인턴 최종 발표 보고서

장연수

목차

1. 진행 경과 2p
2. Google Calendar API 소개 8p
3. 구현한 기능 8p
4. 기능 구현 방법 8p
5. API 사용을 위한 사전 준비 8p
6. 전체 클래스 구조 9p
7. 기능 동작 순서 10p
8. 구글 API 사용을 위한 인증 11p
9. Feauture List 12p
10. 캘린더 리스트 받기 13p
11. 이벤트 리스트 받기 14p
12. 이벤트 하나 정보 요청 16p
13. 이벤트 생성 16p
14. 이벤트 수정 17p
15. 이벤트 삭제 18p
16. 이벤트 캘린더 옮기기 19p
17. 초대 일정의 수정, 삭제 여부 19p
18. 캘린더 생성 19p
19. 캘린더 수정 19p
20. 캘린더 삭제 19p
21. 캘린더 정보 받기 19p
22. 캘린더 정보 수정 20p
23. 캘린더 권한 리스트 받기 20p
24. 캘린더 권한 생성 20p
25. 캘린더 권한 수정 21p
26. 캘린더 권한 삭제 21p
27. 우선순위 정렬 21p
28. Monthly Calendar 일정 그리는 방법 22p
29. Daily, Weekly Calendar 일정 그리는 방법 23p
30. 캘린더 내보내기 기능 구현 방법 24p
31. Google Calendar API의 장단점 25p

1) 진행 경과

1주차(3/20~3/23) -> Google Calendar API 조사. 월별 캘린더 출력. 하루 일정 출력. Java Spring 공부

2주차(3/26~3/30) -> 프로젝트 구조 설계. 모델링. AJAX를 이용한 비동기 전송 방식 이용. 이어지는 일정 출력. 체크된 여러 캘린더들의 일정 출력. 일정의 이름을 클릭하면 일정의 정보가 화면에 나오는 것을 구현.

3주차(4/2~4/6) -> 일정이 날짜 칸을 넘어가는 경우 더보기 기능 구현. 일정 상세 정보 페이지 구현. 일정 생성, 수정, 삭제 구현. UI 개선. 목록 캘린더(한달 동안의 일정 목록) 출력.

4주차(4/9~4/13)-> 일정 알람 생성, 수정, 일정 초대 기능 구현(참석자 목록 보여주기). thread를 이용한 응답속도 개선. 페이지 이동 없앰. 일정을 취소하거나 저장하면 사용자가 전에 봤던 페이지로 이동하게 하는 기능 구현

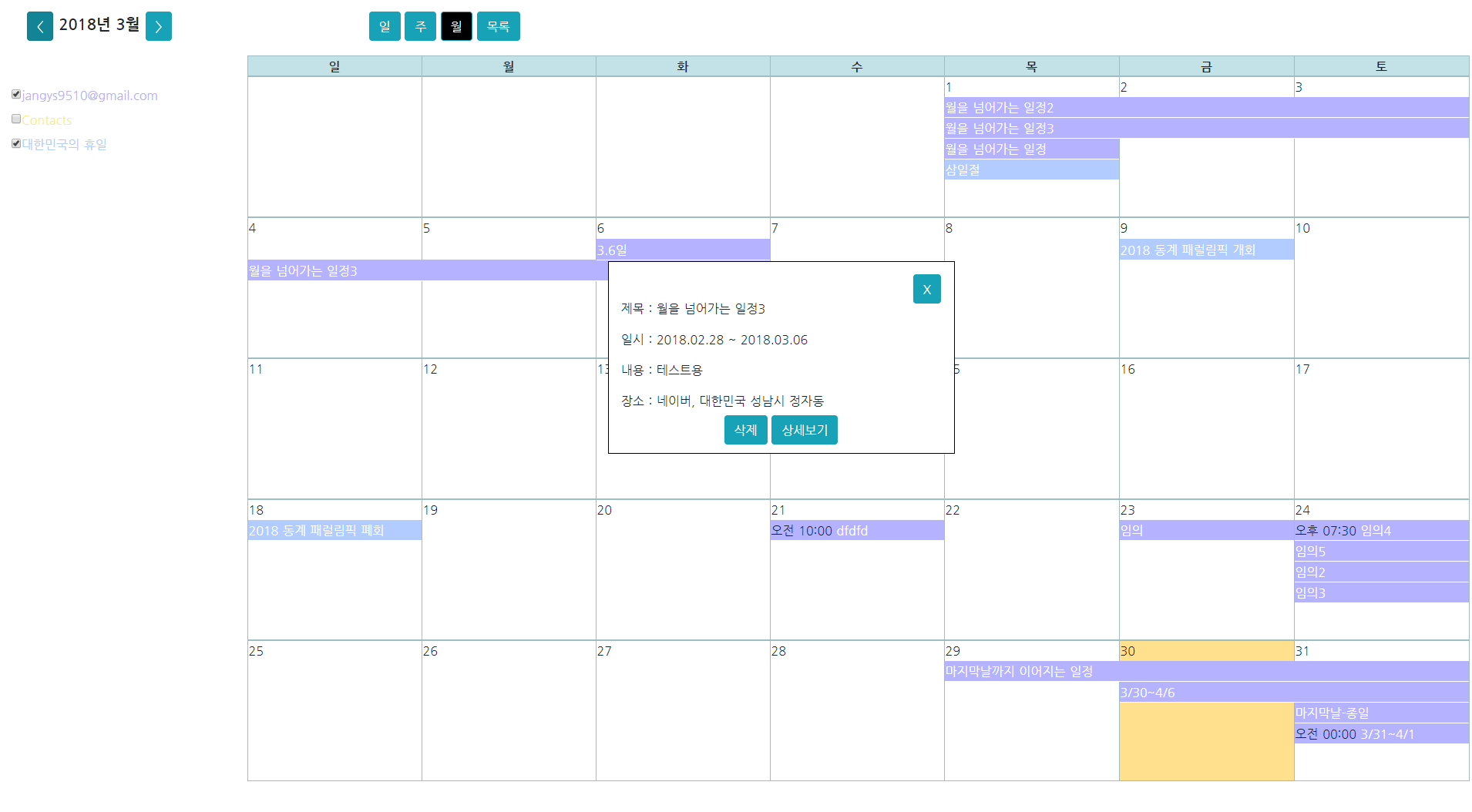
5주차(4/16~4/20)-> 참석 여부를 요약창에서 선택하는 기능 구현. 일정 요약창에 참석자 목록 볼 수 있는 기능 구현. 참석 여부 캘린더에 표시. 반복 일정 처리, 생성. 반복 일정 정보 상세 페이지에 표현. 반복 일정 맞춤 선택지 제공.

6주차(4/23~4/27)-> 반복 일정 수정 구현(3가지 중 선택). 반복 일정 삭제. side에 미니 캘린더 추가. 종일 일정 알림 표시 변경. 캘린더 생성, 수정, 삭제 구현. 캘린더 접근 권한 생성, 수정, 삭제 구현. daily/weekly 캘린더 종일 일정 부분 구현.

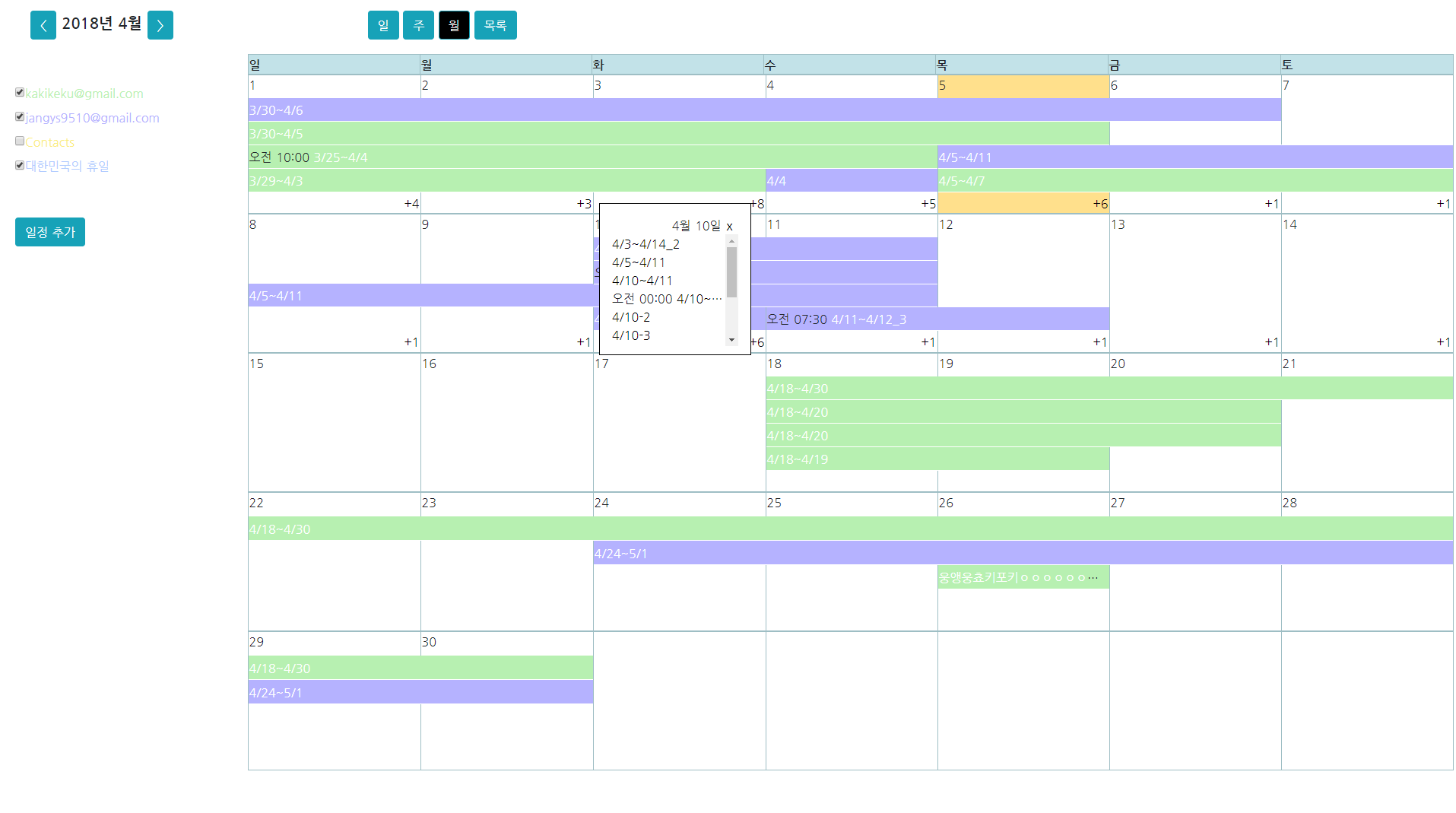
7주차(4/30~5/4)-> daily/weekly 캘린더 시간이 있는 일정 부분 구현. daily/weekly 캘린더에서 드래그하여 일정 생성하는 기능 구현. daily/weekly 캘린더 현재 시간 표시. monthly 캘린더의 일정을 원하는 날짜로 드래그하여 옮기는 기능 구현. daily/weekly 캘린더 시간 일정 드래그 하여 시간대 조정하는 기능 구현.

