

**설계 명세서**

**Team H**

2011314869 서동빈

2011311660 조은성

2012312945 김동민

목차

내용

[1. Preface 16](#_Toc467699571)

[1.1. Readership 16](#_Toc467699572)

[1.2. Document Structure 16](#_Toc467699573)

[A. Preface 16](#_Toc467699574)

[B. Introduction 16](#_Toc467699575)

[C. System Architecture 16](#_Toc467699576)

[D. Item Subsystem 17](#_Toc467699577)

[E. Auction Subsystem 17](#_Toc467699578)

[F. Customer Center Subsystem 17](#_Toc467699579)

[G. User Information Subsystem 17](#_Toc467699580)

[H. Item Admin Subsystem 17](#_Toc467699581)

[I. Auction Admin Subsystem 17](#_Toc467699582)

[J. ItemType/Bland Admin Subsystem 18](#_Toc467699583)

[K. User Admin Subsystem 18](#_Toc467699584)

[L. Order Admin Subsystem 18](#_Toc467699585)

[M. Customer Center Admin Subsystem 18](#_Toc467699586)

[N. Popup Admin Subsystem 18](#_Toc467699587)

[O. Communication Protocol 18](#_Toc467699588)

[P. Database Design 19](#_Toc467699589)

[Q. Testing Plan 19](#_Toc467699590)

[R. Development Environment 19](#_Toc467699591)

[S. Index 19](#_Toc467699592)

[1.3. Versioin of the Document 19](#_Toc467699593)

[A. Version format 19](#_Toc467699594)

[B. Version management policy 19](#_Toc467699595)

[C. Version update history 20](#_Toc467699596)

[2. Introduction 20](#_Toc467699597)

[2.1. Applied Tool 21](#_Toc467699598)

[2.2. Major Constraints 21](#_Toc467699599)

[A. Management and Technical Constraints 21](#_Toc467699600)

[3. System Architecture 21](#_Toc467699601)

[3.1. System Organization 22](#_Toc467699602)

[3.2. SE Shoes System Architecture 23](#_Toc467699603)

[3.3. Deployment Diagram 27](#_Toc467699604)

[3.4. Package Diagram 28](#_Toc467699605)

[4. Item SubSystem 29](#_Toc467699606)

[4.1. Class Diagram 29](#_Toc467699607)

[I. itemController 29](#_Toc467699608)

[A. Attributes 29](#_Toc467699609)

[B. Operations 30](#_Toc467699610)

[II. itemRouter 30](#_Toc467699611)

[A. Attributes 30](#_Toc467699612)

[B. Operations 30](#_Toc467699613)

[III. item 31](#_Toc467699614)

[A. Attributes 31](#_Toc467699615)

[B. Operations 31](#_Toc467699616)

[IV. cartRouter 31](#_Toc467699617)

[A. Attributes 31](#_Toc467699618)

[B. Operations 31](#_Toc467699619)

[V. cart 32](#_Toc467699620)

[A. Attributes 32](#_Toc467699621)

[B. Operations 32](#_Toc467699622)

[VI. orderRouter 32](#_Toc467699623)

[A. Attributes 32](#_Toc467699624)

[B. Operations 32](#_Toc467699625)

[VII. order 33](#_Toc467699626)

[A. Attributes 33](#_Toc467699627)

[B. Operations 33](#_Toc467699628)

[VIII. order\_item 33](#_Toc467699629)

[A. Attributes 33](#_Toc467699630)

[B. Operations 33](#_Toc467699631)

[4.2. Sequence Diagram 34](#_Toc467699632)

[A. 상품 보기 34](#_Toc467699633)

[B. 장바구니 담기 34](#_Toc467699634)

[C. 장바구니 삭제 35](#_Toc467699635)

[D. 상품 구매 35](#_Toc467699636)

[4.3. State Diagram 36](#_Toc467699637)

[A. 상품 보기 36](#_Toc467699638)

[B. 장바구니 담기 36](#_Toc467699639)

[C. 장바구니 삭제 36](#_Toc467699640)

[D. 상품 구매 37](#_Toc467699641)

[5. Auction Subsystem 37](#_Toc467699642)

[5.1. Class Diagram 38](#_Toc467699643)

[I. auctionController 38](#_Toc467699644)

[A. Attributes 38](#_Toc467699645)

[B. Operations 39](#_Toc467699646)

[II. auctionRouter 39](#_Toc467699647)

[A. Attributes 39](#_Toc467699648)

[B. Operations 39](#_Toc467699649)

[III. auction 40](#_Toc467699650)

[A. Attributes 40](#_Toc467699651)

[B. Operations 40](#_Toc467699652)

[5.2. Sequence Diagram 41](#_Toc467699653)

[A. 상품 보기 41](#_Toc467699654)

[B. 입찰하기 41](#_Toc467699655)

[5.3. State Diagram 42](#_Toc467699656)

[A. 상품 보기 42](#_Toc467699657)

[B. 입찰하기 42](#_Toc467699658)

[6. Customer Center Subsystem 42](#_Toc467699659)

[6.1. Class Diagram 43](#_Toc467699660)

[I. boardController 44](#_Toc467699661)

[A. Attributes 44](#_Toc467699662)

[B. Operations 44](#_Toc467699663)

[II. boardRouter 44](#_Toc467699664)

[A. Attributes 44](#_Toc467699665)

[B. Operations 44](#_Toc467699666)

[III. board 45](#_Toc467699667)

[A. Attributes 45](#_Toc467699668)

[B. Operations 45](#_Toc467699669)

[IV. board\_attachment 46](#_Toc467699670)

[A. Attributes 46](#_Toc467699671)

[B. Operations 46](#_Toc467699672)

[6.2. Sequence Diagram 47](#_Toc467699673)

