**웹 프로그래밍 팀 프로젝트**

**최종 보고서**

컴퓨터 공학과

2017112129 이정욱

2017112090 김동욱

2017112125 조현준

송양의 교수님



목차

[**프로젝트 개요** 3](#_Toc58682238)

[**주제** 3](#_Toc58682239)

[**기획 의도** 3](#_Toc58682240)

[**주요기능** 3](#_Toc58682241)

[**외부접속주소 및 DB** 3](#_Toc58682242)

[**파일 구조 및 트리** 4](#_Toc58682243)

[**사이트 맵** 6](#_Toc58682244)

[**로그인 기능 플로우 차트** 7](#_Toc58682245)

[**스토리 보드** 7](#_Toc58682246)

[**로그인 스크린** 13](#_Toc58682247)

[**로그인 화면** 13](#_Toc58682248)

[**프론트엔드** 14](#_Toc58682249)

[**백엔드** 14](#_Toc58682250)

[**회원가입 화면** 15](#_Toc58682251)

[**프론트엔드** 15](#_Toc58682252)

[**백엔드** 16](#_Toc58682253)

[**아이디 / 비밀번호 찾기 화면** 17](#_Toc58682254)

[**프론트엔드** 17](#_Toc58682255)

[**백엔드** 18](#_Toc58682256)

[**캘린더 스크린** 19](#_Toc58682257)

[**프론트엔드** 19](#_Toc58682258)

[**백엔드** 20](#_Toc58682259)

[**일정 검색** 21](#_Toc58682260)

[**팔로우 알림** 21](#_Toc58682261)

[**프론트엔드** 21](#_Toc58682262)

[**백엔드** 21](#_Toc58682263)

[**다가오는 일정** 22](#_Toc58682264)

[**배포** 22](#_Toc58682265)

# **프로젝트 개요**

## **주제**

사용자가 일정을 관리하고 확인할 수 있는 웹 기반 캘린더 서비스 플랫폼

## **기획 의도**

웹에서 쉽고 빠르게 사용할 수 있는 캘린더 서비스가 모바일에 비해 현저히 적어 웹에서도 편하게 일정을 관리하고 공유할 수 있는 새로운 웹 캘린더의 필요성을 느꼈다. 따라서 불필요한 기능들을 없애고 반드시 필요한 기능인 캘린더 기능과 유저간의 팔로우, 일정 확인 기능 위주로 구현하였다.

## **주요기능**

사용자 간 친구추가, 팔로우 기능, 팔로우 시 알림 기능

친구의 캘린더 열람 기능

이름, 상세 내용등 캘린더 및 일정에 대한 검색

일간/주간/월간/연간 단위로 캘린더 확인

드래그 앤 드랍으로 날짜 이동, 날짜 길이 변경

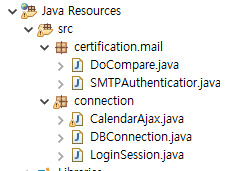
다가오는 일정에 대한 알림

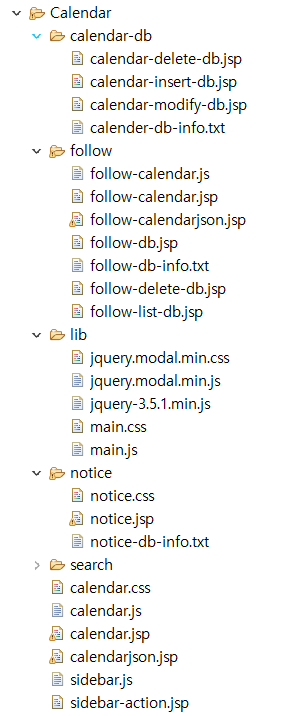
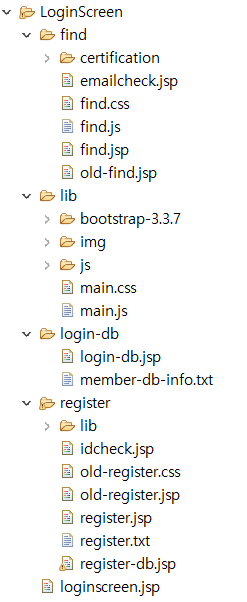
## **외부접속주소 및 DB**

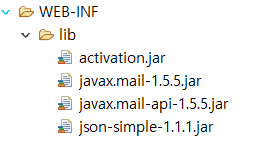
주소: <http://52.78.0.230:8080/WebProgrammingCalendar/LoginScreen/loginscreen.jsp>

사용한 DB의 덤프파일 : 첨부 파일 중 mycal\_dump.sql

## **파일 구조 및 트리**

프로젝트 전체 파일은 다음과 같은 구조이다.



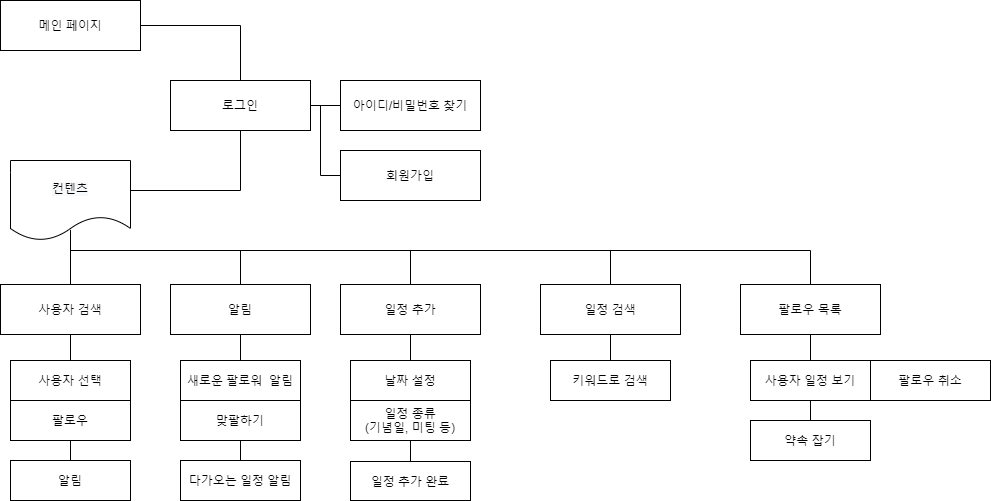


LoginScreen 폴더에는 로그인과 회원가입, ID/PW 찾기, 메일 인증 등을 담당하는 파일들이 위치해있다. 각 하위 폴더를 살펴보면 find폴더는 ID/PW 찾기 페이지와 메일인증 기능을 담당하는 폴더이다. lib폴더는 로그인 스크린의 라이브러리, 이미지, js, css 파일이 위치해 있다. login-db는 로그인 기능을 담당하는 폴더이다. registe는 회원가입 기능을 담당하는 폴더이다.

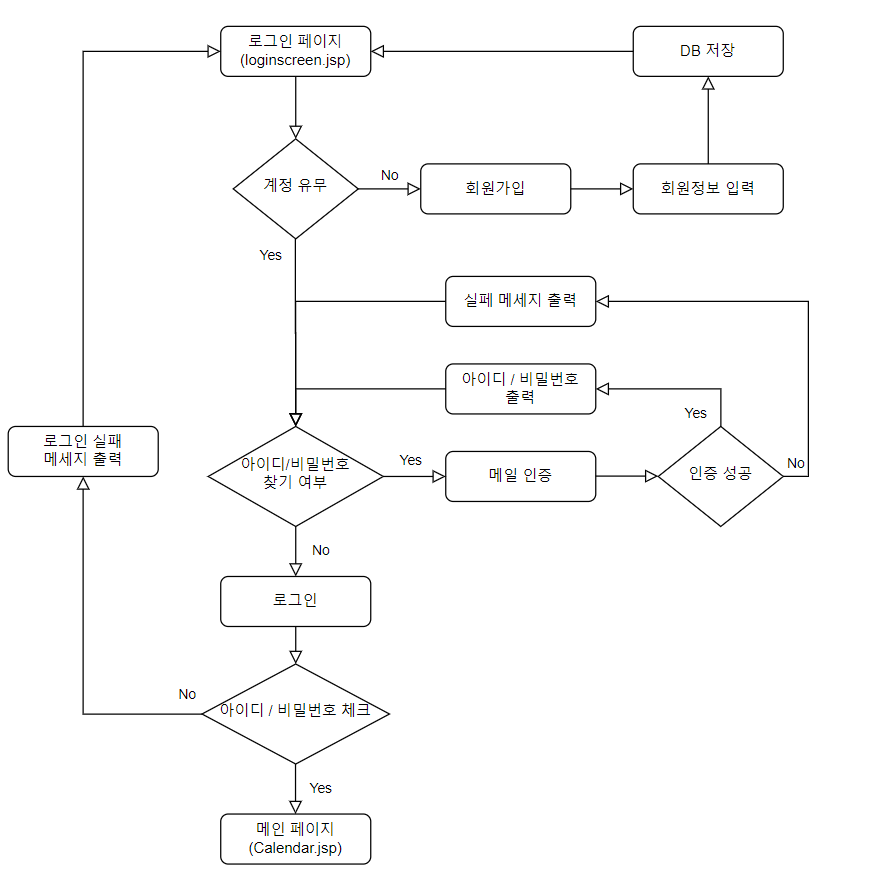
Calendar 폴더에는 실제 캘린더의 기능이 구현된 폴더이다. calendar-db는 일정 DB의 추가, 삭제, 수정 기능을 담당한다. follow 폴더는 친구 팔로우, 팔로우 삭제, 팔로우 한 사람의 캘린더 열람 기능을 담당한다. lib 폴더는 라이브러리, js , css 파일이 위치하고있다. notice는 누가 나를 팔로우 했는지 알려주는 기능을 담당한다. search 폴더는 일정 검색 기능을 담당한다.

