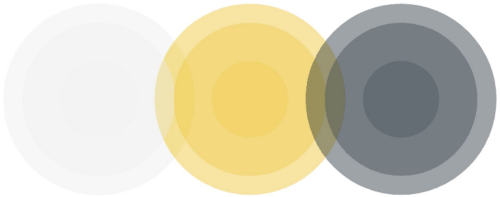


IT JOBS

윤사라

https://github.com/yunsara00/IT_JOBS

개발 목표

IT 관련 기업 대상 구인구직 사이트

IT JOBS

- IT 관련 최신 뉴스
- 실시간 채팅 가능
- 기업 필터링을 위해 멤버십 기능 추가
- 기업페이지와 사용자 페이지 구분
- 직관적이고 심플한 UI

IT JOBS

세부 개발 내용

윤사라

- 아이디어 회의
- 기능 기획
- **전체 DB 설계** 및 구축
- 멀티룸, 멀티 채팅이 가능한 **오픈 채팅** 서버 및 뷰 구현
- **이력서 CRUD**
- **파일 업로드**
- 화면 구현

IT JOBS

개발 환경

FRONT END

SERVER

DATABASE

API

IT JOBS

데이터베이스 설계

IT JOBS

이력서 (Insert)

학력

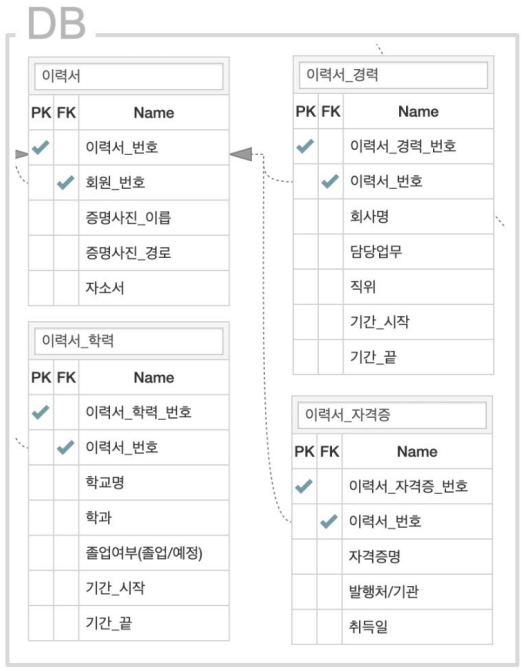
학교명	학과명	졸업여부	재학 기간	
		졸업	2020 년 1 월 ~ 2020 년 1 월	지우기
		졸업	2020 년 1 월 ~ 2020 년 1 월	지우기
		졸업	2020 년 1 월 ~ 2020 년 1 월	지우기

학력추가

경력

회사명	담당업무	직급	근무 기간	
		사원	2020 년 1 월 ~ 2020 년 1 월	지우기

경력추가



- 사용자가 원하는 만큼의 데이터를 받을 수 있도록 학력, 경력, 자격증 테이블을 나누어서 설계하여 각각의 폼으로 **구조화**함
- '추가' 버튼 클릭시 **JQuery**의 **append**를 사용하여 <tr>요소를 추가함

IT JOBS

이력서 (File Upload)

개인정보

이력서 제목	첫번째 이력서
이름	user2
주소	없음
연락처	020000000
이메일	user@user.com

증명사진 수정 : 파일 선택 img03

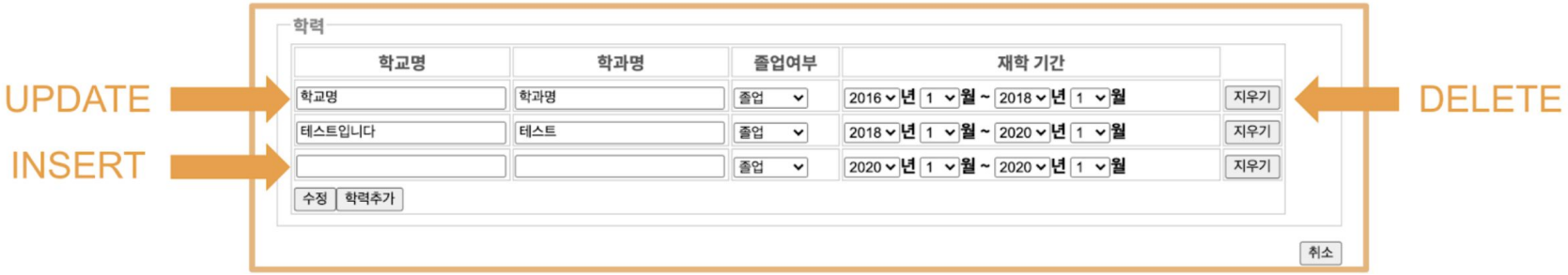
수정

```
MultipartRequest multi =
        new MultipartRequest(request, path, size, "UTF-8", new DefaultFileRenamePolicy());
//                      request객체, 저장될서버경로, 크기, 인코딩 방식, 같은 이름의 파일명 방지 처리
```

