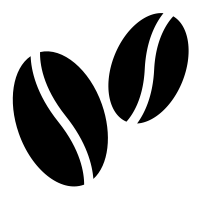
**커피 체인점 관리(ERP) 프로그램**

****

**TripleC**

**Coffee.Chain.Connector**

* **KOSTA Java Framework & 안드로이드 개발자 양성과정 99기**
* **교육 기간: 2015.06.29 ~ 2015.12.11**
* **지도강사 명: 이도연**

|  |
| --- |
| ‘어울림’ 조  김예현(조장), 유성현(부조장),  고준, 박상현, 신승택, 조항덕 |

|  |  |
| --- | --- |
| **차례** |  |
|  |  |
| **PART\_ONE** | **프로젝트 개요** |
|  |  |
|  | * 1. 프로젝트 주제 |
|  | * 1. 주제를 선정한 이유 |
|  | * 1. STP 전략 수립   2. 프로젝트 개발 목표 |
|  |  |
| **PART\_TWO** | **프로젝트 추진 일정** |
|  |  |
|  | 2-1. 프로젝트 개발 일정 |
|  |  |
| **PART\_THREE** | **프로젝트 조직** |
|  |  |
|  | 3-1. 팀원 역할과 책임 |
|  |  |
| **PART\_FOUR** | **시스템 구조** |
|  |  |
|  | 4-1. 적용기술 |
|  | 4-2. S/W 구조 |
| **차례** |  |
| **PART\_FIVE** | **분석/설계 산출물** |
|  |  |
|  | 5-1. Use Case 모델 |
|  | 5-2. Class 모델 |
|  | 5-3. Data 모델 |
|  | 5-4. 메뉴 구조도 |
| **PART\_SIX** | **구현 산출물** |
|  |  |
|  | 6-1. 주요 구현 사항에 대한 코드 설명 |
| **PART\_SEVEN** | **프로젝트 주요 이슈와 해결방안** |
|  |  |
|  | 7-1. 주요 이슈와 해결방안 |
| **PART\_EIGHT** | **결론(lessoned learned, best practices)** |
|  |  |
|  | 8-1. 결론 |

**PART**

**ONE**

|  |
| --- |
| 프로젝트 개요 |

프로젝트 주제를 소개하고 주제를 선정한 동기와 목적,

그리고 개발 목표에 대해 이야기 합니다.

|  |
| --- |
| 1-1. 프로젝트 주제 |

**‘체인점 관리 시스템’**

우리 프로젝트의 주제는 커피 체인점을 \*ERP 시스템과 접목시킨 것으로, 본사와 여러 개의 체인점으로 구성된 기업의 업무 과정을 통합적으로 관리하는 시스템이라고 할 수 있습니다.

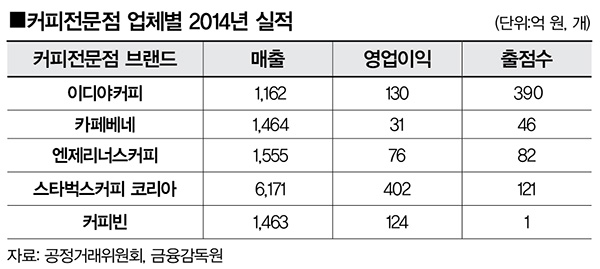
\* ERP는 Enterprise Resource Planning의 약자로, 흔히 전사적 자원 관리를 뜻합니다.

|  |
| --- |
| 1-2. 주제를 선정한 이유 |

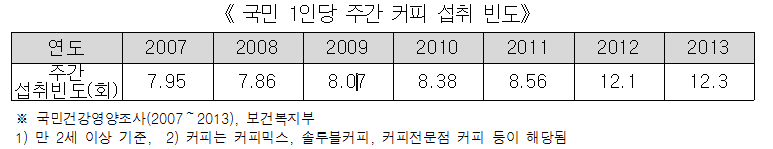
팀원들과 아이디어 회의를 진행하면서 팀원들이 기업의 업무 흐름을 효율적으로 조절해주는 전산 시스템인 ERP시스템에 흥미를 가지고 있다는 것을 알게 되었습니다. 세부적으로 어떤 기업을 선택할까 고민하던 중에는 국내 커피 시장 규모가 점점 증가하는 추세라는 것을 알게 되어 커피 전문점을 선택하게 되었고, USER를 위한 사이트도 추가로 만들어 고객의 참여를 증진시키는 서비스도 만들어 보기로 했습니다.

|  |
| --- |
| 1-3. STP 전략 수립 |

* **Segmentation(시장 세분화): 커피 전문점**



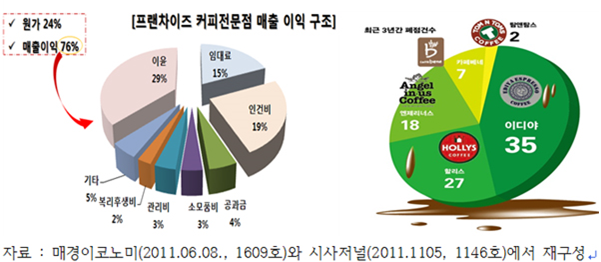
* 녹림축산식품부의 조사에 따르면, 2013년 국민들의 커피 주당 소비빈도는 12.3회, 단일 음식 중 1등(김치, 쌀밥보다 앞선 수치)을 차지 하고 있으며, 국내 커피 시장의 규모는 2012년 약1조 6천억원 이며, 5년간 92%의 성장세를 보이고 있을 정도로 큰 시장규모를 가지고 있다. 하지만, 국내 커피시장은 이미 포화상태라고 할 수 있을 정도로 많은 수의 커피 전문점들이 자리 잡고 있다.



* 국내 커피 전문점들은 크게 개인 커피 전문점과, 체인 커피전문점(브랜드)으로 나뉜다.

|  |
| --- |
| 1-3. STP 전략 수립 |

* **Targeting(목표 시장 설정): 체인 커피전문점**

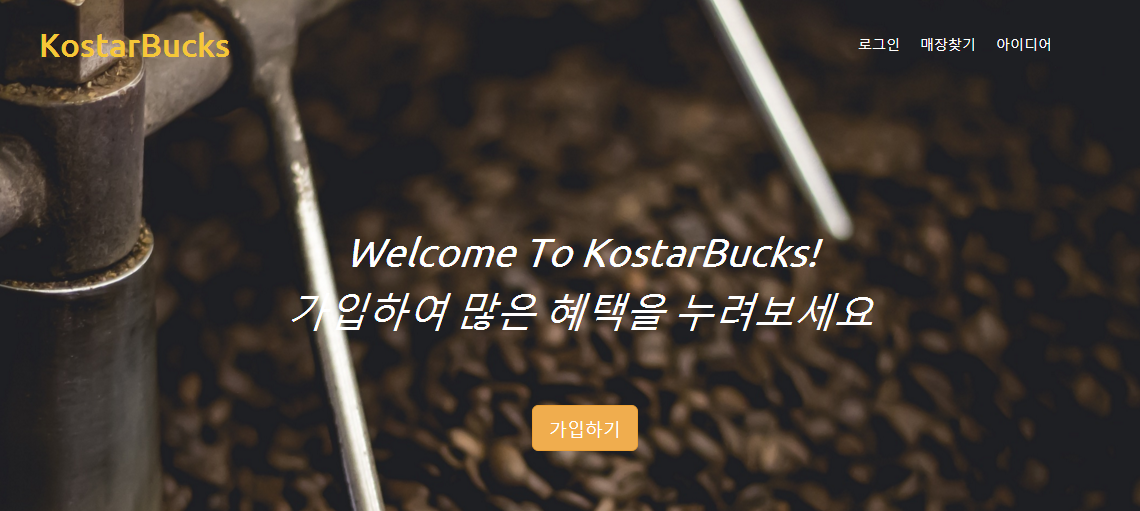




* 개인 커피 전문점을 위한 사이트도 좋지만, 우리의 개발 목표인 ERP시스템 개발을 위해서, 그리고 수익성, 발전 가능성 등을 고려해 봤을 때, 체인 커피 전문점을, 목표 시장으로 설정한다.

|  |
| --- |
| 1-3. STP 전략 수립 |

* **Positioning(포지셔닝): 타사와 다른 조금 더 ERP시스템의 제공, 획일화된 기업이미지, 소비자와 의사 소통, 축척 된 데이터를 이용한 CRM활동**



* 우리는 후발주자로써, 이미 스타벅스, 엔제리너스, 카페베네, 커피빈, 탐앤탐스, 투썸플레이스, 파스쿠찌, 할리스, 이디야와 같은 많은 기업들과 경쟁 하기 위해서, 다음의 목표들을 통하여 포지셔닝을 한다.

