엔진), CSS / Github, SourceTree

- 프로젝트 설명

: 회원가입과 로그인 후 게시글과 댓글의 등록, 수정, 삭제가 가능합니다.

passport 모듈을 사용하여 회원가입, 로그인, 로그아웃 기능을 구현하였고,

bcrypt-nodejs 모듈을 사용하여 비밀번호 암호화를 하였으며,

Github, SourceTree를 통해 프로젝트 관리를 하였습니다.

지속적인 업데이트를 통해 대댓글 기능과 파일업로드 기능을 추가할 예정입니다.

## 화면구성 ( 회원가입 )

### 웹 화면

#### 회원가입

회원가입 화면

A normal membership registration form with a light green background. It contains six white input fields stacked vertically, labeled 'id', 'password', 'repassword', 'name', and 'email'. At the bottom is a green button labeled '회원가입'.

#### 회원가입

아이디 또는 비밀번호를 입력하지 않은 경우

The same membership registration form as above, but with a red error message at the top: '아이디와 비밀번호를 입력해주세요.' (Please enter ID and password).

#### 회원가입

비밀번호와 비밀번호 재입력이 일치하지 않은 경우

The same membership registration form as above, but with a red error message at the top: '비밀번호가 일치하지 않습니다.' (Passwords do not match).

#### 회원가입

이름을 입력하지 않은 경우

The same membership registration form as above, but with a red error message at the top: '이름을 입력해주세요.' (Please enter your name).

#### 회원가입

이메일을 입력하지 않은 경우

The same membership registration form as above, but with a red error message at the top: '이메일을 입력해주세요.' (Please enter your email).

## 화면구성 ( 회원가입 )

```
passport.use('local-join',new LocalStrategy({
  usernameField: 'id',
  passwordField: 'password',
  passReqToCallback : true
}, function(req,id,password,done){
  var repassword = req.body.repassword;
  var name = req.body.name;
  var email = req.body.email;

  var query = connection.query('select * from member where id=?',[id],function(err,rows){
    if(err) return done(err);

    if(repassword==''){
      return done(null,false, {message : '비밀번호를 재입력해주세요.'})
    }

    if(name==''){
      return done(null,false, {message : '이름을 입력해주세요.'})
    }

    if(email==''){
      return done(null,false, {message : '이메일을 입력해주세요.'})
    }

    if(password!==repassword){
      return done(null,false, {message : '비밀번호가 일치하지 않습니다.'})
    }

    if(rows.length){
      console.log('existed member');
      return done(null,false, {message : '사용중인 아이디입니다.'})//message:flash 사용
    } else {
      bcrypt.hash(password, null, null, function(err, hash) {
        var sql = {'id':id,'password':hash, 'name':name,'email':email};
        var query = connection.query('insert into member set ?',sql, function(err, rows) {
          if(err) throw err;
          return done(null, {'id': id});
        });
      });
    }
  });
});
```

회원가입 성공시 로그인 화면으로 이동,  
실패시 회원가입 화면으로 이동

```
router.post('/',passport.authenticate('local-join',{
  successRedirect : '/login',
  failureRedirect : '/join',
  failureFlash : true
})))
```

회원가입 실패시 에러 메시지와 함께  
회원가입 화면으로 이동

```
router.get('/',function(req,res){
  var msg;
  var errMsg = req.flash('error');
  if(errMsg) msg = errMsg;
  if(errMsg=='Missing credentials'){
    msg='아이디와 비밀번호를 입력해주세요';
  }
  res.render('join.ejs',{ 'message': msg});
});
```

비밀번호 재입력, 이름, 이메일, 비밀번호 일치를 확인하여 이상이 있는 경우  
회원가입 화면으로 돌아가 에러 메시지를 같이 출력함

쿼리문에서 아이디 확인 후  
동일한 아이디가 존재하는 경우 회원가입 화면으로 돌아가 에러 메시지 출력,  
동일한 아이디가 존재하지 않는 경우 비밀번호 암호화하여 입력정보를 저장.