- **MultipartRequest**를 이용한 파일 업로드
- JSP 페이지에서 **enctype="*multipart/form-data*"** 로 설정하여 form태그 안의 파일을 포함한 데이터들을 처리함

IT JOBS

이력서 (Update)



- **한 번의 request**를 통해 입력, 수정, 삭제를 처리할 수 있도록 구현
- 컨트롤러로 리퀘스트된 학력의 PK 값들을 기준으로 데이터 처리
  (**Insert** : PK가 없지만 리퀘스트된 값이 있을 경우,
  Update : PK가 리퀘스트된 경우,
  Delete : PK가 DB에 있지만 리퀘스트되지 않은 경우)

IT JOBS

이력서 (Mybatis)

```
<insert id="insertResume" parameterType="RsDto">
    <selectKey keyProperty="rs_no" resultType="int" order="BEFORE">
        SELECT RESUMESEQ.NEXTVAL FROM DUAL
    </selectKey>

    INSERT INTO RESUME
    VALUES(#{rs_no}, #{member_no}, #{rs_img_name}, #{rs_img_path}, #{rs_selfintro}, #{rs_title})
</insert>
```

- insert 한 이력서의 PK 값을 바로 리턴받기 위해 **selectKey** 사용

```
<update id="updateAcademic" parameterType="java.util.List">
    <foreach collection="list" item="item" index="index" separator=";" open="DECLARE BEGIN" close="; END;">
        UPDATE RESUME_ACADEMIC
        <set>
            RS_AC_NAME = #{item.rs_ac_name}, RS_AC_DEPT = #{item.rs_ac_dept}, RS_AC_GRAD = #{item.rs_ac_grad},
            RS_AC_START_YEAR = #{item.rs_ac_start_year}, RS_AC_START_MONTH = #{item.rs_ac_start_month},
            RS_AC_END_YEAR = #{item.rs_ac_end_year}, RS_AC_END_MONTH = #{item.rs_ac_end_month}
        </set>
        WHERE RS_AC_NO = #{item.rs_ac_no}
    </foreach>
</update>
```

- 다중 insert, update 처리를 위해 **foreach** 사용

IT JOBS

채팅 (WebSocket API)

[JSP]

```
// WebSocket 객체 생성
var webSocket = new WebSocket("ws://localhost:8787/IT_JOBS/chatsocket");

// WebSocket 서버와 접속이 되면 호출되는 함수
webSocket.onopen = function(event) {

};

// WebSocket 서버와 접속이 끊기면 호출되는 함수
webSocket.onclose = function(event) {

};

// WebSocket 서버와 통신 중에 에러가 발생하면 요청되는 함수
webSocket.onerror = function(event) {

};

// WebSocket 서버에서 메세지를 보냈을 때 호출되는 함수
webSocket.onmessage = function(event) {

};
```

- HTML5의 **WebSocket API**를 사용하여 서버와 클라이언트 간의 TCP커넥션을 유지하는 양방향 통신을 제공받음
- 클라이언트가 서버에 접속하게 되면 **소켓 세션**에 클라이언트의 정보가 등록됨
- 등록 된 후 서비스를 위한 **함수** 사용 가능 (서버, 브라우저 동일)

IT JOBS

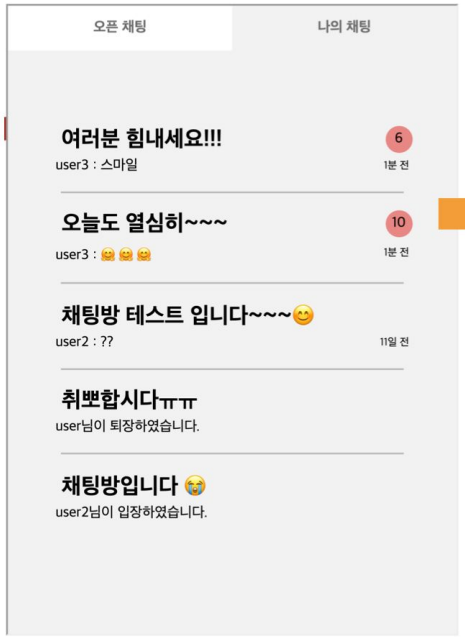
채팅 (오픈 채팅)