[A. 게시글 보기 47](#_Toc467699674)

[B. 환불 문의 작성하기 47](#_Toc467699675)

[C. 환불 문의 수정하기 48](#_Toc467699676)

[D. 환불 문의 삭제하기 48](#_Toc467699677)

[6.3. State Diagram 49](#_Toc467699678)

[A. 게시글 보기 49](#_Toc467699679)

[B. 환불 문의 작성하기 49](#_Toc467699680)

[C. 환불 문의 수정하기 49](#_Toc467699681)

[D. 환불 문의 삭제하기 50](#_Toc467699682)

[7. User Information SubSystem 50](#_Toc467699683)

[7.1. Class Diagram 51](#_Toc467699684)

[I. userController 52](#_Toc467699685)

[A. Attributes 52](#_Toc467699686)

[B. Operations 52](#_Toc467699687)

[II. userRouter 52](#_Toc467699688)

[A. Attributes 52](#_Toc467699689)

[B. Operations 52](#_Toc467699690)

[III. User 53](#_Toc467699691)

[A. Attributes 53](#_Toc467699692)

[B. Operations 53](#_Toc467699693)

[7.2. Sequence Diagram 54](#_Toc467699694)

[A. 회원가입 54](#_Toc467699695)

[B. 로그인 55](#_Toc467699696)

[C. 로그아웃 55](#_Toc467699697)

[D. 회원정보 변경 56](#_Toc467699698)

[E. ID 찾기 56](#_Toc467699699)

[F. PW 찾기 57](#_Toc467699700)

[G. 회원탈퇴 57](#_Toc467699701)

[7.3. State Diagram 58](#_Toc467699702)

[A. 회원가입 58](#_Toc467699703)

[B. 로그인 58](#_Toc467699704)

[C. 로그아웃 58](#_Toc467699705)

[D. 회원정보 변경 59](#_Toc467699706)

[E. ID 찾기 59](#_Toc467699707)

[F. PW 찾기 59](#_Toc467699708)

[G. 회원탈퇴 60](#_Toc467699709)

[8. Item Admin SubSystem 60](#_Toc467699710)

[8.1. Class Diagram 61](#_Toc467699711)

[I. itemAdminController 62](#_Toc467699712)

[A. Attributes 62](#_Toc467699713)

[B. Operations 62](#_Toc467699714)

[II. itemAdminRouter 62](#_Toc467699715)

[A. Attributes 62](#_Toc467699716)

[B. Operations 62](#_Toc467699717)

[III. itemAdmin 63](#_Toc467699718)

[A. Attributes 63](#_Toc467699719)

[B. Operations 63](#_Toc467699720)

[8.2. Sequence Diagram 64](#_Toc467699721)

[A. 상품 보기 64](#_Toc467699722)

[B. 상품 등록 64](#_Toc467699723)

[C. 상품 수정 65](#_Toc467699724)

[D. 상품 삭제 65](#_Toc467699725)

[8.3. State Diagram 66](#_Toc467699726)

[A. 상품 보기 66](#_Toc467699727)

[B. 상품 등록 66](#_Toc467699728)

[C. 상품 수정 66](#_Toc467699729)

[D. 상품 삭제 67](#_Toc467699730)

[9. Auction Admin SubSystem 67](#_Toc467699731)

[9.1. Class Diagram 68](#_Toc467699732)

[I. auctionAdminController 69](#_Toc467699733)

[A. Attributes 69](#_Toc467699734)

[B. Operations 69](#_Toc467699735)

[II. auctionAdminRouter 69](#_Toc467699736)

[A. Attributes 69](#_Toc467699737)

[B. Operations 69](#_Toc467699738)

[III. auctionAdmin 70](#_Toc467699739)

[A. Attributes 70](#_Toc467699740)

[B. Operations 70](#_Toc467699741)

[9.2. Sequence Diagram 71](#_Toc467699742)

[A. 경매 보기 71](#_Toc467699743)

[B. 경매 등록 72](#_Toc467699744)

[C. 경매 수정 72](#_Toc467699745)

[D. 경매 삭제 73](#_Toc467699746)

[9.3. State Diagram 73](#_Toc467699747)

[A. 경매 보기 73](#_Toc467699748)

[B. 경매 등록 74](#_Toc467699749)

[C. 경매 수정 74](#_Toc467699750)

[D. 경매 삭제 74](#_Toc467699751)

[10. ItemType/Bland Admin SubSystem 75](#_Toc467699752)

[10.1. Class Diagram 76](#_Toc467699753)

[I. itemCategoryController 77](#_Toc467699754)

[A. Attributes 77](#_Toc467699755)

[B. Operations 77](#_Toc467699756)

[II. itemCategoryRouter 78](#_Toc467699757)

[A. Attributes 78](#_Toc467699758)

[B. Operations 78](#_Toc467699759)

[III. itemType 79](#_Toc467699760)

[A. Attributes 79](#_Toc467699761)

[B. Operations 80](#_Toc467699762)

[IV. Bland 80](#_Toc467699763)

[A. Attributes 80](#_Toc467699764)

[B. Operations 80](#_Toc467699765)

[10.2. Sequence Diagram 81](#_Toc467699766)

[A. 분류/브랜드 보기 81](#_Toc467699767)

[B. 분류/브랜드 추가 81](#_Toc467699768)

[C. 분류/브랜드 수정 82](#_Toc467699769)

[D. 분류/브랜드 삭제 82](#_Toc467699770)

[10.3. State Diagram 83](#_Toc467699771)

[A. 분류/브랜드 보기 83](#_Toc467699772)

[B. 분류/브랜드 추가 83](#_Toc467699773)

[C. 분류/브랜드 수정 83](#_Toc467699774)

