0222 보고자료

- Springboot 사용 → 교수님께 컨펌
- 웹, 앱 서버 구축 관련
  - react
  - swift
  - springboot → AWS EC2/RDS

- 그룹에서 공유되는 일정은 실시간으로 DB에 요청해서 받아오는 방식으로 진행
  - 근데 이러면 ,, 10명이 최대라고 해도 한명씩 DB에 요청해서 타임테이블에 올리면 시간이 너무 오래 걸리지 않을까?
  - 서버 쪽에서 취합해서 객체 형태로 한번에 보내주고 나중에 바뀌게되어서 사용자의 요청으로 리프레시 해야한다면 그 때 DB에서 다시 요청해서 받아오는 방식이 낫지 않을까 합니다 ,,,
  - 이렇게 하면 클라이언트 쪽에서도 각 사용자의 일정을 받아서 객체의 배열 형태로 갖고 있어서 나중에도 예전에 공유했던 일정을 다시 볼 수 있게 할 수 있음
  - 실시간으로 받아와서 일정을 공유하게 되면 전에 공유했던 일정들을 저장해서 목록 형태로 보여줄 수가 없음!!

- 반복 기능을 서버에서 처리할 것이 아니라 클라이언트 쪽에서 사용자의 버튼 선택에 따라서 로직을 구현해 추가될 일정들을 서버로 보내줘서 서버는 DB에 그대로 집어넣는 게 더 깔끔해질 듯.
- 그룹 사용자들의 일정을 DB에서 뽑아낸다음 뷰로 만들어서 새로운 테이블처럼 만들어 → 다음 그대로 데이터 전달해주면 되지 않을까?
  → 이걸 DB에서 저장하고 있진 않고 보내준 데이터를 클라이언트 쪽에서 저장해둬야 목록 형태로 이전에 공유했던 일정들도 볼 수 있음!

- 서버 쪽 예상 흐름도
  - Springboot에서 JDBC로 DB 서버와 커넥션
    - 쿼리 날리고 결과 받아오는 건 Hibernate 구현체 JPA 이용
  - AWS에서는 아마존 RDS 또는 **EC2** 사용
  - 클라이언트 - 서버 간에는 REST API로 데이터 주고받기
  - 클라이언트 인증은 Firebase Authentication 사용이 아닌, 스프링 시큐리티를 사용할 예정 (JWT, JSON Web Token 이용)
  - Springboot에서 REST API와 레이어드 아키텍처 패턴을 사용해서 Controller, Service, Persistence로 레이어를 나눠 통신할 예정
  - SQL과 연결할 것이므로 테이블과 테이블 항목에 관련된 클래스들도 따로 작성해서 모듈화 → Model, Entity, DTO 등등 ..
  - **결국, 클라이언트에서 데이터를 보내주면 Springboot 서버로 받아서 AWS EC2/RDS로 DB에 저장**

- Springboot를 선택하는 이유
  - Springboot에 React, Swift로 연결하는게 더욱 수월하고 레퍼런스도 많아서 더욱 이점이 많다고 생각함.
  - 강의자료나 공부할 만한 자료도 Spring이 더 많고 node.js와 웹 서비스를 연결하는 레퍼런스는 많지만 iOS와 연동하는 사례가 많지 않아서 프로젝트에 적용하는 데 어려움이 있음.
  - 서버를 띄우는 과정 자체가 node.js나 next.js 보다는 Springboot가 더욱 수월할 것이라고 판단됨. → 검색을 통해 알아본 정보
  - 국내 대다수 IT 기업에서도 next.js 보다는 Spring을 더 많이 쓰는 것으로 보여 취업을 위해서라도 더 좋아보임. 또한 4학년 1학기 수업에서도 Spring을 다루므로 조금 더 도움이 될 수 있을 거라고 생각함.