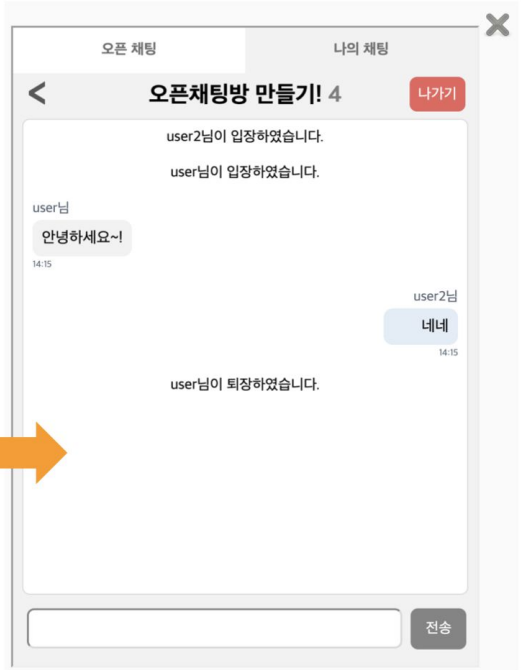
- 채팅페이지는 방을 생성할 수 있고 선택적으로 입장이 가능하고 다수와 채팅할 수 있는 **오픈 채팅** 방식으로 구현함

- **iframe**을 사용하여 aside의 이미지 클릭 시 채팅페이지를 불러오도록 화면 구현

IT JOBS

채팅 (나의 채팅)

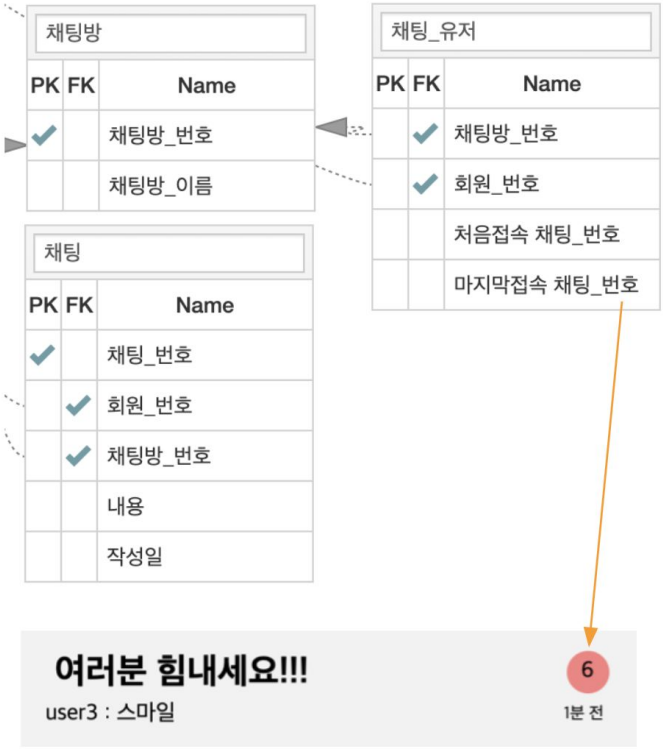




- '나의 채팅'에서 기존에 입장했던 채팅방에 입장할 경우, 처음 입장한 이후부터의 채팅 내용을 가져옴
- 읽지 않은 채팅의 개수 및 마지막 채팅의 내용과 시간 확인 가능
- login 세션의 id를 유저 이름으로 사용함

IT JOBS

채팅 (DB)



- 채팅 방을 삭제하면 해당 방의 채팅내용이 전부 삭제되도록 테이블 설계 시 제약조건에 **ON DELETE CASCADE** 를 사용함

- 처음 접속 시 채팅번호와 마지막 확인한 채팅번호를 채팅_유저 테이블에 저장해두어서 유저마다 가져올 수 있는 데이터에 한도를 두거나, 읽지 않은 채팅의 개수를 가져올 수 있도록 함

IT JOBS

채팅 (Insert / Delete)

Insert

```
private static Map<Integer, List<JsonObject>> chatDataMap = new HashMap<Integer, List<JsonObject>>();
```

1. 채팅내용이 WebSocket 서버로 전달됐을 때,
   채팅방과 채팅내용(유저, 내용, 작성일 등의 데이터가 담긴 Json 객체)을 매핑한 데이터를
   WebSocket **서버에 임시 저장**한다.
2. DB 커넥션 오류를 방지하기 위해서
   클라이언트가 연결을 종료했을 때 **일괄적**으로 채팅 데이터를 저장한다.