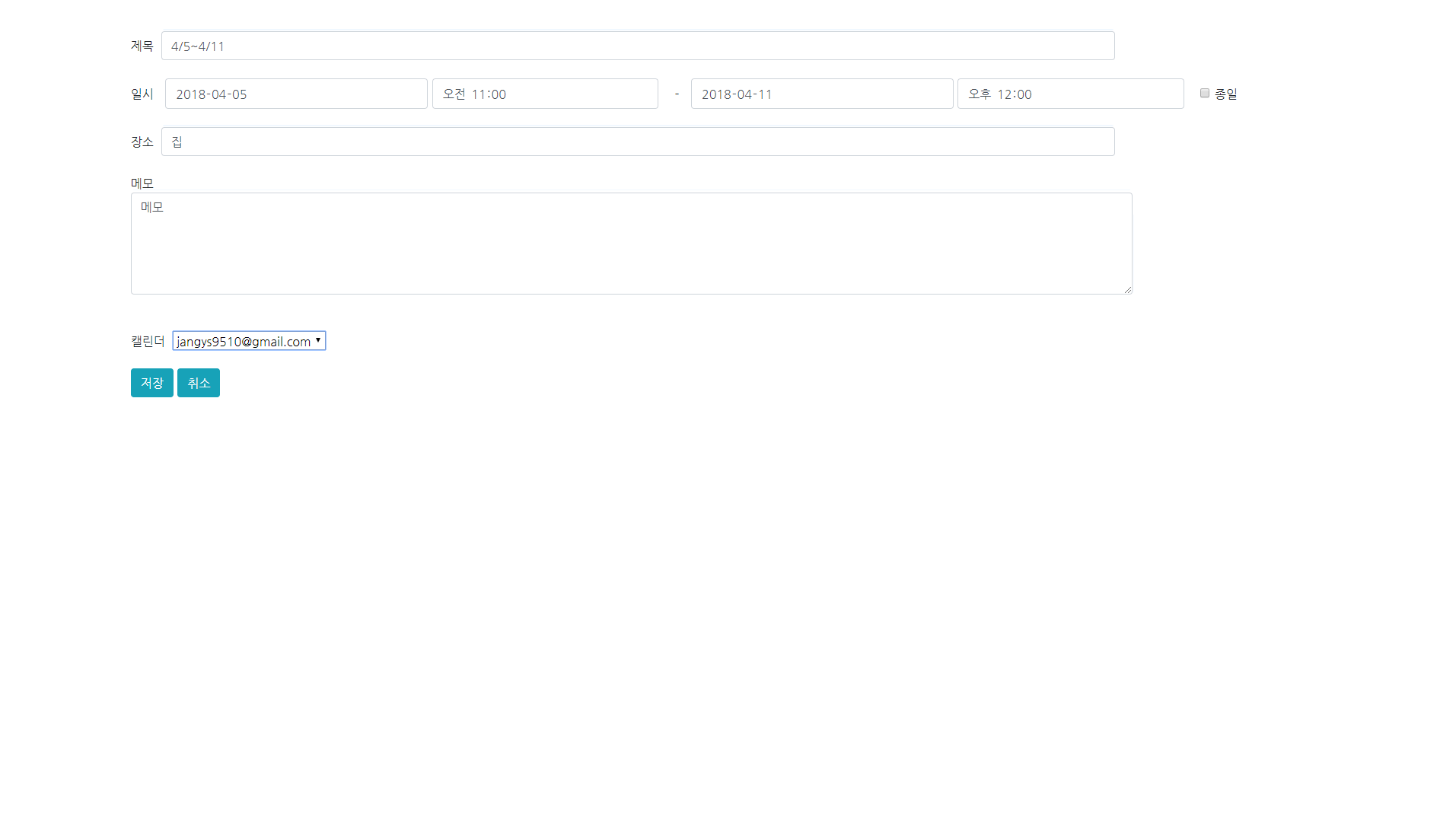
8주차(5/7~5/11)->캘린더 테스트 하며 오류 해결. 일정 상태 변경 구현. 일정 공개 여부 변경 구현. 테스트 코드 일부 작성. 일정에 업로드 된 파일 표시

