알린투(alarm+calendar+todo)

깔끔한 스타일로 작성된 일정 및 업무 관리 웹입니다.
자주 사용하는 일정 관리 어플인 iOS 캘린더, Todo Mate 어플의 단점을 파악하여 장점을 모았습니다.

<iOS 캘린더>
- '미리 알림' 어플과 빠른 전환이 힘들다.
- 여러 기능이 내장되어 UI 의 명확한 전달이 되지 않는다.
- 컴퓨터에서 알림 박스를 통해 일정 리마인드를 받고 싶다.

<To do Mate>
- 할 일 타입을 손쉽게 정의하고 제거하고 싶다.
- '일정'은 해당 시간이 지나면 완료되지만, '할 일'은 완료해야 한다. 이 특성대로 서비스에서 분리해서 확인하고 싶다.

<알린투>
- 필요한 기능만 있기 때문에 일정 관리의 UI 가 명확하다.
- "무슨 일을 완료한 지"는 사실 중요하지 않다. 무슨 일을 해야할 지를 우선 순위에 따라 명확하게 관리할 수 있다.
- 웹에 접속하여 창을 접는다면 오른쪽 하단의 리마인드 메시지를 받고, 웹을 닫으면 리마인드 메시지를 받지 않는다.
- 새벽 4 시가 되면 하루 할 일을 정리하게 하며 밤을 새지 않게끔 유도한다.

기능 소개

- 로그인과 회원 등록으로 캘린더/일정을 유저별 접근 가능
- 일정을 등록할 때 '알림 받기' 여부를 결정 가능
- 알림은 일정 시작 10 분전 우측 하단에 메시지 팝업으로 확인 가능
- 일정의 depth 를 지정하여 일정 사이에 진행되는 세부 일정을 관리
- 일정을 색깔별로 분류하여 시각화
- 할일은 카테고리별로 추가할 수 있으며 클릭으로 완료 처리
- 완료된 일정은 하루의 끝(새벽 4 시)에 삭제되며, 미완료 시 '대기' 풀로 넘어감
- '대기' 풀의 일정은 다음날 다시 할일 사이에 구성할 수 있음
- 일정 순서의 재구성을 통해 우선순위를 지정.
- 일정이 바뀌기 전에 정리하는 시간을 주고자 10 분 전에 알림을 보냄.