[D. 분류/브랜드 삭제 84](#_Toc467699775)

[11. User Admin SubSystem 84](#_Toc467699776)

[11.1. Class Diagram 85](#_Toc467699777)

[I. userAdminController 86](#_Toc467699778)

[A. Attributes 86](#_Toc467699779)

[B. Operations 86](#_Toc467699780)

[II. userAdminRouter 86](#_Toc467699781)

[A. Attributes 86](#_Toc467699782)

[B. Operations 86](#_Toc467699783)

[III. userAdmin 87](#_Toc467699784)

[A. Attributes 87](#_Toc467699785)

[B. Operations 87](#_Toc467699786)

[11.2. Sequence Diagram 88](#_Toc467699787)

[A. 회원 목록 보기 88](#_Toc467699788)

[B. 회원 정보 수정 88](#_Toc467699789)

[C. 회원 삭제 89](#_Toc467699790)

[11.3. State Diagram 89](#_Toc467699791)

[A. 회원 목록 보기 89](#_Toc467699792)

[B. 회원 정보 수정 89](#_Toc467699793)

[C. 회원 삭제 90](#_Toc467699794)

[12. Order Admin SubSystem 90](#_Toc467699795)

[12.1. Class Diagram 91](#_Toc467699796)

[I. orderAdminController 92](#_Toc467699797)

[A. Attributes 92](#_Toc467699798)

[B. Operations 92](#_Toc467699799)

[II. orderAdminRouter 92](#_Toc467699800)

[A. Attributes 92](#_Toc467699801)

[B. Operations 92](#_Toc467699802)

[III. order 93](#_Toc467699803)

[A. Attributes 93](#_Toc467699804)

[B. Operations 93](#_Toc467699805)

[IV. order\_item 93](#_Toc467699806)

[A. Attributes 93](#_Toc467699807)

[B. Operations 93](#_Toc467699808)

[12.2. Sequence Diagram 94](#_Toc467699809)

[A. 판매 내역 보기 94](#_Toc467699810)

[B. 판매 내역 통계 94](#_Toc467699811)

[12.3. State Diagram 95](#_Toc467699812)

[A. 판매 내역 보기 95](#_Toc467699813)

[B. 판매 내역 통계 95](#_Toc467699814)

[13. Customer Center Admin SubSystem 95](#_Toc467699815)

[13.1. Class Diagram 96](#_Toc467699816)

[I. boardAdminController 97](#_Toc467699817)

[A. Attributes 97](#_Toc467699818)

[B. Operations 97](#_Toc467699819)

[II. boardAdminRouter 97](#_Toc467699820)

[A. Attributes 97](#_Toc467699821)

[B. Operations 97](#_Toc467699822)

[III. board 98](#_Toc467699823)

[A. Attributes 98](#_Toc467699824)

[B. Operations 98](#_Toc467699825)

[IV. board\_attachment 99](#_Toc467699826)

[A. Attributes 99](#_Toc467699827)

[B. Operations 99](#_Toc467699828)

[13.2. Sequence Diagram 100](#_Toc467699829)

[A. 게시글 보기 100](#_Toc467699830)

[B. 게시글 등록 100](#_Toc467699831)

[C. 게시글 수정 101](#_Toc467699832)

[D. 게시글 삭제 101](#_Toc467699833)

[E. 환불/교환 문의 답변하기 102](#_Toc467699834)

[13.3. State Diagram 102](#_Toc467699835)

[A. 게시글 보기 102](#_Toc467699836)

[B. 게시글 등록 103](#_Toc467699837)

[C. 게시글 수정 103](#_Toc467699838)

[D. 게시글 삭제 103](#_Toc467699839)

[E. 환불/교환 문의 답변하기 104](#_Toc467699840)

[14. Popup Admin SubSystem 104](#_Toc467699841)

[14.1. Class Diagram 105](#_Toc467699842)

[I. popupController 106](#_Toc467699843)

[A. Attributes 106](#_Toc467699844)

[B. Operations 106](#_Toc467699845)

[II. popupRouter 106](#_Toc467699846)

[A. Attributes 106](#_Toc467699847)

[B. Operations 106](#_Toc467699848)

[III. popup 107](#_Toc467699849)

[A. Attributes 107](#_Toc467699850)

[B. Operations 107](#_Toc467699851)

[14.2. Sequence Diagram 108](#_Toc467699852)

[A. 팝업 보기 108](#_Toc467699853)

[B. 팝업 추가 108](#_Toc467699854)

[C. 팝업 수정 109](#_Toc467699855)

[D. 팝업 삭제 109](#_Toc467699856)

[14.3. State Diagram 110](#_Toc467699857)

[A. 팝업 보기 110](#_Toc467699858)

[B. 팝업 추가 110](#_Toc467699859)

[C. 팝업 수정 110](#_Toc467699860)

[D. 팝업 삭제 111](#_Toc467699861)

[15. Communication Protocol 111](#_Toc467699862)

[15.1. AJAX 111](#_Toc467699863)

[15.2. JSON 111](#_Toc467699864)

[15.3. REST API 112](#_Toc467699865)

[15.4. 상품 시스템 프로토콜 112](#_Toc467699866)

[A. 상품 조회 시 112](#_Toc467699867)

[B. 상품을 장바구니에 추가했을 시 113](#_Toc467699868)

[C. 장바구니의 상품을 구매했을 시 114](#_Toc467699869)

[D. 장바구니의 상품을 삭제했을 시 114](#_Toc467699870)

[15.5. 게시판 시스템 프로토콜 115](#_Toc467699871)

[A. 게시글 조회 시 115](#_Toc467699872)

[B. 게시글 추가 시 117](#_Toc467699873)

[C. 게시글 수정 시 118](#_Toc467699874)