9주차(5/14~)->캘린더 내보내기 기능 구현

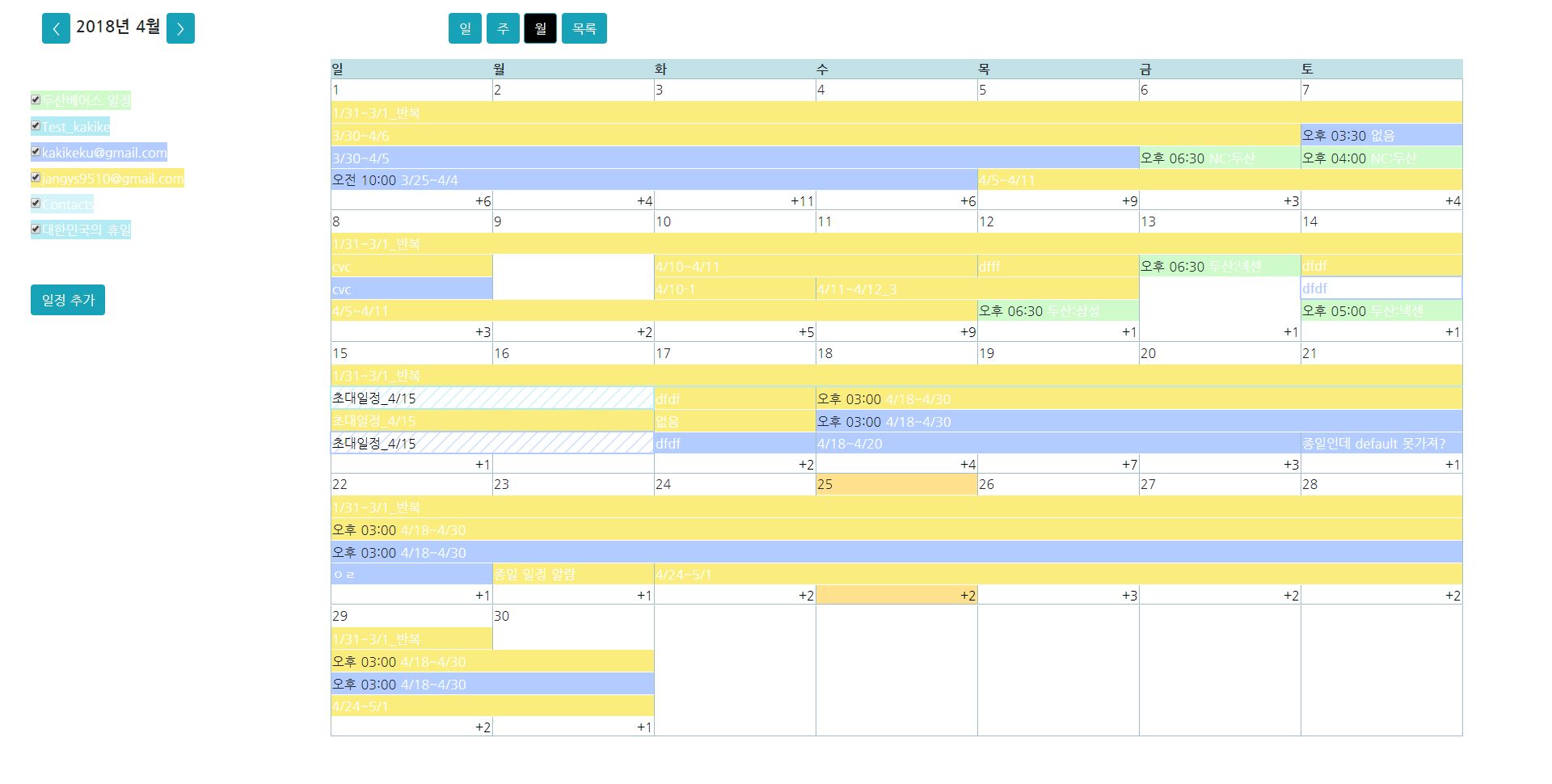


* 3/30 UI 모습

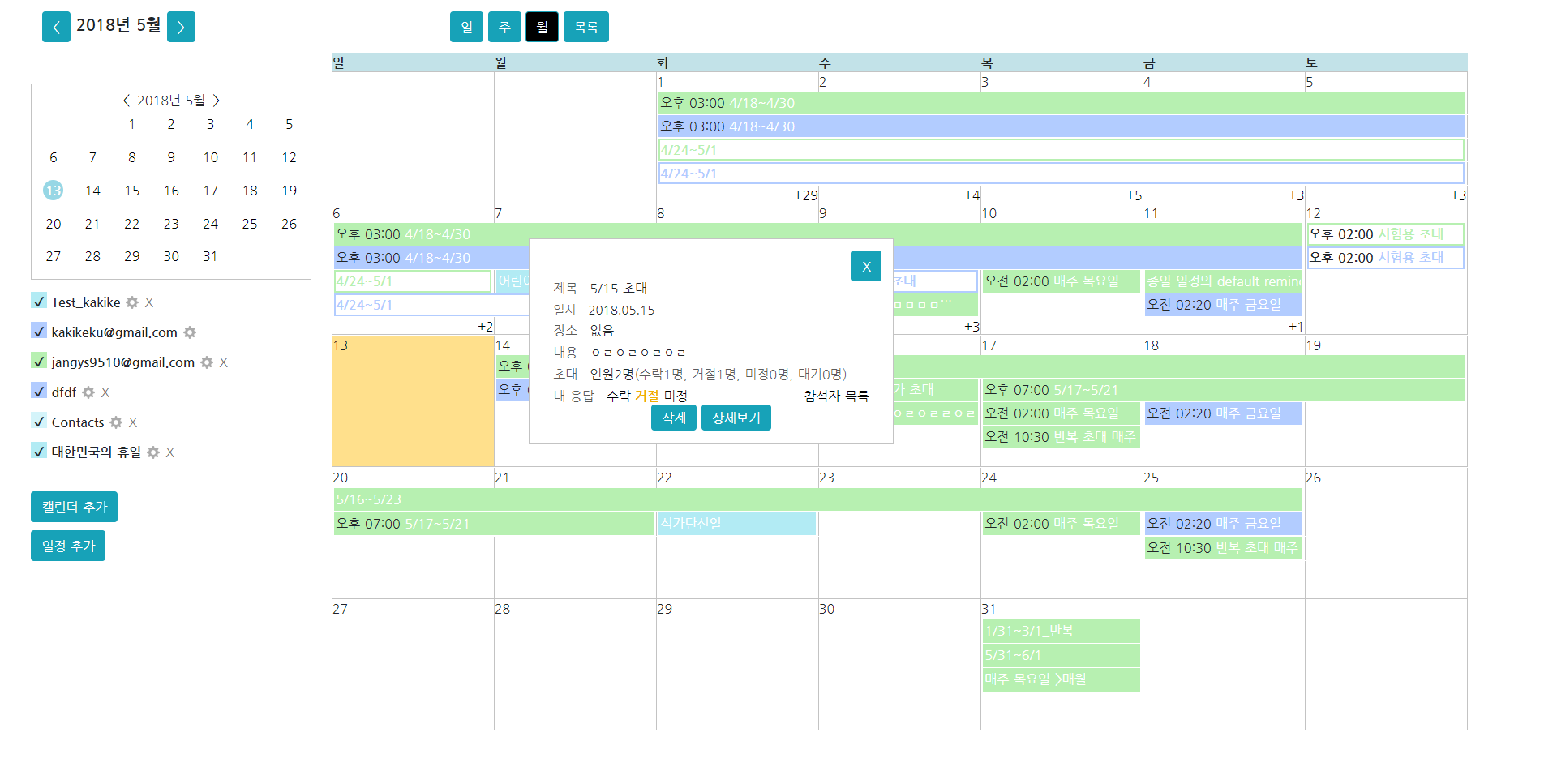
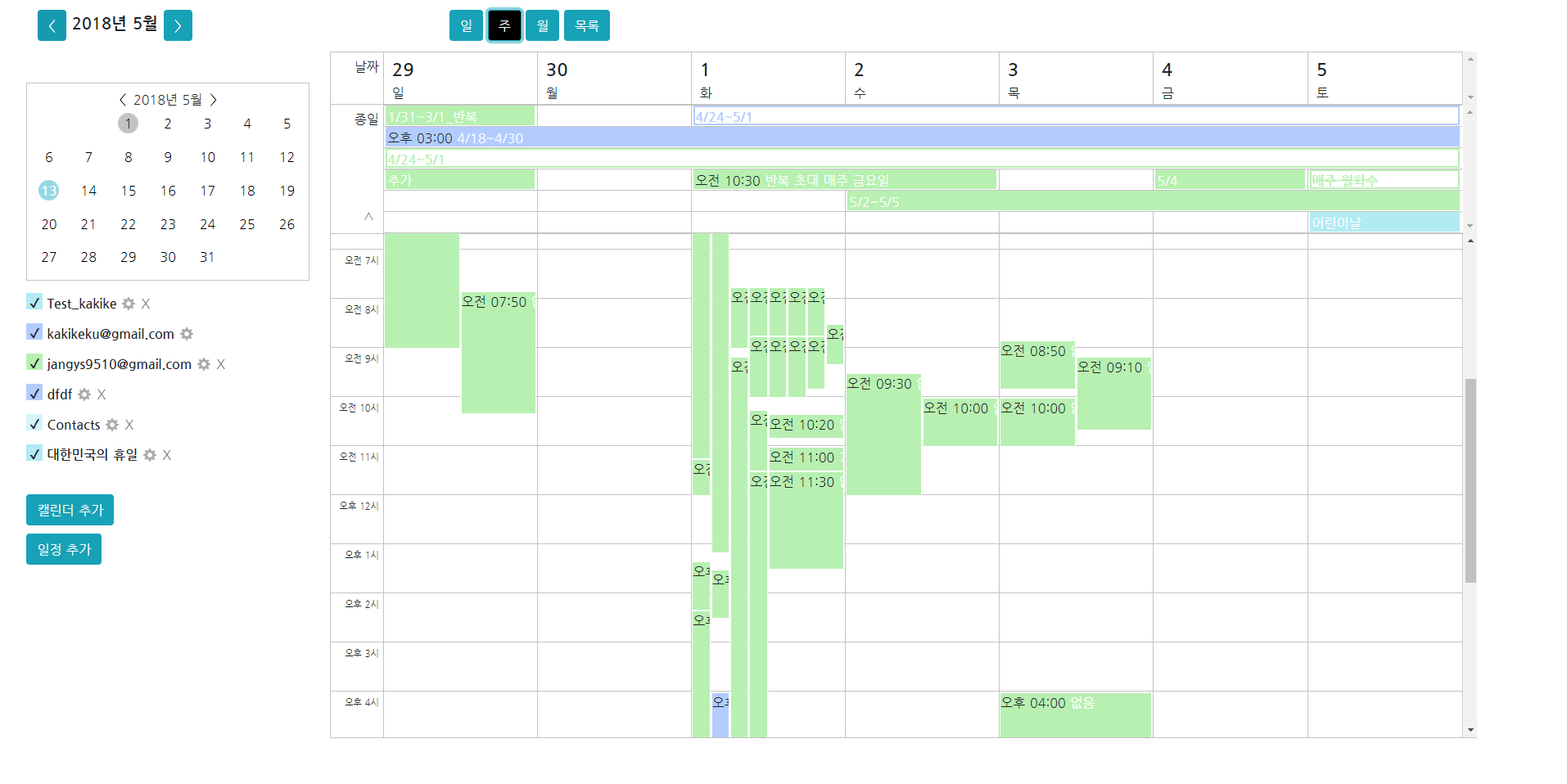
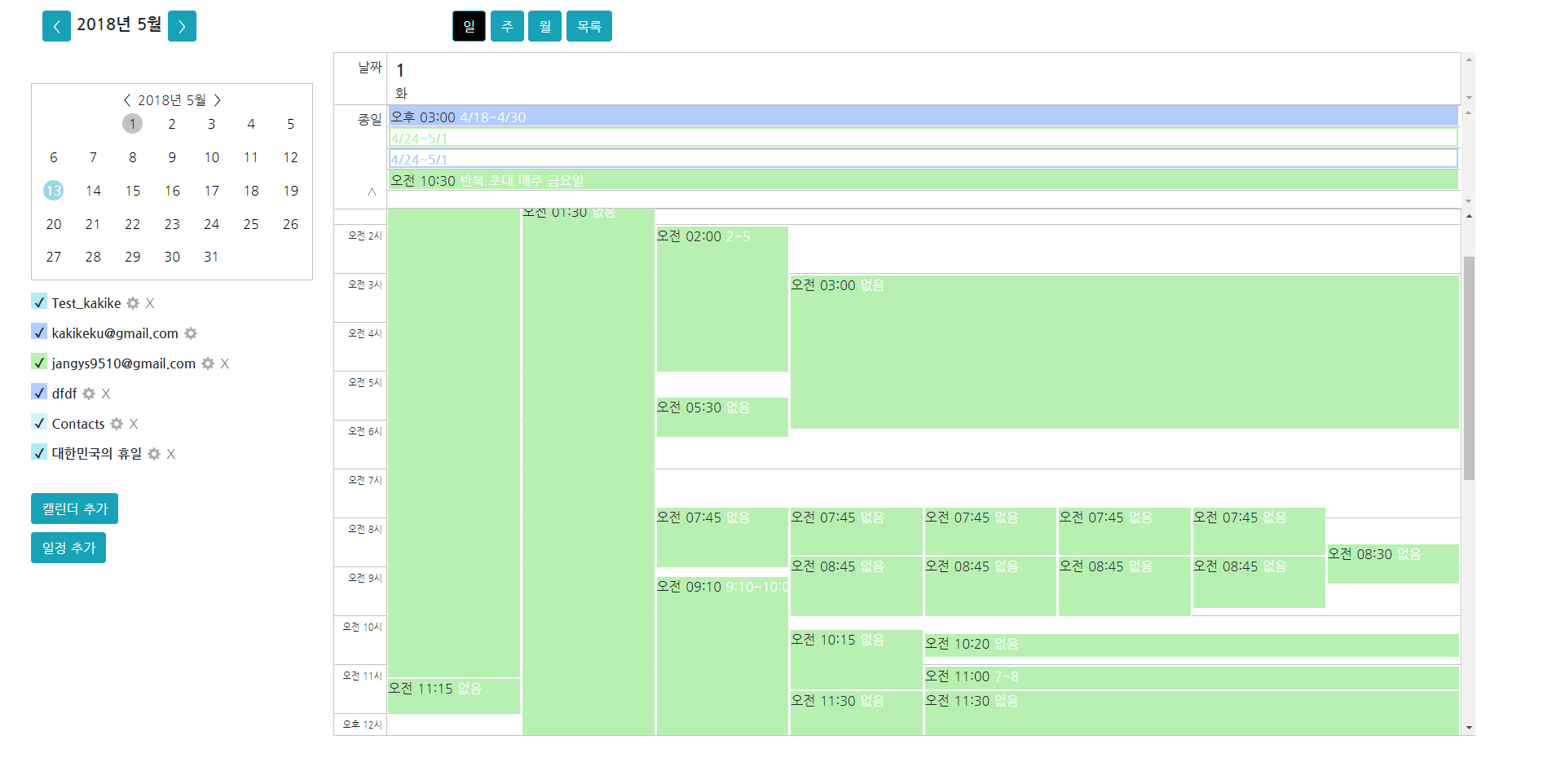


