

2023-11-1

PORTFOLIO

H-SMART 인사 관리 시스템

작성자 문상석

목차

- 01 프로젝트 소개
- 02 개발환경
- 03 프로젝트 일정(WBS)
- 04 관계 다이어그램(ERD)
- 05 담당 파트(구현, 기능)

1. 프로젝트 소개(HSmart)

1. 소개

- 사내 인사관리 시스템(임직원, 프리랜서)

2. 기간 인원

- PM 1인, PL 2인(개발, 문서 각 1인) 포함 총 10인 팀 프로젝트
- 지원자 본인은 개발 PL 담당

3. 프로젝트 기간(23/08/10 ~ 23/10/29)

- 23/08/10 ~ 23/08/29 분석, 설계 단계
- 23/08/30 ~ 23/09/26 개발, 구현 단계
- 23/10/4 ~ 23/10/13 테스트 단계
- 23/10/16 ~ 23/10/29 이행 및 안정화 단계

4. 핵심 기능

- 인사관리 기본 정보의 수집누락 방지, 중복수집, 업데이트 작업 등
의 불편 최소화를 위해 관리기능 개선
- 인적자원 리소스관리를 위한 기초자료 DB화 및 권할 별 조회 기능
구현
- 프로젝트 투입현황 조회 및 경영관리를 위한 대시보드 기능 구현
- 프리랜스 인력풀 관리기능 구현

5. 개인 기여

- 프리랜스 등록
- 프리랜스 리스트 - 리스트 조회, 검색, 페이지, 평가/의견, 전문분
야 조회, 수정, 삭제
- 프리랜스 기술 현황 - 기술 현황 조회, 기술 상세 조회

1. 프로젝트 소개(HSmart)

1. 개발 PL로 써의 수행 한 역할

- 초기 프로젝트 환경 설정
- DB Repository 생성 및 초기 데이터 작성 및 ppt 제작 배포
- 샘플링 자료를 통해 Spring Framework를 구조 ppt 제작 배포
- SVN Server를 통한 형상관리 툴 관리 및 사용법 ppt 제작 배포

3. Project 환경변수 설정

1) pom.xml에서 mybatis와 jdbc 의존성 추가
2) appServlet 폴더에 application-context 이름의 xml을 만들어 DB와 Mapper, mybatis를 연결
3) web.xml에서 application-context가 연결될 수 있게 파라미터 값을 변경

2-2. SQL Server와 DBeaver 연결

1) DBeaver 실행 > 새 데이터베이스 연결 > SQL SERVER 선택
2) HOST와 계정 정보를 통해 SQL Server에 있는 DB를 가져옴
3) SQL Server에 있는 데이터베이스와 연결 성공

1. SpringFramework MVC 순서

1. 화면(View) : jsp, xml로 작성된 View단.
2. 컨트롤러(Controller)
3. 서비스(Service)
4. 서비스 임플(Service Impl)
5. DTO단(데이터 형식 정의)
6. DAO단
7. SQL(쿼리를 날려 데이터를 받음)
8. DAO(DB에서 얻은 데이터를 리스트에 담음)
9. 서비스(Service)
10. 컨트롤러(Controller) : 리스트로 받은 데이터를 맵 형태로 저장.
11. 화면(View) : 출력해줌.

2. Controller 예시

프리랜스 컨트롤러(FreeController.java)

```
FreeController.java
1 package com.java.HSmart.controller;
2
3 import java.util.List;
4
5 @Controller
6 public class FreeController {
7
8     @Autowired
9     FreelanceService freelanceService;
10
11     @RequestMapping(value = "free", method = RequestMethod.GET)
12     public String free(Model model) {
13         try {
14             List<FreelanceDto> list = freelanceService.getList();
15             model.addAttribute("list", list);
16         } catch (Exception e) {
17             e.printStackTrace();
18         }
19         return "freelance";
20     }
21 }
```

@Controller 어노테이션은 주로 View(화면)을 반환하기 위해 사용(return "freelance"는 freelance.jsp화면으로 이동).
@Autowired 필요한 의존 객체의 "타입"에 해당하는 빈을 찾아주입한다.
@RequestMapping 들어온 요청을 특정 메서드와 매핑하기 위해 사용하는 것.
value는 요청별 url을 설정하게 된다.
method는 어떤 요청으로 받을지 정의하게 된다.(주로 GET, POST).
Service에서 메서드를 호출하여 list에 담아 함께 반환.

2. SVN 서버 구축

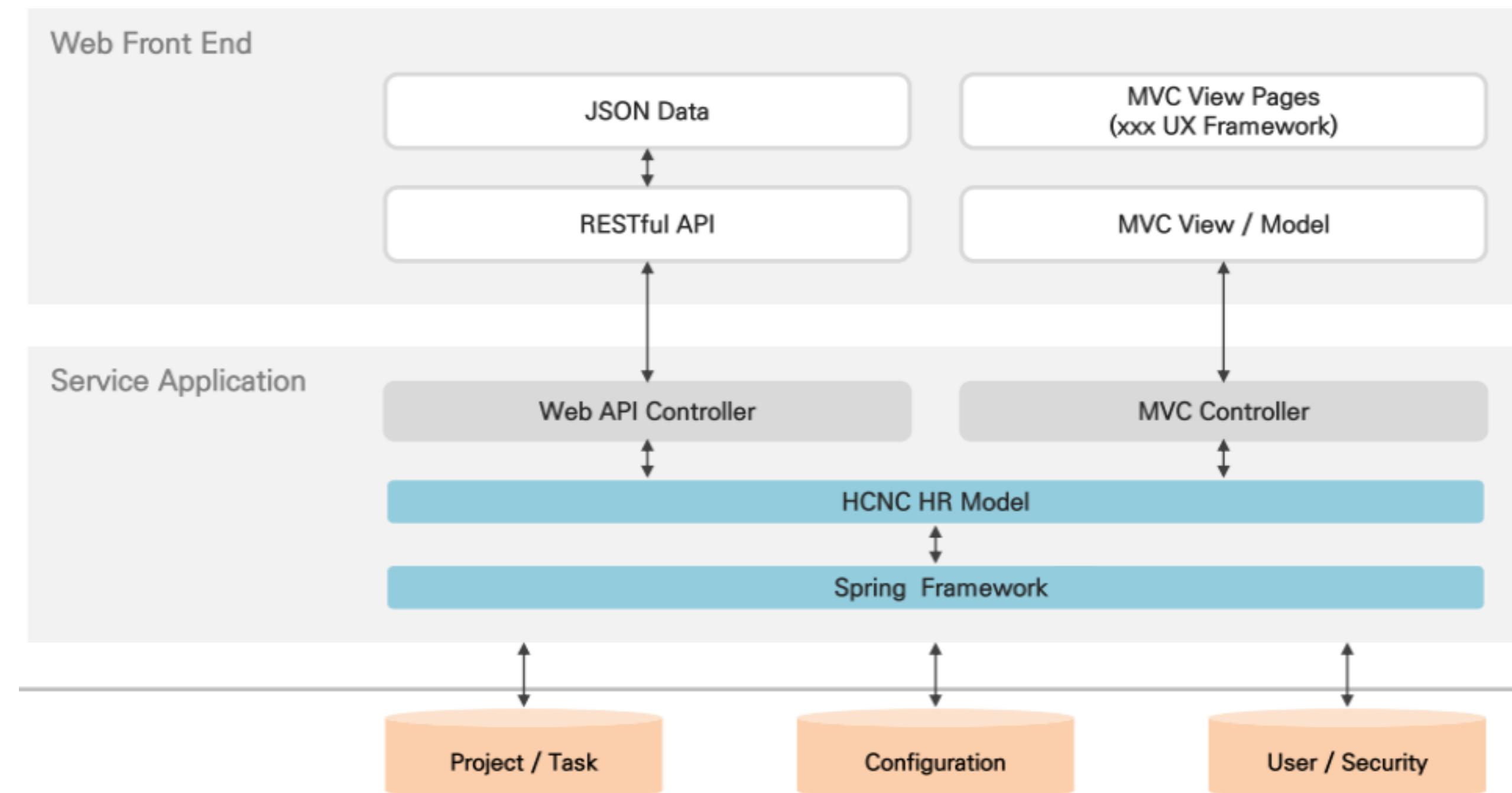
-VisualSVN Server 사용
-HSmart Repository 생성
-User 추가

2. SVN 서버 구축(받아오기)

1) Window > Show View > other..
2) SVN 검색 후 SVN Repositories
3) SVN Repositories에서 우클릭 후 New > Repository Location

1. 프로젝트 소개(HSmart)

1. 초기 HSmart 인사관리 Architecture



2. 개발 환경



프로그래밍 언어	JAVA JDK 1.8.0_202
화면 구현	HTML, CSS, JSP, JS
DB	Sql Server(MSSQL), DBeaver
Framework	Spring Framework 4.3.2.RELEASE
IDE	Eclipse 4.12 (2022-06)
서버	Apache Tomcat 9.0
사용기술	Mybatis, JQuery, AJAX
형상관리	Visual SVN

3. 프로젝트 일정(WBS)

1. 230830~230926(구현, 개발 단계)

- 본격적인 구현 개발 단계에 들어가 개발 PL로써 관리
 - 프리랜서 텁을 맡아 구현 작업 시즈
 - 프로젝트 완성을 앞두고 단위테스트 진행

3. 프로젝트 일정(WBS)