[D. 게시글 삭제 시 119](#_Toc467699875)

[15.6. 회원 정보 시스템 프로토콜 119](#_Toc467699876)

[A. 회원가입 시 119](#_Toc467699877)

[B. 로그인 시 120](#_Toc467699878)

[C. 로그아웃 시 122](#_Toc467699879)

[D. 회원정보 변경 시 122](#_Toc467699880)

[15.7. 팝업 시스템 프로토콜 123](#_Toc467699881)

[A. 팝업 조회 시 123](#_Toc467699882)

[B. 팝업 등록 시 125](#_Toc467699883)

[C. 팝업 수정 시 126](#_Toc467699884)

[D. 팝업 삭제 시 127](#_Toc467699885)

[16. Database Design 127](#_Toc467699886)

[16.1. ER Diagram 128](#_Toc467699887)

[16.2. Entities 128](#_Toc467699888)

[A. User 128](#_Toc467699889)

[B. Cart 129](#_Toc467699890)

[C. Order 129](#_Toc467699891)

[D. Order\_item 130](#_Toc467699892)

[E. Item 130](#_Toc467699893)

[F. Bland 131](#_Toc467699894)

[G. Item\_type 131](#_Toc467699895)

[H. Board 131](#_Toc467699896)

[I. Board\_attachment 132](#_Toc467699897)

[J. Popup 132](#_Toc467699898)

[K. Auction 133](#_Toc467699899)

[16.3. Relationship 133](#_Toc467699900)

[A. Cart 133](#_Toc467699901)

[B. Item 134](#_Toc467699902)

[C. Order 134](#_Toc467699903)

[D. Board 135](#_Toc467699904)

[E. Auction 135](#_Toc467699905)

[16.4. Relational Schema 136](#_Toc467699906)

[A. User 136](#_Toc467699907)

[B. Cart 136](#_Toc467699908)

[C. Order 136](#_Toc467699909)

[D. Order\_item 137](#_Toc467699910)

[E. Item 137](#_Toc467699911)

[F. Brand 138](#_Toc467699912)

[G. Item\_Type 138](#_Toc467699913)

[H. Board 138](#_Toc467699914)

[I. Board\_Attachment 139](#_Toc467699915)

[J. Popup 139](#_Toc467699916)

[K. Auction 140](#_Toc467699917)

[16.5. Normalization 140](#_Toc467699918)

[A. User 141](#_Toc467699919)

[B. Cart 141](#_Toc467699920)

[C. Order 141](#_Toc467699921)

[D. Order\_item 141](#_Toc467699922)

[E. Item 141](#_Toc467699923)

[F. Brand 142](#_Toc467699924)

[G. Item\_type 142](#_Toc467699925)

[H. Board 142](#_Toc467699926)

[I. Board\_Attachment 142](#_Toc467699927)

[J. Popup 142](#_Toc467699928)

[K. Auction 142](#_Toc467699929)

[16.6. SQL DDL 143](#_Toc467699930)

[A. User 143](#_Toc467699931)

[B. Cart 144](#_Toc467699932)

[C. Order 145](#_Toc467699933)

[D. Order\_item 146](#_Toc467699934)

[E. Item 147](#_Toc467699935)

[F. Brand 148](#_Toc467699936)

[G. Item\_type 148](#_Toc467699937)

[H. Board 149](#_Toc467699938)

[I. Board\_attachment 150](#_Toc467699939)

[J. Popup 151](#_Toc467699940)

[K. Auction 152](#_Toc467699941)

[17. Testing Plan 153](#_Toc467699942)

[17.1. Testing Policy 153](#_Toc467699943)

[17.2. Test Case 154](#_Toc467699944)

[I. 관리자 시스템 154](#_Toc467699945)

[A. 로그인 154](#_Toc467699946)

[B. 상품 관리 – 보기 154](#_Toc467699947)

[C. 상품 관리 – 등록 154](#_Toc467699948)

[D. 상품 관리 – 수정 154](#_Toc467699949)

[E. 상품 관리 – 삭제 154](#_Toc467699950)

[F. 회원 관리 – 보기 155](#_Toc467699951)

[G. 회원 관리 – 수정 155](#_Toc467699952)

[H. 회원 관리 – 삭제 155](#_Toc467699953)

[I. 판매 내역 관리 – 보기 155](#_Toc467699954)

[J. 판매 내역 관리 – 통계 155](#_Toc467699955)

[K. 게시판 관리 – 공지사항 155](#_Toc467699956)

[L. 게시판 관리 – FAQ 156](#_Toc467699957)

[M. 게시판 관리 – 환불 / 반품 문의 156](#_Toc467699958)

[N. 경매 관리 – 수정 156](#_Toc467699959)

[O. 경매 관리 – 삭제 156](#_Toc467699960)

[II. 사용자 시스템 157](#_Toc467699961)

[A. 로그인 157](#_Toc467699962)

[B. 로그아웃 157](#_Toc467699963)

[C. 상품 관리 – 조건 별 검색 157](#_Toc467699964)

[D. 상품 관리 – 장바구니 추가 157](#_Toc467699965)

[E. 상품 관리 – 최근 본 상품 157](#_Toc467699966)

[F. 회원 정보 관리 – 회원 가입 158](#_Toc467699967)

[G. 회원 탈퇴 158](#_Toc467699968)

[H. ID / PW 찾기 158](#_Toc467699969)

[I. 회원정보 수정 158](#_Toc467699970)

[J. 마이페이지 – 구매 내역 보기 159](#_Toc467699971)

[K. 마이페이지 – 지갑 159](#_Toc467699972)

[L. 장바구니 – 보기 159](#_Toc467699973)

[M. 장바구니 – 삭제 159](#_Toc467699974)

[N. 장바구니 – 구매 159](#_Toc467699975)