# 화면구성 ( 게시글 상세 )

경로 : localhost:5005/board\_view/4

웹 화면

글상세

글수정

글삭제

리스트로 돌아가기

제목	테스트 4 테스트 4 테스트 4 테스트 4		
작성자	dldbfks		
작성일	2017/9/29		
조회수	78		
내용	테스트 4 테스트 4 테스트 4 테스트 4		
댓글[3]			
이유란 2017/9/30 테스트 1	<div>수정</div> <div>삭제</div>		
이유란 2017/9/30 테스트 2	<div>수정</div> <div>삭제</div>		
이유란 2017/9/30 이유란	<div>수정</div> <div>삭제</div>		
<div></div> <div>이름</div>			

글목록

글쓰기

로그아웃

번호	제목	작성자	작성일	조회수	댓글
1	테스트 1	유란	2017/9/26	30	3
2	테스트 2 테스트 2	유란	2017/9/27	26	1
3	테스트 3	dldbfks	2017/9/28	36	2
4	테스트 4 테스트 4 테스트 4 테스트 4	dldbfks	2017/9/29	78	3
5	테스트 5 테스트 5 테스트 5	이유란	2017/9/30	22	1
6	게시판 테스트	유란	2017/10/7	21	0

4번 게시글 클릭했을 경우



## 화면구성 ( 게시물 상세 )

```
router.get('/:post_num', function(req,res){
  var post_num= req.params.post_num;

  var sql = 'update board set post_hit = post_hit+1 where post_num = ?';
  connection.query(sql, post_num, function(err,result){
    if (err) {
      res.status(500).send('Something broke!');
      console.log(err.code);
    }
    else {
      var sql = 'select post_num, post_id, post_title, post_content, date_format(post_time,"%Y/%c/%e") post_time, post_hit\
        , comm_num, comm_id, comm_content, date_format(comm_time,"%Y/%c/%e") comm_time, (select count(comm_num) from comment where post_num=?) comm_count \
        from board join comment using(post_num) where post_num=?';
      connection.query(sql, [post_num, post_num], function(err,rows){
        if (err) {
          res.status(500).send('Something broke!');
          console.log(err.code);
        }else if(rows[0]){
          res.render('boards/board_view.ejs',{rows:rows,'message':' '});
        }else if(rows[0]==undefined){
          var sql = 'select post_num, post_id, post_title, post_content, date_format(post_time,"%Y/%c/%e") post_time, post_hit\
            from board where post_num=?';
          connection.query(sql, post_num, function(err,rows){
            if (err) {
              res.status(500).send('Something broke!');
              console.log(err.code);
            }else{
              res.render('boards/board_view.ejs',{rows:rows,'message':' '});
            }
          });
        }
      });
    }
  });
});
```

게시글 클릭시 가장 먼저 조회수 1 증가시킴

게시글에 댓글이 존재하는지 쿼리문으로 확인 후 rows[0]이 true 일 경우 게시물 정보와 댓글 정보를 받아서 게시물 상세 화면으로 이동

rows[0]이 undefined 일 경우 쿼리문으로 게시물 정보만 받아서 게시물 상세 화면으로 이동

# 경로정의

/join	get	/join	회원가입 화면
	post	/join	회원가입
/login	get	/login	로그인 화면
	post	/login	로그인
/logout	get	/logout	로그아웃 화면, 로그아웃
/boards	get	/board_list	게시글 목록
	get	/board_view/:post_num	게시글 상세
	post	/board_view	게시글 상세(댓글쓰기)
	get	/board_write	글 쓰기 화면
	post	/board_write	글 쓰기
	get	/board_update/:post_num	글 수정 화면
	post	/board_update	글 수정
	get	/board_delete/:post_num	글 삭제 화면
	post	/board_delete	글 삭제
/comments	get	/comm_update/:post_num/:comm_num	댓글 수정 화면
	post	/comm_update	댓글 수정
	get	/comm_delete/:post_num/:comm_num	댓글 삭제 화면
	post	/comm_delete	댓글 삭제
/find	get	/find_id	아이디 찾기 화면
	post	/find_id	아이디 찾기
	get	/find_pw	비밀번호 찾기 화면
	post	/find_pw	비밀번호 찾기
	get	/re_pw	비밀번호 재설정 화면
	post	/re_pw	비밀번호 재설정



# 앱 게시판 (팀 프로젝트)

- 개발툴 : Node.js(Express 프레임워크) / MySQL/ Github, SourceTree
- 역할 : 백엔드, DB설계
- 프로젝트 설명

: 회원가입과 로그인 후 게시글과 댓글의 등록, 수정, 삭제가 가능합니다.

express-mysql-session 모듈을 사용하여 로그인, 로그아웃 기능을 구현하였고,  
웹과 동일하게 bcrypt-nodejs 모듈을 사용하여 비밀번호 암호화를 하였으며,  
Github, SourceTree를 통해 프로젝트 관리를 하였습니다.  
또한, Postman을 통하여 API 테스트를 하였습니다.