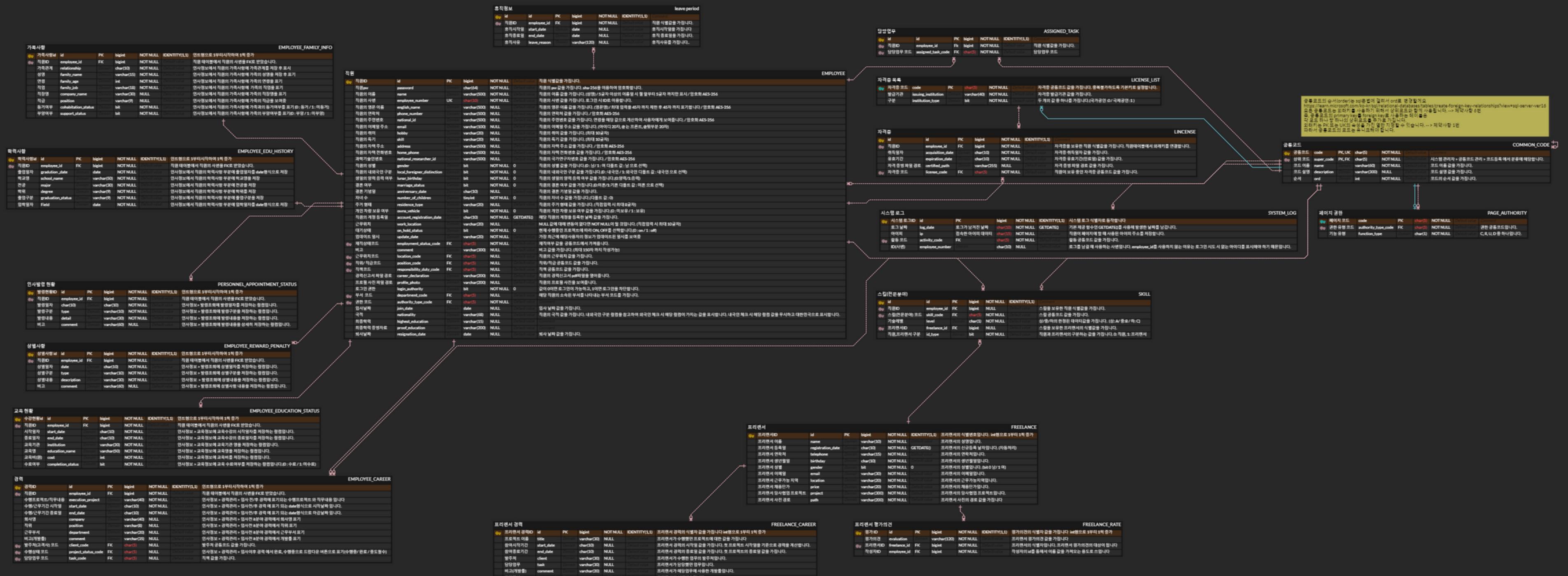
1. 231010~231013(테스트 단계)
2. 231016~231029(이행 및 안정화 단계)

	B	C	D	E	F	G	H	I	J	K	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	
▼ 10	구분	주요 업무	세부 업무	Screen ID	담당자	상태	진척률	시작일	종료일	소요일						2주차														
11											3일	4일	5일	6일	9일	10일	11일	12일	13일	16일	17일	18일	19일	20일	23일	24일	25일	26일	29일	
12																														
64	6. 테스트				문상석	완료	100%	2023.10.04	2023.10.13	7																				
65		6.1 통합테스트 설계서			문상석	완료	100%	2023.10.04	2023.10.05	2																				
66		6.2 통합테스트			문상석	완료	100%	2023.10.05	2023.10.06	2																				
67		6.3 통합테스트 결과서			문상석	완료	100%	2023.10.10	2023.10.11	2																				
68		6.4 성능시험 결과서			문상석	완료	100%	2023.10.11	2023.10.13	3																				
69	7. 이행 및 안정화				오세준	완료	100%	2023.10.04	2023.10.13	7																				
70		7.1 운영자 매뉴얼			오세준	완료	100%	2023.10.16	2023.10.18	3																				
71		7.2 사용자 매뉴얼			오세준	완료	100%	2023.10.16	2023.10.19	4																				
72		7.4 시스템전환(이행) 결과서			오세준	완료	100%	2023.10.19	2023.10.24	4																				
73		7.5 인수테스트 설계서			오세준	완료	100%	2023.10.20	2023.10.24	3																				
74		7.6 인수테스트 결과서			오세준	완료	100%	2023.10.20	2023.10.24	3																				
75		7.7 검수 확인서			오세준	완료	100%	2023.10.25	2023.10.29	3																				
76		7.8 인수 인계서			오세준	완료	100%	2023.10.25	2023.10.29	3																				

- 구현 개발단계가 완료 후 전체적인 통합테스트를 관리, 진행
- 테스트 진행 후 PM의 관리에 따라 운영자, 사용자 매뉴얼 작성

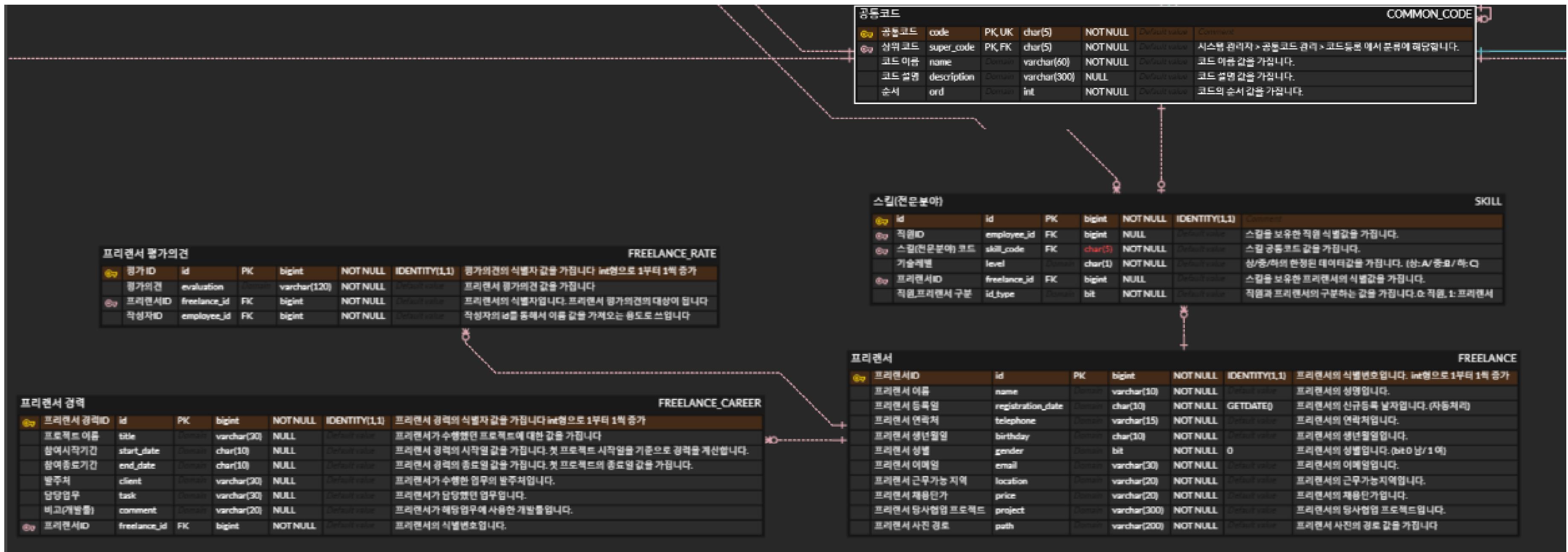
4. 관계 다이어그램(ERD)

1. ERD 전체 화면



4. 관계 다이어그램(ERD)

2. 담당 ERD 화면



5-1. 프리랜스 리스트 조회

6-1-1. 프리랜서 - 프리랜서 리스트

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 프리랜스 리스트 클릭 시 프리랜스 리스트 페이지로 이동함
2. 등록된 프리랜서들의 인적사항과 전문분야, 경력, 평가의견과 수정, 삭제를 할 수 있는 페이지

No	이름	연락처	이메일	전문분야	경력	경력등급	평가의견	관리	
1	강민우	010-2847-8248	2112sd@naver.com	null	1년 4개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
2	고길동	010-2838-2382	awidfy22@naver.com	.NET, ASP, C#,...	6년 9개월	중급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
3	고창석	010-2848-2482	awuehfafw@naver.com	.NET, ASP, C,...	4개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
4	강의영	010-2842-8424	aiwlhasdyf@naver.com	C, JAVA, Java...	2개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
5	리종식	010-2842-8482	aiwfusdfh@naver.com	C, C++	1년 2개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
6	안사유	010-3125-1251	asfwefaf2@gamil.com	HTML5, JAVA,...	경력없음	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
7	전현모	010-2842-8472	awihfsdf@naver.com	ASP, C	경력없음	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
8	기나다	010-4838-4284	awifehawif@naver.com	전자정부 프레...	경력없음	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
9	강호	010-8417-2428	awehfiasdf@naver.com	.NET, ASP, C,...	5개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
10	황조룡이	010-8482-1234	ghkdwhfh@naver.com	.NET, ASP, C,...	1년 10개월	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
11	전현모	010-4274-7274	awifhasdyf@naver.com	C, JAVA	경력없음	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>
12	박철민	010-4827-4274	awufydsf@naver.com	C, JAVA	경력없음	초급	<button>보기/등록</button>	<button>수정</button>	<button>삭제</button>