[O. 고객센터 – 공지사항 160](#_Toc467699976)

[P. 고객센터 – FAQ 160](#_Toc467699977)

[Q. 고객센터 – 환불 / 반품 문의 160](#_Toc467699978)

[18. Development Environment 161](#_Toc467699979)

[18.1. Operating Systems 161](#_Toc467699980)

[18.2. Running Environment 162](#_Toc467699981)

[A. WEB 162](#_Toc467699982)

[B. ubuntu 162](#_Toc467699983)

[C. Digital Ocean 163](#_Toc467699984)

[18.3. IDE 163](#_Toc467699985)

[A. Sublime Text 2.0 163](#_Toc467699986)

[18.4. Programming Language 164](#_Toc467699987)

[A. HTML5+CSS3+JavaScript 164](#_Toc467699988)

[B. Node.js(+Express) 165](#_Toc467699989)

[C. AngularJS 166](#_Toc467699990)

[18.5. Version Control 166](#_Toc467699991)

[A. Github 166](#_Toc467699992)

[18.6. Open Source 167](#_Toc467699993)

[19. Index 168](#_Toc467699994)

[19.1. Figure Index 168](#_Toc467699995)

[19.2. Diagram Index 168](#_Toc467699996)

[19.3. Table Index 173](#_Toc467699997)

## **Preface**

Preface 에서는 이 문서의 대상 독자들과 문서의 전체 구조, 각 부분의 내용에 대해 설명한다. 그리고 버전 관리 정책, 버전 변경 기록, 문서의 변경 사항들을 서술한다.

## **Readership**

시스템 End 유저, 클라이언트 엔지니어, 시스템 아키텍처, 소프트웨어 개발자

설계 명세서의 대상 독자는 다음과 같다. 시스템을 설계하는 시스템 아키텍처, 각 서브 시스템을 구현하는 소프트웨어 개발자 및 클라이언트 엔지니어 및 고객 기술 지원을 위한 서비스 팀도 해당한다.

## **Document Structure**

## Preface

Preface 에서는 이 문서의 대상 독자들과 문서의 전체 구조, 각 부분의 내용에 대해 설명한다. 그리고 버전 관리 정책, 버전 변경 기록, 문서의 변경 사항들을 서술한다.

## Introduction

Introduction 에서는 사용자 관점에서의 시스템에 대해 서술한다.

## System Architecture

System Architecture 부분에서는 목표 시스템에 대한 전체 개요를 서술한다. 전체 시스템의 Architecture와 system organization을 Block diagram으로 도식화한 후, 각각 서브 시스템들을 도식화 하고, 서브 시스템들의 관계와 실제 배포 형태를 Package diagram 및 Deployment diagram으로 표현한다.

## Item Subsystem

System Architecture 부분에서는 목표 시스템에 대한 전체 개요를 서술한다. 전체 시스템의 Architecture와 system organization을 Block diagram으로 도식화한 후, 각각 서브 시스템들을 도식화 하고, 서브 시스템들의 관계와 실제 배포 형태를 Package diagram 및 Deployment diagram으로 표현한다.

## Auction Subsystem

이 부분에서는 사용자가 웹 어플리케이션을 통해 경매 상품을 조회하거나 입찰을 하여 구매하는 역할을 하는 경매 서브시스템에 대해 서술한다.

## Customer Center Subsystem

이 부분에서는 사용자가 웹 어플리케이션을 통해 공지사항, FAQ, 교환/환불 게시판의 게시글을 조회하거나 교환/환불 게시판에 문의글을 작성하는 역할을 하는 고객센터 서브시스템에 대해 서술한다.

## User Information Subsystem

이 부분에서는 사용자가 웹 어플리케이션을 통해 회원가입 또는 탈퇴를 하거나 로그인/로그아웃을 하거나 ID/PW를 모를 때 찾거나 회원정보를 변경하는 역할을 하는 회원정보 서브시스템에 대해 서술한다.

## Item Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 상품을 조회, 등록, 수정, 삭제하는 역할을 하는 상품 관리 서브시스템에 대해 서술한다.

## Auction Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 경매 상품을 조회, 등록, 수정, 삭제하는 역할을 하는 경매 관리 서브시스템에 대해 서술한다.

## ItemType/Bland Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 상품의 분류, 브랜드의 조회, 등록, 수정, 삭제하는 역할을 하는 상품 분류/브랜드 관리 서브시스템에 대해 서술한다.

## User Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 회원을 조회, 수정, 삭제하는 역할을 하는 회원 관리 서브시스템에 대해 서술한다. 회원가입 기능이 있기 때문에 회원 추가 기능은 없다.

## Order Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 회원들을 주문한 정보에 대해 조회, 통계화 하는 역할을 하는 주문 관리 서브시스템에 대해 서술한다.

## Customer Center Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 게시글(공지사항, FAQ, 환불 문의 답변)을 조회, 쓰기, 수정, 삭제를 하는 역할을 하는 게시판 관리 서브시스템에 대해 서술한다.

## Popup Admin Subsystem

이 부분에서는 관리자가 웹 어플리케이션을 통해 팝업을 조회, 쓰기, 수정, 삭제를 하는 역할을 하는 팝업 관리 서브시스템에 대해 서술한다.

## Communication Protocol

Communication Protocol 에서는 서브 시스템들이 상호 통신하기 위하여 필수적으로 준수해야 하는 프로토콜에 대하여 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지의 형식과 용도, 의미를 설명한다.

## Database Design

Database design에서는 요구사항 명세서에 기술하였던 요구사항들을 바탕으로 데이터베이스를 설계하고 설명한다. 데이터베이스에 대한 ER diagram을 제시하고 각 Entity에 대해서 설명한다. 데이터베이스 관리 시스템으로는 MySQL을 사용한다.