# 화면구성 ( 로그인 )

앱 화면

API 테스트 화면

[ id : leeyoulan , password : leeyoulan ] 으로 회원가입 되어있는 경우



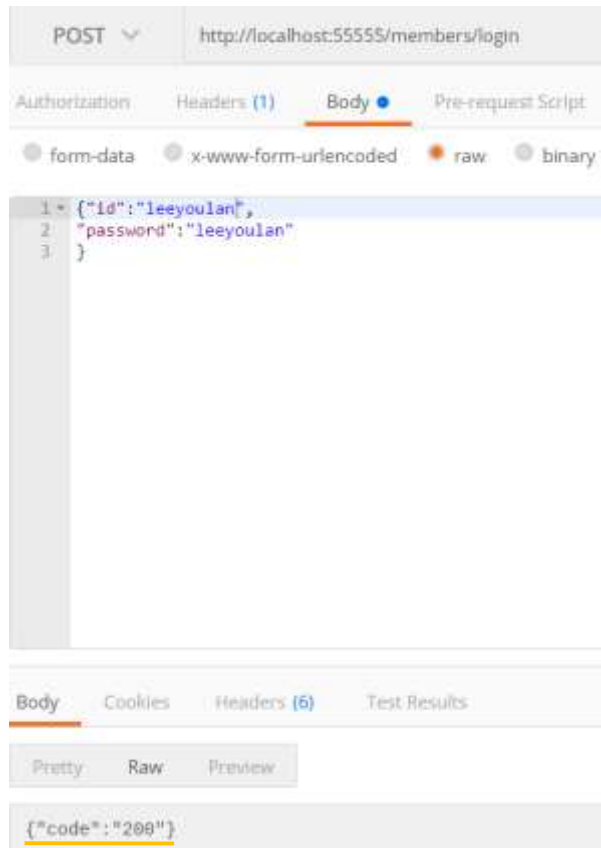
아이디  
비밀번호

- ☐ 아이디 저장  
☒ 자동 로그인

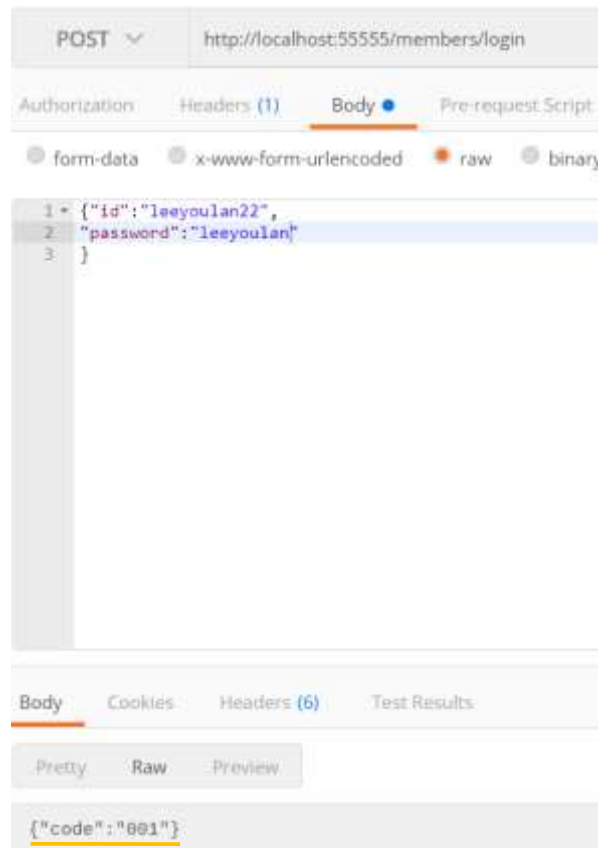
로그인

[아이디 찾기](#) [비밀번호 찾기](#) [회원가입](#)

