7dev 조달 시스템 프로젝트

팀 이름: 7dev

프로젝트 주제: 조달 시스템 설계

자재 입고 기능 담당 : 허재연

깃허브: https://github.com/heo-jaeyeon/7dev

👨‍💼개요

본 프로젝트는 **기업의 자재 수급 및 구매 업무를 디지털화** 하여 **업무 효율성 향상, 업무 효율성 향상, 데이터 기반 의사결정, 투명한 거래 이력 관리**를 실현합니다.

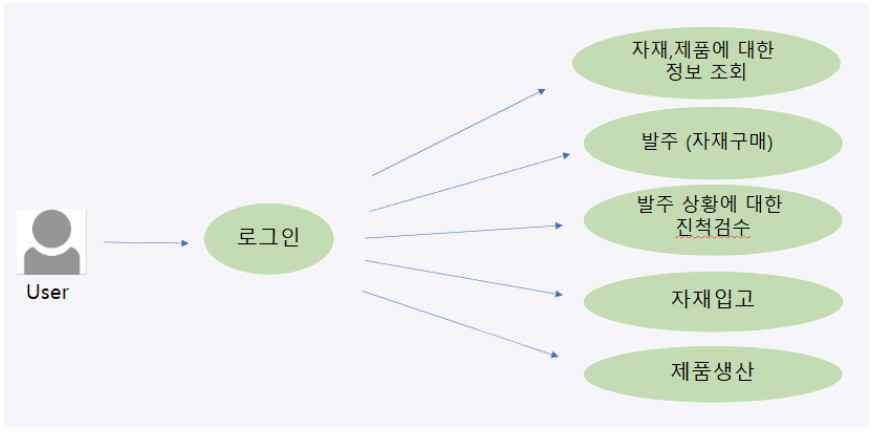
또한, **발주 → 입고 → 계약 → 거래명세서 발행**까지의 전 과정을 하나의 시스템으로 통합함으로써, **운영 비용 절감** 및 **업무 표준화**를 달성하는 것을 목표로 합니다.

📆 개발 일정

단계	기간	주요 활동
요구 사항 분석	2025.03.17 ~ 2025.03.21	업무 프로세스 분석, 요구 사항 정의
설계	03.22 ~ 03.28	시스템 설계, DB 모델링, UI/UX 설계
구현	03.29 ~ 04.15	코드 개발, 단위 테스트
테스트	04.16 ~ 04.18	통합 테스트, 시스템 테스트
문서화 및 발표	04.19 ~ 04.20	개발 문서 작성, 발표 준비

💫기능

주요 기능



담당 기능

- 업체에게 받은 자재 검수 후 **입고 마감 처리**
- 입고 마감 처리가 된 자재들을 **창고에 입고**
- 창고별 **자재 리스트 조회**

👨‍💻기능 구현 화면

1. 로그인 화면

기능 위치: http://localhost:8080/chill/

기능 설명: 사용자가 **로그인하는 기능**입니다. 로그인 시 아이디와 비밀번호는 **DB**에 미리 구성해 놓은 **'emp테이블'**의 존재하는 **'emp_id'**와 **'emp_pw'**의 데이터 값을 이용해 로그인 할 수 있습니다.

nt emp_no (PK)	varchar emp_name	varchar emp_job	varchar emp_id	varchar emp_pw
1	심원섭	시스템 관리자	admin01	admin1234
2	허재연	구매팀장	buyer01	buy1234
3	김효정	자재관리 사원	material01	mat5678
4	박재영	생산부 과장	prod01	prod999
5	진실	품질검수 담당자	qc01	qc2024
6	한예성	R&D 엔지니어	rnd01	rnd8888
7	이수연	총무/회계 팀장	acc01	acc123

핵심 구현 내용:

- 로그인 화면에서 입력한 값을 **'LoginController'**에서 `@RequestParam Map<String, Object> map`으로 받고 `Map<String, Object> user = loginService.login(map)` **'loginService'**를 통해 **DB**에 존재하는 데이터 중 입력 받은 아이디와 비밀번호 값이 **같은지 비교 후** 존재하면 **로그인 성공**, 존재하지 않으면 **로그인 실패**가 됩니다.