등록된
프리랜서
리스트를
열람하거나
검색할 수 있는
페이지

freelance.js

```
// 시작 시 페이지 1, 옵션 키워드
let currentPage = 1;
let option = "";
let keyword = "";

$(document).ready(function () {
    let searchText = localStorage.getItem('keyword');
    if(searchText){
        getList(currentPage, option, searchText);
        localStorage.removeItem('keyword');
    }else{
        getList(currentPage, option, keyword); // 페이지
    }
})
```

```
function getList(page, option, keyword) {
    $.ajax({
        type: "GET",
        url: "/FREEGO/free/list.do",
        data: {
            page: page,
            option: option,
            keyword: keyword,
        },
        dataType: 'json',
        success: function(data) {
            응답 받은 data로 리스트의 정보 조회
            error: function(error) {
                console.error("Error fetching freelance list:", error);
            }
        });
}
```

FreeController.java

```
@RequestMapping("/FREEGO/free")
public class FreeController {
    @Autowired
    FreelanceService freelanceService;

    @RequestMapping(value = "/{path}", method = RequestMethod.GET)
    public String freeStatus(@PathVariable String path) {
        return "freelance_" + path;
    }

    @RequestMapping(value="/list.do", method = RequestMethod.GET)
    public @ResponseBody ModelAndView getFreeList(
        @Param("page") int page,
        @Param("option") String option,
        @Param("keyword") String keyword) {
        ModelAndView mav = new ModelAndView();
    }
```

프리랜스 리스트 조회

- 초기 page 값은 1, option과 keyword는 빈 값으로 ajax를 통해 controller에 GET 방식으로 요청
- Controller에서는 요청을 받아 FreelanceService, FreelanceDao, FreelanceMapper 단 계를 거쳐 DB에서 데이터를 조회 하여 역순으로 응답

5-2-1. 프리랜스 검색, 페이지 기능

6-1-2. 프리랜스 - 프리랜스 리스트(검색)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 프리랜스 검색 옵션 설정
2. 프리랜스 검색 키워드 설정
3. 검색 옵션과 키워드에 따른 프리랜스 리스트를 불러옴
4. 검색 결과의 수에 따른 페이징 기능

[그림 6-1-2] 프리랜스-프리랜스 리스트(검색)

PageHandler.java

```
public class PageHandler {
    private SearchCondition sc;
    private int totalCnt; //총 게시물 개수
    private int naviSize=10; //페이지 네비게이션의 크기
    private int totalPage; // 전체 페이지의 개수
    private int beginPage; // 네비게이션의 첫 페이지
    private int endPage; // 네비게이션의 마지막 페이지
    private boolean showPrev; // 이전페이지로 이동하는 링크 표시 여부
    private boolean showNext; // 다음페이지로 이동하는 링크 표시 여부

    //생성자
    public PageHandler(int totalCnt){
        this.totalCnt = totalCnt;
    }

    public PageHandler(int totalCnt, SearchCondition sc) {
        this.totalCnt = totalCnt;
        this.sc = sc;
        doPaging(totalCnt, sc);
    }

    public void doPaging(int totalCnt, SearchCondition sc) {
        this.totalCnt = totalCnt;
        //전체 페이지의 개수 계산하기
        totalPage = (int) Math.ceil(totalCnt / (double) sc.getPageSize());
        //시작페이지
        beginPage = (sc.getPage()-1) / naviSize * naviSize +1;
        //페이징의 끝
        endPage = Math.min(beginPage + naviSize - 1, totalPage);

        //이전페이지, 다음 페이지 버튼 표시
        showPrev = beginPage != 1;
        showNext = endPage != totalPage;
    }
}
```

SearchCondition.java

```
public class SearchCondition {
    private Integer page = 1; //현재 페이지
    private Integer pageSize = 10; //한 페이지 당 게시물
    private Integer offset = 0;
    private String option = ""; // 검색옵션
    private String keyword = ""; //검색어

    public SearchCondition(Integer page, Integer pageSize) {
        this(page, pageSize, "", "");
    }

    public SearchCondition(String option, String keyword) {
        this.option = option;
        this.keyword = keyword;
    }

    public SearchCondition(Integer page, Integer pageSize, String option, String keyword) {
        this.page = page;
        this.pageSize = pageSize;
        this.option = option;
        this.keyword = keyword;
    }
}
```

프리랜스의
전체 리스트를
검색하여 볼 수
있는 페이지

프리랜스 검색, 페이징 기능

- 페이지 기능을 처리하는 PageHandler를 통해 totalCnt와 sc라는 두 개의 매개변수를 받아 검색기능을 하는 SearchCondition 객체를 활용하여 두 가지 기능을 수행
- Controller에서는 요청을 받아 FreelanceService, FreelanceDao, FreelanceMapper 단계를 거쳐 DB에서 데이터를 조회하여 역순으로 응답

5-2-2. 프리랜스 검색, 페이지 기능

```
freelance.js
//프리랜스 리스트를 불러오는 함수
function getList(page, option, keyword) {
  $.ajax({
    type: "GET",
    url: "/FREEGO/free/list.do",
    data: {
      page: page,
      option: option,
      keyword: keyword,
    },
    dataType: 'json',
    success: function(data) {
```

응답 받은 data로 리스트의 정보 조회...

```
pageStr += '<ul class="page_container">';
if ((data.ph.totalCnt != null && data.ph.totalCnt != 0) && data.ph.endPage != 1) {
  if (data.ph.showPrev) {
    pageStr += '<li class="pprev-page"><a href="#"></a></li>' +
    '<li class="prev-page"><a href="#"></a></li>';
  }
  for (let i = data.ph.beginPage; i <= data.ph.endPage; i++) {
    pageStr += `<li><a class="${i === data.ph.sc.page ? 'on' : 'pageNum'}" href="#">${i}</a></li>`;
  }
  if (data.ph.showNext) {
    pageStr += '<li class="next-page"><a href="#"></a>' +
    '<li class="nnext-page"><a href="#"></a>';
  }
}
pageStr += '</ul>';
// 생성된 HTML 문자열을 .free_list 요소에 추가
$('.free_list').empty().append(str);
$('.free-paging-btn').empty().append(pageStr);
pagination(data);
},
```

프리랜스 검색, 페이지 기능

- 페이지, 검색옵션, 키워드를 매개변수로 받은 getList를 통해 ajax로 Controller에 요청
- 페이지와 검색옵션, 키워드를 담은 SearchCondition을 Mapper까지 보내 DB에서 Data를 조회하여 역순으로 응답

FreeController.java

```
@RequestMapping("/FREEGO/free")
public class FreeController {
  @Autowired
  FreelanceService freelanceService;
  @RequestMapping(value = "/{path}", method = RequestMethod.GET)
  public String freeStatus(@PathVariable String path) {
    return "freelance_" + path;
  }
  @RequestMapping(value="/list.do", method = RequestMethod.GET)
  public @ResponseBody ModelAndView getFreeList(
    @Param("page") int page,
    @Param("option") String option,
    @Param("keyword") String keyword) {
    ModelAndView mav = new ModelAndView();
    SearchCondition sc = new SearchCondition(page, 15, option, keyword);
    List<FreelanceDto> list = freelanceService.getList(sc);
    int totalCnt = freelanceService.searchReusltCnt(sc);
    PageHandler ph = new PageHandler(totalCnt ,sc);
    int freelnace_count = freelanceService.getFreeCount();
    mav.addObject("freelnace_count", freelnace_count);
    mav.addObject("list", list);
    mav.addObject("ph", ph);
    mav.setViewName("jsonView");
    return mav;
  }
},
```

freelanceMapper.xml

```
<sql id="searchCondition">
  <if test="keyword != null and keyword !='"'>
    <choose>
      <when test='option=="N"'>
        AND F.name LIKE '%' + #{keyword} + '%'
      </when>
      <when test='option=="S"'>
        AND <include refid="skill_name"/> LIKE '%' + #{keyword} + '%'
      </when>
      <when test='option=="C"'>
        AND <include refid="career"/> LIKE '%' + #{keyword} + '%'
      </when>
      <when test='option=="G"'>
        AND <include refid="grade"/> LIKE '%' + #{keyword} + '%'
      </when>
      <when test='option=="E"'>
        AND <include refid="etc"/> LIKE '%' + #{keyword} + '%'
      </when>
      <when test='option=="P"'>
        AND F.project LIKE '%' + #{keyword} + '%'
      </when>
    <otherwise>
      AND (
        F.name LIKE '%' + #{keyword} + '%'
        OR F.telephone LIKE '%' + #{keyword} + '%'
        OR F.email LIKE '%' + #{keyword} + '%'
        OR <include refid="career"/> LIKE '%' + #{keyword} + '%'
        OR <include refid="skill_name"/> LIKE '%' + #{keyword} + '%'
        OR <include refid="grade"/> LIKE '%' + #{keyword} + '%'
        OR <include refid="etc"/> LIKE '%' + #{keyword} + '%'
        OR F.project LIKE '%' + #{keyword} + '%'
      )
    </otherwise>
  </choose>
  </if>
</sql>
<select id="selectAll" parameterType="SearchCondition" resultType="FreelanceDto">
  WITH CareerDuration AS (
    SELECT
      FC.freelance_id,
      SUM(DATEDIFF(MONTH, FC.start_date, FC.end_date)) AS total_months
    FROM FREELANCE_CAREER FC
    GROUP BY FC.freelance_id
  ),
  RankedFreelance AS (
    SELECT
      F.id,
      F.name,
      F.registration_date,
      F.telephone,
      F.email,
      <include refid="career"/> AS career,
      <include refid="skill_name"/> AS skill_name,
      <include refid="grade"/> AS grade,
      ROW_NUMBER() OVER (ORDER BY F.registration_date DESC, f.id DESC) AS RowNum
    FROM FREELANCE F
    LEFT JOIN CareerDuration CD ON F.id = CD.freelance_id
    LEFT JOIN SKILL S ON F.id = S.freelance_id
    WHERE 1=1 <include refid="searchCondition"/>
    GROUP BY F.id, F.name, F.registration_date, F.telephone, F.email, CD.total_months
  )
  SELECT * FROM RankedFreelance
  WHERE RowNum BETWEEN ((ISNULL(#{page}, 0) - 1) * #{pageSize} + 1) AND (ISNULL(#{page}, 0) * #{pageSize})
  ORDER BY RowNum;
</select>
```