WEB-INF\lib 폴더에는 메일 인증과 json을 다룰 때 필요한 jar 파일들이 위치해있으며 Java Resources 폴더 아래에 certification.mail 패키지에는 이메일 인증 시 인증번호 비교를 위한 DoCompare과 이메일을 보내기 위한 파일이 있다. connection 패키지에는 로그인 및 캘린더 기능을 위한 파일과 DB접근을 위한 파일이 있다.

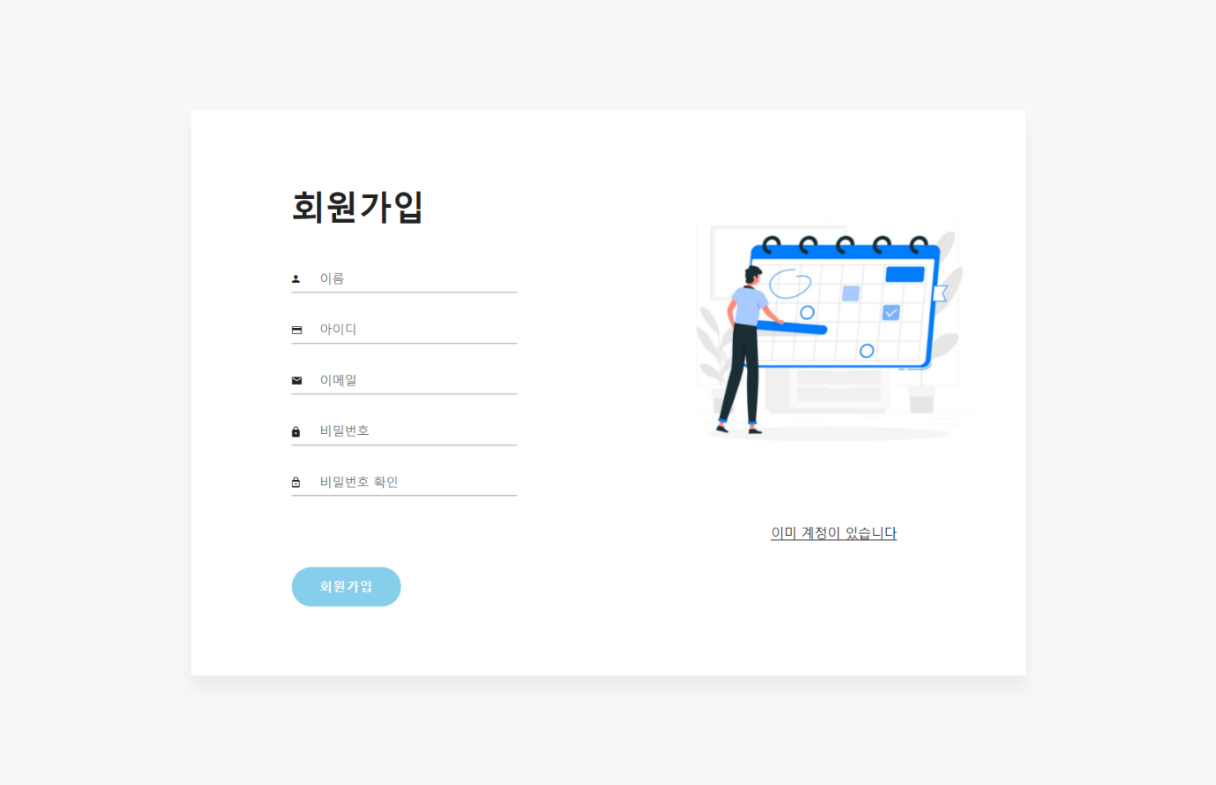
## **사이트 맵**



## **로그인 기능 플로우 차트**



## **스토리 보드**

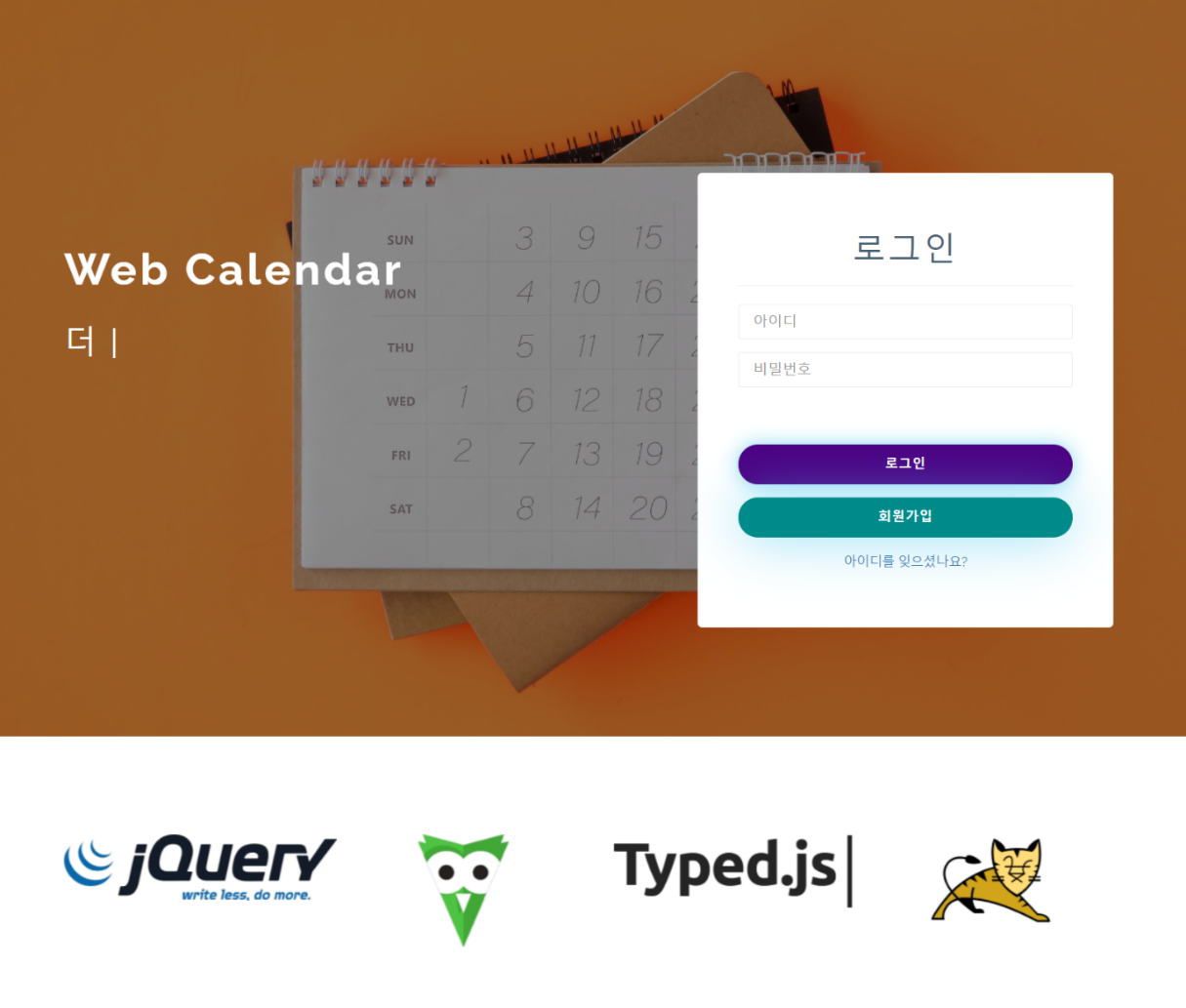


2

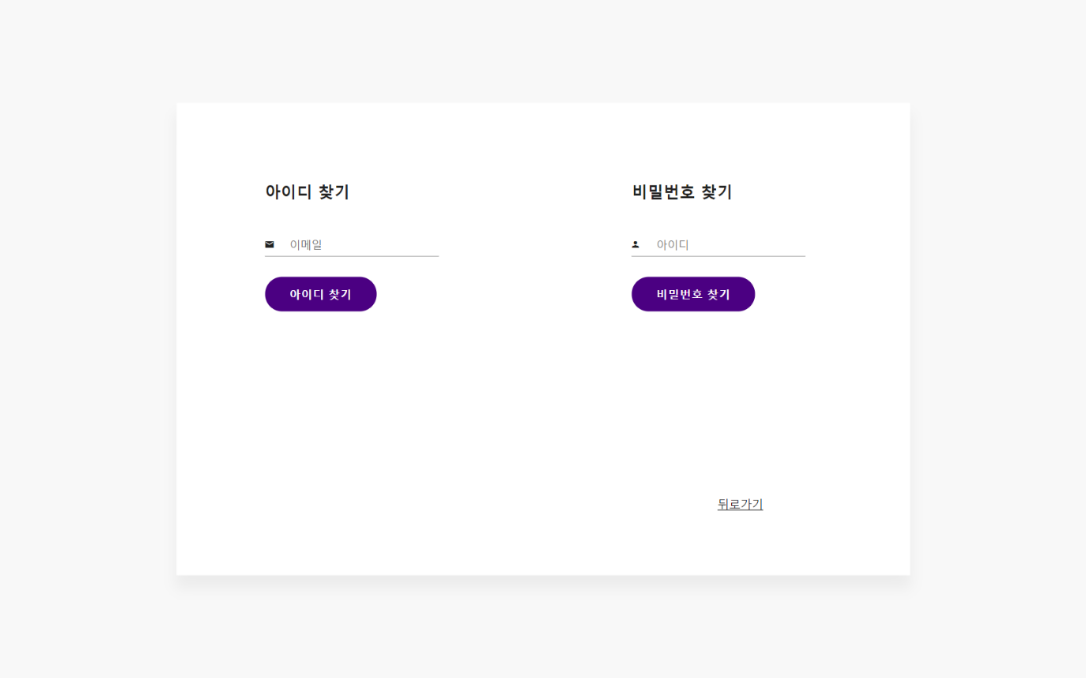
1

1. 모든 입력사항은 필수사항이고 입력사항이 형식에 맞지 않거나 중복되면 경고 메세지가 회원가입 버튼 위에 표시된다. 회원가입이 완료된 후에는 다시 로그인 해야 한다.