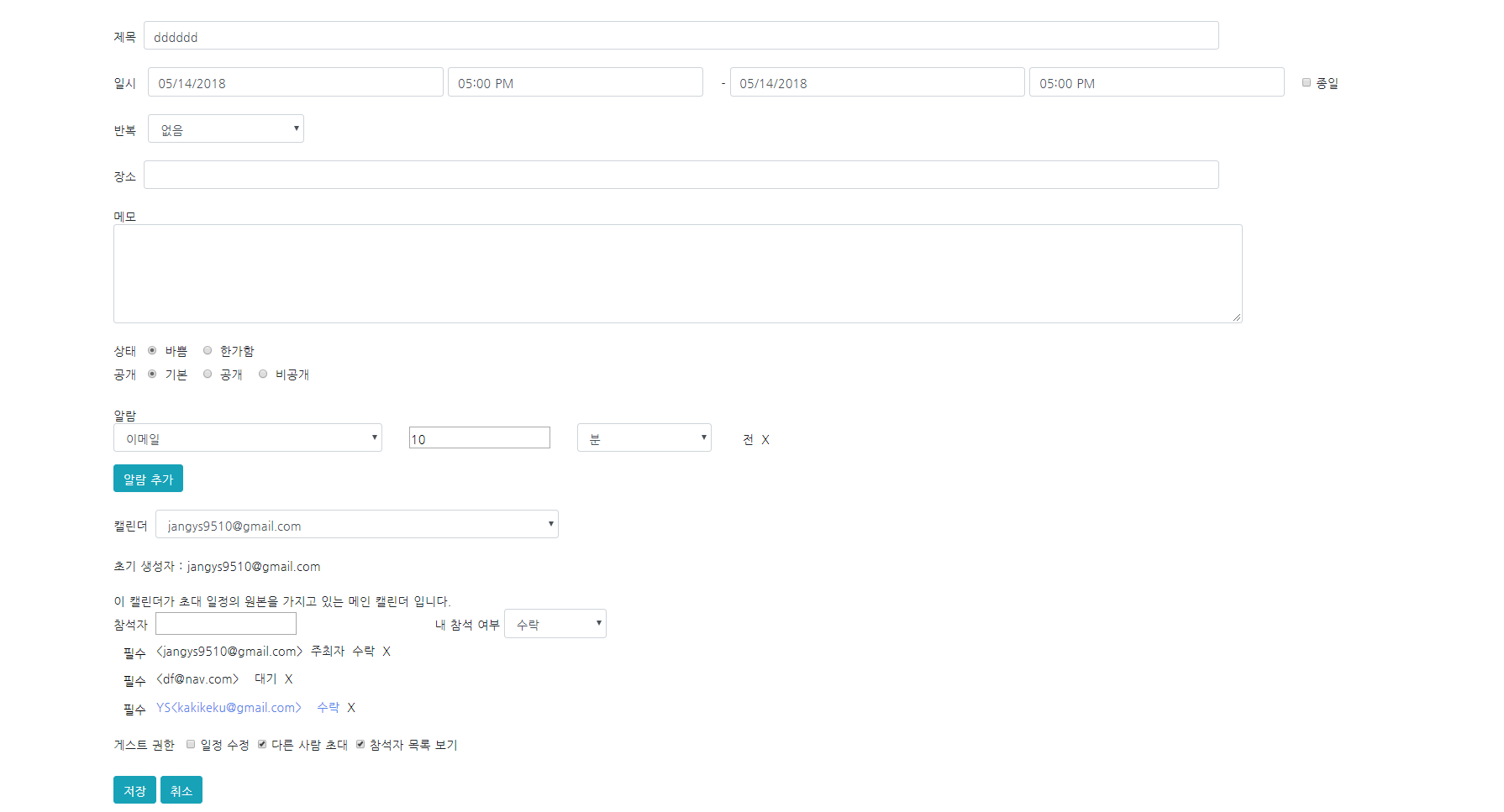


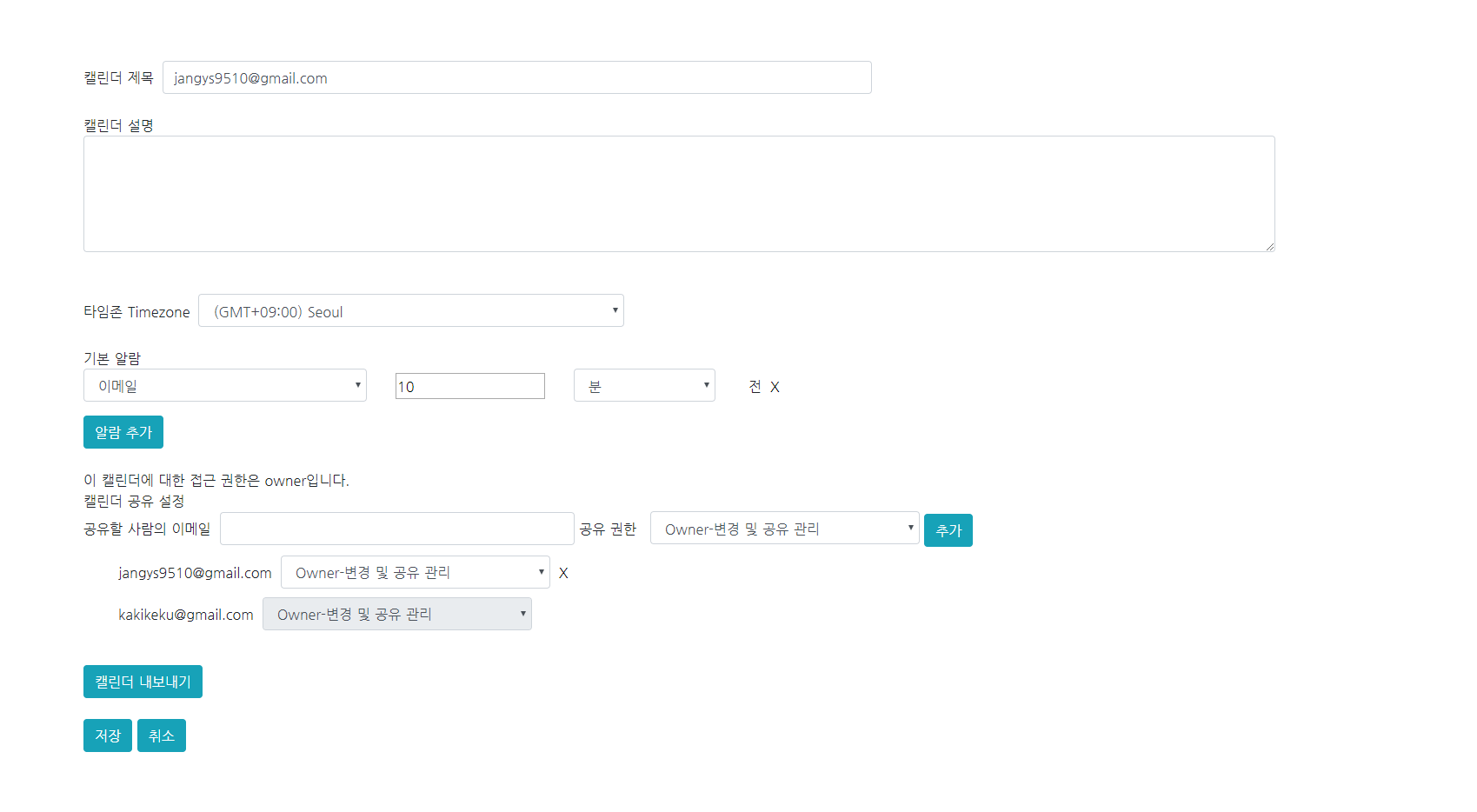
* 4/4 까지 UI



* 4/25까지 UI







* 최종 UI

2) Google Calendar API 소개

Google Calendar API를 쉽게 사용할 수 있도록 라이브러리를 제공

구글 계정 내에 있는 캘린더를 관리할 수 있게 해 줌.

OAuth 2.0 인증 이용

3) 구현한 기능

daily, weekly, monthly, list 캘린더에 일정 표시, 생성, 수정 기능

* 일정 클릭 시 요약창
  + 요약창에서 일정 삭제 가능

일정 상세보기 페이지에서 일정 생성, 수정

* 일정 초대
* 반복 일정 생성, 수정

캘린더 생성, 수정, 삭제

* 캘린더 공유 기능
* 캘린더 내보내기 기능

4) 기능 구현 방법

API 사용을 위한 사전 준비

Pom.xml에 Google Calendar API 라이브러리를 쓰기 위한 라이브러리들을 추가함.

* com.google.api-client -> Google api를 더 쉽게 사용하기 위해 제공. 인증을 쉽게 만들어 줌. 구글 로그인 기능 제공.
* org.springframework.social -> Spring Social extension. 소셜 로그인을 해줌.
* com.google.oauth-client -> OAuth 인증 방식을 이용한 구글 인증.
* com.google.apis -> Calendar API v3-rev235-1.22.0 Calendar API를 사용하기 위한 라이브러리 제공.
* org.apache.httpcomponents -> Apache HttpComponents Client HTTP 관련 low level의 java components를 생성하고 유지할 수 있도록 함.

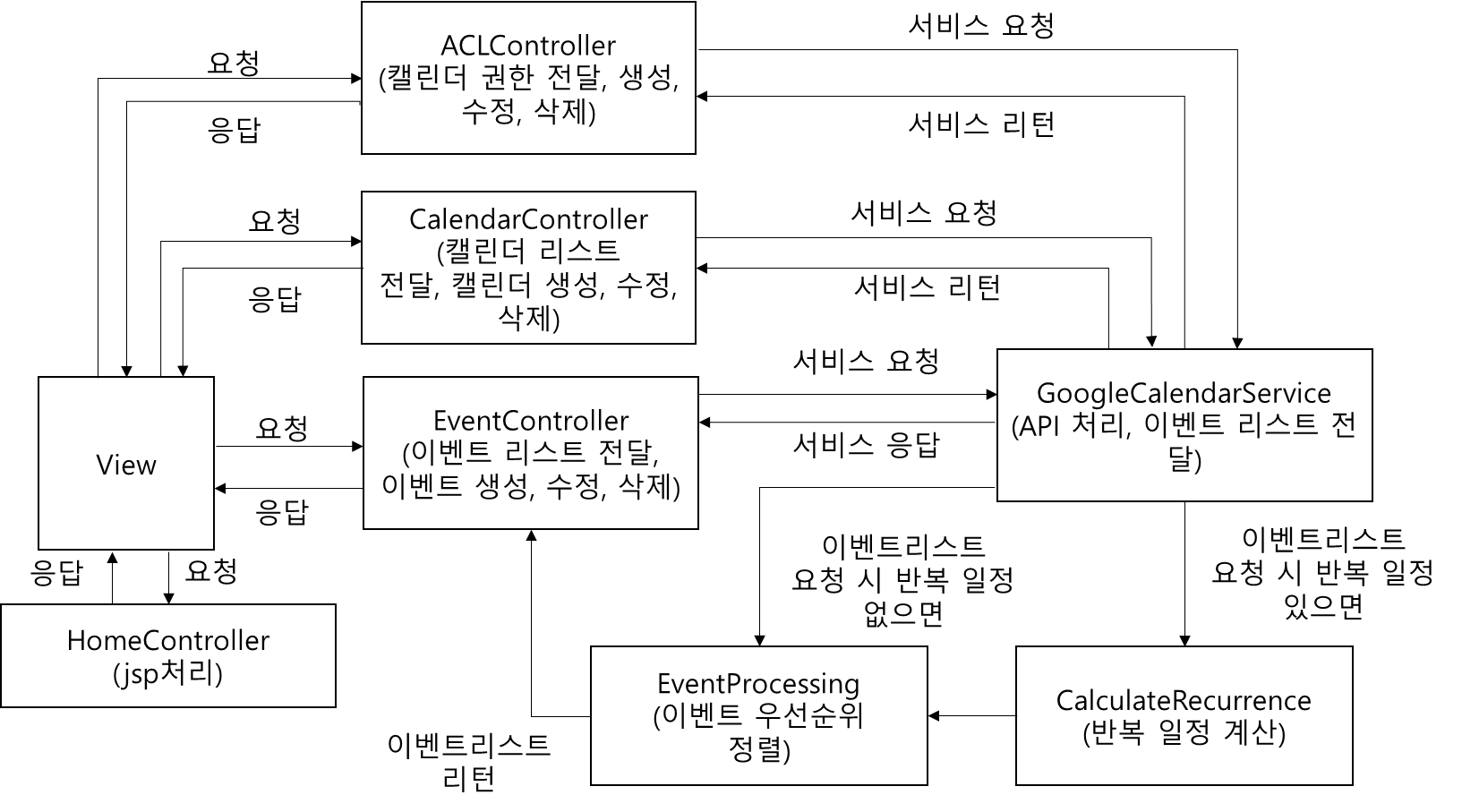
Google Cloud Platform에 접속.