5-3. 고유 ID값 부여

6-1-3. 프리랜스 - 프리랜스 리스트

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 다수의 전문분야를 볼 수 있는 팝업 페이지(6-1-4에서 설명)
2. 프리랜서의 평가를 보고 작성 할 수 있는 팝업 페이지(6-1-5에서 설명)
3. 프리랜서의 정보를 수정 할 수 있는 팝업 페이지(6-1-6에서 설명)
4. 프리랜서를 삭제하는 기능(6-1-8에서 설명)

[그림 6-1-3] 프리랜스-프리랜스 리스트

freelance.js

```
//프리랜스 리스트를 불러오는 함수
function getList(page, option, keyword) {
    $.ajax({
        type: "GET",
        url: "/FREEGO/free/list.do",
        data: {
            page: page,
            option: option,
            keyword: keyword,
        },
        dataType: 'json',
        success: function(data) {
            let str = '' // HTML 문자열을 담을 변수
            let pageStr = '';
            $('#freelancer_num').text(data.freelance_count + '명'); //프리랜스 현황 인원 수
            // 리스트 데이터를 동적으로 생성
            for (let i = 0; i < data.list.length; i++) {
                let postNumber = (data.ph.sc.page - 1) * data.ph.sc.pageSize + i + 1;
                str += '<tr>' +
                    '<td>' + postNumber + '</td>' +
                    '<td>' + data.list[i].name + '</td>' +
                    '<td>' + data.list[i].telephone + '</td>' +
                    '<td>' + data.list[i].email + '</td>' +
                    '<td class="skill_hidden"><a href="#" onclick="getSkillList(' + data.list[i].id + ')">' + data.list[i].skill_name + '</td>' +
                    '<td>' + data.list[i].career + '</td>' +
                    '<td>' + data.list[i].grade + '</td>' +
                    '<td><a href="#" class="rate_read btn-manage" onclick="getRate(' + data.list[i].id + ')>보기/등록</a></td>' +
                    '<td>' +
                    '<div class="free-fbtn-manage">' +
                        '<a href="#" class="btn-manage" onclick="getFreeInfo(' + data.list[i].id + ')>수정</a>' +
                        '<a href="#" class="btn-manage" onclick="deleteFreelance(' + data.list[i].id + ')>삭제</a>' +
                    '</div>' +
                    '</td>' +
                    '</tr>';
            }
        }
    });
}
```

프리랜스의
상세 정보 및
평가의견, 수정,
삭제를 할 수
있는 팝업
페이지

프리랜스 전문분야 상세, 평가 의견, 수정, 삭제에 id값 부여

- 리스트를 불러오며 각 getSkillList(), getRate(), getFreeInfo(), deleteFreelance() 함수에 id 값을 부여

5-4. 프리랜스 전문분야 상세정보

6-1-4. 프리랜스 - 프리랜스 리스트(전문분야 상세정보)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

- 선택한 프리랜스 전문분야 상세정보를 보여주는 팝업(이름, 전문분야, 기술등급)

The screenshot shows a web application interface. On the left, there's a sidebar with various menu items like '로그인', '프리랜스', '프리랜스 신고 등록', etc. The main content area has a title '프리랜스 리스트' and a sub-section '프리랜스 전문분야 상세정보 팝업'. A modal window is open, displaying a table with columns: No, 이름, 전문분야, 기술등급. The table contains 5 rows of data. A red circle with the number '1' is drawn around the first row of the table in the modal. The background page shows a list of 15 entries with columns: No, 이름, 연락처, 이메일, 전문분야, 경력, 경력등급, 평가의견, 관리. The bottom right of the screenshot has the text '프리랜스 전문분야 상세정보 팝업 페이지'.

[그림 6-1-4] 프리랜스-프리랜스 전문분야 상세정보 팝업

freelance.js

```
//프리랜스 리스트 스킬 상세보기
function getSkillList(freelance_id){
    $.ajax({
        type: "GET",
        url: "/FREEGO/free/skillList.do?id=" + freelance_id,
        dataType:'json',
        success : function(data){
            $('#skillModal').show();

            let str = '' // HTML 문자열을 담을 변수
            let no = 1; //평가 리스트의 순서 용도
            let inputRow = '' ; //평가 의견 작성란

            for (let i = 0; i < data.skillList.length; i++){
                // 각 데이터를 행으로 추가
                str += '<tr>' +
                    '<td>' + (i+1) + '</td>' +
                    '<td>' + data.skillList[i].name + '</td>' +
                    '<td>' + data.skillList[i].skill_name + '</td>' +
                    '<td>' + data.skillList[i].skill_level + '</td>' +
                    '</tr>';
                no++;
            }

            $('.skillListTbody').html(str);
        },
        error : function(data){
            console.error("Error fetching freelance rate:", error);
        }
    });
}
```

프리랜스 전문분야 상세정보

- 리스트에서 받은 ID값을 매개변수로 ajax를 통해 GET방식으로 요청 후 조회

5-5-1. 프리랜스 평가의견 보기

6-1-5. 프리랜스 - 프리랜스 리스트(평가의견 보기/등록)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

- 선택한 프리랜서의 이름과 당사 협업 프로젝트가 나옴
- 선택한 프리랜서에 대한 평가 의견을 보기
- 프리랜서에 대한 평가의견을 작성 후 버튼을 누르면 평가의견이 등록된다

The screenshot shows the H-Smart HR system interface. On the left, there's a sidebar with various menu items like 'Employee', 'Leave Record', 'Attendance', etc. The main area displays a table of freelancers with columns: No, Name, Contact, Email, Major, Experience, Rating Level, Review, and Action buttons. A modal window titled 'Free Rate List' is overlaid on the page. It contains a table of reviews with columns: No, Review, Reviewer, Reviewer, and Date. The first review is highlighted with a red border and circled with a red number 1. The second review is highlighted with a red border and circled with a red number 2. The third review is highlighted with a red border and circled with a red number 3. At the bottom of the modal, there's a '등록' (Register) button.

[그림 6-1-5] 프리랜스-프리랜스 평가의견 보기/등록

freelance.js

```
// 평가 의견 보기 할수
function getRate(freelance_id){
    $.ajax({
        type: "GET",
        url: "/FREEGO/free/rate.do?id=" + freelance_id,
        dataType:'json',
        success : function(data){
            $('#editModal').show();
            $('.scrollable-table').scrollTop(0);
            let str = ''; // HTML 문자열을 담을 변수
            let no = 1; // 평가 리스트의 순서 증도
            let inputRow = ''; // 평가 의견 작성 판
            $('.rate_name').text(data.freelanceInfo.name+'(' + (data.freelanceInfo.project == "" || data.freelanceInfo.project == null ? "해당없음" : data.freelanceInfo.project) +')');

            for (let i = 0; i < data.rateList.length; i++){
                // 각 데이터를 행으로 추가
                str += '<tr>' +
                    '<td>' + (i+1) + '</td>' +
                    '<td style="text-align:left;">' + data.rateList[i].evaluation + '</td>' +
                    '<td>' + dateString(data.rateList[i].registration_date) + '</td>' +
                    '<td>' + data.rateList[i].e_name + '</td>' +
                    '</tr>';

                no++;
            }
            // 마지막 행 추가
            inputRow = '<tr>' +
                '<td>' + no + '</td>' +
                '<td style="text-align:left;"><input type="text" name="free_evaluation" class="wp100 hp100" placeholder="평가 내용을 입력해 주세요" maxlength="50"></td>' +
                '<td>' + todayDate() + '</td>' +
                '<td href="#" id="free_rate_btn" class="table-btn wp90" style="margin:0 auto" onclick="evaluationCheck() ? modal10(' + freelance_id + ') : false"><a>등록</a></td>';
            $('.free_rateList').empty().append(str);
            $('.free_rateList_input').empty().append(inputRow);

        },
        error : function(data){
            console.error("Error fetching freelance rate:", error);
        }
    });
}
```

FreeController.java

```
@RequestMapping(value="/rate.do", method = RequestMethod.GET)
public @ResponseBody ModelAndView getRateList(@RequestParam("id") int freelance_id){

    ModelAndView mav = new ModelAndView();
    List<FreelanceRateDto> rateList = freelanceService.getRateList(freelance_id);
    FreelanceDto freelanceInfo = freelanceService.getFreeUpdate(freelance_id);

    mav.addObject("rateList", rateList);
    mav.addObject("freelanceInfo", freelanceInfo);
    mav.setViewName("jsonView");
    return mav;
}
```