2. 전체 메뉴

기능 위치: 로그인 화면 → 전체 메뉴

기능 설명: **'emp_main.jsp'**에 결과화면으로 로그인하면 원하는 기능을 선택할 수 있는 메뉴가 나옵니다.

핵심 구현 내용:

- 원하는 메뉴를 선택하면 `<a href="${contextPath}/...">` 을 통해 해당 기능을 사용할 수 있는 화면이 나오게 됩니다.

3. 입고 전 자재 통합 관리

기능 위치: 전체 메뉴 → 자재 통합 관리

기능 설명: 업체로부터 도착한 **자재들의 상태를 확인**하고, **이상 여부 검수 후 구매 마감**을 처리하는 기능입니다.

핵심 구현 내용:

- `mstorage_in_date` 컬럼 값을 기준으로 **삼항 연산자**를 사용하여, **null**이면 **'배송 중'**, **값이 존재**하면 **'도착 완료'** 상태로 표시되도록 구현하였습니다.
- 체크박스를 통해 마감할 자재들을 선택한 후 **[구매 마감]** 버튼 클릭 시, `purc_order_status`의 값이 **'0→1'**로 변경되며 구매 마감 상태가 **'완료'**로 표시됩니다.
- 마감된 자재는 이후 자재 입고 화면에서 입고 처리를 할 수 있도록 흐름이 연결됩니다.
- 이외에도 자재가 많을 경우를 대비하여 **검색 필터 기능**을 추가해 사용성을 개선하였습니다.

4. 자재 입고

기능 위치: 전제 메뉴 → 자재 입고 관리

기능 설명: '구매 마감' 단계에서 '**구매 마감'된 자재들만** 창고에 **입고**가 가능하며, 입고할 자재들을 **체크 박스**로 선택 후 **'자재 등록'** 버튼을 누르면 선택했던 자재들의 정보들이 **팝업 창**으로 나타나고 입고 시키고자 하는 **창고의 코드 번호를 선택**해 **입고**를 처리할 수 있습니다.

핵심 구현 내용:

- `<input type="checkbox" class="materialChk">`를 통해 **자재별로 체크박스 선택**이 가능합니다.
- `data-*` **속성을 이용**하여 자재 번호, 자재명, 코드 등의 상세 정보를 HTML 요소에 **바인딩**해 선택된 자재 정보를 **팝업창으로 전달**합니다.
- `${incoming.mstorage_in_date eq null ? '배송중' : '도착완료'}`를 통해 **입고일 여부에 따라** 자재 상태를 동적으로 표시함으로써 UI 상에서 실시간 상태 확인 가능합니다.
- 체크된 자재들의 정보를 **자바스크립트로 수집**하여 `openModal()` **함수**로 **팝업창에 전달**합니다.
- 자재 목록은 `data-no`, `data-name`, `data-code` 등으로 구성되어 `dataset`을 통해 접근합니다.

5. 창고별 자재 리스트 조회

기능 위치: 전제 메뉴 → 창고별 자재 현황

기능 설명: '자재 입고'단계에서 입고 된 자재들을 창고마다 어떤 자재들이 있는지 검색 기능을 통해 좀 더 빠르고 정확하게 확인할 수 있도록 구현했습니다.

핵심 구현 내용:

- 아래와 같이 옵션을 통해 창고 코드별로 검색할 수 있게 구성했습니다. `<select name="mstorage_code" class="form-select" required> <option value="MW001" <` **`c:if`** `test="${mstorage_code eq 'MW001'}">selected</` **`c:if`** `>>MW001</option>...</select>`

🚀트러블 슈팅

문제상황 1

'자재 입고'에서 입고 하고자 하는 자재들을 체크 박스로 선택 후 **팝업창**에 선택한 **자재들의 정보가 띄워져야 하는데 보여지지 않는 문제**가 생겼습니다.