* 사용자 인증 정보를 만들어서 Oauth 클라이언트 ID(특정 사용자의 구글 안에서의 정보를 가져오는 경우)를 받음.
* client\_secret.json 파일을 다운 받아 프로젝트에 저장.
* 프로젝트를 생성하고 사용할 API 설정.

캘린더 공유 설정

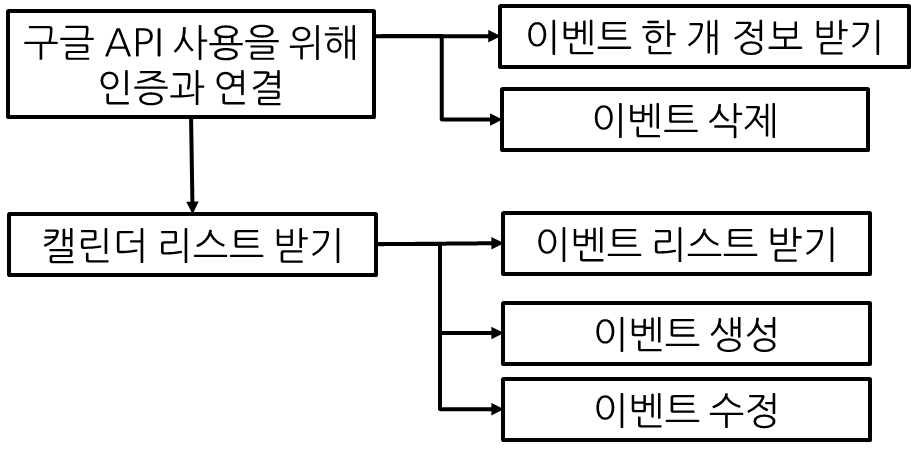
* 캘린더 공개 설정 체크

전체 클래스 구조

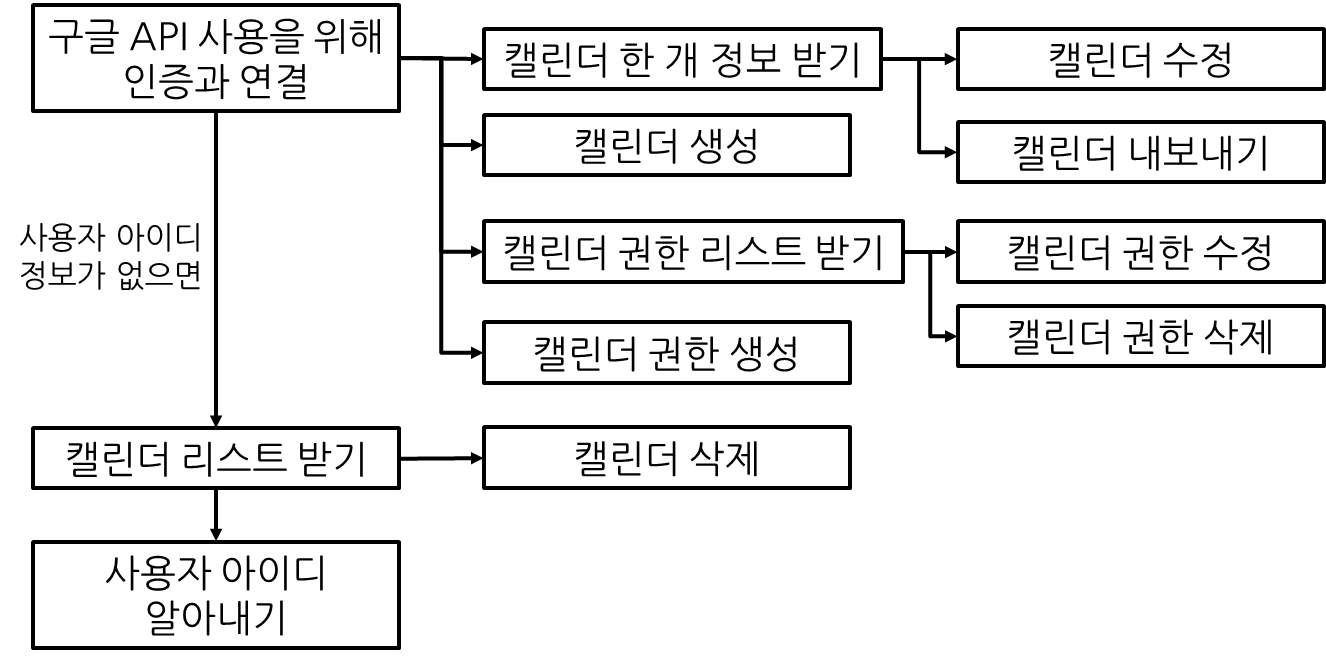


* 전체적인 동작은 View에서 정보를 요청하거나 생성, 수정, 삭제를 요청하면 그에 맞는 Controller에서 GoogleCalendarService를 통해 서비스(인증과 연결 정보가 있는 Calendar 객체)를 받아 API를 사용하여 처리한 후 응답해 줌.
* View의 jsp 파일 처리는 HomeController에서 해 줌.
* 이벤트 리스트를 요청 받으면 이벤트를 그리기 위한 편의성을 위해 정해 놓은 우선 순위에 맞게 정렬하여 이벤트를 보냄.
  + 이벤트 리스트에 반복 일정이 있으면 반복 일정을 계산해 반복 일정 리스트를 얻어낸 후 이벤트 리스트에 추가하고 정렬함.

기능 동작 순서



* 여러 캘린더들에 대해 이벤트를 받거나 생성, 수정이 가능하기 때문에 캘린더 리스트를 받고 이벤트 리스트를 받거나 생성, 수정할 수 있음.
* 이벤트 한 개 정보를 받거나 이벤트를 삭제할 때도 캘린더의 정보(캘린더 아이디)가 필요로 하지만 이벤트 한 개에 대한 캘린더 정보는 바뀌지 않기 때문에 태그에 속성 값으로 저장하여 사용함.

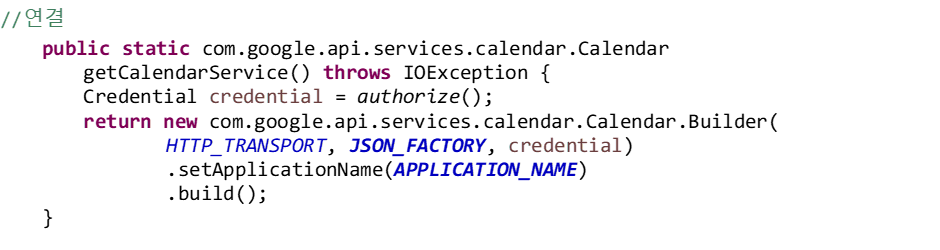


* 캘린더 권한 수정, 삭제, 캘린더 내보내기 기능을 수행하기 위해서는 사용자의 아이디를 알아야 함.
  + 사용자 아이디가 가지고 있는 캘린더 권한은 변경 불가
  + 캘린더 내보내기 시 ics 파일을 작성할 때 사용자의 아이디 정보가 필요
* 캘린더 한 개에 대한 정보를 받아서 캘린더를 수정하고 내보내기 기능 수행.
* 캘린더 권한 리스트를 받아 캘린더 권한 수정, 삭제 기능 수행
* 캘린더 삭제는 캘린더 리스트를 받아온 후에만 가능. 캘린더 리스트 체크 박스 부분에 삭제 기능을 추가했기 때문

구글 API 사용을 위한 인증



* HttpTransport는 GET, POST, PUT, DELETE만 지원해 줌.
* 프로젝트에 저장된 client\_secret.json 파일을 통해 client secret 값을 얻음.
* SCOPES = 캘린더의 읽기/쓰기 권환
* Access type = offline으로 해서 access token이 만료 되어 사용하지 못하게 되면 리소스 서버에 refresh token으로 다시 요청해 access token을 받을 수 있음.
* LocalServerReceiver에서 redirectUri 생성. Host(localhost), port는 -1로 해서 사용하지 않은 port 선정 verification code receiver
* authorize(“user”) 이 부분에서 브라우저를 열어서 승인을 받고 code를 받아서 access token을 받음. 그리고 credential 저장.

JsonFactory -> JSON 파싱 기능 제공.(input stream에서 파싱)

* builder안에 캘린더 정보 요청할 때 사용하는 url 가지고 있음.
* 기본 url: https://www.googleapis.com/calendar/v3
* 만약 캘린더 리스트에 대한 요청을 했다면 위의 url 뒤에
  + - /users/me/calendarList 붙어서 요청.

Feature List

|  |  |
| --- | --- |
| feature | method |
| 캘린더 리스트 받기 | calendarList().list() |
| 이벤트 리스트 받기 | events().list(calendarId)  (.setTimeMin(DateTime).setTimeMax(DateTime)) |
| 이벤트 생성 | events().insert(calendarId, Event) |
| 이벤트 수정 | events().update(calendarId, eventId, Event) |
| 이벤트 삭제 | events().delete(calendarId, eventId) |
| 이벤트 캘린더 옮기기 | events().move(originalCalendarId, eventId, newCalendarId) |
| 캘린더 생성 | calendars().insert(Calendar) |
| 캘린더 수정 | calendars().update(calendarId, Calendar) |
| 캘린더 삭제 | calendars().delete(calendarId) |
| 캘린더 정보 받기 | calendarList().get(calendarId) |
| 캘린더 정보 수정 | calendarList().update(calendarId, calendarListEntry) |
| 캘린더의 권한 리스트 받기 | Acl().list(calendarId) |
| 캘린더의 권한 생성 | Acl().insert(calendarId,rule) |
| 캘린더의 권한 수정 | Acl().update(calendarId, ruleId, rule) |
| 캘린더의 권한 삭제 | Acl().delete(calendarId, ruleId) |

캘린더 리스트 받기

* 사용자가 보고 싶은 캘린더를 체크할 수 있도록 사용자의 캘린더 리스트를 받아 옴.