프리랜스 평가의견 보기

- 리스트에서 받은 ID값을 매개변수로 ajax를 통해 GET방식으로 요청 후 조회

5-5-2. 프리랜스 평가의견 등록

```
freelance.js
//평가 의견 등록 할수
function insertRate(freelance_id){
    let evaluation = $('input[name="free_evaluation"]').val();
    $.ajax({
        type: "POST",
        url: "/FREEGO/free/insertRate.do",
        data: {
            freelance_id : freelance_id,
            evaluation : "" + evaluation + ""
        },
        dataType:'json',
        success: function(data) {
            getRate(freelance_id);
            $('.btn-popup-check').removeClass('processing'); // 버튼 다시 활성화
        },
        error: function(error) {
            console.error("Error while inserting evaluation:", error);
            // 실패 시에도 버튼을 다시 활성화
            $('.btn-popup-check').removeClass('processing');
        }
    });
}
```

```
FreeController.java
@RequestMapping(value = "/insertRate.do", method = RequestMethod.POST)
public @ResponseBody ModelAndView insertRate(
    @RequestParam("evaluation") String evaluation,
    @RequestParam("freelance_id") int freelance_id,
    HttpSession session) {
    ModelAndView mav = new ModelAndView();
    MemberDto member = (MemberDto) session.getAttribute("member");
    int id = (int) member.getId();
    FreelanceRateDto dto = new FreelanceRateDto();
    dto.setFreelance_id(freelance_id);
    dto.setEvaluation(evaluation);
    dto.setEmployee_id(id); // 나중에 세션으로 값 가져오기
    freelanceService.insertRate(dto);
    mav.setViewName("jsonView");
    return mav;
}
```

```
freelanceMapper.xml
<!-- 5. 프리랜스 평가 입력 -->
<insert id="insertRate" parameterType="FreelanceRateDto">
    INSERT INTO
        FREELANCE_RATE(evaluation, freelance_id, employee_id)
        VALUES(${evaluation}, ${freelance_id}, ${employee_id})
</insert>
```

프리랜스 평가의견 등록

- ID값과 input에 입력한 평가 의견 값을 변수에 담아 ajax를 통해 POST방식으로 요청
- controller에서 freelance_id와 평가의견, 현재 로그인 되어 있는 인원의 session값을 가져와 id를 통해 FreelanceRateDto dto 인스턴스에 담아 보낸다

5-6-1. 프리랜스 정보 조회/수정

6-1-6. 프리랜스 - 프리랜스 리스트(프리랜서 정보 수정)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 프리랜서의 정보를 수정하는 프리랜서 정보 수정 팝업
2. 해당 프리랜서의 사진 수정기능
3. 등록할 프리랜서의 기본 인적사항을 수정하는 부분 (각 부분에 해당하는 조건에 맞지 않으면 경고와 함께 재확인 요청을 함)
4. 해당 프리랜서의 전문분야를 수정하는 팝업 페이지(6-1-7.에서 설명)
5. 해당 프리랜서의 이전 수행한 프로젝트들을 수정하는 부분
6. 해당 프리랜서의 경력이 많을 경우 공백란을 채운 뒤 추가 가능 버튼 (추가 시 필수 입력 조건: 수행프로젝트 명, 참여기간, 종료기간)
7. 해당 프리랜서의 경력을 삭제하는 부분
8. 해당 버튼을 클릭하면 수정한 프리랜스의 정보들을 업데이트한다

The screenshot shows the 'Freelancer Information Update' modal. It includes fields for basic information like name, phone, email, gender, birthdate, address, and skills. It also shows a preview image of the freelancer and a section for previous projects. Buttons for saving changes (수정) and canceling (취소) are at the bottom.

[그림 6-1-6] 프리랜스-프리랜스 정보 수정

기존에 등록된 프리랜서의 정보 수정을 위한 팝업 페이지

```
// 프리랜스 정보 불러오기
function getFreeInfo(freelance_id){
    $('#free_profile_Img').remove();
    $('#e_profile_sample_image').remove();
    e_callProfile').remove();
    Skills = [];

    {
        e: "GET",
        : "/FREEGO/free/getFreeInfo.do",
        a:{freelance_id:freelance_id},
        aType: "json",
        cess: function(data) {
            $('#freeUpdateModal').show();

            // 변환한 날짜를 입력 요소에 설정
            $('#name1').val(data.getFreelance.name);
            $('#registration_date1').text(dateString(data.getFreelance.registration_date));
            $('#telephone1').val(data.getFreelance.telephone);
            $('#birthday1').val(dateString(data.getFreelance.birthday));
            $('#gender1').val(data.getFreelance.gender);
            $('#email1').val(data.getFreelance.email);
            $('#location1').val(data.getFreelance.location);
            $('#price1').val(data.getFreelance.price);
            $('#project1').val(data.getFreelance.project);
            $('#careeर1').text(data.getFreelance.career);
            $('#grade1').text(data.getFreelance.grade);
            $('#skill1').val(data.getFreelance.skill_name);
            $('#mainPro').val(data.getFreelance.path);

            showFreeProfileImg();
            getFreeCareer(freelance_id);
            getUpdateSkillList(freelance_id)
        }
    }

    // 수정버튼을 입력 요소에 설정
    $('#updateButton').off('click');
    $('.exprt-search').off('click');

    $('#skill1').click(function(){
        $('.exprt-search').click();
    });

    $('.exprt-search').click(function() {
        $('#freeExprtModal').show();
    });
    // 클릭 이벤트 핸들러를 등록
    $('#updateButton').click(function () {
        if(freeUpdateCheck()){
            if(freeCareerUpdateCheck()){
                if(freeCareerInsertCheck()){
                    freeUpdate(freelance_id);
                }
            }
        }
    });
}

//프리랜스 정보 수정
function freeUpdate(id){

let formData = new FormData();
let profileUpload = $('input[name="profileUpload"]')[0].files[0];

let updateData = {
    "id": id,
    "name": $('#name1').val(),
    "registration_date": $('#registration_date1').val(),
    "telephone": $('#telephone1').val(),
    "birthday": $('#birthday1').val(),
    "gender": $('#gender1').val(),
    "email": $('#email1').val(),
    "location": $('#location1').val(),
    "price": $('#price1').val(),
    "project": $('#project1').val(),
    "path": $('#mainPro').val()
};

formData.append('profileUpload', profileUpload);
formData.append('key', new Blob([ JSON.stringify(updateData) ], {type : "application/json"}));

$.ajax({
    type: "POST",
    url: "/FREEGO/free/freeUpdate.do",
    data: formData,
    processData: false, // 데이터 처리 방식 설정
    contentType: false, // 컨텐츠 타입 설정
    enctype: 'multipart/form-data',
    success: function(data) {
        updateCareer(id);
        deleteCareer(id);
        insertCareer(id);
        getCheckedSkills(id, checkedSkills);
        alert("프리랜스 정보 수정 완료");
        getList(currentPage, option, keyword);
        $('#loading_page').hide();
        $('#freeUpdateModal').hide();
        getFreeInfo(id);
    },
    error: function(error) {
        console.log(error);
    },
    beforeSend: function(){
        $('#text').text('저장 중입니다.');
        $('#loading_page').show();
    }
});
}
}
```

프리랜스 정보 조회/수정

- 리스트에서 받은 ID값을 매개변수로 ajax를 통해 GET방식으로 요청 후 조회
- 프로필 사진 업로드 파일을 선택하는 profileUpload와, 프리랜서 정보를 담고 있는 updateData를 FormData 객체에 추가하고 JSON 데이터를 Blob으로 변환하여 key로 추가하여 전송
- ajax 요청이 성공하면 프리랜스 정보를 변경하면서, updateCareer(경력 수정), deleteCareer(경력 삭제), insertCareer(경력 추가), getCheckedSkills(전문분야 수정)을 일괄 진행

5-6-2. 프리랜스 정보 수정

```
@RequestMapping(value = "/freeUpdate.do", method = RequestMethod.POST)
public ModelAndView freeUpdate(@RequestParam(value = "key") Map<String, Object> param,
    @RequestParam(value = "profileUpload", required = false) MultipartFile profileUpload,
    HttpSession session, HttpServletRequest request){

    ModelAndView mav = new ModelAndView();
    int id = (int) param.get("id");
    FreelanceDto dto = freelanceService.getFreeupdate(id);

    /* ===== 프로필사진 업로드 영역 ===== */
    // 프로필사진 DB 존재 유무 확인
    FreelanceDto checkUploadFile = freelanceService.checkProfileUpload(id);

    // 프로필 사진을 저장할 실제 파일 시스템 경로 설정
    String profilepath = request.getSession().getServletContext().getRealPath("/resources/uploadFiles/profile/freelance");
    String profileFilePath = "\\\\"resources\\\\uploadFiles\\\\profile\\\\freelance\\\\";

    // 폴더 생성
    File proFileUploadPath = new File(profilepath);

    if (checkUploadFile == null || checkUploadFile.getPath().equals("")) {
        // DB에 프로필 사진이 존재하지 않을 경우 또는 업로드된 파일이 없을 경우
        if (profileUpload != null) {
            // 업로드된 파일이 존재하는 경우
            String originalProfilename = profileUpload.getOriginalFilename(); // 파일원본명
            String uuid = UUID.randomUUID().toString(); // UUID 생성
            String proFileName = uuid + "_" + originalProfilename; // 새로운 파일 이름 생성

            if (proFileUploadPath.exists() == false) {
                // 프로필 사진 저장 폴더가 존재하지 않으면 생성
                proFileUploadPath.mkdirs();
            }
            // 업로드 경로 설정
            String Db_saveFile = profileFilePath + proFileName;
            dto.setPath(Db_saveFile);
            File saveFile = new File(proFileUploadPath, proFileName);
            // 파일 저장
            try {
                profileUpload.transferTo(saveFile);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    } else {
        // 업로드된 파일이 없는 경우, 기존 경로를 사용
        dto.setPath((String) param.get("path"));
    }
} else {
    // DB에 프로필 사진이 존재하는 경우
    if (profileUpload != null) {
        // 업로드된 파일이 존재하는 경우
        String[] parts = checkUploadFile.getPath().split("\\\\");
        // DB에 프로필 사진이 존재하는 경우
        if (profileUpload != null) {
            // 업로드된 파일이 존재하는 경우
            String[] parts = checkUploadFile.getPath().split("\\\\");
            String fileName = parts[parts.length - 1];
            String deleteFilePath = profilepath + '/' + fileName;
            File file = new File(deleteFilePath);
            if (file.exists()) {
                // 기존 파일 삭제
                file.delete();
            } else {
                System.out.println("삭제할 파일이 존재하지 않습니다.");
                if (proFileUploadPath.exists() == false) {
                    proFileUploadPath.mkdirs();
                    System.out.println("저장경로 : " + proFileUploadPath);
                }
            }
            // 새로운 파일 이름 생성
            String originalProfilename = profileUpload.getOriginalFilename();
            String uuid = UUID.randomUUID().toString();
            String proFileName = uuid + "_" + originalProfilename;
            // 새로운 경로 설정
            String Db_saveFile = profileFilePath + proFileName;
            System.out.println("Db_saveFile : " + Db_saveFile);
            dto.setPath(Db_saveFile);
            File saveFile = new File(proFileUploadPath, proFileName);
            // 파일 저장
            try {
                profileUpload.transferTo(saveFile);
                System.out.println("저장 : " + saveFile);
            } catch (Exception e) {
                e.printStackTrace();
            }
        } else {
            // 업로드된 파일이 없는 경우, 기존 경로를 사용
            dto.setPath((String) param.get("path"));
        }
    }
}

dto.setName((String)param.get("name"));
dto.setTelephone((String)param.get("telephone"));
dto.setEmail((String)param.get("email"));
dto.setLocation((String)param.get("location"));
dto.setProject((String)param.get("project"));
if (param.get("price") instanceof Integer) {
    dto.setPrice((Integer)param.get("price"));
} else if (param.get("price") instanceof String) {
    dto.setPrice(Integer.parseInt((String)param.get("price")));
}
if (param.get("gender") instanceof Integer) {
    dto.setGender((Integer)param.get("gender"));
} else if (param.get("gender") instanceof String) {
    dto.setGender(Integer.parseInt((String)param.get("gender")));
}
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
try {
    String birthdayValue = (String) param.get("birthday");
    if (birthdayValue != null && !birthdayValue.isEmpty()) {
        // birthdayValue가 null이 아니고 비어있지 않을 때만 파싱을 시도합니다.
        Date birthday = dateFormat.parse(birthdayValue);
        dto.setBirthday(birthday);
    }
}
String registrationDateValue = (String) param.get("registration_date");
if (registrationDateValue != null && !registrationDateValue.isEmpty()) {
    // registrationDateValue가 null이 아니고 비어있지 않을 때만 파싱을 시도합니다.
    Date registrationDate = dateFormat.parse(registrationDateValue);
    dto.setRegistration_date(registrationDate);
}
} catch (ParseException e) {
    System.out.println("날짜 오류");
    e.printStackTrace();
}
int freeUpdate = freelanceService.freeUpdate(dto);
mav.setViewName("jsonView");
return mav;
}
```