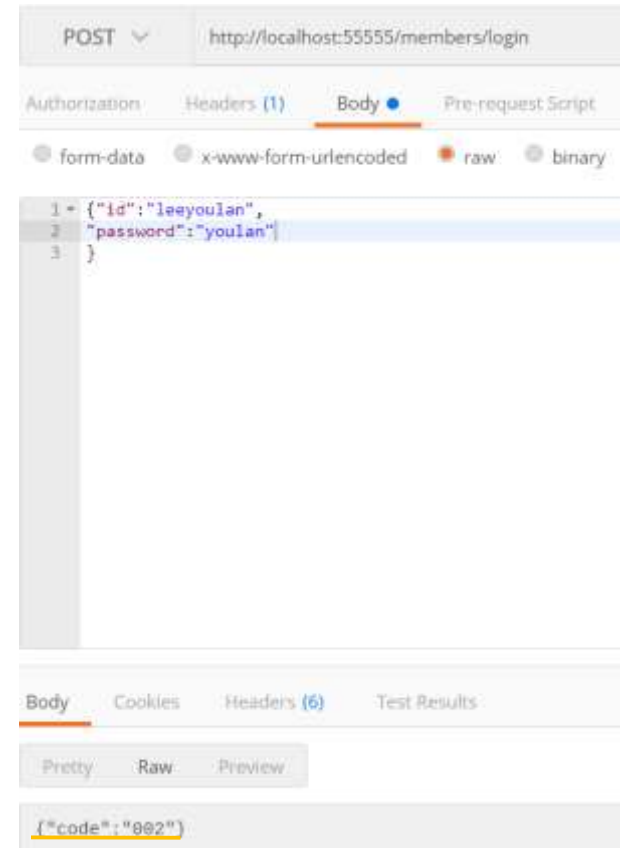
Id와 password 정보가 같은 경우  
-> {code : 200} response 함  
= 요청성공



Id 정보가 존재하지 않는 경우  
-> {code : 001} response 함  
= 존재하지 않는 사용자



Id와 password 정보가 다른 경우  
-> {code : 002} response 함  
= 비밀번호 불일치



## 화면구성 ( 로그인 )

```
router.post('/login',function(req,res){
  var id = req.body.id;
  var password = req.body.password;
  pool.getConnection(function(err,connection){
    if(err) {
      res.json({'code':'500'});
      console.log(err.code);
    }

    var sql = 'select * from member where id=?';

    connection.query(sql,[id],function(err,rows){
      if (err) {
        res.json({'code':'500'});
        console.log(err.code);
        connection.release();
      }

      if(rows.length){
        bcrypt.compare(password, rows[0].password, function(err, result) {
          if (result) {
            req.session.id = id;
            res.json({'code':'200'});
            console.log('200');
            connection.release();
          } else {
            res.json({'code':'002'});
            console.log('002');
            connection.release();
          }
        })
      } else {
        res.json({'code':'001'});
        console.log('001');
        connection.release();
      }
    })
  })
}
```

회원가입시 암호화 되어 저장된 비밀번호와  
사용자가 입력한 비밀번호 비교

result 값이 true이면 id를 session에 저장 후  
{code : 200} response 함  
false이면 {code : 002} response 함

## 회원가입 ( 비밀번호 암호화 부분 )


```
bcrypt.hash(password, null, null, function(err, hash) {
  var post = {'id':id,'password':hash,'name':name,'email':email};
  var sql = 'insert into member set ?';
```

## 화면구성 ( 게시글 목록 )

## 앱 화면

[illegible]

## API 테스트 화면



The screenshot shows a web browser with a REST client interface. The address bar displays the URL `http://localhost:55555/posts`. The 'Body' tab is selected, showing a JSON response. The response is a JSON array containing four objects, each representing a post. The objects have the following fields: `post_num`, `post_id`, `post_title`, `post_time`, `post_hit`, and `comm_count`.

A yellow callout box on the right side of the image contains the following text:

```
get http://localhost:55555/
접속했을 때 response 5
json 형태의 데이터
```

**get** http://localhost:55555/posts로  
접속했을 때 response 되는  
json 형태의 데이터

## 화면구성 ( 게시글 목록 )

```
router.get('/', function(req, res){

pool.getConnection(function(err, connection){
  if (err) {
    res.json({'code': '500'});
    console.log(err.code);
  }

  var sql = 'select post_num, post_id, post_title, post_content, date_format(post_time,"%Y/%c/%e") post_time, post_hit,\
    (select count(comm_num) from comment c where b.post_num=c.post_num) comm_count from board b;';
  connection.query(sql, function(err, result){
    if (err) {
      res.json({'code': '500'});
      console.log(err.code);
      connection.release();
    } else {

      var result_sql = []; ← 쿼리문 결과를 넣어줄 result_sql 배열 생성

      for (var i = 0; i < result.length; i++) {
        result_sql.push({'post_num' : result[i].post_num,
          'post_id' : result[i].post_id,
          'post_title' : result[i].post_title,
          'post_time' : result[i].post_time,
          'post_hit' : result[i].post_hit,
          'comm_count' : result[i].comm_count
        });
      }
      res.json(result_sql); ← 결과값이 담겨있는 result_sql 배열을
      console.log('200');    json 형태로 response 함
      connection.release();
    }
  });
}
```