문제 해결

'expected.jsp'에서 **input type="checkbox"**를 사용 할 때 자재 번호를 나타내는 **'material_no'**을 기준으로 선택했는데 데이터를 넘길 때, **'material_no'**뿐만 아니라 **'material_name', 'material_code'** 등의 데이터를 함께 넘겼어야 했는데 **'material_no'** 값만 **팝업창**에 보내려 해서 자재 정보가 보이지 않았던 것입니다. 그래서 저는 아래와 같이 필요한 데이터를 묶어 **"materialChk"**라는 **클래스 이름을 지정**했습니다.

```
<td><input type="checkbox" class="materialChk"
data-no="${incoming.material_no}"
data-name="${incoming.material_name}"
data-code="${incoming.material_code}"
...></td>
```

그 후 <script>부분에 선택한 자재 정보들을 **팝업창**에 불러왔고,

```
function openModal() { let checked = document.querySelectorAll('.materialChk:checked');
```

'forEach'를 활용해 자재 정보를 성공적으로 나타나게 할 수 있었습니다.

```
checked.forEach(chk => {let no = chk.dataset.no;
let name = chk.dataset.name;
let code = chk.dataset.code;...});
```

문제상황 2

'자재 입고'단계에서 파업 창에 띄웠던 자재들을 **'최종 입고'**를 하면 **"입고 완료"**라고 안내가 되지만 **실제 DB에는 데이터가 넘어 오질 않는 문제**가 있었습니다.

문제해결

'expected.jsp'에서 요청한 데이터 값을 **리스트**로 받기 위해 **'IncomingController'**에서 아래와 같이 구성했었습니다.

```
public String materialInProcess(@ModelAttribute("incomingDTOList") List<IncomingDTO>) {...}
```

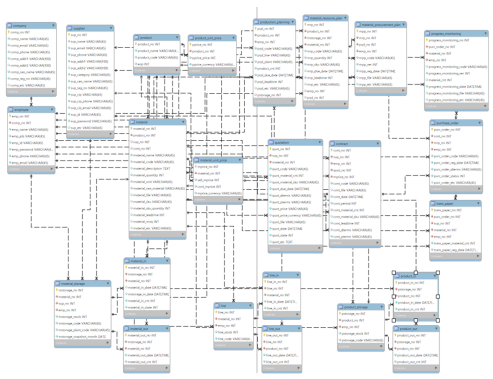
하지만 `@ModelAttribute`는 내부적으로 빈 객체를 생성한 뒤 거기에 데이터를 채워 넣는 구조이고 **Spring**은 `List`라는 인터페이스 타입을 직접 생성할 수 없었기 때문에 **'Wrapper'클래스**를 만들어 아래와 같이 구성했습니다.

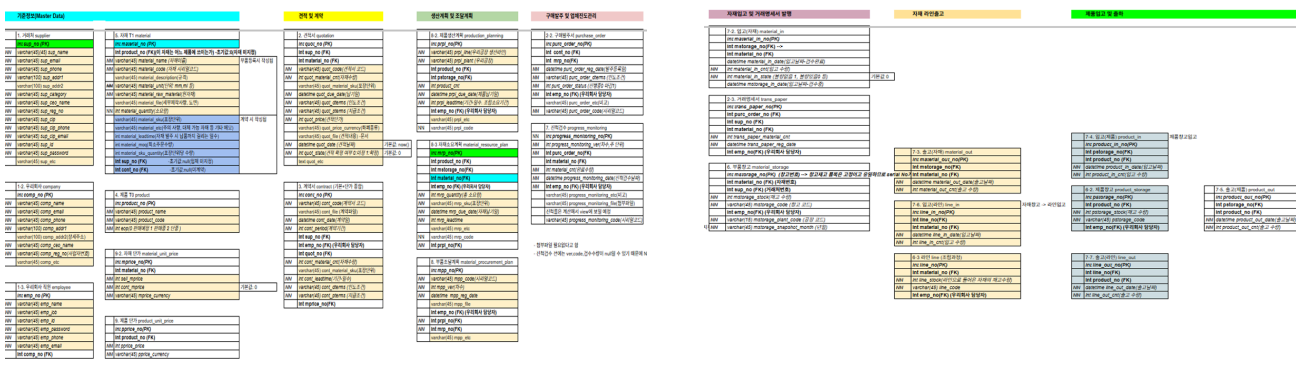
```
public class IncomingWrapperDTO {
private List<IncomingDTO> incomingDTOList;
// getter/setter
public List<IncomingDTO> getIncomingDTOList() {
return incomingDTOList; }
public void setIncomingDTOList(List<IncomingDTO> incomingDTOList) {
this.incomingDTOList = incomingDTOList; }
}
```