Delete

1. 클라이언트가 '나가기' 버튼을 클릭해 채팅방을 나갔을 경우,
   채팅_유저 테이블에서 해당 유저를 삭제한다.
2. 채팅_유저 테이블에서 해당 방에 남은 유저가 있는지 체크한 후,
   유저가 있을 때는 퇴장 메시지를 전송하고, 없을 때는 해당 방을 삭제한다.

IT JOBS

채팅 (HandShake)

[HandShake]

```
public class HttpSessionConfigurator extends Configurator {

    // HandShake : Http에서 WebSocket으로 프로토콜 전환
    // EndPointConfig에 HttpSession을 넣는다.
    @Override
    public void modifyHandshake(ServerEndpointConfig config,
            HandshakeRequest request, HandshakeResponse response) {

        // HttpRequest로부터 Session을 받는다.
        HttpSession session = (HttpSession) request.getHttpSession();

        config.getUserProperties().put("session", session);
    }
}
```

[WebSocket Server]

```
// WebSocket server를 생성하기 위한 URL, handshake 설정하기 위한 클래스 지정
@ServerEndpoint(value="/chatsocket",
                configurator = HttpSessionConfigurator.class)
```

1. HTTP에서 WebSocket으로 **프로토콜 전환**을 하기 위해 지정해둔 HandShake 클래스가 호출됨

2. HandShake 과정에서 HttpSession 객체를 **Endpoint** 객체의 속성으로 넣어줌

3. HandShake를 거친 후 웹소켓 서버와 연결됨

4. 웹소켓 세션과 Endpoint로 전달받은 HttpSession을 서버에 만들어둔 맵에 매핑하여 유저관리함
   (login 세션의 id값을 가져와서 채팅 유저 id로 사용함)

IT JOBS

채팅 (Json)

[Server]

```
// Json타입 메세지 받기
JsonElement element = JsonParser.parseString(msg);
JsonObject jsonData = element.getAsJsonObject();
jsonData.addProperty("member_no", member_no);
jsonData.addProperty("member_id", member_id);
```

[Client]

```
webSocket.onmessage = function(event) {
    var data = JSON.parse(event.data);

    var $chat = $("<div class='my-chat-area'><div class='chat-writer'>"
                + data.member_id
                + "님</div><div class='chat-content'>"
                + data.content
                + "</div><div class='chat-date'>"
                + data.data
                +"</div></div>");

    $("#chat-container").append($chat);

}
```

- 클라이언트와 서버간에 메시지를 주고 받기 위해 Json 객체 사용
- **gson**에서 제공하는 JsonParser, JsonObject, JsonElement 등 사용
- **JsonObject**를 만들 때에는 addProperty() 메서드를 사용하여 gson.toJson() 으로 처리함
- **JsonParser**를 사용하여 Json형태의 문자열을 JsonElement로 파싱함

IT JOBS

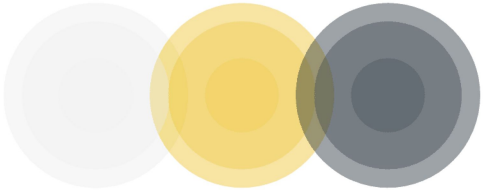
참여 소감

윤사라

기획의 중요성을 느꼈습니다. 초기 기획 과정에서 많은 시간을 들여 결정했기 때문에 프로젝트 구조를 익히는 데에 도움이 되었고, 프로젝트를 진행하면서도 목표를 확실히 할 수 있었습니다. 그렇기 때문에 힘들지만 중요한 과정이라고 생각합니다.

한 가지의 아쉬운 점은 협업이었습니다. 비대면으로 프로젝트를 진행해야 했기 때문에 팀원들과 소통할 때에 불편함이 있었습니다. 프로젝트 중 여러가지 문제들에 있어서 팀원들과 의사소통이 잘 되었을 경우 쉽게 해결할 수 있는 문제들이 많았기 때문에 줌, 행아웃 등의 화상회의를 통해서 잦은 소통을 하기 위해 노력했습니다.

협업을 잘 하기 위해서는 팀워크와 의사소통에 적극적이어야 한다는 것을 다시 한 번 느꼈습니다.



IT JOBS

감사합니다