## API 정의

### //member

member 리소스	/members		
member URL	post	/members	회원가입
		/members/login	로그인
	get	/members/logout	로그아웃
		/members/id	아이디 찾기
		/members/pw	비밀번호 찾기
	put	/members/pw	비밀번호 재설정

### //post

post 리소스	/posts		
post URL	get	/posts	게시글 목록
		/posts/details	게시글 상세
		/posts/comments	댓글 상세
	post	/posts	글 쓰기
		/posts/comments	댓글 쓰기
	put	/posts	글 수정
		/posts/comments	댓글 수정
	delete	/posts	글 삭제
		/posts/comments	댓글 삭제

## Status Code 정의

category	코드	설명
전체	200	요청 성공
	500	서버 오류
로그인	001	존재하지 않는 사용자
	002	비밀번호 불일치
회원가입	003	이미 가입된 아이디
아이디 찾기	004	일치하는 아이디가 없음
비밀번호 찾기	005	일치하는 비밀번호가 없음

## BODY ( IN\_JSON / OUT\_JSON ) 정의

Key	상황	Req/Res	Field명	Field Desc
body	로그인 <b>post</b> /members/login	Req	id	아이디
			password	비밀번호
		Res	Status_code	상태코드 참조
	로그아웃 <b>get</b> /members/logout	Res	Status_code	상태코드 참조
	회원가입 <b>post</b> /members	Req	id	아이디
			password	비밀번호
			name	이름
			email	이메일
		Res	Status_code	상태코드 참조
	아이디 찾기 <b>get</b> /members/id	Req	name	이름
			email	이메일
		Res	id	아이디
		Res	Status_code	상태코드 참조
	비밀번호 찾기 <b>get</b> /members/pw	Req	id	아이디
			name	이름
			email	이메일
		Res	Status_code	상태코드 참조
	비밀번호 재설정 <b>put</b> /members/pw	Req	id	아이디
		Res	password	비밀번호
			Status_code	상태코드 참조
	게시글 목록 <b>get</b> /posts	Res	post_num	게시글 번호
			post_id	아이디
			post_title	게시글 제목
			post_time	작성일
			post_hit	조회수
			comm_count	댓글 수 (=count(comm_num))

body	게시글 상세 <b>get</b> /posts/details	Req	post_num	게시글 번호
		Res	post_id	(게시글)아이디
			post_title	게시글 제목
			post_content	게시글 내용
			post_time	작성일
		Res	post_hit	조회수
	댓글 상세 <b>get</b> /posts/comments	Req	post_num	게시글 번호
		Res	comm_num	댓글 번호
			comm_count	댓글 수
			comm_id	(댓글)아이디
			comm_content	댓글 내용
			comm_time	댓글 작성일
	글 쓰기 <b>post</b> /posts	Req	post_id	아이디
			post_title	게시글 제목
			post_content	게시글 내용
			post_time	작성일
		Res	Status_code	상태코드 참조
	(답)댓글 쓰기 <b>post</b> /posts/comments	Req	post_num	게시글 번호
			comm_id	(댓글)아이디
			comm_content	댓글 내용
			comm_time	댓글 작성일
		Res	Status_code	상태코드 참조
	글 수정 <b>put</b> /posts	Req	post_num	게시글 번호
			post_title	게시글 제목
			post_content	게시글 내용
		Res	Status_code	상태코드 참조
	(답)댓글 수정 <b>put</b> /posts/comments	Req	post_num	게시글 번호
			comm_num	댓글 번호
			comm_content	댓글 내용
		Res	Status_code	상태코드 참조
	글 삭제 <b>post</b> /posts	Req	post_num	게시글 번호
		Res	Status_code	상태코드 참조
	(답)댓글 삭제 <b>delete</b> /posts/comments	Req	post_num	게시글 번호
			comm_num	댓글 번호
		Res	Status_code	상태코드 참조