|  |
| --- |
| 1-4. 프로젝트 개발 목표 |

* **전반적인 목표**

기업 전반의 업무 과정을 통합적으로 관리하고, 경영상태를 실시간으로 파악하는 것

* **세부적인 목표**

1. 본사와 체인점을 연계한 전산망 구축

2. 체인점 별 매출분석, 유통관리, 손익분석

3. 재고 관리를 통한 신속, 정확한 물량공급

4. 축적된 통계자료를 이용한 고객분석 제공

5. 각 지점과의 커뮤니케이션 제공

**PART**

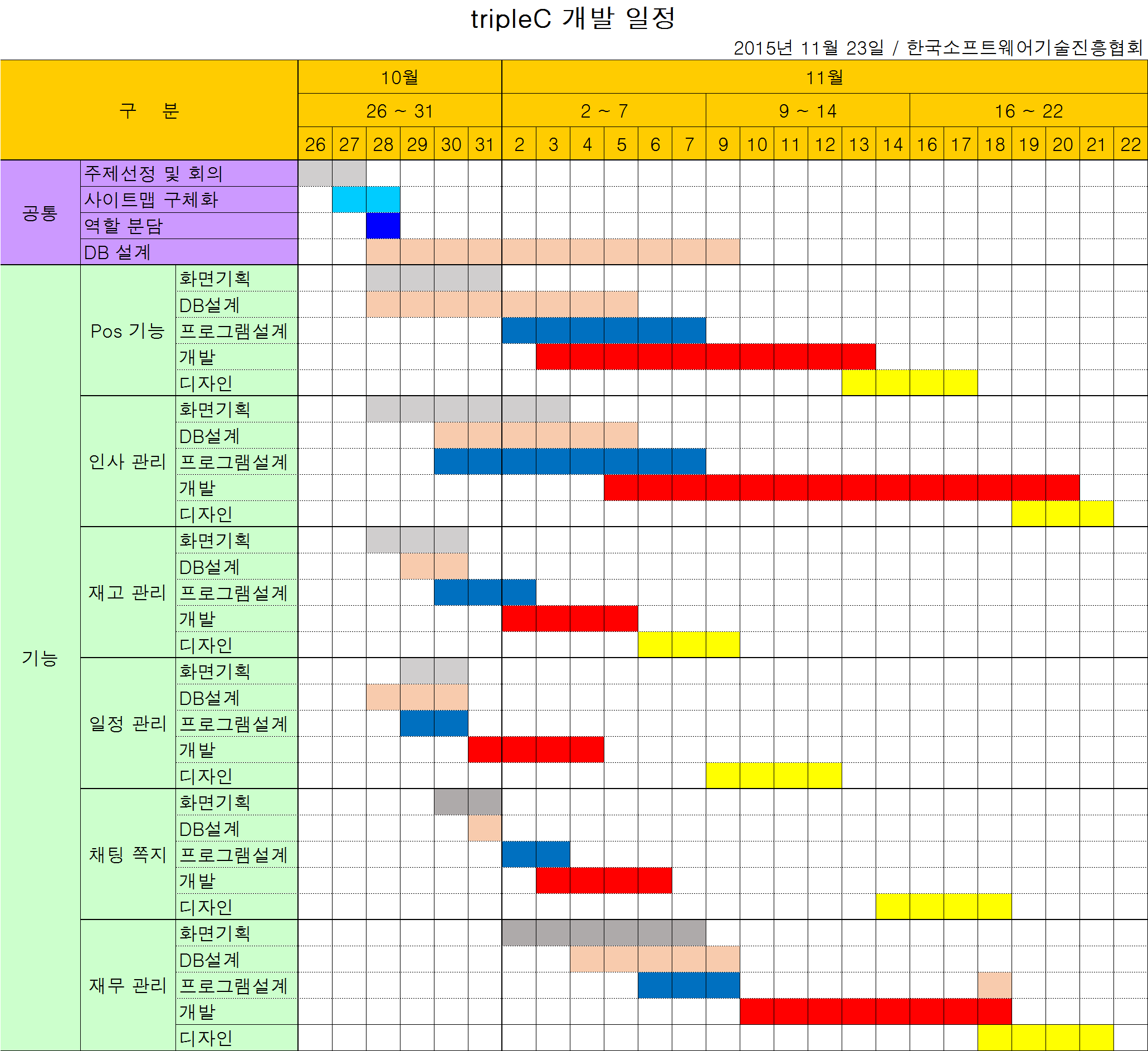
**TWO**

|  |
| --- |
| 프로젝트 추진일정 |

프로젝트의 계획부터 발표까지의

모든 일정을 간략하게 소개합니다.

|  |
| --- |
| 2-1. 프로젝트 개발일정 |



**PART**

**THREE**

|  |
| --- |
| 프로젝트 조직 |

팀원의 역할분담과 책임에 대해서 소개합니다.

|  |
| --- |
| 3-1. 팀원 역할과 책임 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **역할 1** | **역할 2** | **역할 3** | **비고** |
| **김예현** | 재고관리 | 기초 정보 등록 | KostarBucks  사이트 | 조장 |
| **유성현** | 포스기 | 대화형 쪽지 | 경영/재무관리 | 부조장 |
| **고 준** | 공지사항  게시판 | 다음 지도api | kostarBucks  매장소개 | 조원 |
| **박상현** | 고객관리 | 일정관리 | 전체적인 CSS  디자인 | 조원 |
| **신승택** | Q&A 게시판 | 체인점 통계 | 본사 통계 | 조원 |
| **조항덕** | 인사관리 | 관리자 페이지 | BGM관리 | 조원 |

**PART**

**FOUR**

|  |
| --- |
| 시스템 구조 |

프로젝트 개발을 위해 구성한 개발 환경을 소개하고

시스템이 어떻게 구성되어있는지 봅니다. 또한 프로젝트에 사용된 기술에 대해

살펴보면서 시스템의 구성 요소를 파악합니다.

|  |
| --- |
| 4-1. 적용기술 |

* **프레임워크**



전자정부프레임워크인 스프링 프레임워크(Spring Framework) 3.0를 사용하여 모델2 개발방식으로 개발하였고, 더불어 MVC 패턴을 사용해 서비스를 구현하였습니다. 이를 통해 뷰(View)와 비즈니스 로직(Business Logic)을 분리하여 복잡한 개발로 인한 혼선을 방지하였고, 스프링 프레임워크가 제공하는 Annotation 기술을 활용하여 조금 더 간결한 개발을 할 수 있었습니다.



또한 퍼시스턴스 프레임워크(Persistance Framework)인 마이바티스를 함께 사용해 데이터 베이스에 대한 고급 맵핑을 지원할 수 있도록 하였고, 코드의 간결성과 가독성을 높였습니다.

* **형상관리**



형상관리 툴로는 SVN을 사용하여 작업이력관리, 문제점 파악, 예전 버전 복원 등 조원들 간의 손쉬운 협업을 할 수 있었습니다.

|  |
| --- |
| 4-1. 적용기술 |

* **HTML5 + CSS3, jQuery**

.

HTML5가 갖는 장점인 시맨틱 태그를 적극 활용하고, HTML5의 표준을 지키기 위해 노력했으며 HTML 문서의 구조와 스타일, 기능을 분리하여 페이지를 구성하는데 초점을 맞추었습니다. 또 한 CSS3를 사용하여 디자인을 역동적으로 표현하였고, jQuery를 사용하여 자바스크립트의 생산성을 높여주었습니다.

* **API**

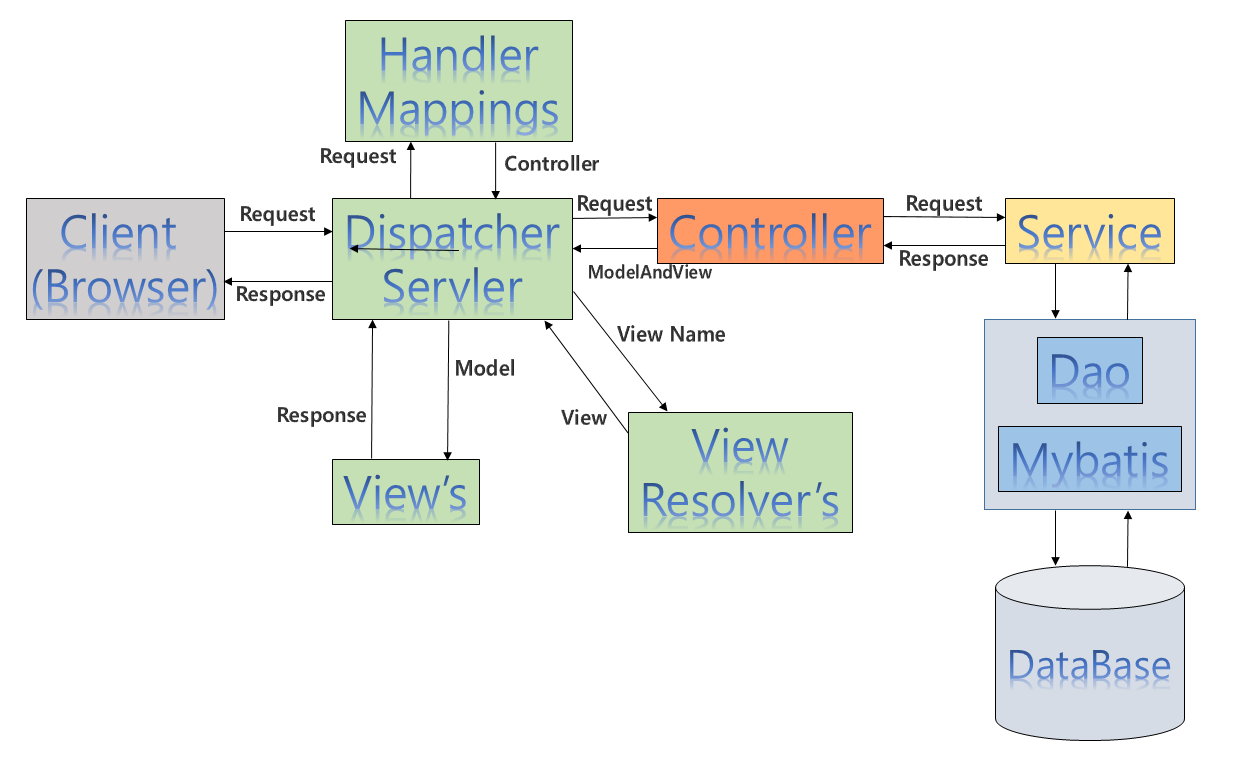
****

Highcharts를 사용하여 통계를 손쉽게 이미지화 함으로써 전문지식이 없어도 통계를 한눈에 파악하기 쉽게 하였고, 각 자료 간 데이터비교가 용이하게 되었습니다.