## Testing Plan

Testing Plan 에서는 테스트 정책과 테스트 케이스에 대해 설명한다.

## Development Environment

Development Environment 에서는 실제 시스템 개발을 위하여 필요한 개발 환경을 설명한다. 개발 과정에서의 버전 관리 도구를 제시하고, 프로그래밍을 더 쉽게 해주는 프레임워크에 대하여 서술한다.

## Index

Index 에서는 문서를 빠르게 색인 할 수 있도록 인덱스를 제공한다. 다이어그램 및 표 등의 인덱스를 제공한다.

## **Versioin of the Document**

## Version format

버전 번호는 major. Minor[.maintenance]로 구성되며, 문서의 버전은 0.1부터 시작한다.

## Version management policy

설계명세서를 수정할 때 마다 버전을 업데이트 한다. 다만 변경 간의 간격이 1 시간 이내 일 때 에는 버전 번호를 업데이트 하지 않고 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경 할 때에는 minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있는 경우 major number를 변경한다. 이미 작성한 부분에 대해서 오타 수정을 하거나, 문서의 구조를 변경할 경우 maintenance number를 추가하거나 변경한다.

## Version update history

|  |  |  |
| --- | --- | --- |
| Version | Modified date | Explanation |
| 0.1 | 2016-11-11 | 문서의 초안과 표지작성. |
| 1.0 | 2016-11-11 | Preface 작성. |
| 2.0 | 2016-11-12 | Introduction 작성 |
| 3.0 | 2016-11-13 | Glossary 작성 |
| 4.0 | 2016-11-15 | User Requirements Definition 작성 |
| 5.0 | 2016-11-20 | System Architecture, System Requirements Specification 작성 |
| 6.0 | 2016-11-21 | System Models 작성 |
| 7.0 | 2016-11-22 | System Evolution, Index, Appendices, References 작성 |

## **Introduction**

SE Shoes 시스템은 넓은 고객층과 더불어 의류제품들 중 매니아 층을 형성하기 좋은 신발을 모바일 멀티 샵에서 판매 하는 시스템이다. 오프라인 매장이나 PC쇼핑에서는 이루어질 수 없었던 조건부 검색, 장바구니와 같은 Personalization과 가상지갑을 통한 전자결제 그리고 푸시 알림을 통한 고객의 거래 유도 등의 기능들을 모바일에서 구현함으로써 고객들의 편리한 거래를 유도할 수 있을 것이라고 예상된다.

## **Applied Tool**

본 문서에 첨부되어 있는 모든 시스템 설계도와 diagram의 작성에는 ‘StarUML’ tool이 사용되었다. ‘StarUML’ tool은 다양한 diagram 및 flow chart를 그리는 데에 최적화 되어있는 소프트웨어이다. 특히, 이 소프트웨어는 대부분의 diagram tool 을 지원하며 완성된 diagram 을 pdf, png, jpeg 등 다양한 형식의 확장자로 저장이 가능하다. 특별한 가입 절차 없이 홈페이지에서 다운 받을 수 있어 자유로운 작업이 가능하다.

## **Major Constraints**

## **Management and Technical Constraints**

일반적인 쇼핑몰에서 상품 구매 시 결제 방법이 다양하다. 신용카드, 무통장, 간편결제 등

여러가지가 있지만 결제 모듈을 적용하기에는 어려움이 있어 가상지갑을 이용한 결제 방법으로 구매를 한정하였다.

## **System Architecture**

System Architecture 부분에서는 목표 시스템에 대한 전체 개요를 서술한다. 전체 시스템의 Architecture와 system organization을 Block diagram으로 도식화한 후, 각각 서브 시스템들을 도식화 하고, 서브 시스템들의 관계와 실제 배포 형태를 Package diagram 및 Deployment diagram으로 표현한다.