그 후 아래와 같이 수정해서 **성공적으로 자재 데이터를 DB에 저장**할 수 있었습니다.

```
public String materialInProcess(@ModelAttribute IncomingWrapperList wrapper) {
List<IncomingDTO> list = wrapper.getIncomingDTOList();
...}
```

📝DB 설계





👨‍💻사용 스택

백엔드	Java 17, Spring Framework, MyBatis, Apatch Tomcat9.0
프론트엔드	JSP, JavaScript / jQuery
데이터베이스	MySql
개발 도구	STS3, Git

- **Java**는 제가 처음 접한 언어로, 객체 지향적 구조 덕분에 **재사용성과 유지보수에 유리**하다는 점에서 선택했습니다.
- **Spring Framework**는 기존에 구현되어 있는 **구조(MVC 패턴)를 활용**하여 개발자 역량과 관계없이 효율적인 개발이 가능한 점에서 사용했습니다.
- **MyBatis** 개발자가 직접 SQL 쿼리를 작성하고 매핑하는 방식으로 직접적인 SQL제어로 JOIN과 같이 **복잡한 쿼리나 성능 튜닝에 유용한 점**에서 사용했습니다.
- **Apatch**는 정적인 자원을 처리, **Tomcat**은 동적인 자원을 처리하는 **WAS**로 **빠른 실행**과 **쉬운 설정**, **Spring MVC와의 호환성**이 뛰어난 점에서 사용했습니다.
- **JSP**는 **Java 코드와 HTML을 함께 사용**할 수 있어 **동적 데이터 표현**에 유리하고 **Model객체와 쉽게 연동 가능**하다는 점에서 사용했습니다.
- **JavaScript/jQuery**는 **모달창, 유효성 검사, 체크박스 일괄 처리 등 사용자 인터랙션 기능을 구현**하였고, **jQuery**의 AJAX 기능을 활용하여 **비동기 통신**을 처리했습니다.
- **MySql**은 **Spring+MyBatis와의 연동이 용이**하고 조달 시스템에서 발생하는 모든 데이터를 **저장 및 조회**하기 위해 사용했습니다.
- **STS3**는 **Spring 개발에 최적화된 통합 개발 환경**으로, XML 기반 설정과 Bean 관리 등 프로젝트 구성을 직관적으로 관리할 수 있었습니다.
- **Git**은 팀원 간 **협업**을 위한 버전 관리 도구로 사용하였으며, GitHub를 통해 브랜치 전략, 이슈 관리, 코드 리뷰 등 **협업 프로세스를 체계화**했습니다.

👨‍💼이 프로젝트를 하며 느낀 점

이번 프로젝트는 제가 처음으로 팀원들과 함께 진행한 **협업 기반 팀 프로젝트**였습니다. 처음에는 '조달 시스템'이라는 주제가 익숙하지 않아 전체적인 흐름을 이해하는 데 어려움이 있었지만, 팀원들과 함께 자료를 조사하고 정보를 공유하면서 점차 시스템 구조와 업무 흐름을 파악할 수 있었습니다. 개발 과정에서도 문제가 발생할 때마다 서로의 코드를 확인하고 의견을 나누며 **효율적으로 디버깅**을 진행할 수 있었고, 그 덕분에 프로젝트 발표에서 우수상을 받으며 성공적으로 마무리할 수 있었습니다. 이번 프로젝트를 통해 **협동심과 의사소통 능력의 중요성**을 다시 한 번 느꼈으며, 이 두 가지 역량이 실무에서도 핵심적인 역할을 한다는 점을 체감했습니다. 앞으로도 이 경험을 바탕으로 더욱 원활하게 팀 프로젝트를 수행할 수 있을 것이라는 자신감을 얻었습니다.