DHTMXL의 Scheduler를 이용하여 체인점 및 본사의 일정을 원활하고 일관성 있게 관리하였습니다.

|  |
| --- |
| 4-2. S/W구조 |



**PART**

**FIVE**

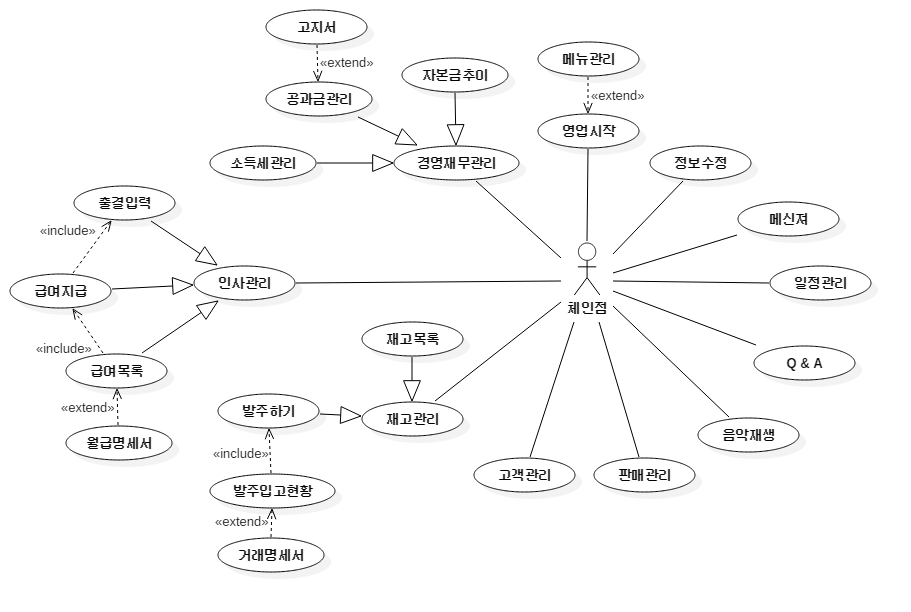
|  |
| --- |
| 분석/설계 산출물 |

프로젝트 UseCase모델, Class모델, Data모델, 메뉴구조도와 같이

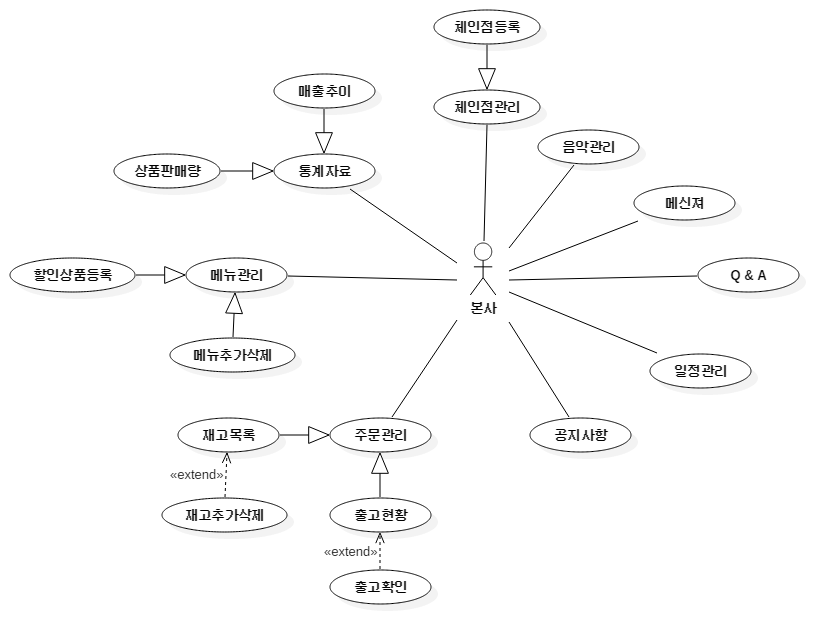
프로젝트의 큰 틀과 구조의 전반적인 내용을 소개합니다.