프리랜스 정보 수정(프로필 사진, 프리랜스 정보 수정)

- key 이름의 form-data 요청 파라미터를 param이라는 맵으로 받아옴
profileUpload 이름의 form-data 요청 파라미터를 MultipartFile로 받아옴
프로필 사진이 DB에 이미 존재하는지 확인하고, checkUploadFile 객체에 저장
프로필 사진을 저장할 경로를 설정하고 해당 경로를 폴더로 생성
업로드된 프로필 사진 파일의 원본 이름과 UUID를 사용하여 새로운 파일 이름을 생성
DB에 프로필 사진이 존재하지 않거나 새 파일이 업로드된 경우, 파일을 저장하고 dto
객체의 "path" 필드에 새로운 파일 경로를 설정
기존 프로필 사진이 있는 경우, 해당 파일을 삭제하고 새로운 파일을 저장
FreelanceDto 객체에 프리랜서 정보(이름, 전화번호, 이메일, 위치, 프로젝트, 가격,
성별, 생일, 등록일)를 설정
freelanceService를 사용하여 프리랜서 정보 수정

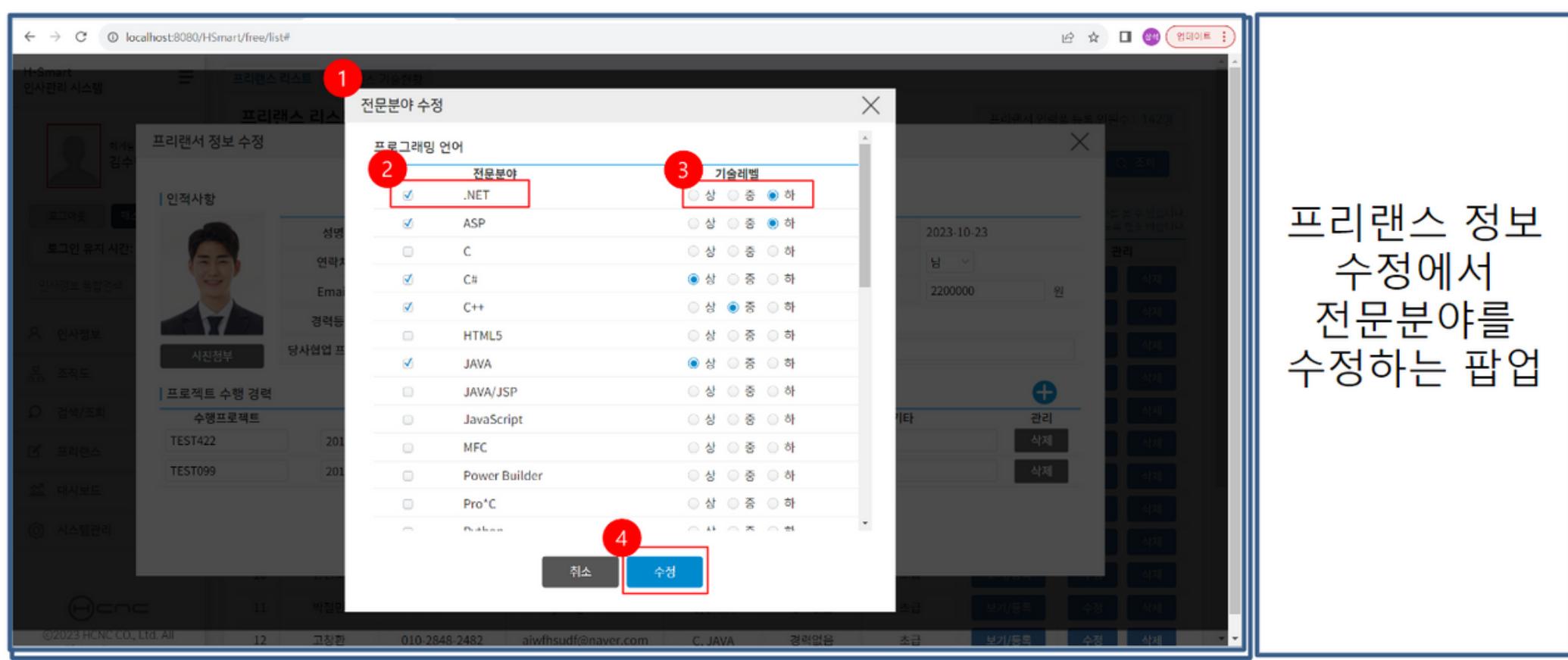
5-7-1. 프리랜스 전문분야 조회

6-1-7. 프리랜스 - 프리랜스 리스트(전문분야 수정)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 프리랜스의 전문분야를 수정하는 팝업 페이지
 2. 각 전문분야들을 불러오며 체크 시 자동으로 기술레벨(상)이 체크 되고, 체크 해제 시 기술레벨도 함께 해제 된다
 3. 각 전문분야마다 기술레벨을 뜻하며 상 중 하 체크시 전문분야도 자동 체크 된다
 4. 전문분야를 선택 후 수정을 누르면 다시 프리랜서 정보 수정 창으로 이동된 후 전문분야 란의 정보가 변경된다

프리랜스 전문분야 수정



- 프리랜스 수정 창에서 전문분야를 클릭하면 공통 코드에 등록되어 있는 스킬들을 카테고리 별로 조회
 - 각 전문분야에는 skill_\${code}, 기술레벨에는 레벨 별 A_\${code}, B_\${code}, C_\${code} 의 값을 가짐
 - checked() 함수를 통해 전문분야 리스트를 조회 할 때 프리랜서가 보유한 스킬과 리스트 상의 전문분야가 일치한다면 체크되어 조회됨
 - checked2() 함수를 통해 프리랜서의 전문분야의 기술레벨과 리스트 상의 기술레벨이 A,B,C에 맞게 자동 체크
 - checkRadio() 함수를 통해 각 전문분야 checkbox에 체크를 하면 radio에도 자동 체크 되고 checkbox 체크를 해제하면 radio도 자동해제

5-7-2. 프리랜스 전문분야 조회

```
//프리랜스 수정 스킬 불러오기
function getUpdateSkillList(freelance_id){
    $.ajax({
        type: "GET",
        url: "/FREEGO/free/getUpdateSkillList.do",
        dataType: "json",
        success: function(data) {
            $('#free_skill_list').empty();
            let prevCategory = null;

            for(let i =0; i < data.skillListData.length; i++){
                let skillCategory = data.skillListData[i].skill_category.replace(/\s+/g, '_'); // 공백을 _로 대체
                if(prevCategory !== skillCategory){
                    prevCategory = skillCategory;

                    let skillTable =
                        `<p>${data.skillListData[i].skill_name}</p>
                        <table class="table-type02 free_skill_table">
                            <caption>${data.skillListData[i].skill_name} 테이블</caption>
                            <col class="wp50">
                            <col class="wp50">
                        </table>
                    `;
                    $('#free_skill_list').append(skillTable);
                }

                let skillTbody = `<tr>
                    <td>
                        <input type="checkbox" name="check_${data.skillListData[i].code}" id="check_${data.skillListData[i].code}" value="${data.skillListData[i].name}" ${checked(i, data) === 'Y' ? 'checked' : ''}>
                        <label for="check_${data.skillListData[i].code}">${data.skillListData[i].name}</label>
                    <td>
                        <input type="radio" name="skill_${data.skillListData[i].code}" id="A_${data.skillListData[i].code}" value="A" ${checked2(i, data) === 'A' ? 'checked' : ''}>
                        <label for="A_${data.skillListData[i].code}">상</label>
                        <input type="radio" name="skill_${data.skillListData[i].code}" id="B_${data.skillListData[i].code}" value="B" ${checked2(i, data) === 'B' ? 'checked' : ''}>
                        <label for="B_${data.skillListData[i].code}">중</label>
                        <input type="radio" name="skill_${data.skillListData[i].code}" id="C_${data.skillListData[i].code}" value="C" ${checked2(i, data) === 'C' ? 'checked' : ''}>
                        <label for="C_${data.skillListData[i].code}">하</label>
                    </td>
                `;
                $('#free_' + skillCategory).append(skillTbody);
            }
        }
    });
}
```