2. 가입된 계정이 있을 경우 클릭 시 로그인 페이지로 이동한다.



아이디와 비밀번호 입력 이후 로그인 버튼 클릭 후에는 캘린더 메인 페이지로 이동한다.. 회원가입 또는 바로 아래의 버튼을 클릭하여 가입하거나 아이디 비밀번호 찾기를 할 수 있다.

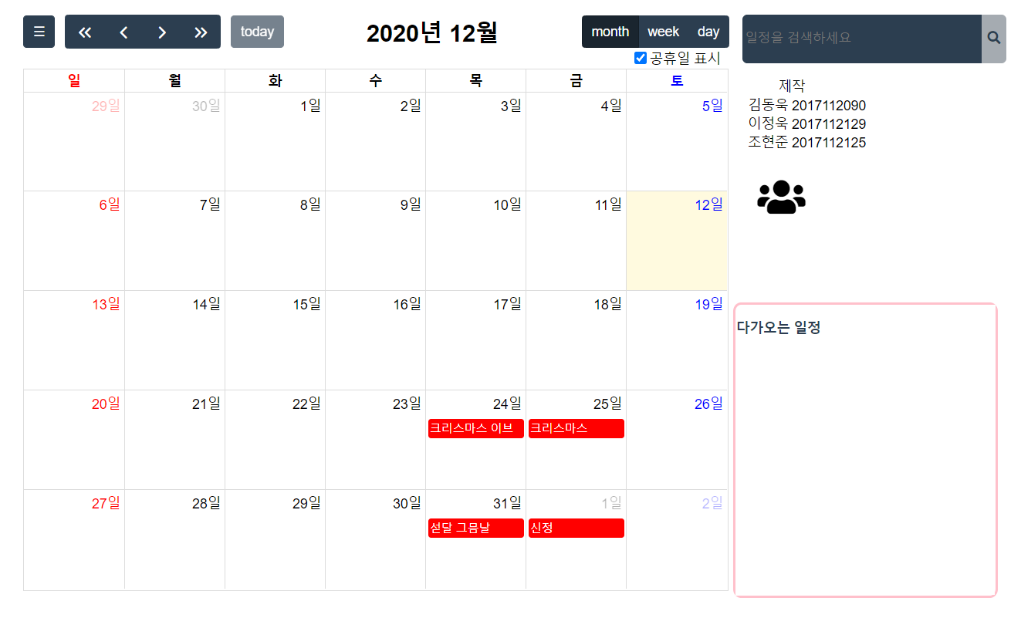


2

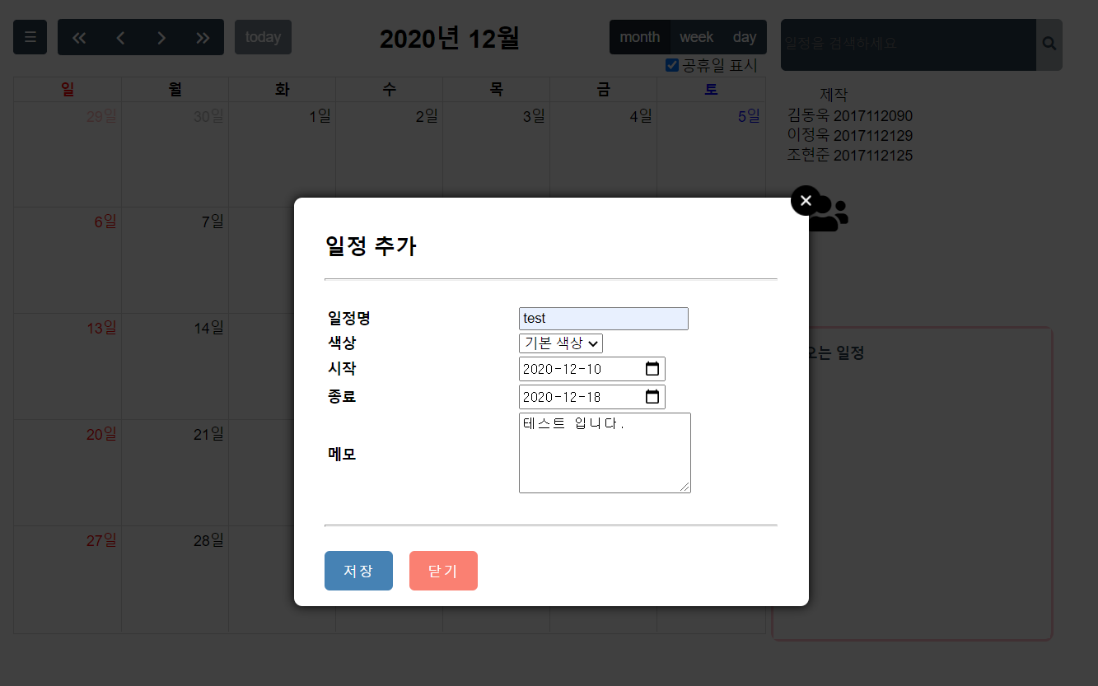
1

1. 이메일의 형식이 맞지 않거나(아이디 찾기) 가입되지 않은 아이디(비밀번호 찾기)일 경우 입력창 아래에 경고 메시지 표시함. 버튼 클릭 시 입력된 이메일 또는 가입된 이메일로 인증번호를 보냄. 사용자는 인증번호를 입력해야한다.

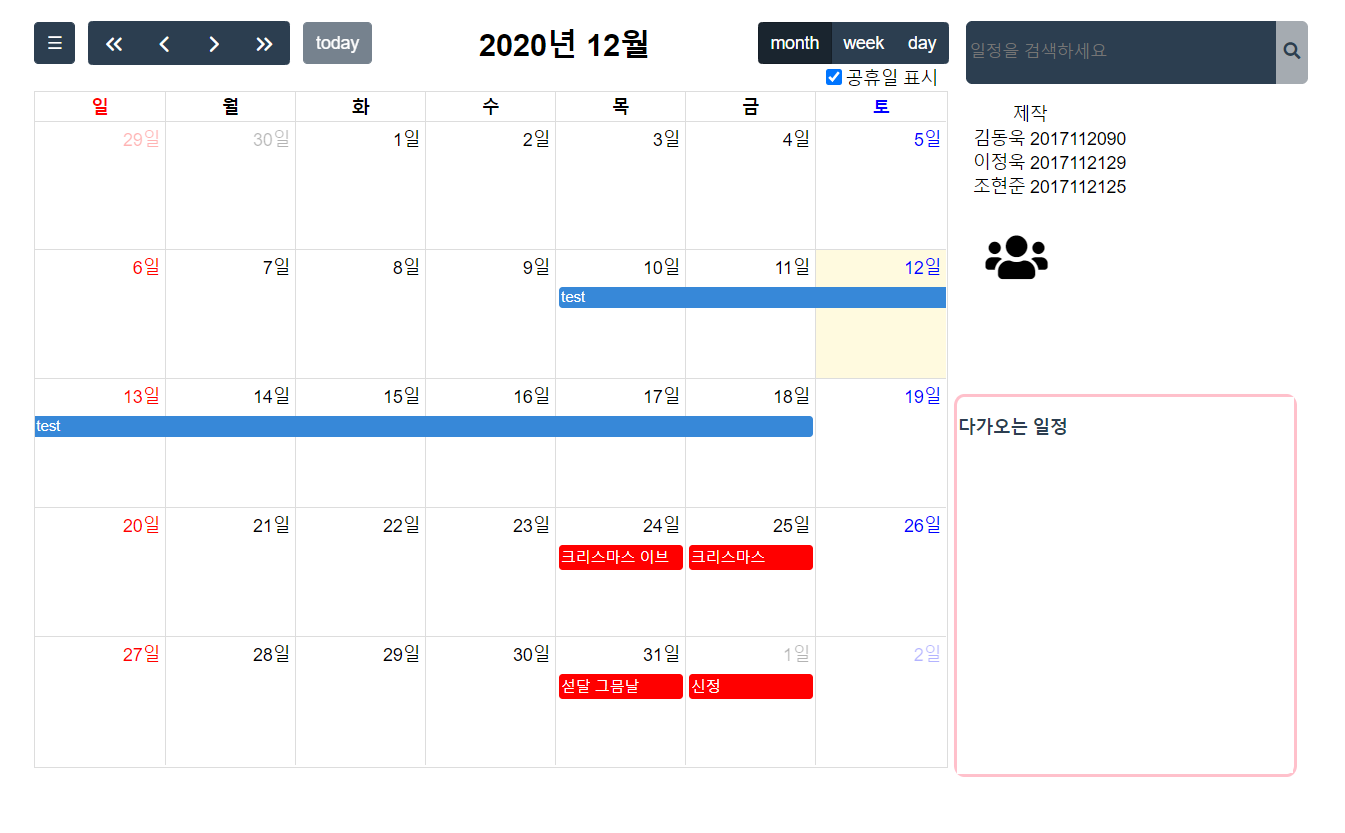
2. 뒤로가기 버튼 클릭 시 로그인 페이지로 이동한다.