|  |
| --- |
| 5-1. Use Case 모델 |

* **Triple C에 체인점으로 로그인 한 경우**



* **Triple C에 본사로 로그인 한 경우**



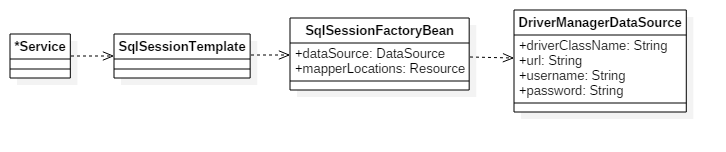
|  |
| --- |
| 5-1. Use Case 모델 |

* **KostarBucks에 회원/관리자로 로그인 한 경우**

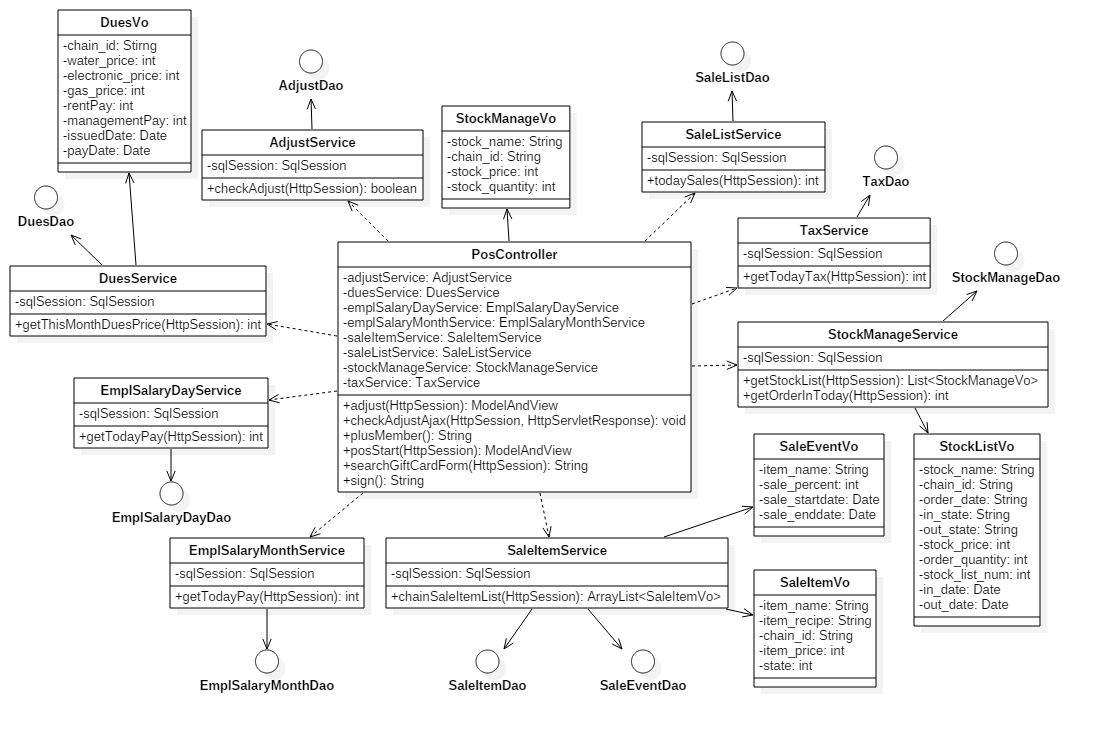


|  |
| --- |
| 5-2. Class 모델 |

* **Mybatis와 서비스의 Dependency 관계**

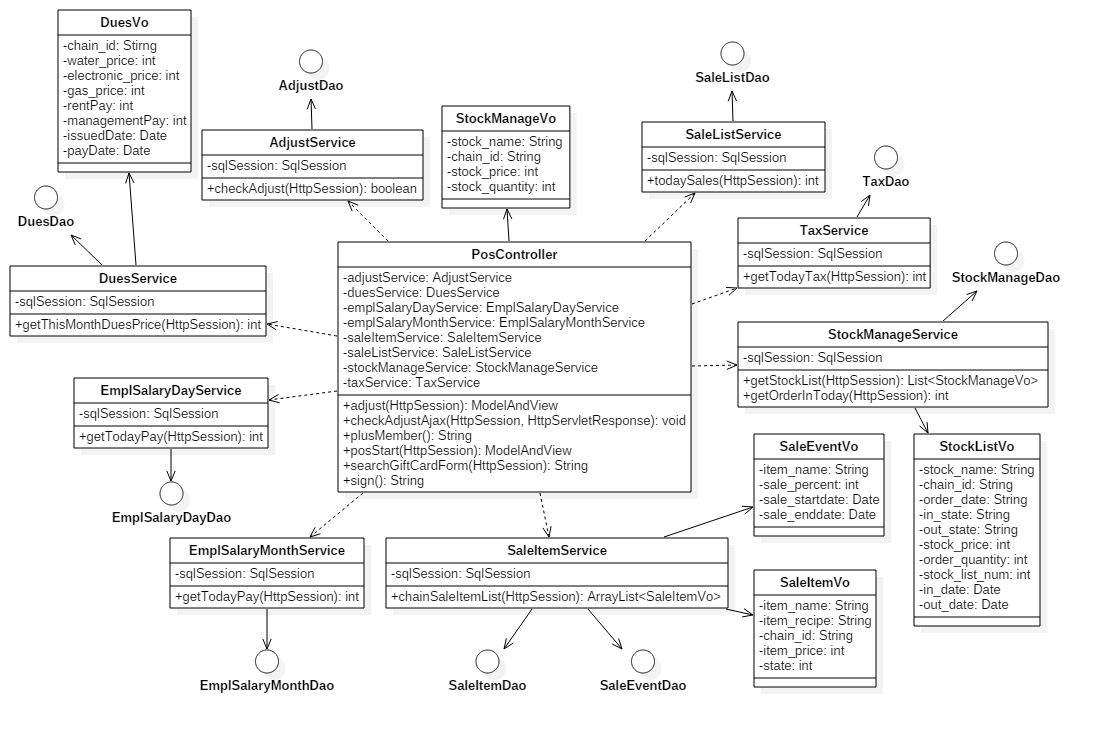


* **포스 관련 Class Diagram**

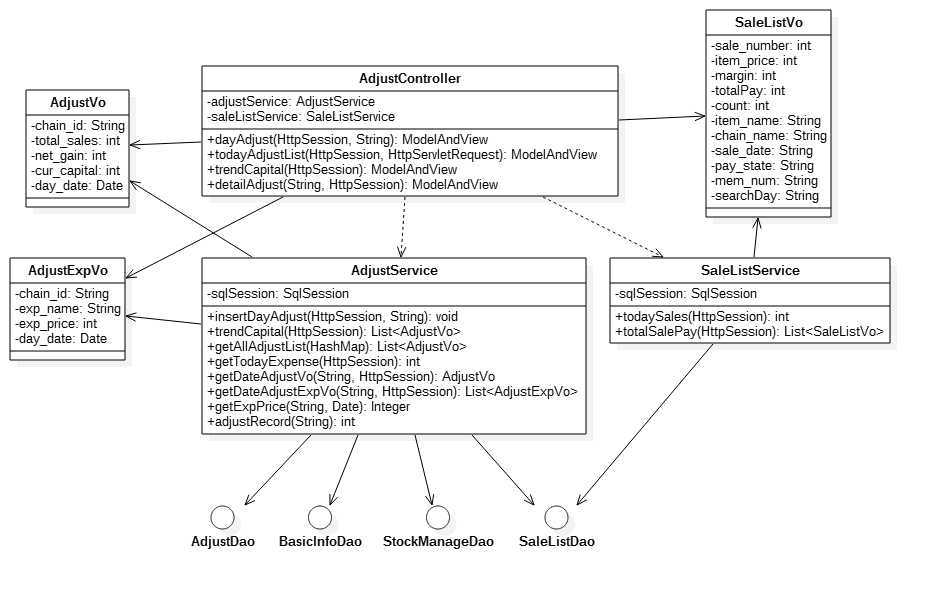


|  |
| --- |
| 5-2. Class 모델 |

* **판매 관련 Class Diagram**

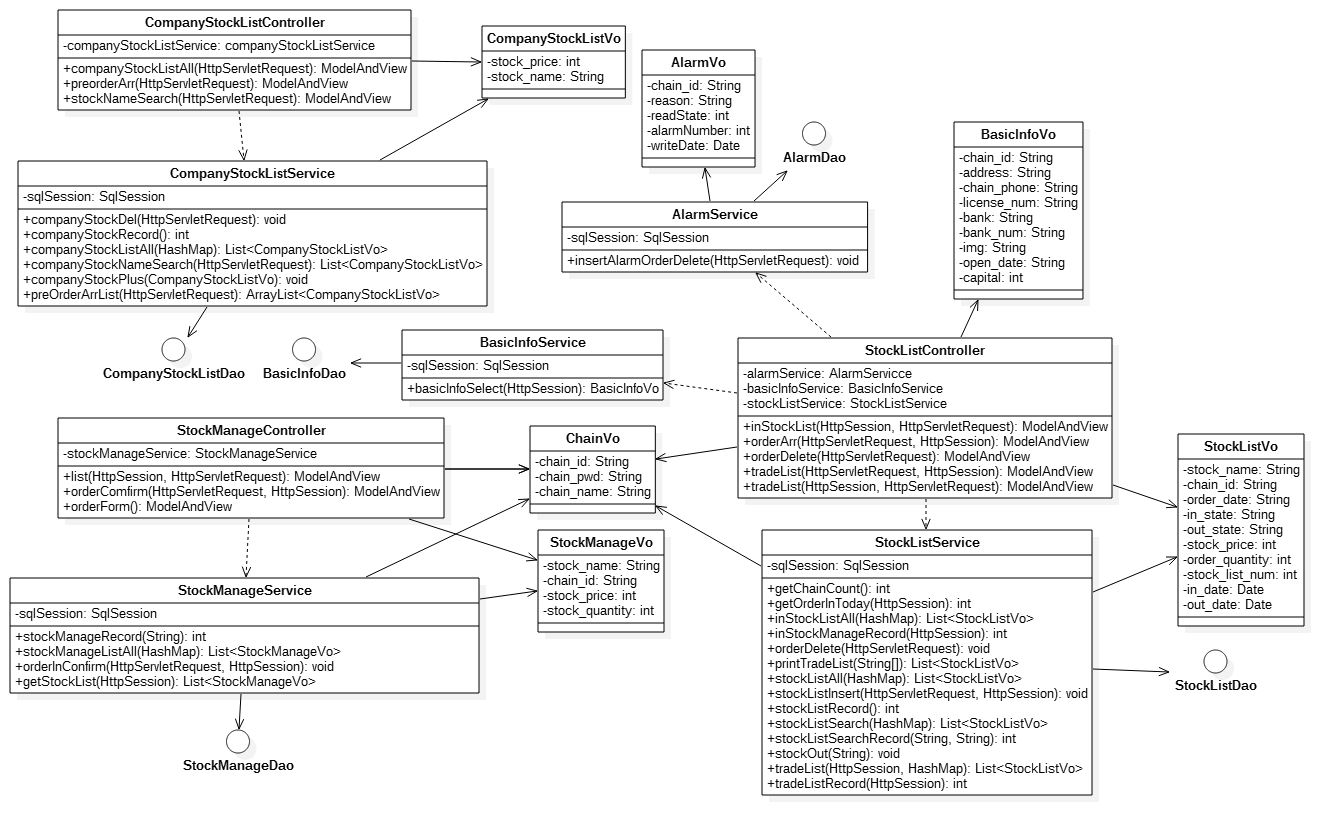


* **정산 관련 Class Diagram**

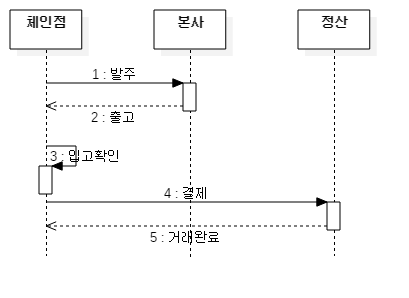


|  |
| --- |
| 5-2. Class 모델 |

* **재고 관련 Class Diagram**