스킬카테고리를 불러와 전에 있던 스킬카테고리와
다르다면 새로 카테고리를 생성

전문분야와 기술레벨을 불러옴

프리랜스 전문분야 조회 기능

- ID값을 매개변수로 ajax를 통해 GET방식으로 요청 후 데이터를 받는다
- 스킬카테고리를 불러와 전에 있던 스킬카테고리와 다르다면 새로 카테고리를 생성
- checked(), checked2() 함수를 통해 프리랜스의 전문분야에 따라 체크되어 조회

```

        }
        return 'N';
    }

    //레벨 체크
    function checked2(i, data) {
        for (let j = 0; j < data.skillSelectList.length; j++) {
            if (data.skillListData[i].code == data.skillSelectList[j].code) {
                switch (data.skillSelectList[j].level) {
                    case 'A':
                        return 'A'; // 'A'에 해당하는 경우 상 라디오 버튼을 체크
                    case 'B':
                        return 'B'; // 'B'에 해당하는 경우 중 라디오 버튼을 체크
                    case 'C':
                        return 'C'; // 'C'에 해당하는 경우 하 라디오 버튼을 체크
                    default:
                        return 'N'; // 다른 경우는 'N'을 반환하여 체크하지 않음
                }
            }
        }
        return 'N';
    }

    checked(), checked2() 함수를 통해 프리랜스의 전문분야에 따라 체크되어 조회
}

//체크박스 라디오 동시 선택
function checkRadio(){
    $('input[type="checkbox"]').on('change', function () {
        let code = $(this).attr("id").replace("check_", "");

        if ($(this).is(":checked")){
            // 체크박스가 선택된 경우 해당 스킬 라디오 버튼도 선택
            $('input[name="skill_' + code + '"]').first().prop('checked', true);
        } else {
            // 체크박스가 선택 해제된 경우 해당 스킬 라디오 버튼도 선택 해제
            $('input[name="skill_' + code + '"]').prop('checked', false);
        }
    });

    $('input[type="radio"]').on('change', function() {
        let code = $(this).attr('id').split('_')[1]; // _ 다음에 오는 부분을 코드로 주출
        let level = $(this).attr('id').split('_')[0]; // _ 이전에 오는 부분을 레벨로 주출
        if ($(this).is(':checked')){
            // 라디오 버튼이 선택된 경우 해당 스킬 체크박스도 선택
            $('input[name="check_' + code + '"]').prop('checked', true);
        }
    });
}
checkbox와 radio 동시 선택

```

5-7-3. 프리랜스 전문분야 수정

```

프리랜서 수정 시 체크된 전문분야와 기술레벨을 담을 배열을
checkedSkills 전역변수로 선언

전문분야 수정에서 체크를 하고 취소를 눌렀을 때 그 전의 전문분야
와 기술레벨을 담을 배열을 originalSkillStates 전역변수로 선언

//체크되어 있는 스킬들 배열 담아 보내기
function skillCheckBox(freelance_id){
    let skillView = [];
    checkedSkills = [];
    originalSkillStates = [];

    //체크되어 있는 스킬들 배열 담아 보내기
    $('input[type="checkbox"]').each(function() {
        if ($(this).is(":checked")) {
            let code = $(this).attr("id").replace("check_", "");
            let level = $('input[name="skill_' + code + '":checked]').attr("id").replace("_"+code,"");
            checkedSkills.push({ code: code, level: level, freelance_id : freelance_id });
            if (!skillView.includes($(this).val())) {
                skillView.push($(this).val());
            }
        }
    });
    let selectedSkillsString = skillView.join(", ");
    $("#skill1").val(selectedSkillsString);
}

// 전역 변수인 checkedSkills에 담겨져 있는 걸 보냄
function getCheckedSkills(freelance_id, checkedSkills){
}

```

```

$.ajax({
    type: "POST",
    url: "/FREEGO/free/updateCheckSkill.do",
    data: JSON.stringify(checkedSkills),
    contentType: "application/json;charset=UTF-8",
    dataType: "json",
    success: function(data) {
    },
    error :function(error) {
        console.log(error);
    }
});

```

```

//원래의 전문분야 상태를 저장해주는 함수
function saveOriginalSkillState() {
    originalSkillStates = [];
    $('input[type="checkbox"]').each(function() {
        if ($(this).is(":checked")) {
            let code = $(this).attr("id").replace("check_", "");
            let level = $('input[name="skill_' + code + '":checked]').attr("id").replace("_"+code,"");
            originalSkillStates.push({ code : code, level : level });
        }
    });
}

//원래의 전문분야 상태를 부르는 함수
function restoreOriginalSkillState() {
    $('input[type="checkbox"]').prop('checked', false);
    $('input[type="radio"]').prop('checked', false);

    originalSkillStates.forEach((state) => {
        $('input[name="check_'+state.code+']').prop('checked', true);
        $('input[name="skill_'+state.code+'][value="'+state.level+'"]').prop('checked', true);
    });
}

```

```

@RequestMapping(value = "/updateCheckSkill.do", method = RequestMethod.POST)
public ModelAndView updateCheckSkill(@RequestBody List<FreelanceSkillDto> checkedSkills ) {
    ModelAndView mav = new ModelAndView();

    int deleteFreeSkill = freelanceService.deleteFreeSkill(checkedSkills.get(0).getFreelance_id());
    freelanceService.deleteFreeSkill(checkedSkills.get(0).getFreelance_id());
    for (int i =0; i<checkedSkills.size(); i++) {
        freelanceService.insertFreeSkill(checkedSkills.get(i));
        int insertFreeSkill = freelanceService.insertFreeSkill(checkedSkills.get(i));
    }
    mav.setViewName("jsonView");
    return mav;
}

```

프리랜스 전문분야 수정 기능

- 프리랜서 수정 시 체크된 전문분야와 기술레벨을 담을 배열을 checkedSkills 전역변수로 선언
- 전문분야 수정에서 체크를 하고 취소를 눌렀을 때 그 전의 전문분야와 기술레벨을 담을 배열을 originalSkillStates 전역변수로 선언
- 반복문을 통해 체크되어 있는 전문분야와 기술레벨을 key value 형식으로 배열에 담는다
- getCheckedSkills() 함수를 통해 checkedSkills 배열을 보냄
- 배열을 Controller에서 받아 freelance_id에 있는 skill들을 삭제하고 checkedSkills에 있는 전문분야와, 기술레벨을 추가

5-8. 프리랜스 삭제

6-1-8. 프리랜스 - 프리랜스 리스트(프리랜스 삭제)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

- 삭제 버튼을 누르면 해당 프리랜서의 정보를 삭제 할 것인지를 묻는 알림창이 뜬다
- 확인 버튼을 누르면 프리랜스 정보는 삭제되고 취소를 누르면 원래대로 돌아온다

No	이름	연락처	이메일	전문분야	경력	경력등급	평가의견
1	dsf	122-1212-2112	2112sd@saf.cd	null	1년 4개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
2	고길동	010-2838-2382	awidfy22@naver.com	.NET, ASP, C#...	6년 9개월	중급	<button>보기/등록</button> <button>수정</button> 삭제
3	고정석	010-2848-2482	awuehfafw@naver.com	.NET, ASP, C, ...	4개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
4	강의명	010-2842-8424	aiwfhasdyf@naver.com	C, JAVA, Java...	2개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
5	리종석	010-2842-8482	aiwfusdfh@naver.com	C, C++	1년 2개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
6	안사유	010-3125-1251	asfwef2@gmail.com	HTML5, JAVA, ...	경력없음	초급	<button>보기/등록</button> <button>수정</button> 삭제
7	전현모	010-2842-8472	awihfsdf@naver.com	ASP, C	경력없음	초급	<button>보기/등록</button> <button>수정</button> 삭제
8	가나다	010-4838-4284	awifehawif@naver.com	전자정부 프레...	경력없음	초급	<button>보기/등록</button> <button>수정</button> 삭제
9	강호	010-8417-2428	awehfiasdf@naver.com	.NET, ASP, C, ...	5개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
10	황조동이	010-8482-1234	ghkdwhfhd@naver.com	.NET, ASP, C, ...	1년 10개월	초급	<button>보기/등록</button> <button>수정</button> 삭제
11	전현모	010-4274-7274	awifhasdyf@naver.com	C, JAVA	경력없음	초급	<button>보기/등록</button> <button>수정</button> 삭제
12	비질민	010-4827-4274	awufydsf@naver.com	C, JAVA	경력없음	초급	<button>보기/등록</button> <button>수정</button> 삭제