# DB 구성 (웹 / 앱 동일)

## member 테이블

	Field	Type	Comment
🔑	id	varchar(20) NOT NULL	아이디
	password	varchar(70) NOT NULL	비밀번호
	name	varchar(10) NOT NULL	이름
	email	varchar(30) NOT NULL	이메일

## board 테이블

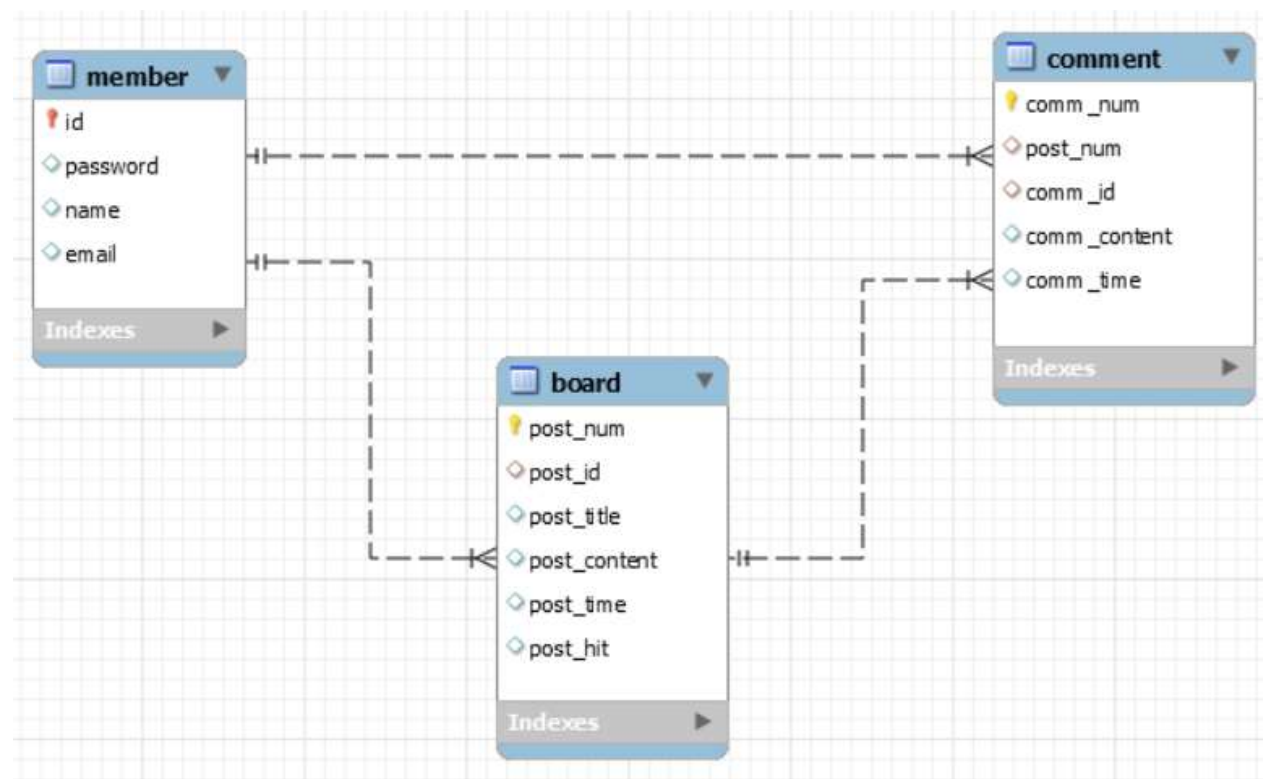
	Field	Type	Comment
🔑	post_num	int(5) NOT NULL	게시글 번호
	post_id	varchar(20) NOT NULL	아이디
	post_title	varchar(255) NOT NULL	게시글 제목
	post_content	mediumtext NOT NULL	게시글 내용
	post_time	datetime NOT NULL	작성일
	post_hit	int(5) NOT NULL	조회수

## comment 테이블

	Field	Type	Comment
🔑	comm_num	int(5) NOT NULL	댓글 번호
	post_num	int(5) NOT NULL	게시글 번호
	comm_id	varchar(20) NOT NULL	아이디
	comm_content	mediumtext NOT NULL	댓글 내용
	comm_time	datetime NOT NULL	댓글 작성일

## ERD

MySQL Workbench 6.3 CE 버전으로 작성





**감사합니다.**