CalendarList calendarList = service.calendarList().list().setPageToken(pageToken).execute();

* 위의 코드로 리스트를 얻어 옴.
* CalendarDTO에 맞게 데이터 전처리
* DTO를 사용하는 이유는 전송하는 데이터의 사이즈를 작게 해서 전송 시간을 줄이기 위함
  + API에 요청하고 받아오는 시간이 최소 500ms 걸림
* CalendarDTO

|  |  |  |  |
| --- | --- | --- | --- |
| 자료명 | 자료형 | 설명 | 값 |
| id | String | 사용자의 캘린더 아이디 | CalendarListEntry.getId() |
| summary | String | 사용자의 캘린더 제목 | CalendarListEntry.getSummary() |
| check | boolean | 사용자가 캘린더의 체크박스 체크 여부 | Calendar의 id를 name으로 갖는 세션의 value  (초기는 true) |
| colorId | String | 사용자의 캘린더 색 아이디 | CalendarListEntry.getColorId() |
| primary | Boolean | 캘린더가 사용자의 메인  캘린더인지에 대한 여부 | calendarListEntry.getPrimary() |
| accessRole | String | 캘린더의 Access Role(일정에 대한 접근 권한 정의) | calendarListEntry.getAccessRole() |
| defaultReminders | List<EventReminder> | 캘린더의 알림 기본 값 | caelndarListEntry.getDefaultReminders() |

* Access Role = freeBusyReader < reader < writer < owner
* writer, owner만이 일정을 쓸 수 있는 권한이 있음.
* 캘린더마다 기본 알림 값을 설정할 수 있음.
  + 일정이 캘린더의 기본 알림을 사용했으면 이 알림 값을 보여주면 됨.
  + Google Calendar에서 시간이 있는 일정에 대한 기본 알림 값만 API로 알 수 있음.
  + Google Calendar에서는 종일 일정에 대한 기본 알림 값 설정도 제공해주지만 API를 통해서는 알 수 없음.

이벤트 리스트 받기

* 사용자가 요청한 구간에 해당하는 이벤트들을 View에 전달해서 일정들을 보여주기 위함.

Events events = service.events().list(calendarId).setTimeMin(now).setTimeMax(next).execute();

* setTimeMin과 Max로 이벤트를 받아올 날짜의 범위를 지정. 그 안에 포함된 이벤트들을 모두 가져온다.
  + setTimeMin은 이벤트의 끝 날짜를 기준으로 파라미터 날짜 보다 늦은 이벤트들만 뽑는다. inclusive
  + setTimeMax는 이벤트의 시작 날짜를 기준으로 파라미터 날짜 보다 빠른 이벤트들만 뽑는다. exclusive
  + 둘다 RFC3339 형태. 밀리초 단위는 무시.
* 시작 날짜와 끝 날짜를 받을 때 일정인 경우(getDate()), 시간이 있는 일정인 경우(getDateTime())로 나눠짐.
  + 시작 날짜가 없는 경우
    - 일정이 삭제된 경우에 해당. status = cancelled
  + 반복 일정의 개별 일정인 경우
    - recurringEventId 값으로 원래 반복 일정의 eventId값을 가짐.
    - 개별 일정의 날짜가 이벤트를 받아올 때 정해진 구간에 해당하지 않아도 원래 반복 일정일 때의 날짜가 구간에 있으면 이벤트를 가져옴.
      * 이 경우 구간에 개별 일정의 날짜가 포함되는지 확인해야함.
      * 일정이 삭제된 경우에도 존재하므로 status가 cancelled인지 확인해야함.
* EventDTO 형태로 저장
* 반복 일정이 있는 경우 반복 일정 계산 후 이벤트 리스트와 합쳐 우선순위에 맞게 정렬하여 리턴
  + 없으면 이벤트 리스트만 우선순위에 맞게 정렬하여 리턴
* 여러 캘린더의 일정을 가져와야 하므로 응답 속도 개선을 위해 각 캘린더마다 쓰레드를 사용하여 일정을 동시에 받도록 함.
* EventDTO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 자료명 | 자료형 | 자료형 크기 | 설명 | 값 |
| calendarID | String |  | 사용자의 캘린더 아이디 | CalendarListEntry.getID() |
| eventID | String |  | 일정 아이디 | Event.getId() |
| summary | String |  | 일정 제목 | Event.getSummary() |
| start | long |  | 일정 시작 날짜 | Event.getStart().getDateTime().getValue() |
| startTime | Int | 5 | 일정 시작 날짜를 년, 월, 일, 시, 분으로 나눔. | start를 년, 월, 일, 시, 분으로 나눔. |
| end | long |  | 일정 끝 날짜 | Event.getEnd().getDateTime().getValue() 여기서 종일 일정인 경우는 하루 빼줘야 함. |
| endTime | Int | 5 | 일정 끝 날짜를 년, 월, 일, 시, 분으로 나눔. | end를 년, 월, 일, 시, 분으로 나눔. |
| location | String |  | 일정의 장소 | Event.getLocation() |
| description | String |  | 일정의 상세 내용 | Event.getDescription() |
| attendees | List<EventAttendee> |  | 일정의 참석자 목록 | Event.getAttendees() |
| organizer | String |  | 일정의 주최자의 이메일 주소 | Event.getOrganizer().getEmail() |
| recurrence | List<String> |  | 일정의 반복 규칙 목록 | Event.getRecurrence() |
| guestsCanSee OtherGuests | Boolean |  | 일정의 참석자가 다른 참석자 목록 볼 수 있는지에 대한 여부 | Event.getGuestsCanSeeOtherGuests() |

* start나 end를 저장할 때 startTime, endTime의 값을 함수 내에서 파싱해서 세팅.
* 종일 일정의 경우 end는 하루 뒤의 값을 가지므로 하루치 값을 뺌.
  + 종일 일정의 경우 start는 현재 timezone에 의해 그날 9시의 값을 가지게 됨.
  + 추후 이벤트들을 정렬하기 위해서 종일 일정인 경우 start의 값을 그날 오전 12시에서 1밀리초를 뺀 값을 저장.(추후 날짜 정보는 startTime으로만 다루기 때문에 값이 달라져도 상관 없음)

이벤트 하나 정보 요청

Event event = service.events().get(calendarId, eventId).execute();

* 이벤트 상세보기 페이지에서 이벤트 정보 보여주기 위해 사용.

이벤트 생성

Event event = **new** Event()

.setSummary(dto.getSummary())

.setLocation(dto.getLocation())

.setDescription(dto.getDescription())

.setStart(start)

.setEnd(end)

.setReminders(reminders)

.setAttendees(dto.getAttendees())

.setGuestsCanInviteOthers(dto.getGuestsCanInviteOthers())

.setGuestsCanModify(dto.getGuestsCanModify())

.setGuestsCanSeeOtherGuests(dto.getGuestsCanSeeOtherGuests())

.setVisibility(dto.getVisibility())

.setTransparency(dto.getTransparency());

service.events().insert(calendarId,event).execute();

* setStart(시작 날짜 설정), setEnd(끝 날짜 설정)의 파라미터 데이터 타입은 EventDateTime을 가짐.
  + EventDateTime을 생성하기 위해서는 DateTime이 필요.
  + 사용자의 인풋을 받아 Date->DateTime->EventDateTime으로 저장.
  + 종일 일정인 경우 DateTime으로 저장할 때 DateOnly값을 true로 지정.
    - end값에 하루를 더 더해줘야 함.(exclusive)
    - 종일 일정을 Date에서 DateTime으로 변경할 때는 long value값을 사용하는데 timezone 때문에 현재 +9시간이므로 시간을 0으로 세팅하지 않고 9로 세팅해야 함.
* setReminders(알림 설정)의 파라미터 데이터 타입은 Reminders
  + Reminders -> useDefault, List<EventReminders>
  + useDefault == true이면 캘린더의 기본 알림 값을 사용하겠다는 의미
    - 사용자가 default값을 사용하였는지 알기 위해 CalendarList 리소스를 이용해 캘린더가 설정한 기본 알림 값을 얻어온 후 비교
  + EventReminder -> method(popup or email), minutes(0~40320, 몇 분 전인지. 최대 4주 전까지 가능)
* setAttendees(참석자 설정)의 파라미터 데이터 타입은 List<EventAttendee>
  + EventAttendee로 참석자 세부 내용 저장.(ex. 이메일, 참석 상태 등)
* setGuestsCanInviteOthers(참석자가 다른 참석자를 초대할 지 여부, default=true), setGuestsCanModify(참석자가 일정을 수정할 수 있는지 여부, default=false), setGuestsCanSeeOtherGuests(참석자가 다른 참석자 목록 볼 수 있는지 여부, default=true)
  + Boolean 값을 파라미터로 가짐
* setVisibility(다른 사람들에게 일정 공개 여부)는 4가지 String 값을 가질 수 있음
  + default, public(all readers), private(일정 참석자만), confidential(private과 동일, 호환성)
* 나머지는 String
* Insert의 파라미터로는 이벤트를 넣을 calendarId와 이벤트 객체.

이벤트 수정

* 일정 상세보기 페이지에서 일정 수정, 요약창에서 내 응답 상태 수정, 일정 드래그하여 날짜 수정할 때 사용.

Event updatedEvent = service.events().update(calendarId, event.getId(), event).execute();

* 생성과 거의 동일. 함수만 다름.
* 이벤트 객체를 새로 생성하지 않고 calendarId, eventId로 수정할 이벤트 객체를 얻어와 수정 후 반영
* Update의 파라미터는 현재 이벤트의 calendarId, eventId, 이벤트 객체
* 반복 일정 수정
  + 이 일정만 수정
    - 이벤트 수정 시 이벤트 객체에 시작과 끝 날짜는 사용자가 현재 선택한 날짜
      * recurringEventId 추가 (반복 일정의 eventId)
      * originalStartTime 추가 (원래 반복 일정일 때의 시작 날짜)
        + 종일이었으면 종일, 시간이 있는 일정이었으면 시간이 있는 날짜의 값을 가져야 함.
    - insert를 통해 개별 일정으로 추가
  + 모든 일정 수정
    - 날짜가 변경된 경우
      * 현재 사용자가 선택한 시작과 끝 날짜가 기존의 날짜와 얼마나 차이 나는지 계산
      * 그 차이 만큼을 반복 일정 첫 번째 일정의 시작 날짜와 끝 날짜에 적용시켜 수정에 반영
    - 그 외의 경우는 이벤트 수정과 동일
  + 이 일정과 향후 일정 수정
    - 현재 반복 일정의 날짜보다 하루 전까지를 UNTIL로 하여 RRULE(반복 규칙)에 추가
      * 바뀐 RRULE을 원래 반복 일정에 반영
    - 수정된 내용을 바탕으로 새로운 이벤트를 생성
  + 반복 일정을 드래그 해서 날짜를 옮기는 경우 그 일정만 수정 됨

이벤트 삭제

service.events().delete(dto.getCalendarId(), dto.getEventId()).execute();

* 지우고자 하는 이벤트의 calendarId, eventId를 파라미터로 넣어 실행시킴.
* 반복 일정 삭제
  + 이 일정만 삭제
    - 원래 반복 일정의 RRULE에 EXDATE로 추가
    - 현재 사용자가 삭제한 일정의 날짜를 추가
  + 모든 일정 삭제
    - 원래 반복 일정 삭제
  + 이 일정과 향후 일정 삭제
    - 현재 반복 일정의 날짜보다 하루 전까지를 UNTIL로 하여 원래 반복 일정의 RRULE에 추가

이벤트 캘린더 옮기기

이벤트를 수정할 때 사용자가 원래 이벤트가 있던 캘리더에서 다른 캘린더로 옮기는 경우

service.events().move(calendarId, updateEvent.getId(), newCalendarId).execute();

* 파라미터로 원래 있던 캘린더의 calendarId, 이동하고자 하는 이벤트의 eventId, 새로운 캘린더의 calendarId

초대 일정의 수정, 삭제 여부

* ‘초대 일정 관련 스펙’과 ‘참석자 변경 여부’ 문서에 자세히 정리되어 있음.
* 일정 변경 삭제는 주최자가 할 시 모든 참석자에 반영, 참석자인 경우 본인의 캘린더에만 반영
* 주최자만 참석자를 수정, 삭제 가능.
* 참석자 응답 상태 변경은 사용자의 응답 상태만 변경 가능하도록 함.