사용된 라이브러리

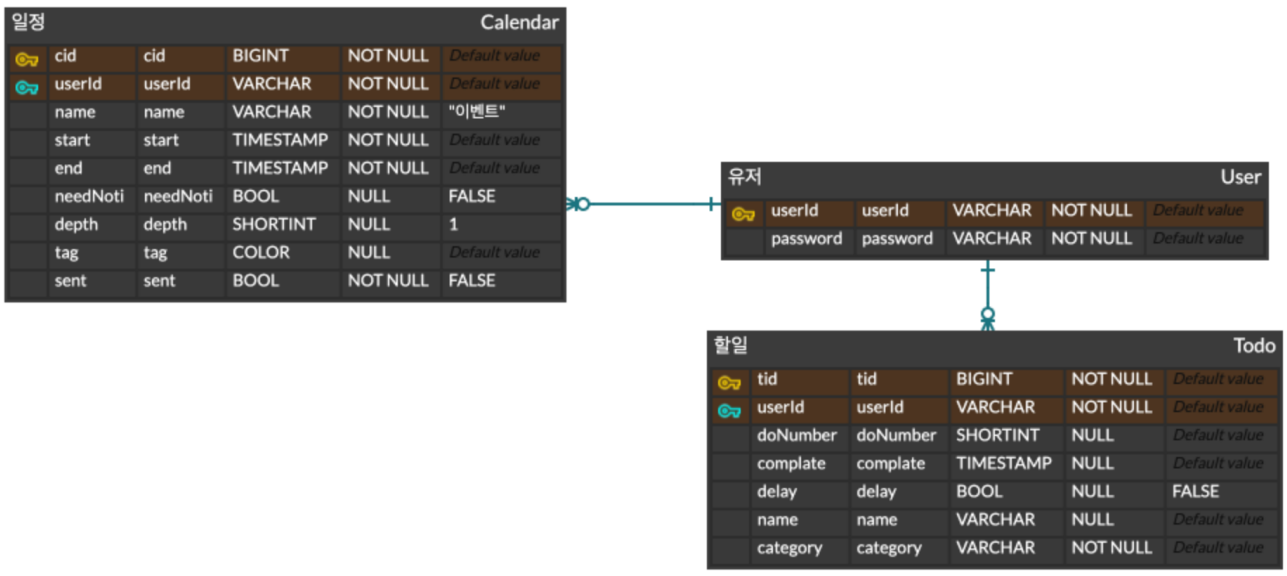
BackEnd

1. Spring Web
   기본적인 CRUD 기능을 제공하기 위해 RESTful API 를 구현했습니다. 이를 통해 클라이언트가 할 일과 일정을 추가, 수정, 삭제, 조회할 수 있도록 했습니다.
2. Server-Sent Events (SSE)
   일정이 가까워지면 리마인더를 제공하기 위해 SSE 를 사용했습니다. 클라이언트와 서버 간의 지속적인 연결을 유지하여 사용자에게 알림을 즉시 전달할 수 있도록 했습니다.
3. Spring JPA
   MySQL 데이터베이스와의 상호작용을 단순화하기 위해 사용했습니다. JPA 의 ORM 기능을 활용하여 생산성을 높이고 유지보수를 용이하게 했습니다.
4. Spring Batch
   한 일을 정리, 알림 리마인더를 위해 Spring Batch 를 사용했습니다. 특정 시간마다 메소드를 실행하고, 1 분마다 현재 연결된 유저에게 리마인드 해야하는 알림이 있는지 확인하는 역할을 합니다.

FrontEnd

1. Axios
2. Styled-components
3. React-Calendar

데이터베이스 설계



API 명세

HTTP 메서드	API 이름	경로	요청 파라미터 (바디) {경로}	응답 바디 {객체} [배열]
POST	로그인	/user/login	(id, pw)	-
POST	일정 추가하기	/schedule	(userId schedule(name,start,end,needNoti,depth,tag))	id
GET	한달 일정 조회하기	/schedule/month	date, userId	[{name,start,end,tag}]
GET	일정 상세 조회하기	/schedule/day	date, userId	[{cid,name,start,end,needNoti,depth,tag}]
PUT	일정 수정하기	/schedule/{id}	{id}, (userId schedule(name,start,end,needNoti,depth,tag))	id
DELETE	일정 삭제하기	/schedule/{id}	{id}	-
POST	할일 추가하기	/todo	userId, todo(priority, name, category)	id
PATCH	할일 완료하기	/todo/{id}/complete/t	{id}	-
PATCH	할일 완료취소하기	/todo/{id}/complete/f	{id}	-
PATCH	할일 미루기	/todo/{id}/delay	{id}	-
DELETE	할일 지우기	/todo/{id}	{id}	-
PUT	할일 수정하기	/todo/{id}	{id}, userId, todo(priority, name, category)	id
GET	할일 조회하기	/todo	userId	[{tid,priority,delay,name,category}]
GET	다한 일 조회하기	/todo/complete	userId	[{tid,priority,delay,name,category}]
GET	못한 일 조회하기	/todo/delay	userId	[{tid,priority,delay,name,category}]
PATCH	할일 순서 바꾸기	/todo/{id}/order	[{tid,priority}]	-
GET (SSE)	알림 연결하기	/sse/connect	userId	
(batch)	일정 10 분 전 알림			

(batch)	미완료 할 일 안내			
(batch)	매일 할일 정리하기.			

소스 확인 메뉴얼

의존성

- docker-compose : https://www.docker.com/ docker-desktop 다운로드 혹은
- jq : https://jqlang.github.io/jq/download/ => Curl 응답값 파싱 후 활용 위해 필요

공통

깃허브를 클론하여 프로젝트를 로컬에 불러온 후, 프로젝트 루트 경로에서 docker-compose up

[Mac OS (ARM64) 환경에서 빌드]

```
sudo docker-compose up
```

명령어 실행 후 약 2 분 뒤 test_shell_script.sh 파일의 경로에서

```
./test_shell_script.sh
```

를 통해 API 가 전부 동작할 수 있는 상태인지 확인할 수 있습니다. (jq 설치 필요)

최초 compose up 할 때 spring-app 이 오류로 종료되는 경우가 있고 두 번 실행했을 때 문제가 없었습니다.

또한 0.0.0.0:3000 으로 접속했을 때 프론트엔드가 일부 구현되어있습니다.

예제 코드가 있으면 금방 학습할 수 있기에 학습 용도로 chatGPT 를 이용하여 구상했던 프로젝트의 프론트엔드를 구현했습니다만, 직접 수행한 과제가 아니기에 참고용도로만 넣었습니다.