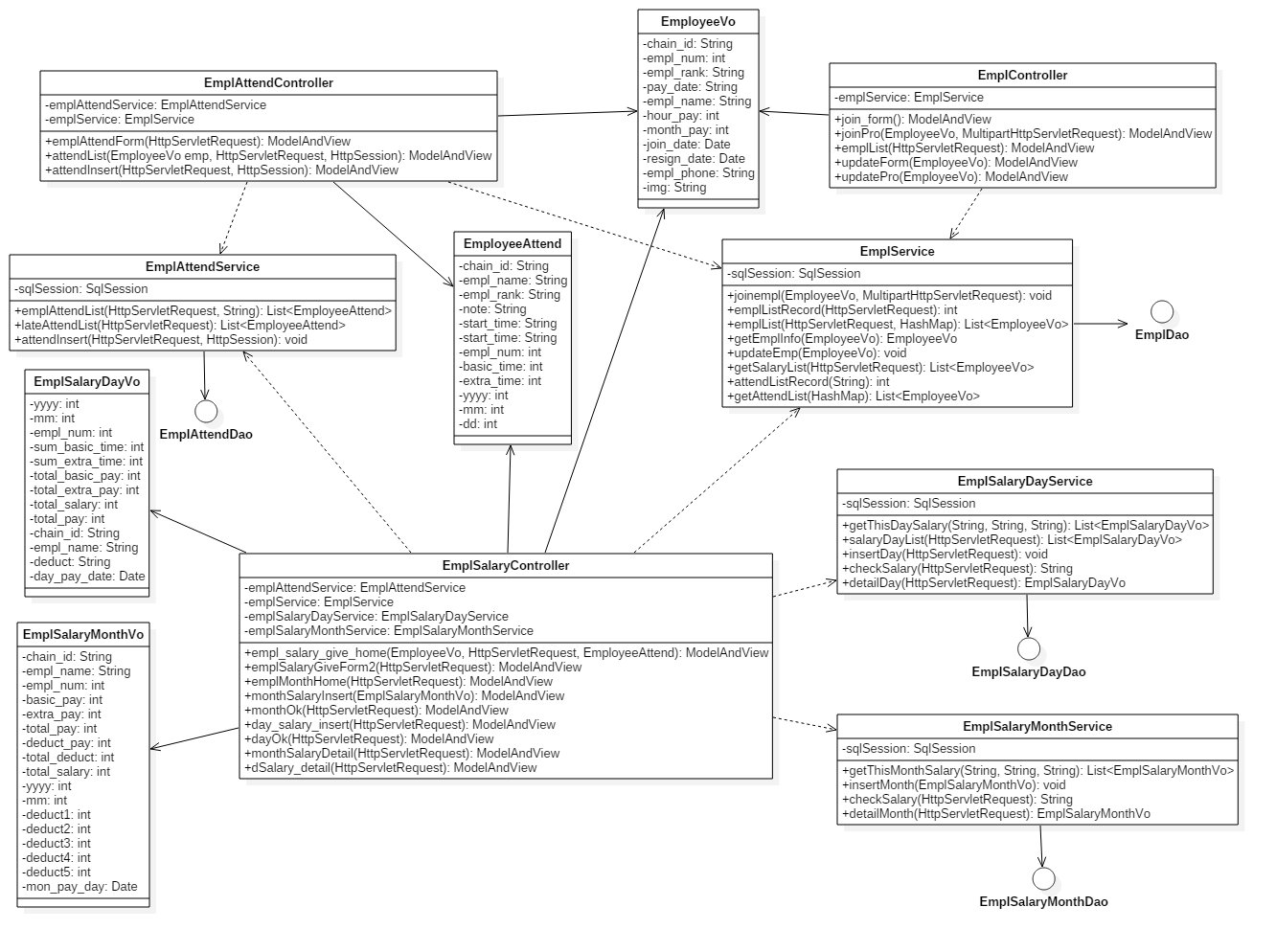


* **재고 관련 Sequence Diagram**

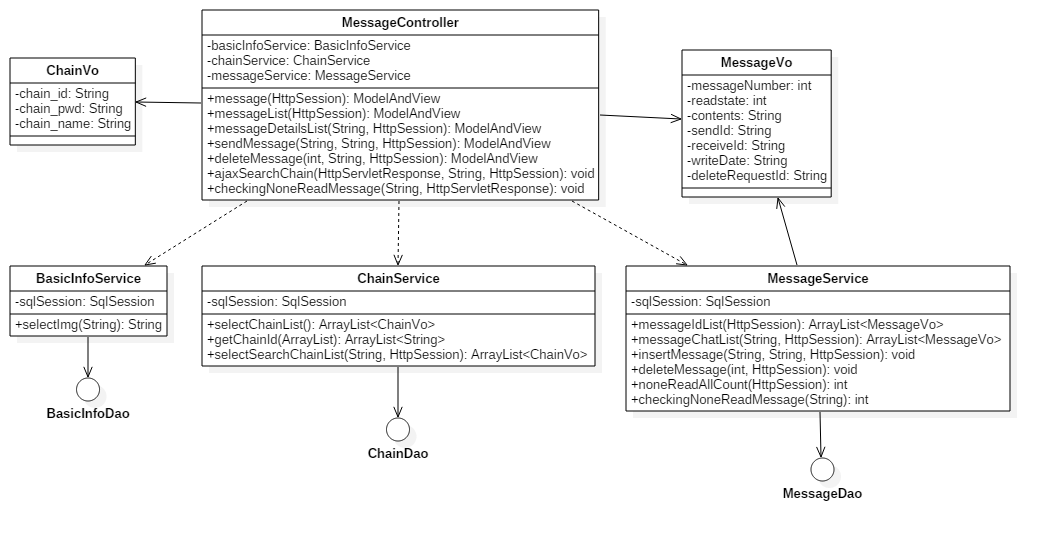


|  |
| --- |
| 5-2. Class 모델 |

* **인사 관련 Class Diagram**

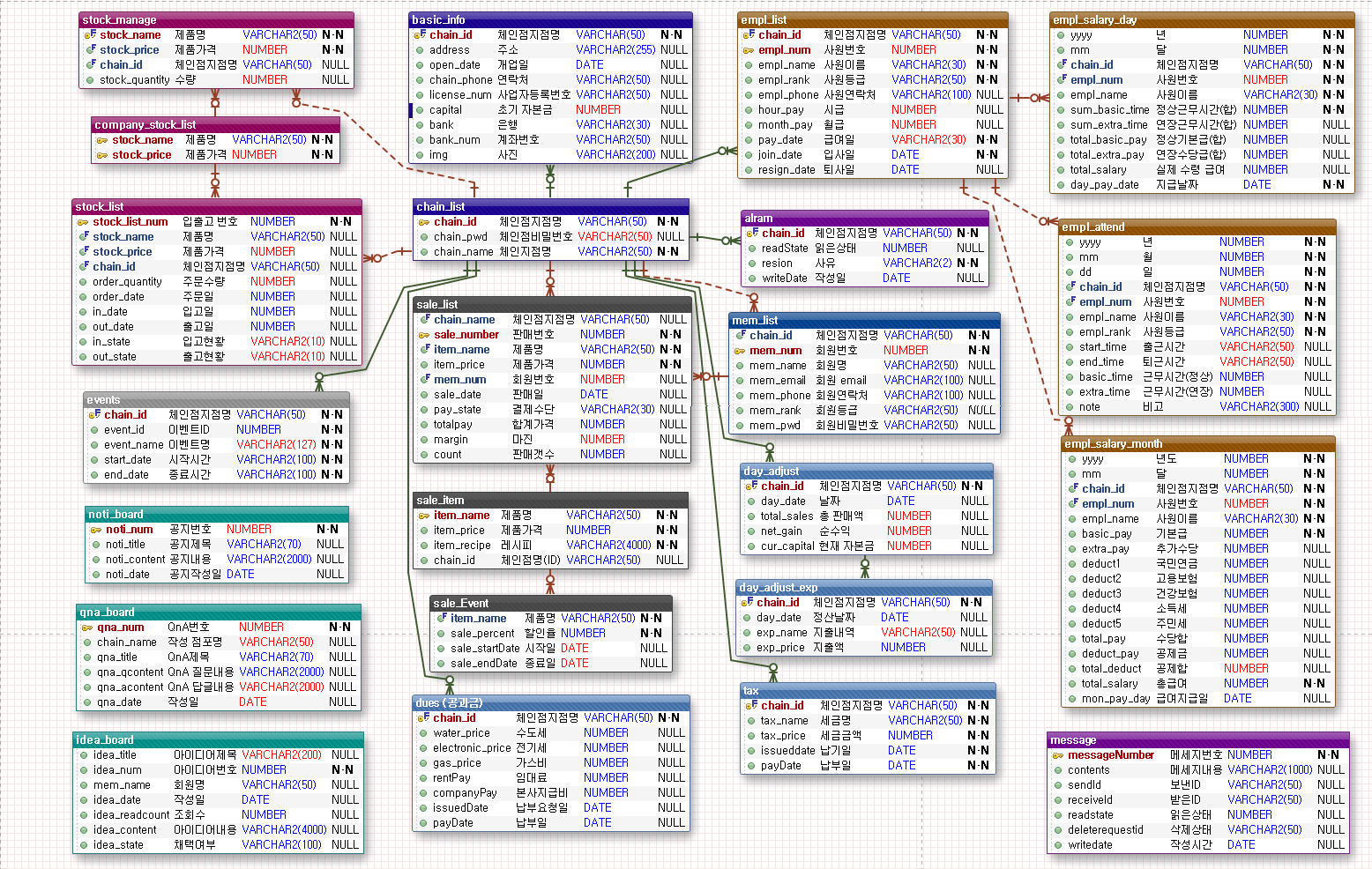


* **메신져 관련 Class Diagram**



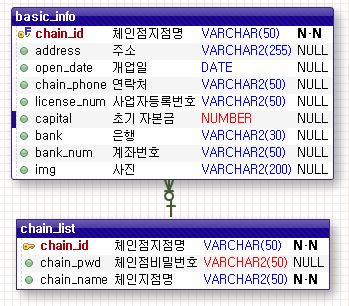
|  |
| --- |
| 5-3. Data 모델 |

* **ER Diagram**



|  |
| --- |
| 5-3. Data 모델 |

* **체인점관리**



**체인점목록(chain\_list)**

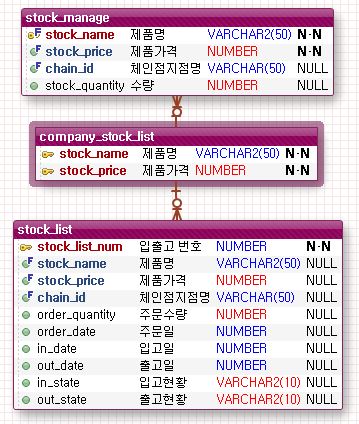
새로 문을 여는 체인점은 본사가 체인점 목록에 각 항목을 추가해줌으로써 로그인을 할 수 있다.

**기초정보등록(basic\_info)**

체인점주는 첫 번째 로그인 시 필수로 기초정보등록을 하여야 한다.

|  |
| --- |
| 5-3. Data 모델 |

* **재고관리**



**본사재고목록(company\_stock\_list)**

본사의 재고를 추가/삭제 할 수 있는 테이블이다.