캘린더 생성

calendar.setSummary(dto.getSummary())

.setDescription(dto.getDescription())

.setTimeZone(dto.getTimezone());

service.calendars().insert(calendar).execute();

* Calendars 리소스를 이용하면 위의 3가지 정보와 location에 대해서만 반영할 수 있음.
  + Location은 Google Calendar에서 사용하지 않는 정보라 생략함

캘린더 수정

캘린더 수정은 캘린더 상세보기 페이지에서 가능

service.calendars().update(calendarId, calendar).execute();

* 생성과 거의 동일. 함수만 다름
* 수정하고자 하는 캘린더의 객체를 받아서수정한 후 update 파라미터에 넣음. calendarId는 수정하고자 하는 캘린더의 아이디
* 캘린더 수정은 이 캘린더에 대한 사용자의 권한이 owner일 때만 가능

캘린더 삭제

캘린더 삭제는 캘린더 리스트 체크박스 옆에 X 버튼으로 삭제

service.calendars().delete(calendarId).execute();

* 파라미터는 삭제하고자 하는 캘린더의 아이디
* 캘린더 삭제는 primary calendar(Google에서 처음 계정을 생성 시 생기는 캘린더)를 제외하고 모두 가능

캘린더 정보 받기

캘린더의 기본 알림 값을 받기 위해 사용

CalendarListEntry entry = service.calendarList().get(calendarId).execute();

* 파라미터는 정보를 받고자 하는 캘린더의 아이디

캘린더 정보 수정

캘린더를 생성할 때 기본 알림 값을 설정할 때와 캘린더 수정 시 기본 알림 값 수정할 때 사용

service.calendarList().update(calendarId, calendarListEntry).execute();

* 파라미터는 수정하고자 하는 캘린더의 아이디, 수정한 정보가 담겨져 있는 calendarListEntry 객체
* calendarListEntry안에 캘린더의 summary, timezone, description 정보가 있지만 생성, 수정 불가능

캘린더 권한 리스트 받기

캘린더 상세보기 페이지에서 이 캘린더에 대해 어떤 사용자들이 권한을 가지고 있는지 보여줄 때 사용

Acl acl = service.acl().list(calendarId).execute();

* 파라미터는 리스트를 받고자 하는 캘린더의 아이디
* Acl.getItems()를 통해 List<AclRule>를 얻을 수 있음. 이 리스트로 사용자들의 권한 확인
* 해당 캘린더에 대한 사용자의 권한이 Owner일 때만 화면에 보여줌. Owner가 아닌 경우는 권한을 생성, 수정,삭제가 불가능하기 때문

캘린더 권한 생성

캘린더 상세보기 페이지에서 새로운 사용자에 대한 권한을 생성할 때 사용

Scope scope = new Scope().setType("user").setValue(dto.getValue());

AclRule input = new AclRule().setRole(dto.getRole()).setScope(scope);

acl = service.acl().insert(dto.getCalendarId(), input).execute();

* Scope의 setType으로 사용자가 어떤 타입인지 결정.
  + user(사용자 한 명), group, domain
  + default 로 정하면 권한으로 owner이나 writer을 가질 수 없음.
* Scope의 value 값으로 새로운 사용자의 이메일 주소를 넣어 줌.
* AclRule 객체를 통해 setRole로 ACL 권한 설정
  + owner, writer, reader, freeBusyReader
* insert의 파라미터는 권한을 생성하고자 하는 캘린더의 아이디, AclRule 객체
* 생성 후 반환되는 AclRule 객체를 View로 보내 View에 추가된 ACL을 표시하도록 함.

캘린더 권한 수정

캘린더 상세보기 페이지에서 등록된 권한들을 수정할 때 사용

service.acl().update(calendarId, roleId, acl).execute();

* 생성과 거의 동일. 파라미터로는 수정하고자 하는 캘린더의 아이디, 수정하는 권한에 대한 아이디, AclRule 객체
* 본인 권한에 대해서는 수정이 불가능함.

캘린더 권한 삭제

캘린더 상세보기 페이지에서 등록된 권한들을 삭제할 때 사용

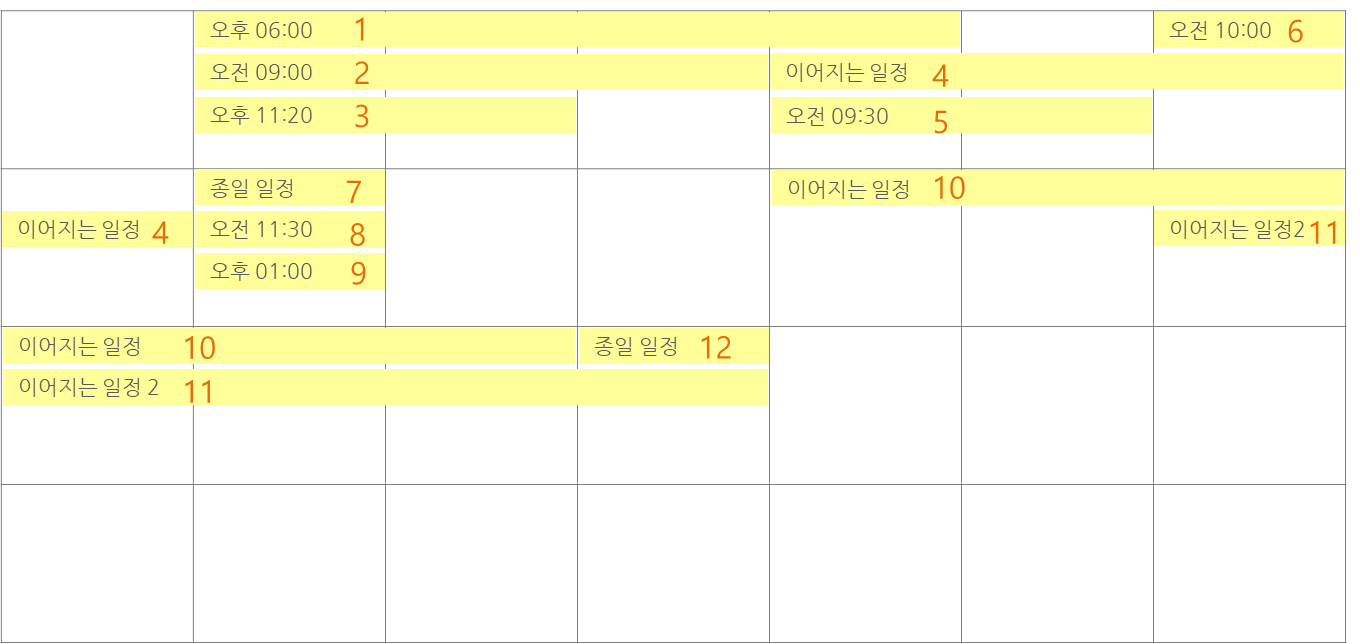
service.acl().delete(calendarId, roleId).execute();

* 파라미터로는 권한을 삭제하고자 하는 캘린더의 아이디, 삭제하는 권한에 대한 아이디
* 본인 권한에 대한 삭제는 불가능함.

5) 우선순위 정렬

* 각 캘린더에서 받아온 이벤트들을 시작 날짜가 빠른 순으로 정렬.
  + 시작 날짜가 같은 경우에는 끝 날짜가 늦은 순으로 정렬
* 현재 사용자가 보고 있는 날짜보다 시작 날짜가 전인 이벤트들을 모아
  + 끝 날짜가 늦은 순으로 정렬
* 사용자가 보고 있는 날짜보다 시작 날짜가 같거나 후인 경우
  + 시작 날짜가 같은 경우
    - 끝 날짜도 같으면 시작 시간이 빠른 순으로 정렬(종일 > 시간)
      * 시작 시간이 일치하면 끝 시간이 늦은 순으로 정렬
    - 끝 날짜가 같지 않은 경우
      * 끝 날짜가 늦은 순으로 정렬
* 이렇게 정렬을 하면 그릴 때 끝 날짜가 늦은 순으로 정렬되어 칸을 많이 차지하는 순서대로 시작 날짜 순서에 맞게 들어감. 미리 자리를 차지하게 만들 수 있음.