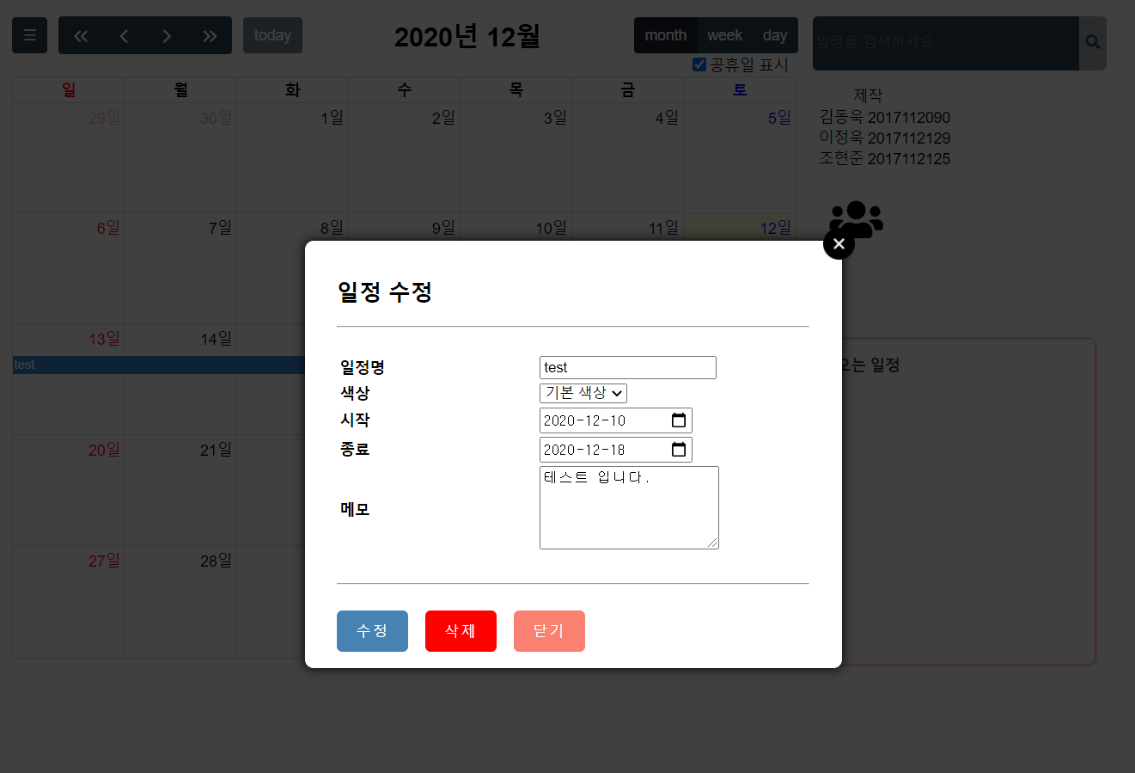
위는 로그인 성공시 나오는 Calendar.jsp 화면이다.



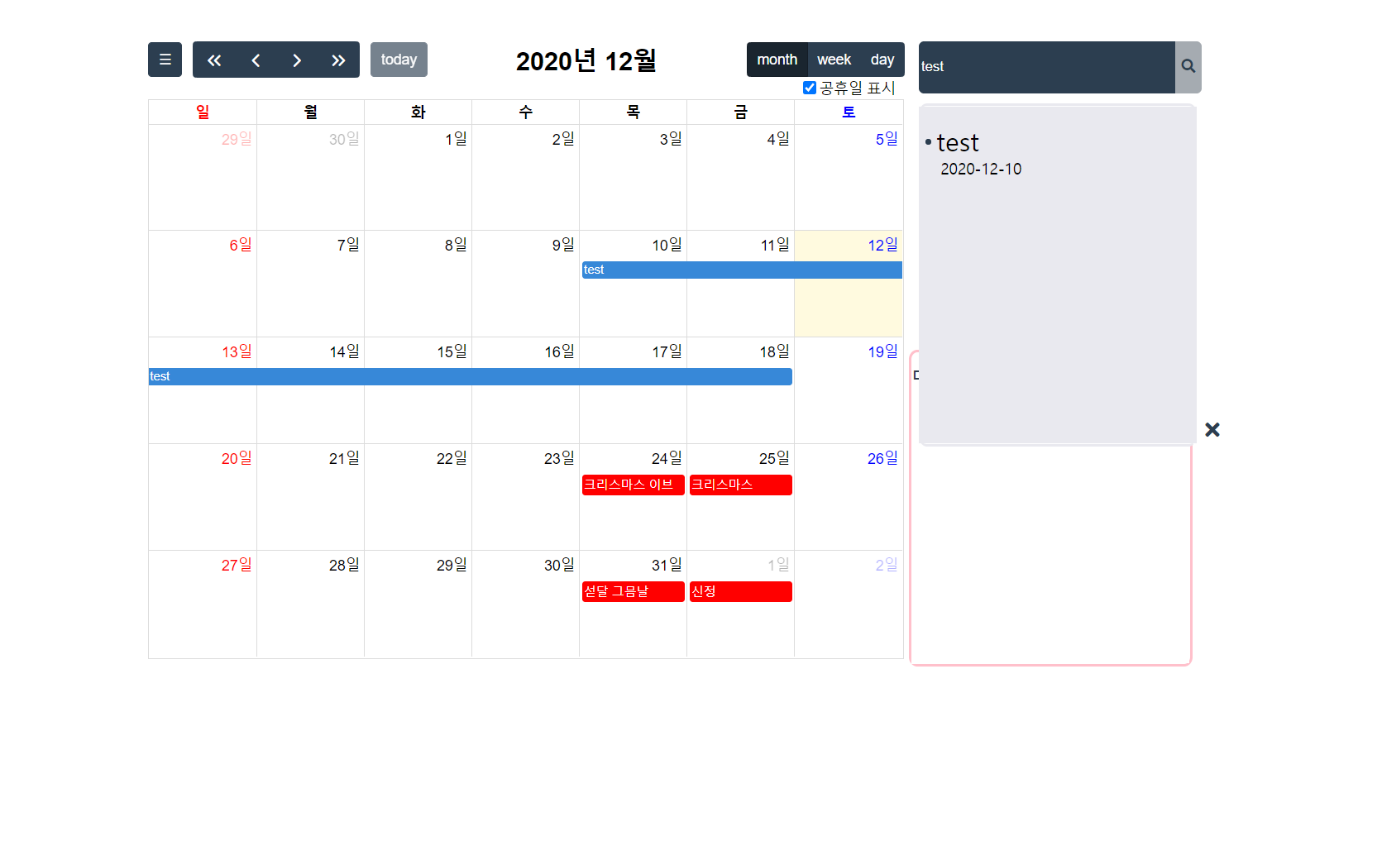
날짜를 클릭하면 위와 같은 창이 나오고 정보를 입력하고 저장을 누르면 일정을 추가할 수있다.



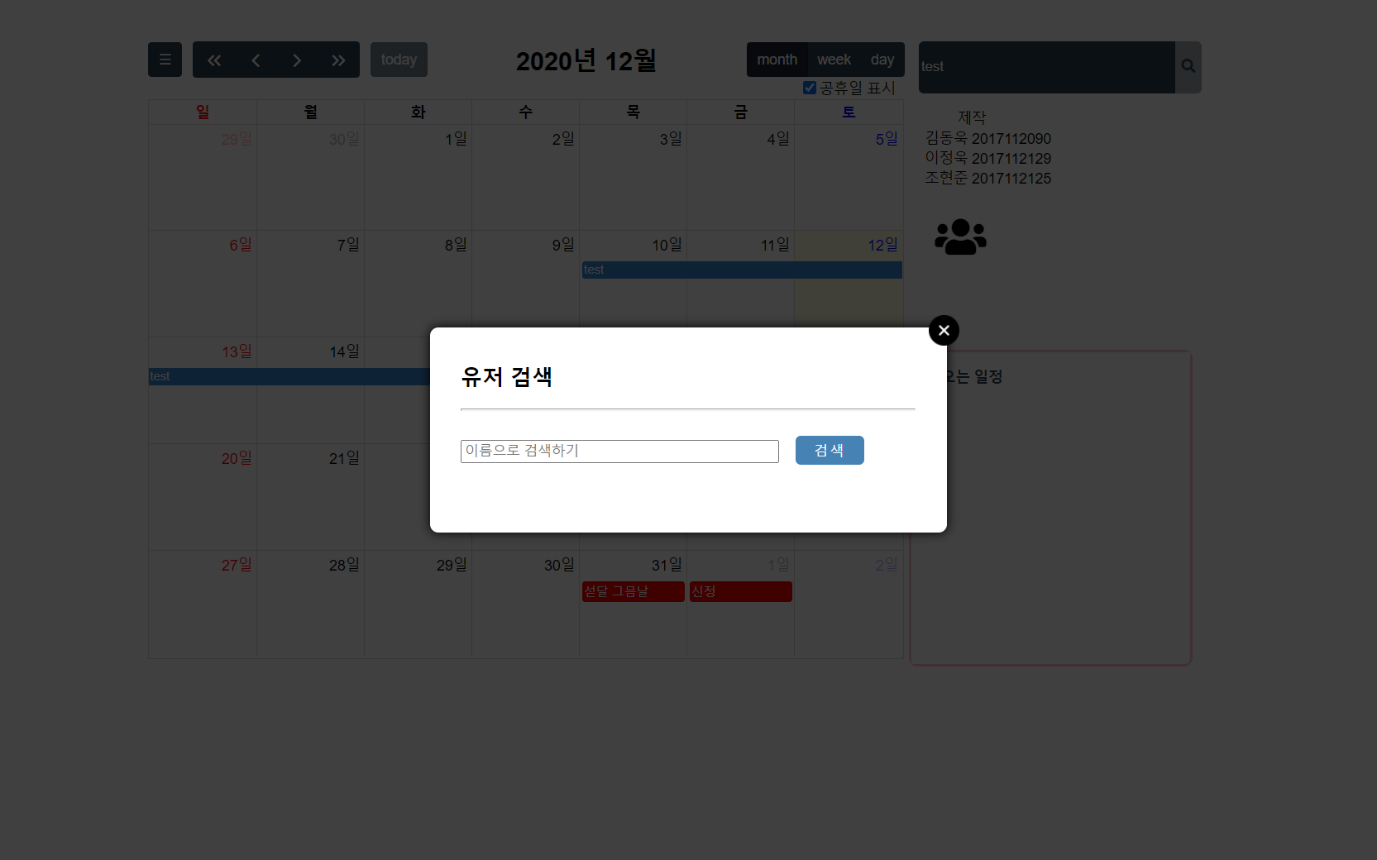
위는 캘린더에 일정이 추가된 모습이다. 드래그 드랍으로 날짜를 이동시킬 수 있고 길이 조절도 가능하다.