**체인점별 재고목록(stock\_manage)**

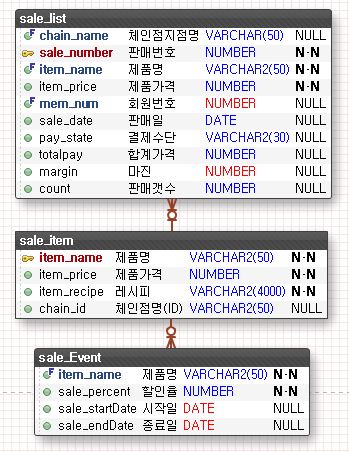
체인점에서 발주한 상품을 본사에서 출고해 주었을 때, 정말 체인점에 입고가 잘 되었다면 입고확인을 통해 체인점별 재고목록에 재고가 추가된다.

**입출고현황(stock\_list)**

어느 체인점에서 얼만큼을 발주했는지, 입출고현황을 알 수 있는 테이블이다.

|  |
| --- |
| 5-3. Data 모델 |

* **판매관리**



**판매상품(sale\_item)**

본사에서 판매상품을 지정한 후 각 체인점에서 원하는 상품을 골라서 판매할 수 있다.

**판매목록(sale\_list)**

포스에서 현금이나 카드결제버튼을 눌렀을 때 정보가 담기는 테이블이다.

**행사상품(sale\_Event)**

본사에서 행사를 등록 할 시 모든 체인점의 해당제품의 가격이 행사가격으로

변경되며, 전체 일정관리에도 기간이 표시된다.

|  |
| --- |
| 5-3. Data 모델 |

* **인사관리**



**사원목록(empl\_list)**

체인점 별 사원 등록을 할 수 있는 테이블이다.

**사원출결등록(empl\_attend)**

매일 사원의 출결정보를 입력할 수 있는 테이블이다.

**비정규직급여(empl\_salary\_day)**

사원출결등록 테이블의 정보를 바탕으로 비정규직사원의 급여를 관리할 수

있는 테이블이다.

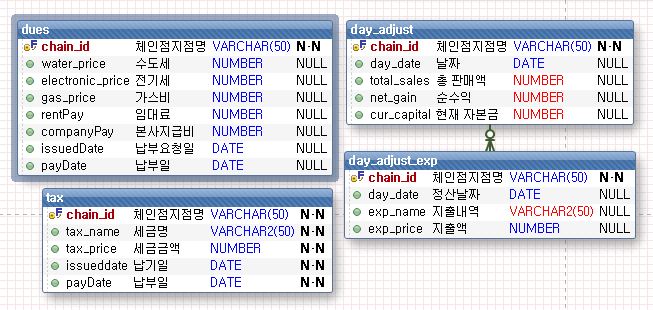
**정규직급여(empl\_salary\_day)**

사원출결등록 테이블의 정보를 바탕으로 정규직사원의 급여를 관리할 수

있는 테이블이다.

|  |
| --- |
| 5-3. Data 모델 |

* **경영재무관리**



**공과금(dues)**

매달 낸 공과금을 입력할 수 있는 테이블이다.

**소득세(tax)**

매년 낸 종합소득세를 입력할 수 있는 테이블이다.

**일일지출(day\_adjust\_exp)**

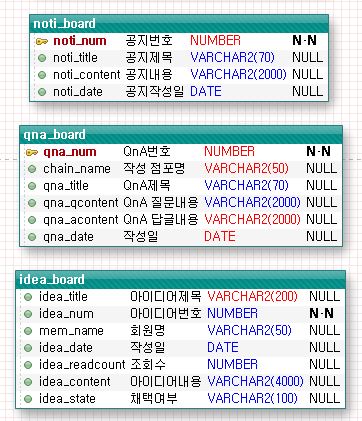
매일의 사원의 급여, 공과금, 소득세, 발주내역이 등록되는 테이블이다.

**일일정산(empl\_salary\_day)**

매일의 순이익과 남은 자본금이 등록되는 테이블이다.

|  |
| --- |
| 5-3. Data 모델 |

* **게시판**



**공지사항(noti\_board)**

본사에서 작성한 공지사항은 모든 체인점에서 확인 할 수 있다.

**Q & A(qna\_board)**

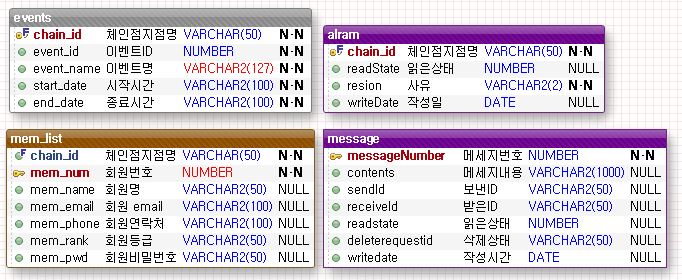
각 체인점에서 궁금한 점을 작성하면 본사에서 답변을 등록해 줄 수 있다.

**아이디어게시판(idea\_board)**

KostarBucks 사이트에서 사용되는 테이블이다. 회원이 아이디어를 올리면 관리자가 채택할 수 있고, 채택 시 해당 회원의 이메일로 5만원 상품권이 발송된다.

|  |
| --- |
| 5-3. Data 모델 |

* **기타**



**일정관리(events)**

본사의 일정관리는 모든 체인점에서 볼 수 있으며 체인점의 일정관리는 해당 체인점 에서만 볼 수 있다.

**알람(alram)**

본사에서 재고를 출고했을 시, 체인점에서 발주를 취소했을 시, 체인점에서

11시까지도 정산을 하지 않았을 시 알람이 발송된다.

**메신져(message)**

각 체인점 별로 커뮤니케이션이 가능하도록 만든 테이블이다.

**회원(mem\_list)**

POS에서 발급받은 회원번호로 KostarBucks에 회원가입할 수 있으며 먹은

횟수에 따라 등급이 나뉘고 할인율이 적용된다.

|  |
| --- |
| 5-4. 메뉴구조도 |

* **TripleC에 체인점으로 로그인 했을 때**
* **TripleC에 본사로 로그인 했을 때**
* **KostarBucks 회원/관리자로 로그인 했을 때**

**PART**

**SIX**

|  |
| --- |
| 구현산출물 |

시스템의 상세 기능에 대해 설명하고,

사용한 기술과 적용 시 주의사항에 대해서 소개합니다.