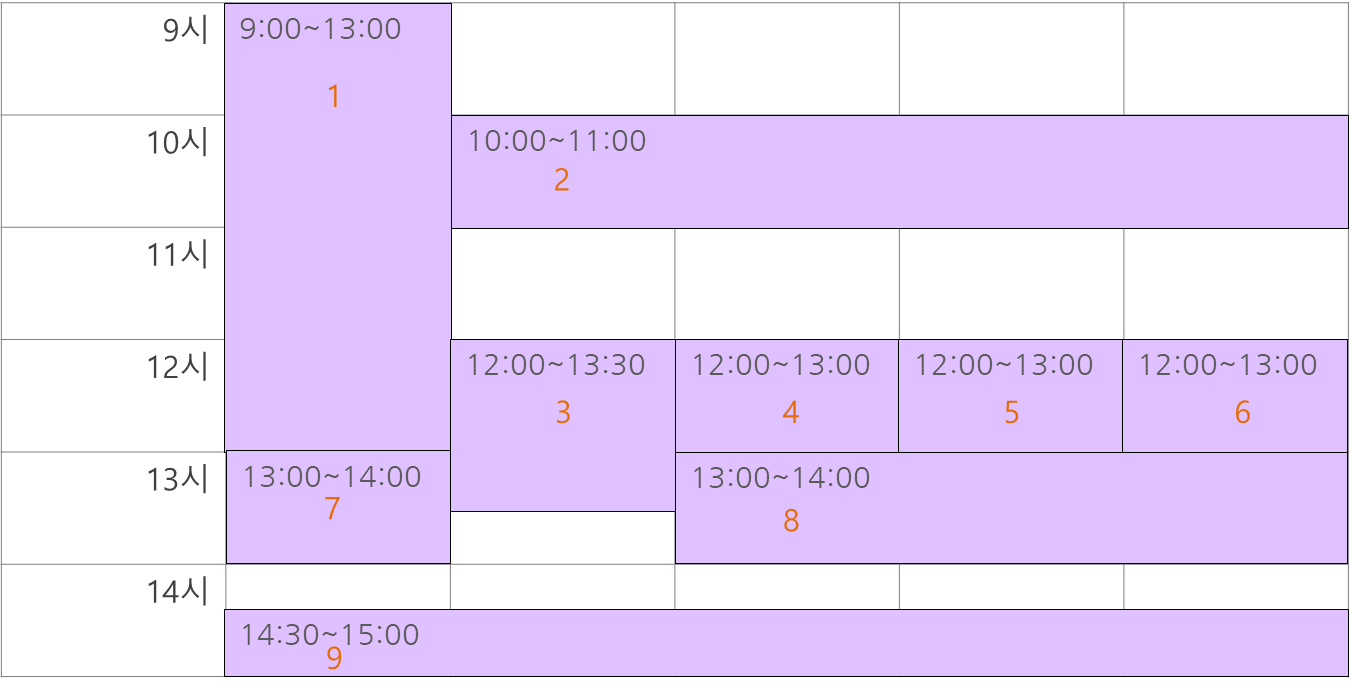
6) Monthly Calendar 일정 그리는 방법



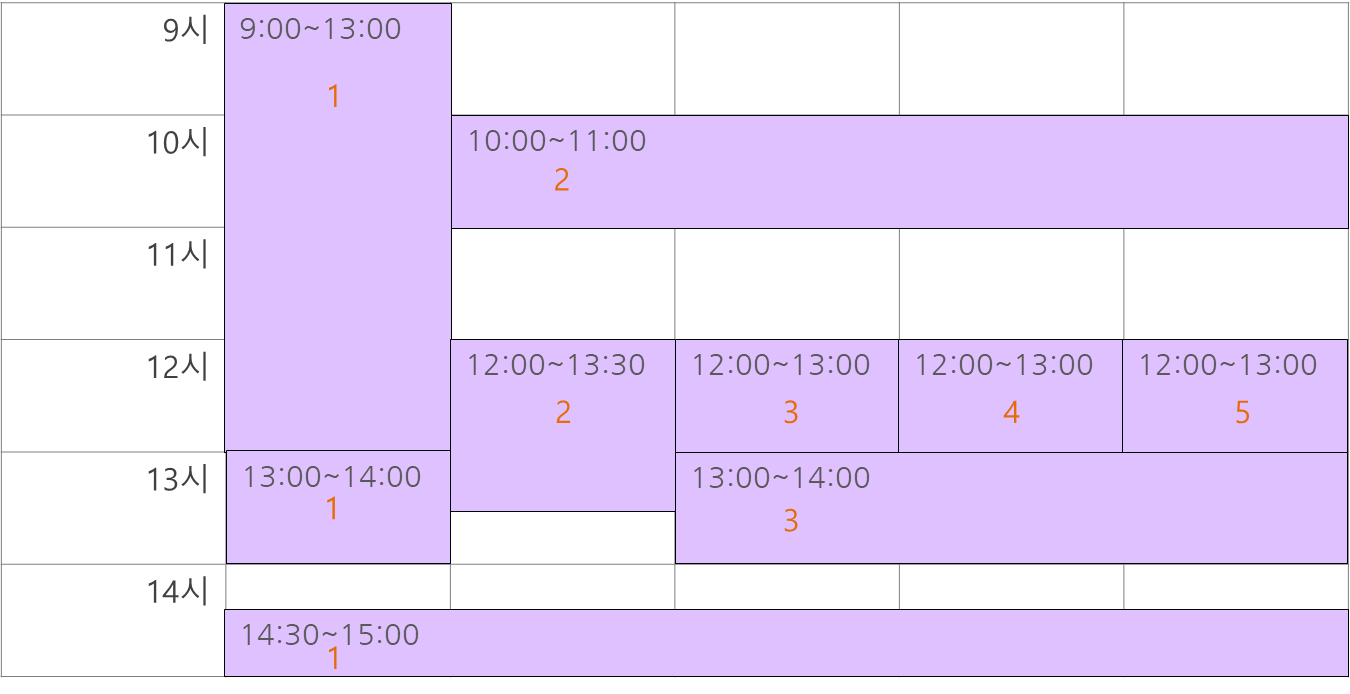
* 사진의 빨간색 숫자 순서대로 일정이 그려 짐
* 일정이 들어갈 날짜 칸에 어느 위치에 들어갈 수 있는지 확인하고 들어갈 수 있는 제일 처음 위치로 일정이 들어감.
* 4번 일정인 경우 이미 위에 일정이 있으므로 아래 칸으로 내려 감
  + 4번 일정이 그려질 당시에는 4번 일정 끝 부분 칸에 아무 일정도 없으므로 들어감
  + 만약 해당 줄에 이미 일정 있다면 그 아래 칸으로 내려 감
  + 일정이 들어갈 수 있는 줄을 초과하면 더보기로 추가 됨.
* 6번 일정인 경우 맨 위 칸이 비어 있으므로 첫번째 칸에 들어감.
* 7, 8, 9번 일정을 보면 종일 일정 > 시간이 있는 일정 순으로 배치되는 것을 볼 수 있음.
  + 우선순위 정렬 시 종일 일정이 앞으로 나오게 했기 때문

7) Daily, Weekly Calendar 일정 그리는 방법

일정들의 여러 케이스를 포괄적이게 커버할 수 있도록 함.



* 일정 그려지는 순서



* index
* 최대 칸 수를 셀 때 index값(시작 시간 기준으로 왼쪽에서부터 몇 번째 인지) 정함
  + longerIndex에 현재 index값이 설정된 일정들 중 index값을 설정할 일정보다 긴 일정의 index를 저장
  + index 값을 0~지금까지 최대 칸 수 만큼 하나씩 증가해가면서​
    - 이 index 값을 가지는 일정들 중 마지막 일정을 뽑아​
    - 그 일정의 bottom이 index값을 설정할 일정의 top보다 크거나 같은 경우 기록(beforeIndex 값 갱신)​
      * 같은 경우는 index-1을 기록(같은 경우는 바로 아래에 넣을 수 있다는 것을 의미하기 때문)
* index = beforeIndex + 1
* 만약 index=0인 일정 중 마지막 일정의 bottom이 top보다 작으면
* 넣을 일정이 맨 첫번째 자리에 들어갈 수 있는 것을 의미(beforeIndex == -1)
* index값을 설정할 일정의 index가 0이 아닌 경우
* longerIndex를 조사해 현재 들어갈 index와 같은 index가 있는지 조사
* 있으면 그 다음 index로 설정
* 최종적으로 넣을 일정의 index값을 속성으로 저장
* 현재 index값을 보고 최대 칸 수 값보다 크면 최대 칸 수 갱신
* index값을 이용해서 일정의 left, width 값 설정
* longerIndex 구하기(전체 일정 중 자신보다 긴 일정의 index 저장)
* 일정의 정해진 index값 가져오기
* 일정의 바로 왼쪽에 위치한 일정 구하기
* index-1을 가지는 일정들을 구해 현재 일정의 top보다 bottom이 크면 옆에 있는 것을 의미
* 있으면 왼쪽 일정의 width를 (100/최대 칸 수)%로 설정​​​
* 일정이 들어갈 수 있는 최대 너비 계산
* longerIndex가 존재하면
* longerIndex- 현재 index한 값 중 양수이고 제일 작은 값을 찾아 그 값을 (100/최대 칸 수)에 곱해 width를 구함
* 아닌 경우 width = (최대 칸수 - 현재 index값) \* (100/최대 칸 수) %
* 일정의 너비 적용
* 일정의 left(가로 위치)는 index\*(100/최대 칸 수)%

8) 캘린더 내보내기 기능 구현 방법(Google Calendar -> ical)

Ical4j 라이브러리를 이용해서 calendar 객체를 생성하고 이벤트, 알림 들을 넣어 ics 파일을 만듦.

* 캘린더의 전체 일정을 Google Calendar API를 통해 받아와서 ical 형태로 변환
  + VCALENDAR property 세팅
  + PRODID = -//Google Inc//Google Calendar 70.9054//EN

VERSION:2.0

CALSCALE:GREGORIAN

X-WR-CALNAME:jangys9510@gmail.com

X-WR-TIMEZONE:Asia/Seoul

* VTIMEZONE은 Calendar의 timezone으로 세팅
* VEVENT 세팅
  + DTSTART, DTEND, SUMMARY, CREATED, LAST-MODIFIED, SEQUENCE, ORGANIZER, UID, TRANSP, STATUS, CLASS, DESCRIPTION, LOCATION, ATTACH, ATTENDEE, RRULE, EXDATE, RECURRENCE-ID 세팅
  + Google Calendar API의 Recurrence에는 ‘RRULE:’ 이 포함되어 있어 이 뒤부터 RRULE에 넣음
    - Recurrence에 EXDATE도 같이 있음.
  + Google Calendar API의 Visibility는 ical에서의 CLASS에 해당.
    - Ical4j에서는 Clazz 객체로 제공
* VALARM 세팅
  + Google Calendar API의 알림 종류
    - popup(alert으로 알려줌.)
      * ical-> ACTION:DISPLAY
    - email
      * ical-> ACTION:EMAIL
      * email 보낼 때 무조건 primay(사용자 본인)의 아이디로 보냄. 캘린더에 상관 없음.
  + TRIGGER, ACTION, DESCRIPTION, SUMMARY, ATTENDEE 세팅
    - DESCRIPTION:This is an event reminder

SUMMARY:Alarm notification

* + - ATTENDEE는 사용자의 아이디

9) Google Calendar API의 장단점

* 장점
  + Java Library로 제공해주기 때문에 인증이나 연결, 데이터를 받아오는 과정이 수월함.
    - 대부분 코드 한 줄이면 실행이 됨.
  + Google Calendar의 거의 모든 기능을 사용할 수 있음.
  + DB를 설계하지 않고 사용할 수 있음.
  + 처음 캘린더를 개발하는 사람들도 쉽게 개발할 수 있게 해 줌.
* 단점
  + 코드 한 줄을 실행하기 위해 세팅 해야 하는 과정이 복잡한 경우가 있음
    - Ex. 일정 생성을 위해 View에서 받아온 값을 Google Calendar API에 맞게 수정해야하는 번거로움이 있음.
  + Reference에 설명되지 않은 부분은 직접 여러 시도 끝에 찾아야 함.
    - Ex. 반복 일정의 개별 일정을 설정하는 부분에 대한 설명이 없어서 여러 방법을 시도한 끝에 찾을 수 있었음
    - Google Calendar 내에서 정해 놓은 규칙에 대한 설명이 없어서 이 또한 여러 실험을 통해 찾아야 함.
      * Google Calendar에서는 삭제된 일정을 DB에서 삭제하지 않고 남겨 두는 경우가 있음.
      * 이럴 때 이벤트의 시작 날짜 값은 없지만 status가 cancelled인 것을 보고 삭제된 일정임을 확인해야 함.
  + 응답 속도가 느림. 최소 500ms