일정을 클릭하면 위처럼 일정을 수정할 수 있는 창이 나온다.



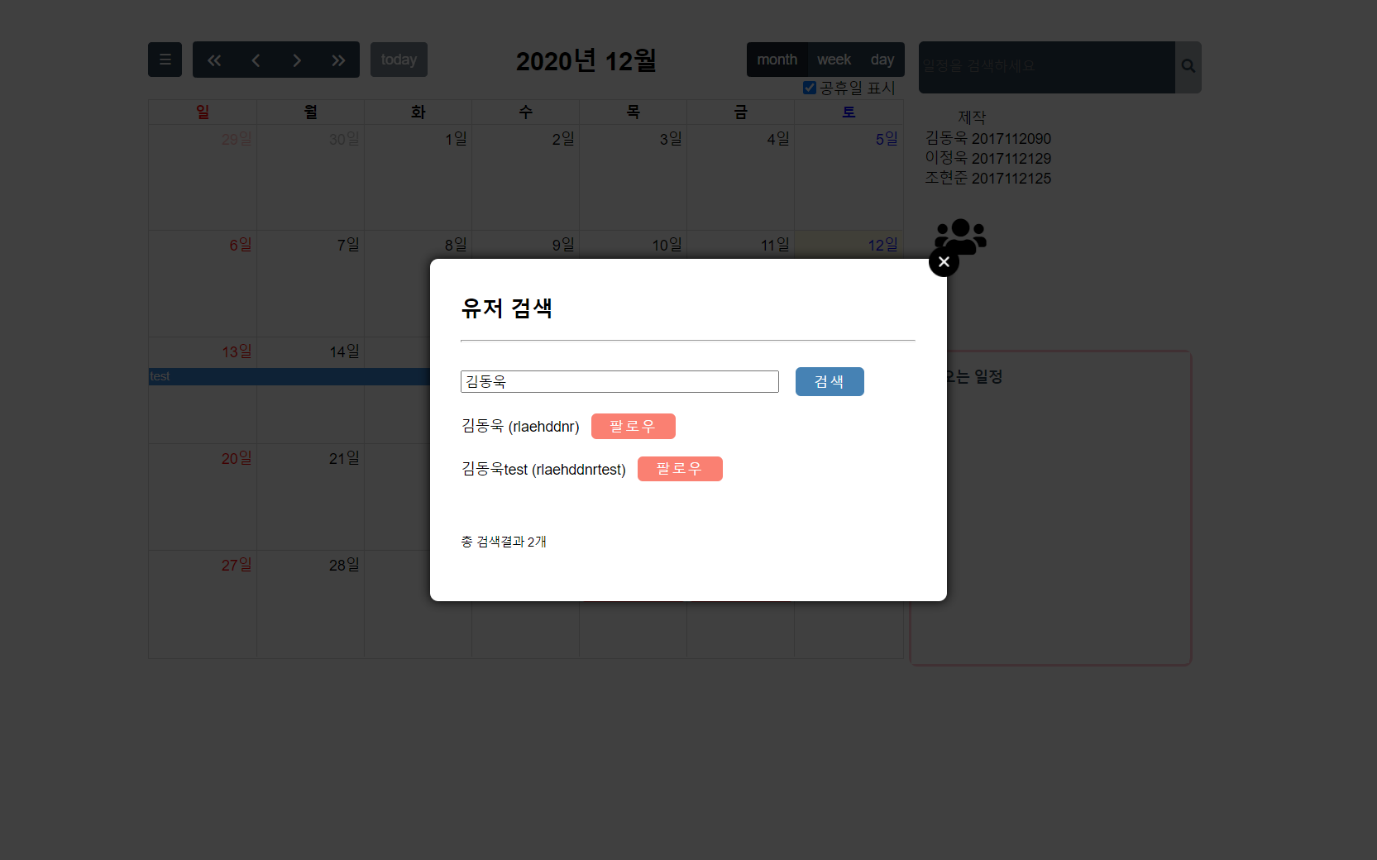
검색창에 원하는 일정의 이름을 입력하면 그에 맞는 결과를 출력해준다.



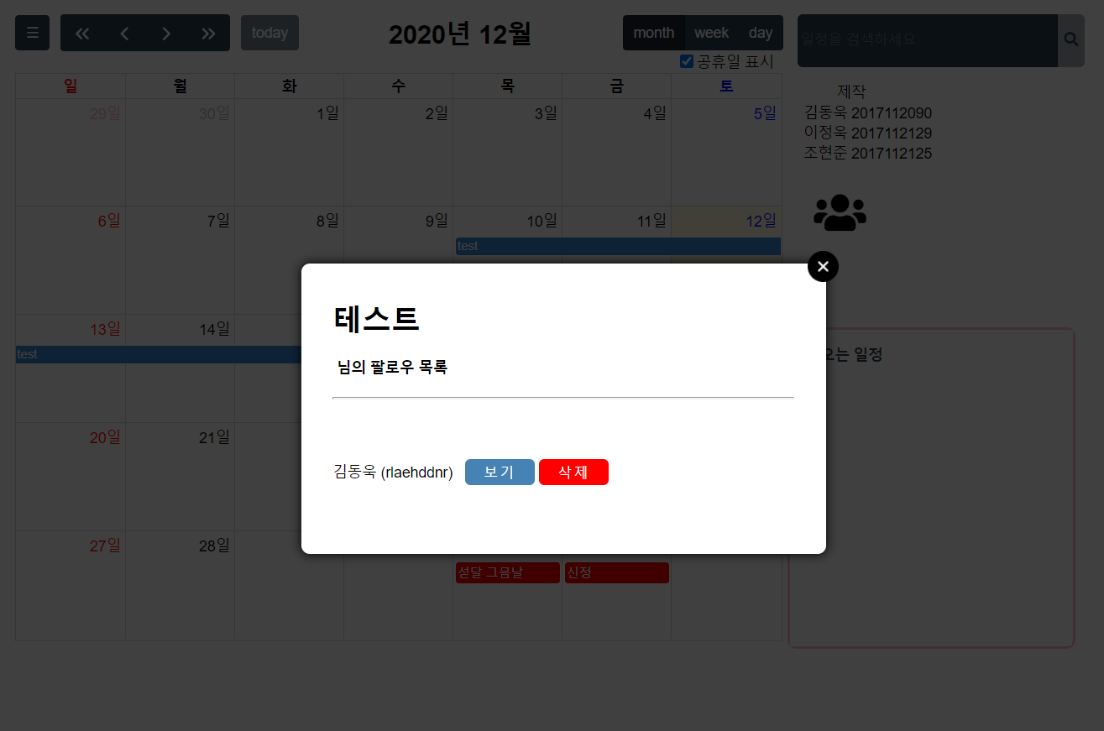
좌측의 ☰(사이드바 버튼)을 클릭 후,

사용자 검색을 클릭하면 위와 같이 유저

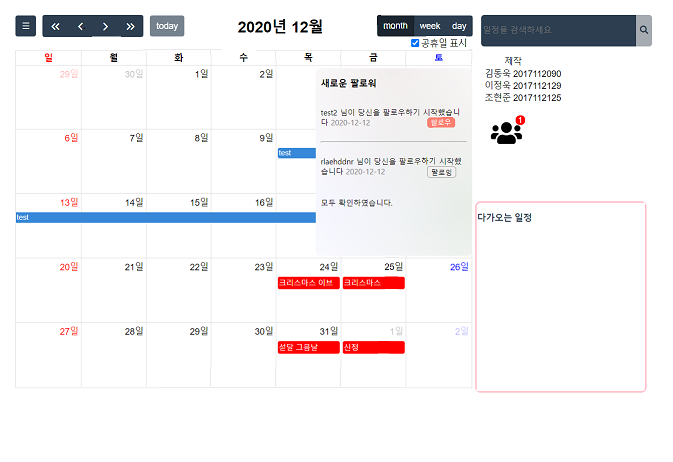
를 검색할 수 있는 창이 출력된다.



웹 캘린더 사용자의 이름을 기준으로 검색하여 유저가 원하는 사용자를 팔로우할 수 있도록 한다.