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (1) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 재고관리  (체인점에서 발주→본사에서 출고→체인점에서 입고확인→거래완료) | | |
| **Table** | stock\_list, stock\_manage, company\_stock\_list | **Controller** | StockController.java, AdminController.java |
| **페이지** | orderForm.jsp, inStockList.jsp,  stockListAll.jsp | **주요사용기술** | request객체, myBatis, JSTL |
| 1. **체인점에서 발주하기**      1. **본사에서 출고하기**      1. **체인점에서 입고확인하기** | | | |
| **주요코드** | | | |
| 1. **StockManageService.java**      1. **StockManageDao.xml**      1. **inStockList.jsp**      1. **StockManageService.java** | | | |
| **코드설명** | | | |
| 1. 발주를 몇 개를 할지 모르기 때문에 request객체의 getParameterValues를 이용하여 배열로 발주내역을 가져온 후 배열에 담겨있는 정보를 하나씩 insert한다. 2. insert문에서 in\_date, out\_date는 ‘’로 in\_state, out\_state는 ‘미입고’, ‘대기’로 하여 상태를 구분한다. 3. in\_date와 out\_date가 비어있는지의 여부로 버튼의 value를 각각 발주취소, 입고확인, 거래완료로 하며 거래완료일 경우엔 버튼의 속성을 disabled로 바꾼다. 4. 거래완료가 되었을 때 해당 재고가 재고목록에 있었다면 수량만 증가시켜주고, 재고목록에 없다면 입고내역을 insert시켜준다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (2) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 회원등록 (포스에서 번호발급→KostarBucks에서 회원가입) | | |
| **Table** | mem\_list | **Controller** | MemberController.java  (TripleC / KostarBucks) |
| **페이지** | pos.jsp  index.jsp(KostarBucks) | **주요사용기술** | JAVA(for, charAt) |
| 1. **포스에서 새로운 회원번호 발급**      1. **회원번호로 KostarBucks에 회원가입** | | | |
| **주요코드** | | | |
| 1. **(TripleC) MemberService.java**      1. **(KostarBucks) MemberController.java** | | | |
| **코드설명** | | | |
| 1. member시퀀스의 nextVal을 가져와서 회원번호패턴으로 재정렬 하기 위해 sequenceNumber라는 변수를 새로 잡고 for문과 if문으로 패턴을 만들어 회원테이블의 회원번호로 insert한다. 이때 member시퀀스는 100000000000부터 시작하도록 설정되어있으며, 회원번호 패턴은 ‘0000-0000-0000-0000’이다. 2. KostarBucks의 회원 가입 시 부여 받은 회원 번호를 입력하고 확인버튼을 누르면 회원 테이블에서 확인하여 해당번호가 없는지, 해당번호가 있는데 이름까지 등록되어있는지를 확인하여 각각 다른 상태번호를 리턴 해 준다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (3) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 출결관리 및 급여지급 | | |
| **Table** | empl\_list, empl\_attend, empl\_salary\_day, empl\_salary\_month | **Controller** | EmplController.java  EmplAttendController.java EmplSalaryController.java |
| **페이지** | emplAttend.jsp  emplAttendList.jsp emplSalaryGiveForm.jsp | **주요사용기술** | request객체, myBatis, JSTL,  jQuery |
| 1. **사원출결 페이지에서 검색을 눌러 사원 선택**      1. **선택한 사원의 정보를 불러온 뒤, 출근시간과 퇴근시간을 입력하면 자동으로 오 늘 근무시간 계산 후 출결목록에 저장**      1. **사원급여지급 페이지에서 지급 할 월과 사원명을 선택한 뒤 근무내역을 보고 추가수당과 공제액 입력 후 급여 지급** | | | |
| **주요코드** | | | |
| 1. **EmplAttendService.java**      1. **EmplSalaryController.java**      1. **EmplSalaryService.java**      1. **EmplSalaryController.java** | | | |
| **코드설명** | | | |
| 1. 한번에 여러 명의 사원출결을 입력 받아 배열로 출결 데이터를 저장한다. 출결입력창을 여러 줄 생성하고도 입력하지 않고 등록을 할 수 있기 때문에 사원명으로 비교하며 실질적으로 데이터가 있는 배열만 저장한다. 2. 급여를 지급할 사원리스트를 가져오기 위해 getSalaryList서비스에서 사원 목록을 가져온 뒤, 퇴사한 사원을 제외한 모든 사원 중 급여를 지급 할 사원을 선택하도록 한다. 사원을 선택하면 emplAttendList서비스에서 해당 년, 월의 출결 정보를 가지고 온다. getEmplInfo서비스에서 해당 사원의 등급과 급여 정보를 가지고 온다. 그 후 jsp페이지에서 사원등급에 따른 정규직, 비정규직 급여지급 폼이 나타난다. 3. 급여지급 페이지에서 지급을 누를 때 EmplSalaryservice.java의 checkSalary서비스에서 해당년, 월의 급여를 이미 지급 했는지 확인한다. 4. EmplSalaryController.java에서 급여 여부에 따른 리턴 값으로 jsp에서 급여 성공, 실패 여부를 출력한다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (4) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 포스기 상품 판매 | | |
| **Table** | sale\_list,  sale\_item | **Controller** | PosController.java,  SaleListController.java,  SaleItemController.java |
| **페이지** | pos.jsp,  sign.jsp,  memberList.jsp | **주요사용기술** | myBatis, JSTL, Query, JSON |
| 1. **메인 페이지에서 영업 시작 버튼을 클릭**   포스기   1. **구매 요청이 들어온 상품을 선택**   상품선택   1. **만약 구매 요청한 고객이 회원이라면 고객 선택 버튼을 이용하여**   **고객의 카드번호 혹은 전화번호를 이용해 고객을 등록**  고객선택   1. **카드 & 현금 버튼을 이용해 결제를 진행**   **(해당 사진은 카드 결제를 진행하였을 때)**  **카드결제**   1. **결제 완료되면 일일결제No 가 증가**   **결제완료** | | | |
| **주요코드** | | | |
| 1. **SaleItemService.java**   **체인세일리스트**   1. **posScript.js**   **계산1**  **C:\Users\seonghyunVv\AppData\Local\Microsoft\Windows\INetCache\Content.Word\계산2.png**  **계산3**   1. **SaleListService.java**   **인서트세일리스트1**  **인서트세일리스트2**   1. **postscript.js**   **현재시간** | | | |
| **코드설명** | | | |
| 1. 해당 지점에서 어떠한 상품을 판매하는지 상품리스트를 가져오는 메소드   판매 상품들은 지점마다 다르기 때문에, % % 를 사용하여  해당 지점만의 상품리스트를 가져온다.  해당 지점의 상품리스트를 가져온 후 checkInsertSaleEvent 메소드를 이용,  만약 상품의 할인이 진행중이라면 할인율 적용한 가격을  상품 객체에 넣어준다.(saleItemVo.setItemPrice)   1. 상품을 판매할 때 어느 상황에서 어느 상품을 몇 개 판매할지 알 수 없기 때문에, 판매할 상품 목록을 JSON 형태로 만들어 자바로 넘겨주는 스크립트   넘겨줄 JSON 데이터가 몇 단어가 될지 모르기 때문에  해당 DOM에 form 태그를 생성, form 태그를 이용하여 JSON 데이터를 넘겨준다  (get 방식으로 넘겨주면 최대 255자 밖에 전송이 안되기 때문에 post 방식으로 전달하려고 form 태그를 생성한 것이다.)   1. 자바스크립트에서 넘어온 JSON 데이터를 처리해주는 메소드   JSONArray 와 JSONObject 를 통해 넘어온 JSON 데이터를 자바에서 사용하는  자료형으로 변환시킨 후(jsonObject.getInt, jsonObject.getString …) 변환 시킨 결과를  saleListVo 에 담아 DB 에 저장한다. 그 후 구매한 고객의 구매횟수에 따라서  고객 등급을 조정해준다(50번 이상이면 Gold Level, 30번은 Silver Level …)   1. 포스기의 맨 상단에 현재 시간이 찍히게 해주는 스크립트   재귀 호출을 이용, 1초마다 현재 시간을 가져온다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (5) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 대화형 쪽지 | | |
| **Table** | chain\_list,  message | **Controller** | MessageController.java |
| **페이지** | messageIndex.jsp,  messageList.jsp,  messageDetailsList.jsp | **주요사용기술** | myBatis, JSTL, jQuery, JSON, Ajax |
| 1. **헤더부분의 쪽지 모양 이미지를 클릭(모든 체인점 리스트 출력)**   **체인메신저**   1. **새로 뜬 창의 대화창 이미지를 클릭하면 어떤 체인점과 대화를 하였는지 리스트 출력(오늘 대화를 하였다면 시간이, 오늘 이전이라면 날짜가 출력)**   **본사메신저** | | | |
| **주요코드** | | | |
| 1. **MessageController.java**   **메세지1**   1. **MessageService.java**   **메세지리스트1**  **메세지리스트2**  **C:\Users\seonghyunVv\AppData\Local\Microsoft\Windows\INetCache\Content.Word\메세지리스트3.png**   1. **MessageService.java**   **상세메세지1**  **상세메세지2**   1. **messageIndex.js**   **에이젝스체인점검색**   1. **MessageController.java**   **에이젝스체인점검색자바** | | | |
| **코드설명** | | | |
| 1. 메시지를 처음 시작했을 때 나오는 체인점 리스트를 가져오는 메소드   해당 session 에 저장되어 있는 아이디가 읽지 않은 메시지수도 가져와 띄워준다.   1. 어떤 체인점과 대화를 나누었는지 그 리스트를 가져오는 메소드   우선 내가 어느 체인점과 대화를 나누었는지  보낸 체인점 리스트와 받은 체인점 리스트를 모두 가져와 하나로 합친다.  합친 리스트를 중복 제거 해준 후 해당 체인점과의 보낸 메세지의 최근 시간, 받은 메세지의 최근 시간을 가져온다.  둘 중 가장 최근 시간을 writeDate 에 담는다.  체인점 아이디와 대화 중 삭제 요청을 한 적이 있는지가 담겨 있는 deleteRequestIdList 를 통해 삭제 요청이 들어온 적이 없다면 실제로 보여줄 리스트에 담는다.  최종 적으로 보낼 리스트를 날짜순으로 정렬해준다.   1. 상세 대화로 들어갔을 때 리스트를 가져오는 메소드   현재 읽지 않은 메세지가 있다면 그 메세지의 상태는 0 이기 때문에 1 로 update 해준다.  session 에 저장된 아이디와 대화한 체인점의 아이디가 같으면 보낸 메세지 리스트만 담고,  다르다면 보낸 메세지 리스트와 받은 메세지 리스트를 모두 담는다.  2번과 마찬가지로 deleteId 를 전부 가져온 후 삭제 요청이 들어온 적이 있는지 확인하고 삭제 요청이 들어온 적이 있다면  chatList 에서 해당 대화 내용을 삭제한다.   1. 가장 먼저 나오는 체인점 리스트에서 체인점을 검색하기 위한 스크립트   messagIndex.jsp 의 검색 창에 체인점 이름을 검색했을 때,  ajax 를 이용하여 입력한 데이터를 실시간으로 자바에 넘겨준다.  넘겨준 데이터로 처리한 결과(JSON) 를 이용하여  messageIndex.jsp 의 체인점 리스트의 값에 결과를 넣어준다.   1. 4번과 연관된 소스의 자바 코드   검색한 이름을 이용하여 검색 키워드에 해당하는 체인점 이름이 있는지 찾고,  존재하는 체인점 이름들이 몇 개일지 모르기 때문에  JSON 을 이용하여 넘겨줄 데이터를 만든다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (6) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 정산 기능 | | |
| **Table** | day\_adjust,  day\_adjust\_exp | **Controller** | PosController.java,  AdjustController.java |
| **페이지** | adjust.jsp,  dayAdjust.jsp,  adjustList.jsp | **주요사용기술** | myBatis, JSTL, jQuery, JSON, Ajax |
| 1. **영업이 끝난 후 포스기에서 정산 버튼을 클릭**   **정산**   1. **재고 사용 후 남은 재고입력하고 정산완료버튼 클릭**   **정산완료**   1. **경영.재무관리의 자본금 추이 탭에서 지금까지의 자본금 변화 확인 가능**   **체인자본금추이**   1. **만약 오늘 정산을 하였다면 오늘은 영업 시작이 불가**   **정산재시작불가** | | | |
| **주요코드** | | | |
| 1. **PosController.java**   **adjust.do컨트롤러**   1. **AdjustService.java**   **정산처리서비스1**  **정산처리서비스2** | | | |
| **코드설명** | | | |
| 1. 포스기에서 정산 버튼을 눌렀을 때, 현재 해당 지점이 가지고 있는 재고리스트와 그 지점의 오늘 지출 내역을 가지고 오는 메소드   (지출 목록은 총 4가지(발주, 공과금, 급여, 세금)   1. 정산 완료 버튼을 눌렀을 때 실제 정산 처리를 해주는 메소드   이 메소드에서 재고 수량을 변경해주고, 자본금도 변경해준다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (7) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 일정관리 | | |
| **Table** | events, saleEvents | **Controller** | CalendarController.java  Calendar2Controller.java |
| **페이지** | Events.jsp,  Javacalendar.jsp,  adminCalendar.jsp,  events2.jsp,  javacalendar2.jsp | **주요사용기술** | request객체, session,  javaplanner-1.5.jar(라이브러리) |
| 1. **체인점 Calendar에서 원하는 날짜 선택해 일정 추가, 수정, 삭제하기 (체인점에 따라 다르게)**      1. **본사에서 상품 할인을 등록하면, 자동으로 본사 Calendar에 일정 추가되게 하기**   **(단, 본사 Calendar의 일정 추가, 수정, 삭제는 본사 계정에서만 가능하게 하기)** | | | |
| **주요코드** | | | |
| 1. **EventsManager.java**      1. **EventsManager2.java , EventsManager.java** | | | |
| **코드설명** | | | |
| 1. 각 체인점마다 일정이 다르기 때문에, 체인점 index페이지의 달력이 다 달라야 한다. 그렇기 위해서는 로그인 정보를 Session에 담아야 하므로, Session 객체를 얻어오기 위한 방법으로 인자 값으로 HttpServletRequest를 받고, chain\_id라는 변수에 넣어준다. 2. chain\_id가 admin일때와 그렇지 않을 때 나누어서 admin일때만 insert, delete, update되게 한다. 그리고 할인등록은 admin 계정일때만 할 수 있는 것이므로, 할인 시작일자를 start\_date, 할인 끝 일자를 end\_date에 추가하고 할인내용을 text에 추가해, 할인 등록을 하면 자동으로 본사 Calendar에 추가되도록 한다. | | | |