프리랜스 정보
수정에서
전문분야를
수정하는 팝업

```
//프리랜스 삭제
function deleteFreelance(freelance_id){
    let result = confirm("프리랜스 정보를 삭제하시겠습니까?");
    if(result == true){
        $.ajax({
            type: "POST",
            url: "/FREEGO/free/deleteFreelance.do",
            data:{freelance_id:freelance_id},
            dataType: "json",
            success: function(data) {
                alert("프리랜스 삭제 완료")
                getList(currentPage, option, keyword);
            },
            error: function(data){
                console.log(error);
            }
        })
    }else{
        return false;
    }
}
```

freelanceMapper.xml

```
<!-- 21. 프리랜스 삭제 -->
<delete id="deleteFreelance" parameterType="int">
    DELETE FROM FREELANCE_RATE WHERE freelance_id = #{id};
    DELETE FROM FREELANCE_CAREER WHERE freelance_id = #{id};
    DELETE FROM SKILL WHERE freelance_id = #{id};
    DELETE FROM FREELANCE WHERE id = #{id};
</delete>
```

프리랜스 평가의견 보기

- 리스트에서 받은 ID값을 매개변수로 ajax를 통해 POST방식으로 요청
- ID를 통해 프리랜스 관련 데이터를 FK > PK 순으로 삭제

5-9. 프리랜스 기술현황

6-2-1. 프리랜스 - 프리랜스 기술현황

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

1. 좌측 메뉴 탭에 프리랜스 -> 프리랜스 기술현황을 클릭하거나 프리랜스 탭에서 프리랜스 기술현황을 누르면 해당 화면이 나옴
2. 프리랜스 신규 등록(6-3-1.)
3. 스킬분야, 구분, 기술레벨(상, 중, 하), 기술인력 합계를 클릭하면 해당하는 기술조건에 맞는 프리랜서의 정보가 나온다

스킬분야	구분	기술레벨(상)	기술레벨(중)	기술레벨(하)	기술인력 합계
	.NET	14	2	2	18
	ASP	11	4	3	18
	C	26	15	4	45
	C#	18	9	2	29
	C++	14	10	4	28
	HTML5	2	8	2	12
	JAVA	38	24	8	70
	JAVA/JSP	18	12	6	36
	JavaScript	8	13	1	22
	MFC	1	1	1	3
	Power Builder		1	1	2
	Pro*C	1	1		2
	Python	4	4	2	10
	VB	8	1		9
	DB2		1		1
	MS-SQL	1	2	3	

[그림 6-2-1] 프리랜스 - 프리랜스 기술현황

```
//프리랜스 경향에 스킬과 스킬등급별로 데이터를 불러옴
$(document).ready(function() {
  $.ajax({
    type: "GET",
    async: true,
    url: "/FREEGO/free/status.do",
    dataType: "json",
    success: function(data) {
      let tableBody = $('#free_status_table tbody');
      tableBody.empty();

      let categoryCounts = {} // 각 카테고리 별 행 수를 저장할 객체
      for (let i = 0; i < data.skillData.length; i++) {
        let item = data.skillData[i]; //스킬 데이터 item에 스킬데이터 담음
        let category = item.skill_category;

        if (!categoryCounts.hasOwnProperty(category)) {
          categoryCounts[category] = 0;
        }

        categoryCounts[category]++; // 해당 카테고리의 행 수 증가
      }

      let prevCategory = null; // 이전 스킬 분야 카테고리
      for (let i = 0; i < data.skillData.length; i++) {
        let item = data.skillData[i];
        let row = $(`<tr>`);

        if (prevCategory !== item.skill_category) {
          prevCategory = item.skill_category;
          var rowspanAttr = `rowspan="${categoryCounts[item.skill_category]} + ""`;
          row.append(`<td ${rowspanAttr} onclick="getFreeSkill('${item.code}', '${item.level_A}', '${item.level_B}', '${item.level_C}')>${item.skill_name}</td>`);
        }

        row.append(`<td class="skill_name">${item.skill_name}</td>`);
        row.append(`<td onclick="getFreeSkill('${item.code}', '${item.level_A}', '${item.level_B}', '${item.level_C}')>${item.skill_level_A}</td>`);
        row.append(`<td onclick="getFreeSkill('${item.code}', '${item.level_A}', '${item.level_B}', '${item.level_C}')>${item.skill_level_B}</td>`);
        row.append(`<td onclick="getFreeSkill('${item.code}', '${item.level_A}', '${item.level_B}', '${item.level_C}')>${item.skill_level_C}</td>`);

        row.append(`<td>${item.skill_sum}</td>`);
        tableBody.append(row);
      }
    }
  })
})
```

프리랜서들의
보유 중인
기술들을 레벨
별 기술 별로
보여주는
페이지

프리랜스 기술현황

- 스킬 카테고리 별 행 수를 저장하기 위해 categoryCounts 객체를 만들어 각 카테고리가 몇 개의 행을 가지는지 추적
- 이전 스킬 분야 카테고리를 추적하는 prevCategory 변수를 만들어 새로운 카테고리가 시작될 때, rowspan 속성을 설정하기 위한 기준으로 사용
- 첫 번째 for문은 스킬 데이터를 반복하여 스킬 카테고리별로 테이블 행을 생성
- 두 번째 for문은 스킬 데이터를 다시 반복하며, 각 스킬 항목을 테이블에 추가
- 이전 카테고리와 현재 카테고리를 비교하여 rowspan 속성을 설정하여 카테고리 이름이 여러 행을 병합
- 각 셀마다 getFreeSkill() 함수를 통해 눌렀을 때 해당 전문분야를 가진 프리랜스를 조회

5-10. 프리랜스 기술현황 상세정보 팝업

6-2-2. 프리랜스 - 프리랜스 기술현황(프리랜스 등록인원 상세정보 팝업)

부서장 이상의 권한을 가진 사용자가 로그인 해야 함

- 선택한 정보에 맞는 프리랜스 등록인원 상세정보 팝업을 보여준다(전문분야, 기술등급, 이름, 연락처, 이메일)

No	전문분야	기술등급	이름	연락처	이메일
1	.NET	상	강호	010-8417-2428	awehfiasdf@naver.com
2	.NET	상	곤잘레스	010-9092-0090	tkemfkdl@naver.com
3	.NET	상	감안해	010-8428-4828	aifwhfia@naver.com
4	.NET	상	구만해	010-4942-8284	awkefhawifw@naver.com
5	.NET	상	정진총	010-2484-2842	insane guy@naver.com
6	.NET	상	김덕배	011-1113-1434	nmkrmkmt@gmail.com
7	.NET	상	김종우	010-2482-1234	awifhwf@naver.com
8	.NET	상	문동규	010-2948-2472	rmlcsakror@naver.com
9	.NET	상	김순자	232-3623-3333	236326326@gmail.com
10	.NET	상	강형욱	010-4400-0201	010das@nate.com
11	NET	사	기기기	010-7419-1024	awifhwf30@naver.com

[그림 6-2-2] 프리랜스 - 프리랜스 등록인원 상세정보 팝업

전문분야
조건에 맞는
프리랜서들을
보여주는
상세정보 팝업
페이지

```
//스킬 현황 불러오기
function getFreeSkill(skill_category, skill_level, skill_name){

$.ajax({
    type: "GET",
    async: true,
    url: "/FREEGO/free/getSkill.do",
    data: {
        skill_category: skill_category,
        skill_level: skill_level,
        skill_name: skill_name
    },
    dataType: "json",
    success: function(data) {
        $('#premenModal').show();
        let str = ''; // HTML 문자열을 담을 변수
        let no = 1; //프리랜서 순서 몇번
        $('.scrollable-table').scrollTop(0);
        for (let i = 0; i < data.skillDetailList.length; i++){
            // 각 데이터를 행으로 추가
            str += '<tr>' +
                '<td>' + (i+1) + '</td>' +
                '<td>' + data.skillDetailList[i].skill_name + '</td>' +
                '<td>' + data.skillDetailList[i].skill_level + '</td>' +
                '<td>' + data.skillDetailList[i].name + '</td>' +
                '<td>' + data.skillDetailList[i].telephone + '</td>' +
                '<td>' + data.skillDetailList[i].email + '</td>' +
                '</tr>';
        }
        $('.freeSkillDetail').html(str);
    },
    error: function(data) {
        console.log(data);
    }
})
}
```

```
freelanceMapper.xml
<!-- 8. 프리랜스 현황 상세보기 -->
<select id="selectSkillDetail" resultType="FreelanceDto">
    SELECT
        cc.super_code AS skill_category,
        cc.name AS skill_name,
        f.name AS name,
        (
        CASE
            WHEN s.level = 'A' THEN '상'
            WHEN s.level = 'B' THEN '중'
            WHEN s.level = 'C' THEN '하'
            ELSE '스킬없음'
        END
        ) AS skill_level,
        f.telephone AS telephone,
        f.email AS email
    FROM COMMON_CODE sc
    JOIN COMMON_CODE cc ON sc.code = cc.super_code
    LEFT JOIN SKILL s ON cc.code = s.skill_code
    LEFT JOIN FREELANCE f ON f.id = s.freelance_id
    WHERE
        sc.super_code = 'SOFT' and s.id_type = 1
        AND cc.super_code = #{skill_category}
        AND s.level like '%' + #{skill_level} + '%'
        AND cc.super_code IN ('DB', 'LANGU', 'FRAME', 'SERVE')
        <if test="skill_name != null and skill_name != ''">
            AND cc.name = #{skill_name}
        </if>
        ORDER BY
            skill_name asc, skill_level asc, f.id desc;
</select>
```

프리랜스 기술현황 상세정보 팝업

- skill_category, skill_level, skill_name 매개변수를 가지고 ajax를 통해 GET방식으로 요청
- skill_category, skill_level, skill_name을 통해 해당하는 스킬 카테고리, 스킬 이름, 프리랜스의 이름, 연락처, 이메일을 조회

감사합니다