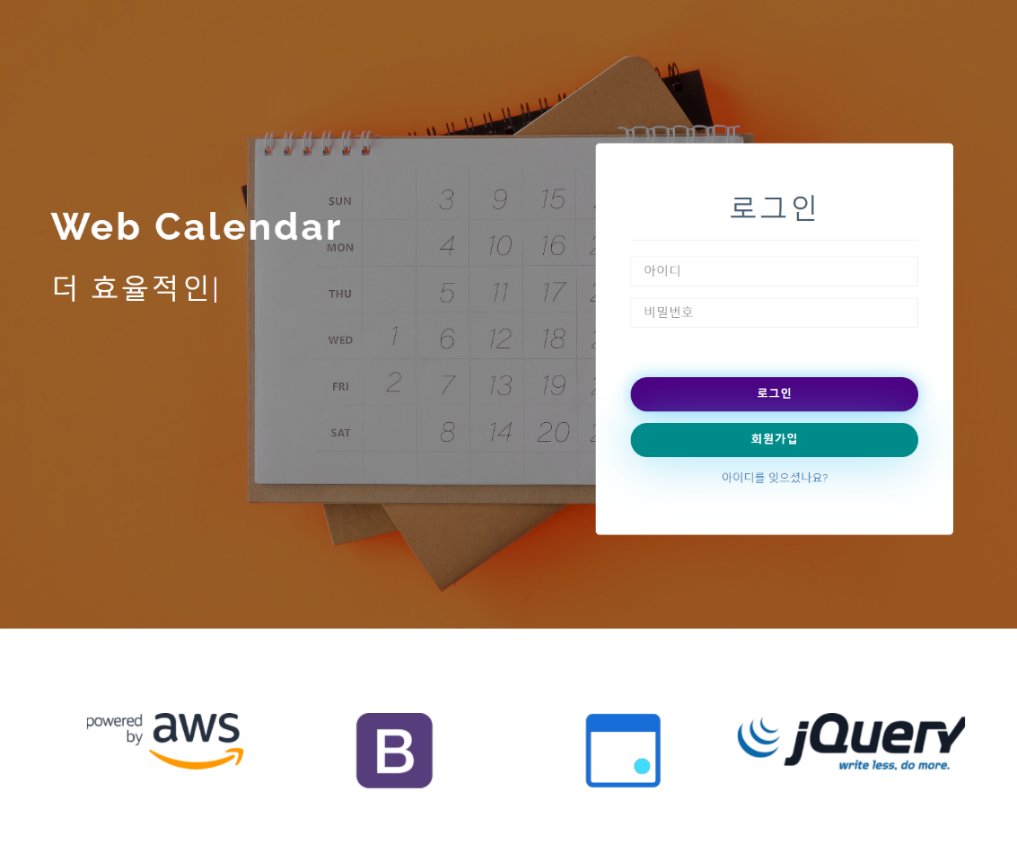
팔로우를 완료한 후에, 좌측 메뉴에서 팔로우 목록을 클릭하여 아까 팔로우한 사람을 확인할수있고, 보기로 그 사람의 일정을 확인할수있다.



누군가 나를 팔로우하면 위처럼 알림창이 생긴다. 서로 팔로우한 상태라면 팔로잉으로 뜨지만 자신이 팔로우한 상대방이 아니라면 팔로우 버튼이 나타난다.

# **로그인 스크린**

## **로그인 화면**



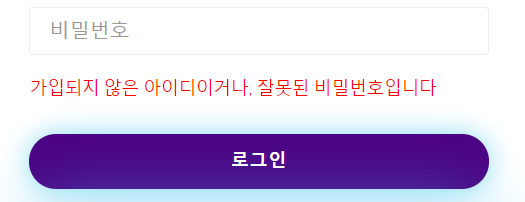
loginscreen.jsp 실행시 출력되는 화면이다.

### **프론트엔드**

프론트엔드 라이브러리인 Bootstrap, Owl Carousel, Typed.js를 이용하여 로그인 화면의 웹 디자인을 구성하였다. 전체적인 디자인은 Bootstrap의 그리드 시스템과 버튼 클래스를 활용하여 디자인했고 LoginScreen/lib/main.css로 저장하였다. 버튼에는 hover 액션을 적용시켜서 좀더 동적인 버튼을 만들었다. 좌측의 큰 제목 Web calendar 밑에 있는 설명은 Typed.js를 이용하여 글쓰는 효과를 적용시켰고 하단에 사용 라이브러리들의 로고를 Owl Carousel을 이용해 슬라이드 형태로 보여준다. 슬라이드 형태는 자동으로 하나씩 넘어가지만 마우스 드래그를 이용해 넘겨볼 수 도있다. Typed.js와 Owl Carousel을 이용한 자바스크립트 코드는 LoginScreen/lib/main.js로 저장하였다.

### **백엔드**

아이디와 비밀번호 입력부분은 <form> 태그로 되어있다. 로그인 버튼은 submit 타입이고 회원가입 버튼은 button타입이다. 로그인 버튼을 누르면 아이디는 id이름의 파라미터로, 비밀번호는 pw 파라미터로 login-db.jsp 에 전달된다. login-db.jsp 에서는 DB와 연결하여 SQL문을 통해 아이디와 비밀번호를 검증한다. 검증에 성공하면 로그인이 되어 calendar.jsp로 redirect 시키지만, 로그인에 실패하면 실패 메시지를 parameter에 담아서 loginscreen.jsp로 전달하여 아래와 같이 출력한다.



## **회원가입 화면**

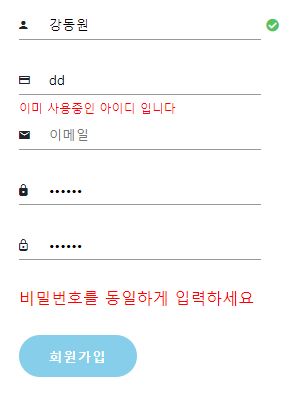
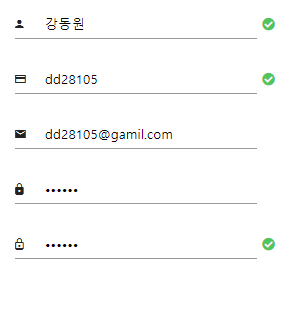
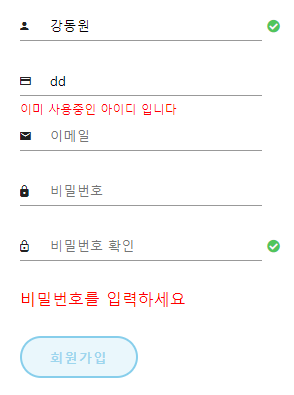


loginscreen.jsp에서 회원가입 버튼 클릭시 이동하는 페이지이다.

### **프론트엔드**

LoginScreen과 마찬가지로 Bootstrap 라이브러리를 적용하여 디자인을 구성하였다. 배경을 약간 어둡게하고 회원가입 부분을 흰색으로 만든뒤 그림자를 적용시켰다. 회원가입의 전체 부분은 세로로 반을 나눠 한쪽은 입력, 한쪽은 그림과 다시 로그인 화면으로 돌아가는 <a>태그 버튼이 있다. 각 input 창의 아이콘은 Material Design Iconic Font라는 오픈소스 아이콘을 적용시켰고 버튼도 hover 이벤트를 적용시켜서 동적으로 표현했다. 또한 font awesome의 체크모양 아이콘도 사용하여 input의 입력이 유효한지 표시하는 용도로 사용하였다. 회원가입 페이지는 register.jsp가 메인이고 css 파일은 register/lib/css/style.css에 저장하였다.

### **백엔드**

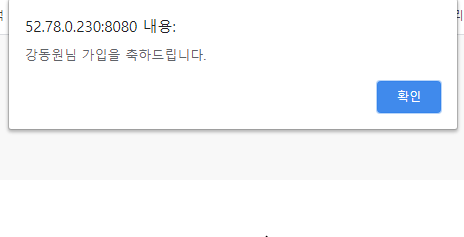
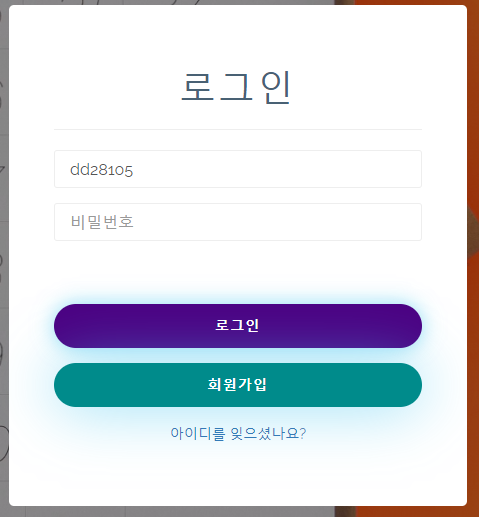
 

회원가입 정보입력은 구체적인 제한은 두지 않았으며 입력된 정보의 유효성만을 확인한다. 모든 칸이 입력되었는지, 아이디가 중복되는지, 비밀번호를 두 칸 동일하게 입력했는지를 확인한다. 검사는 실시간으로 반영되며, 모두 통과해야만 회원가입 버튼이 활성화된다.