|  |
| --- |
| 6-1. 주요 구현 사항에 대한 코드 설명 (8) |

|  |  |  |  |
| --- | --- | --- | --- |
| **기능** | 판매관리(매출추이, 제품별판매량) | | |
| **Table** | sale\_list, sale\_item | **Controller** | SaleListController.java  SaleItemController.java AdminController.java |
| **페이지** | todaySalePay.jsp  yearPrice.jsp itemSaleCount.jsp  admin\_todaySalePay.jsp  admin\_yearPrice.jsp  admin\_ChainTotalSaleCount.jsp  admin\_itemSaleCount.jsp | **주요사용기술** | request객체, myBatis, JSTL,  jQuery |
| 1. **매출추이 페이지**      1. **제품별 판매량**      1. **전체 지점 매출 추이**      1. **년도별 전체 지점 매출 추이**      1. **전체 지점 제품 판매량**      1. **전체 지점 제품별 판매 비율** | | | |
| **주요코드** | | | |
| 1. **SaleListController.java**      1. **SaleListService.java**          1. **AdminController.java**      1. **AdminService.java** | | | |
| **코드설명** | | | |
| 1. 해당 지점의 제품별 판매량을 비율로 가져오는 메소드   해당 지점의 판매하는 상품을 가져온다.  그 후 모든 상품들의 판매량(총 판매량)을 totalCount 에 담는다  (여기서 totalCount 에 담는 이유는 총 판매량을 미리 알고 있어야 퍼센트를 구할수 있다)  totalCount 를 이용하여 상품의 퍼센티지를 구하고 구한 퍼센티지를 반올림해준다(소수점 ~~자리에서 올림)   1. 해당 지점의 오늘의 판매금액을 시간별로 가져오는 메소드   session에서 지점 chain\_name을 가져온다.  for문을 0~23까지 돌리면서 해당시간에 판매금액을 넣어주고 해당시간에  판매한 금액이 없다면 0원을 넣는다.   1. 해당 지점의 연매출을 가져오는 메소드   session에서 지점 chain\_name을 가져온다.  for문을 1~12까지 돌리면서 해당월의 판매금액을 가져온다.   1. 해당 지점의 판매개수와 상품명을 가져오는 메소드   session에서 지점 chain\_name을 가져온다.  for문을 돌리면서 해당 지점의 판매하는 item\_name을 가져온다.  해당 지점에 상품이 있다면 판매개수를 넣어주고 없다면 0을 넣는다. | | | |

**PART**

**SEVEN**

|  |
| --- |
| 프로젝트 주요 이슈와 해결방안 |

프로젝트를 진행하면서 힘들었던 부분과,

어떻게 해결해 나갔는지에 대해 정리하였습니다.

|  |
| --- |
| 7-1. 주요 이슈와 해결방안 |

1. **사원관리**

프로젝트 초기에 이루어지는 설계의 중요성을 알고 있었기에 설계단계에서 세부적인 내용까지 미리 기획하였습니다. 코딩단계에서 순조롭게 계획한대로 진행이 되었는데 인사관리의 출결부분을 추가하면서 예상치 못하게 기존에 설계했던 부분들까지 수정하게 되었고 그로 인해 기존의 부분들을 사용할 수 없게 되었습니다. 이미 다른 부분들이 설계되어 있는 상태에서 그 틀에 맞춰 설계하는 것은 쉽지 않았고, 일주일 동안 같은 부분을 4번이나 지우고, 새로 만들고를 반복하였습니다. 일주일 동안 방향을 잡지 못 하고 방황하고 있을 때 모든 팀원들이 코딩을 잠시 멈추고 회의를 통해 다시 설계에 들어가 그 부분들 재정의 함으로써 문제를 해결하였습니다. 이런 문제를 겪고 나서 설계의 중요성과 팀원간의 소통의 중요성을 깨닫는 계기가 되었습니다.

1. **포스기 데이터 전달**

매장 영업을 위해 포스기를 만들었는데, 포스기에서 물건을 선택하여 계산 할 때, 한번의 결제에서 몇 개의 상품을 결제 할 지, 또 어떤 방법으로 결제 할 것인지 미리 알 수 없기 때문에 어떠한 방법으로 데이터를 넘겨야 하는지 3일 밤낮으로 고민에 빠지게 되었습니다. 하지만 그때 뇌리를 스친 것은 jSon데이터 방식이었습니다. jSon을 사용함으로써 몇 개의 데이터를 전송하든지 하나의 데이터로 전송 할 수 있게 되었습니다. jSon의 사용은 포스기에서 뿐만 아니라 다른 부분에서도 데이터 전송을 원활하게 해주는 중요한 라이브러리였습니다.

**PART**

**EIGHT**

|  |
| --- |
| 결론(lessoned learned, best practices) |

프로젝트를 진행하면서

느낀 점, 알게 된 점을 정리하였습니다.

|  |
| --- |
| 8-1. 결론 |

**‘좋은 팀워크가 만든 시너지’**

지금 실력으로 우리가 만들 수 있을까? 프로젝트 주제 선정이나 개발 문서, 혼자가 아닌 팀으로 만들면서 서로의 의견차이로 다툼은 없을까? 생각한 대로 기능을 구현 할 수 있을까? 처음 프로젝트를 시작 할 때는 막막했습니다. 프로젝트를 진행하면서 개발 문서를 만들고 정의한 기능을 구현 할 때 많은 부족함을 느꼈습니다.

하지만 팀원들 모두 열심히 노력하여 모르는 부분이 있을 때는 서로 공부하고 물어보며 조금씩 발전하며 마침내 기능을 구현 했을 때는 혼자 수업을 들으며 배울 때보다 더 기억에도 남고 할 수 있다는 마음을 가질 수 있었습니다. 생각한 대로 화면에 출력 되었을 때 재미와 또 다른 것도 해보고 싶다라는 생각이 들었습니다.

혼자라면 불가능 했을지도 모르는 많은 것들이 좋은 팀워크를 바탕으로 팀원 서로간에 많은 시너지 효과를 얻어 해결 할 수 있었던 점이 팀 프로젝트의 본질이란 생각이 들었습니다.

아직 많이 부족하지만 프로젝트를 진행하면서 ‘나는 할 수 있다’라는 자신감이 생겨났습니다. 앞으로도 어떤 어려움을 겪는다 하여도 이 순간을 기억하며 ‘나는 할 수 있다’는 마음가짐으로 임하겠습니다.

이번 프로젝트를 무사히 마칠 수 있던 이유를 뽑자면

1. SVN 사용으로 팀원들이 했던 작업을 합치는 시간을 줄일 수 있었다.

2. 팀원 각각에게 주어진 역할은 그저 문서화 하기 위함이고 누구라고 말 할 것

없이 모두가 서로서로 도와주며 윈윈하였다.

3. 처음부터 끝까지 어느 사소한 것이라도 의사소통하여 현재의 상황파악을 하였고

이를 바탕으로 회의를 통해 서로의 의견을 맞춰 좋은 팀워크를 유지 할 수 있었다.