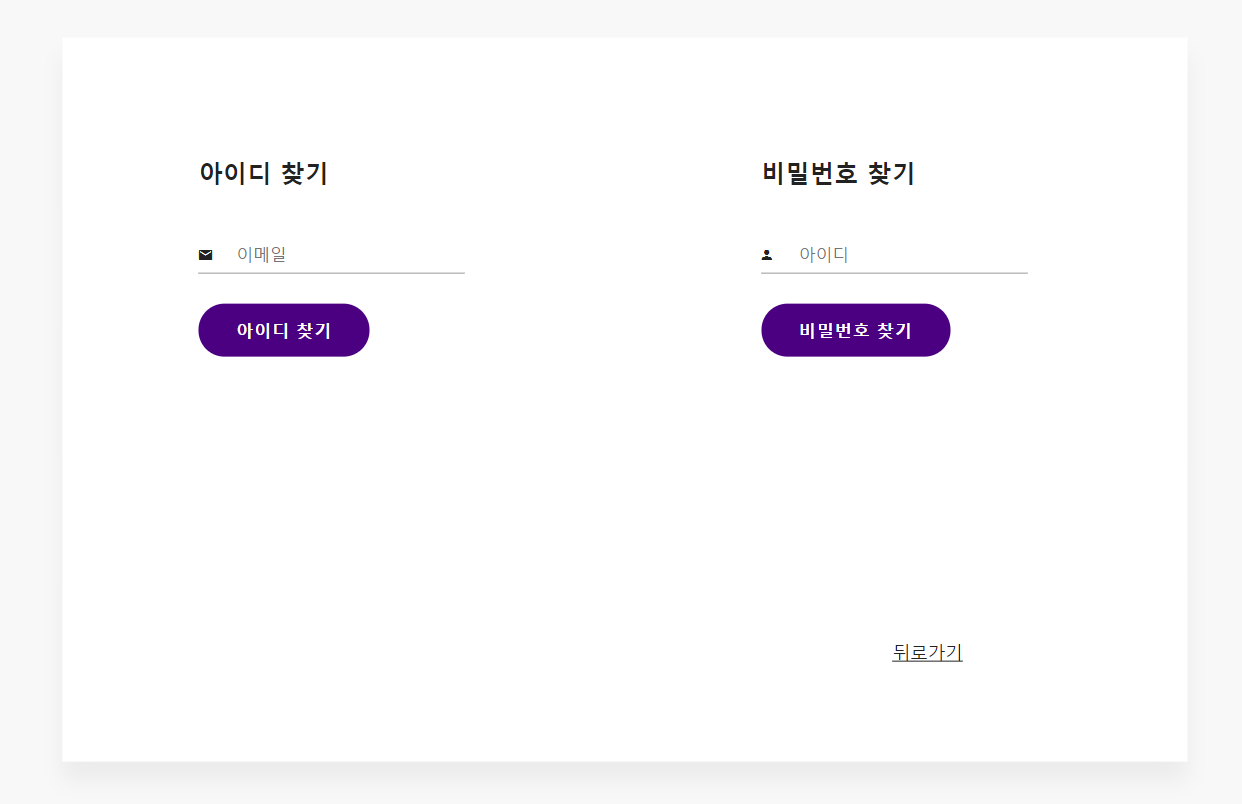
<form>태그는 이름, 아이디, 이메일, 비밀번호 등의 입력칸과 회원가입 버튼으로 구성되며 <form>의 onsubmit 속성, <input>의 onkeyup 속성을 이용하여 각각 제출할 때, 정보를 입력할 때 자바스크립트 함수를 호출하며, 유효성을 검사한다. 유호성 검사를 통과하여 제출이 되면 name, id, email, pw 파라미터가 register-db.jsp로 전달되며 member

테이블에 새로운 정보로 추가한다.

정보 추가가 완료되면 가입축하 알림을 띄우고 id 파라미터와 함께 로그인 스크린으로 전달한다. 로그인스크린으로 돌아오면 아이디 입력부분이 가입한 아이디로 설정되어있다.



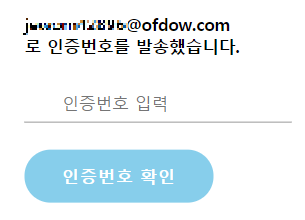
## **아이디 / 비밀번호 찾기 화면**



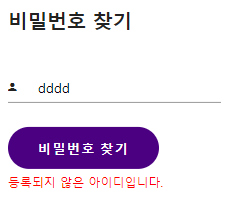
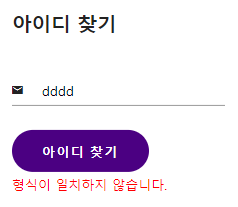
find.jsp 실행시 출력되는 화면이다.

### **프론트엔드**

아이디 / 비밀번호 찾기 화면은 회원가입 페이지와 거의 비슷하게 제작하였다. 전체적인 틀은 동일하며, 이번에는 페이지의 왼쪽 반은 아이디 찾기, 오른쪽 반은 비밀번호 찾기로 구성하였다. 우측 하단에는 뒤로가기 버튼을 넣었다. find.jsp와 동일한 경로에 find.css를 만들었다. 그리고 아이디 / 비밀번호 찾기 시 새롭게 iframe 페이지가 생성되는데, 여기에도 동일한 css를 적용시켜서 화면 구성이 어색하지 않도록 하였다.

****

### **백엔드**

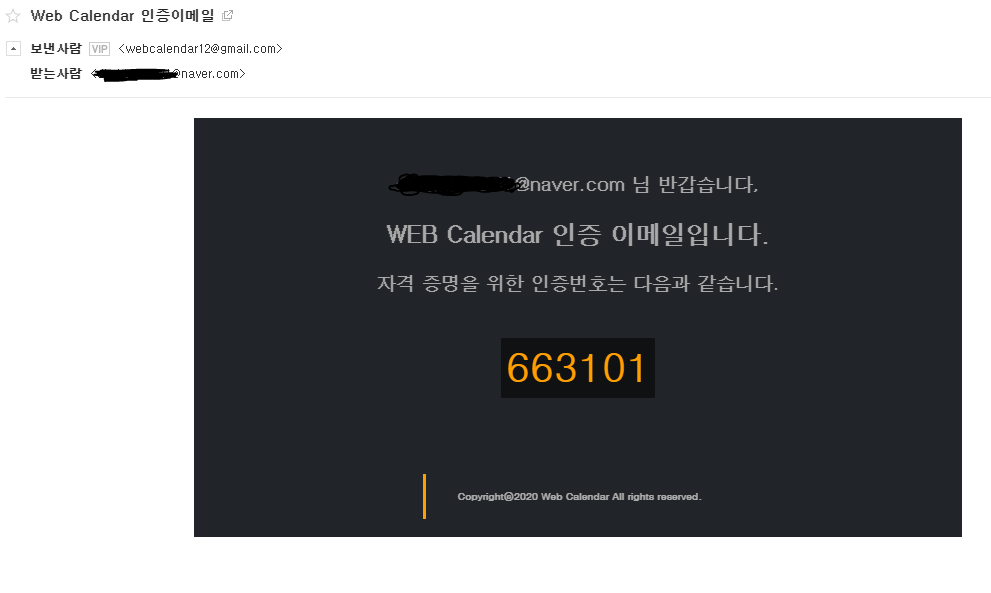


아이디 찾기는 이메일을, 비밀번호 찾기는 아이디를 입력해 진행하며, 회원가입과 마찬가지로 onkeyup속성을 이용하여 이메일의 경우 입력한 이메일의 형식이 올바른지, 아이디의 경우 등록되어 있는 아이디인지 실시간으로 검사하며, 오류가 없을 때 버튼이 활성화된다.

아이디 찾기와 비밀번호 찾기 모두 등록된 이메일(아이디 찾기는 입력된 이메일)로 인증번호가 포함된 이메일을 발송하며 사용자는 인증번호를 iframe창에 생성된 페이지에 인증번호를 입력하는 방식으로 구현하였다.

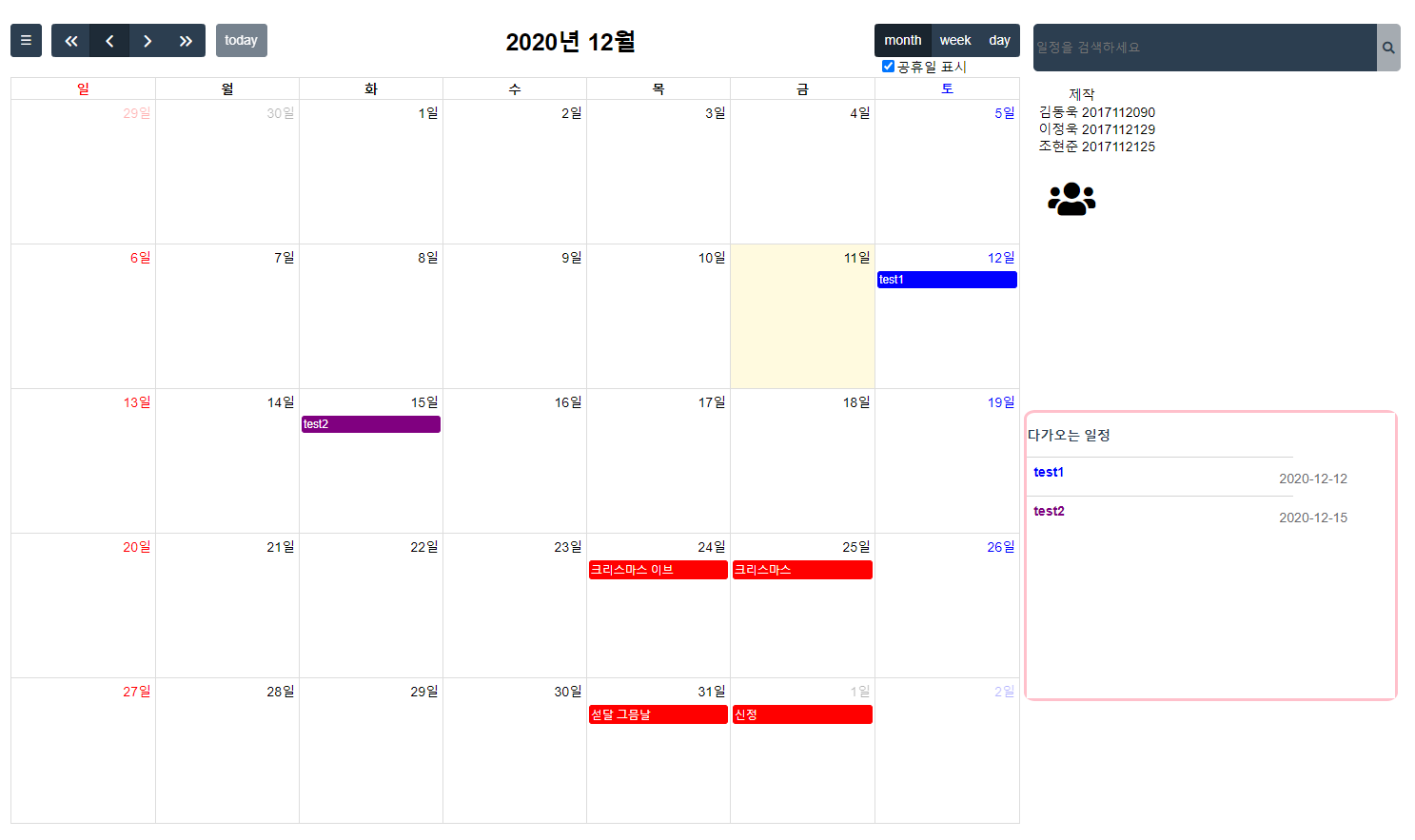
먼저 아이디 찾기는 버튼을 클릭하면 certification.jsp로 파라미터 email을 전달하며 certification.jsp에서 6자리 난수를 생성한 후, 자바 메일 라이브러리와 gmail stmp api를 이용하여 해당 메일로 인증번호를 발송한다. iframe 창에서 진행한다.

비밀번호 찾기는 findpw.jsp로 파라미터 id를 전달하며 DB에서 해당 id로 등록된 이메일을 찾아 해당 이메일로 메일을 발송한다.



메일 발송이 완료되면 test.jsp로 forward하여 인증번호 검증 페이지를 구현하며 이 페이지에서 사용자는 인증번호를 입력하게 된다. 인증번호를 입력하여 확인버튼을 클릭하면 compare.jsp로 email, id를 <input type=’hidden’>으로, 사용자가 입력한 번호와 함께 전달한다. compare.jsp에서는 인증번호가 맞는지 검사하여 맞으면 DB에서 정보를 찾아 표시하고 틀리면 인증번호를 다시 입력받는다.

# **캘린더 스크린**



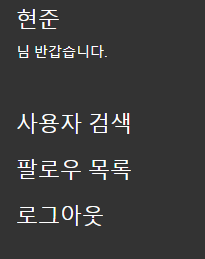
calendar.jsp의 출력 화면이다.

### **프론트엔드**

로그인한 유저의 모든 일정을 출력한다. 사용자가 일정을 추가하고 싶은 날의 빈칸을 클릭하면 다음과 같이 일정의 세부내용을 입력할 수 있는 칸이 출력된다. 또한 기존에 있는 일정을 클릭하여 일정을 수정하거나 삭제할 수 있다. 이때 사용자가 직접 색상을 지정하여 일정의 종류를 정할 수 있다.

* 일정 추가 - 일정 수정

좌측 상단의 사이드바를 클릭하면 오른쪽의 사진과 같이 사용자 검색, 팔로우 목록, 로그아웃을 할 수 있게하는 버튼이 있다.

사이드바 버튼 우측에 있는 버튼은 각각 1년 전, 이전 달(혹은 주, 일), 다음 달(혹은 주, 일), 1년 후로 이동하는 버튼이고 today를 클릭하면 오늘로 다시 돌아온다. 우측 상단의 month, week, day를 선택하여 달력에 표시할 날의 단위를 선택할 수 있고, 공휴일 표시 체크박스의 체크 여부에 따라 공휴일의 출력 여부가 결정된다.

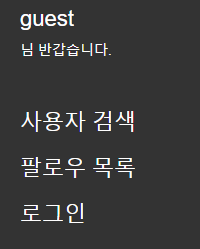
### **백엔드**

대부분의 작업은 Calendar/calendar.js에 있는 코드로 진행된다. 우선, 처음 로드될 때 Fullcalendar에서 제공하는 Calendar 객체를 이용하여 달력을 출력할 수 있도록 한다.

calendarjson.jsp에서는 데이터베이스의 plan 테이블에서 해당 유저가 가지고 있는 일정의 목록을 json으로 파싱하는 작업을 한다. Fullcalendar에서 제공하는 eventSources를 통해서 이 json을 달력의 이벤트로 입력받는다. 또한, 구글 캘린더에서 제공하는 공휴일 달력도 eventSources로 받는다. 프론트엔드의 공휴일 표시 체크박스를 해제하면 eventSource에서 해당 이벤트를 지우고 체크박스를 체크하면 다시 추가한다.

만약, 프론트엔드에서 이벤트를 드래그하여 옮기거나 사이즈를 변경해 일정을 변경한다면, ajax를 통해 CalendarAjax.java로 변경된 내용을 전송하여 데이터베이스 상의 일정 또한 수정할 수 있도록 한다.

사용자가 일정 추가 버튼을 클릭했다면 calendar에 사용자가 입력한 일정을 추가하고 calendar\_insert\_form를 calendar-db/calendar-insert-db.jsp로 전달하여 데이터베이스에도 추가한다. 일정 수정 및 삭제 버튼을 클릭할 때도 캘린더 상의 이벤트를 먼저 수정 혹은 삭제하고, 데이터베이스 상의 일정을 수정, 삭제한다.

사용자가 검색어를 입력하여 일정을 검색한다면 search/search.jsp로 전달해서 데이터베이스 조건절 중 like를 이용하여 해당 문자열을 포함한 일정을 출력한다.

사이드바 버튼 중 로그아웃 버튼을 클릭하면 세션을 만료시킨 후 로그인 화면(loginscreen.jsp)으로 이동한다. 만약 세션이 없는 상태로 calendar.jsp에 접속하면 다음과 같이 guest인 상태로 접속된다.

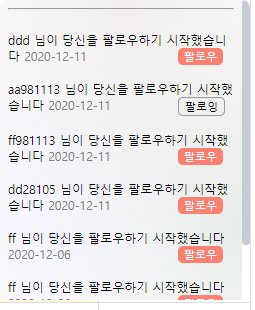
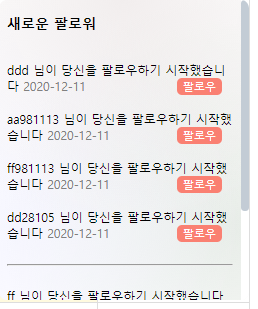
## **일정 검색**

검색어를 포함하는 일정의 제목을 모두 검색한다. 일정은 가까운 순서대로 표시되며 일정의 제목을 클릭하면 메모를 볼 수 있다.

검색어를 입력하고 검색하면 search\_word 파라미터가 search.jsp로 전달되며 쿼리문을 %"+search\_word+"% 와 같이 작성하여 검색어를 포함한 일정을 모두 가져오며 제목 클릭 시 메모표시는 search.js에서 진행한다.

## **팔로우 알림**

### **프론트엔드**



팔로우 알림1

팔로우 알림2

다른 사용자가 나를 팔로우하면 위와같이 (우측 상단 아이콘) 새로운 팔로워가 있다고 알려주며 아이콘을 클릭하면 알림창을 iframe에 띄우며 <form>태그를 이용하여 notice.jsp 로 이동한다. 알림창에서 바로 팔로워를 맞팔로우 할 수 있으며 이미 팔로우한 사람은 **팔로우 알림2**와 같이 ‘팔로잉’으로 표시된다.

### **백엔드**

notice.jsp에서는 세션에서 id를 얻고 notice DB에서 해당 id의 알림을 가져와 출력해주며, target\_ID를 변수에 저장한다. 이후 follow DB에서 id(나의 id)가 target\_ID를 팔로우하지 않았다면 <form>태그를 만들며 follow-db.jsp에 두 아이디를 파라미터로 전달하여 팔로우를 할 수있도록 하였다.

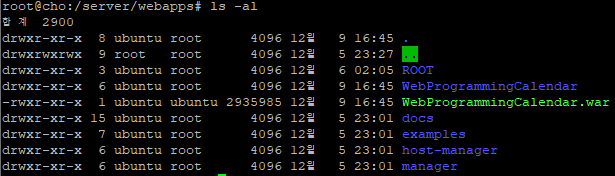
알림에는 팔로워의 id가 표시되며 팔로우 하기 시작한 날짜도 옆에 표시된다. 알림의 번호를 매겨 DB에 저장하므로 가장 최근 알림부터 위에 표시되고 확인한 알림보다 새로운 알림이 먼저 표시된다.

## **다가오는 일정**

캘린더 스크린 우측 하단에는 다가오는 일정을 보여주며 일주일 뒤까지의 일정을 가까운 순서로 보여준다. calendar.jsp파일 내에서 일정의 색상에 맞게 css를 설정하여 내가 어떤 색 일정으로 표시하였는지 한눈에 확인이 가능하다. 제목과 메모 모두 표시되며, 제목이나 메모가 길어서 박스를 넘어갈 경우 … 으로 표시된다.

# **배포**

해당 프로젝트를 배포하기 위해 AWS EC2를 이용하였다. AWS EC2에서 Ubuntu 18.04 인스턴스를 생성하여 Tomcat 및 mysql 환경을 구축하고, 인스턴스의 외부 접속 IP에서 일부 포트를 외부에서 접속 가능하게 설정했다.



* webapps 폴더에 있는 프로젝트의 war 파일



* conf/server.xml의 일부