## **System Organization**

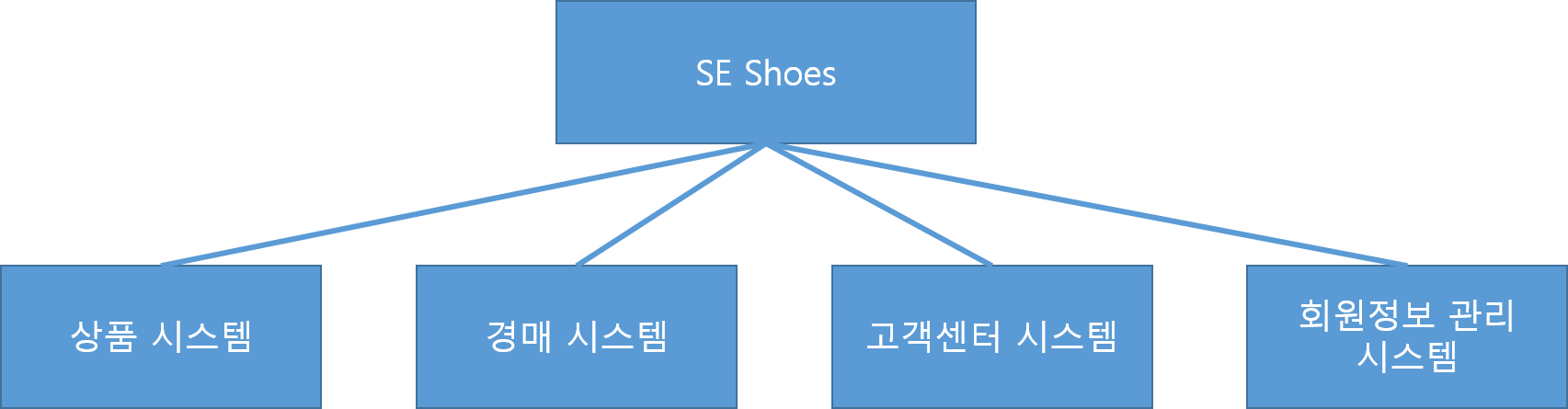


Diagram 3.1 사용자 관점 System Organization Block Diagram

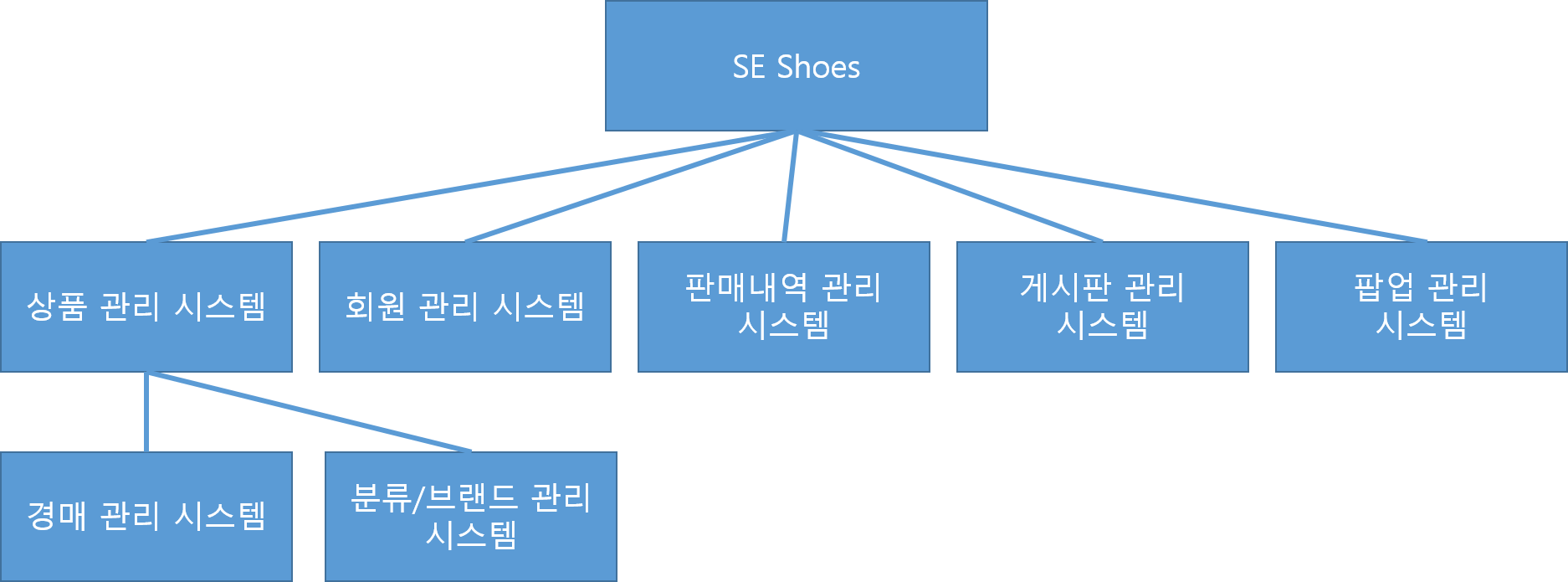


Diagram 3.2 관리자 관점 System Organization Block Diagram

## **SE Shoes System Architecture**

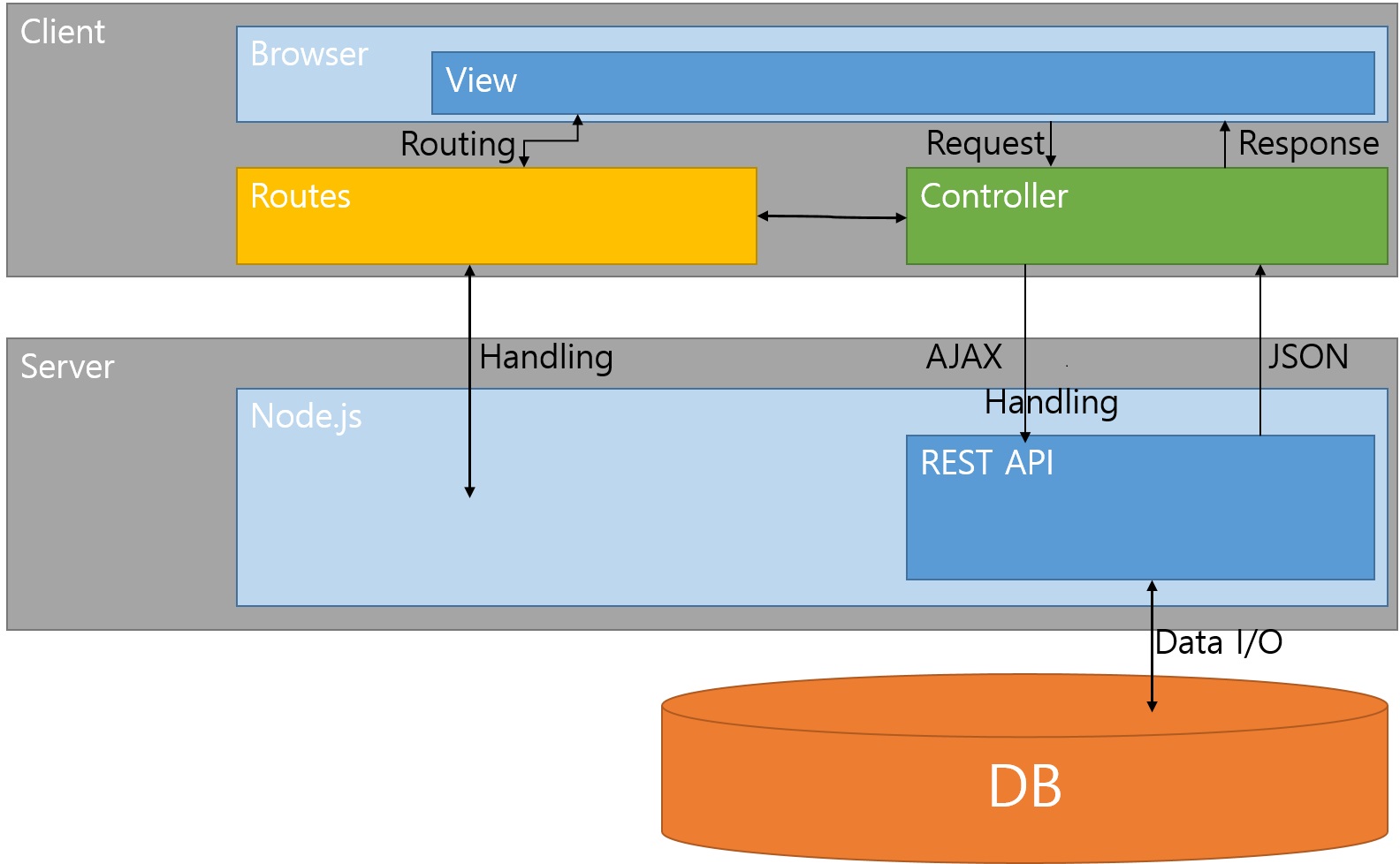
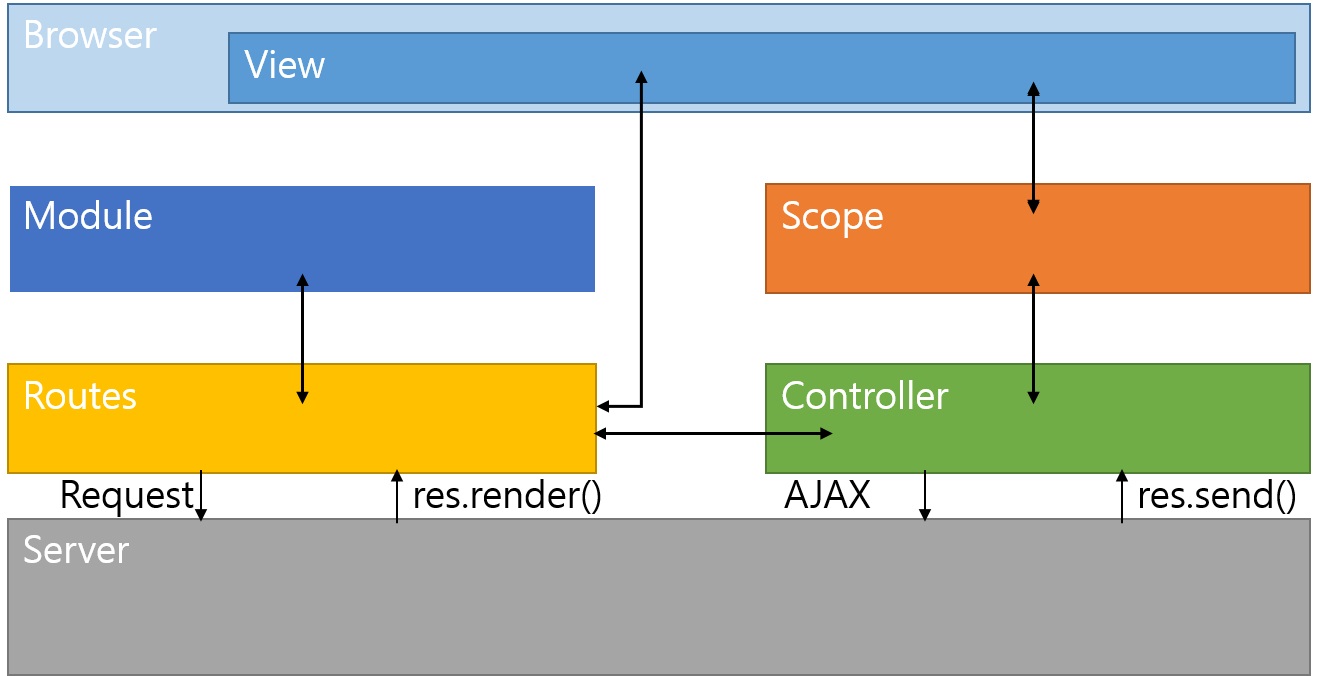


Figure 3.1 전체 시스템 아키텍처

Client는 자신의 모바일 기기에서 웹 브라우저를 통해 SE Shoes 서비스에 접속한다. Routes는 Client의 접속 URL에 따라 맞는 View를 제공하고, Controller와 View에 사용되는 module을 관리한다. 그리고 Controller는 Client와 Server사이에서 브라우저를 통해 입력 및 전달되는 데이터를 받아서 Server로 전송하게 되고 DB통신을 통해 주고 받은 데이터를 다시 Client로 제공한다.

Node.js는 URL을 통한 Client의 접근을 돕고 REST API를 통해 Client와 DB사이의 데이터 통신을 하는 역할 및 예외처리를 담당한다.

Client, DB 사이의 통신은 REST API라는 아키텍처를 이용하는데, 서버로 AJAX를 통해 요청을 보내고 서버는 DB에 요청에 알맞는 데이터를 주고받은 후 JSON 형식으로 결과물을 반환하는 역할을 한다.

**Figure 3.2 클라이언트 부분 시스템 아키텍처**

SE Shoes 서비스의 시스템 중 클라이언트 사이드의 구조도를 나타낸다. 클라이언트의 뷰 템플릿 엔진으로 AngularJS를 이용하였으며 구조는 크게 View, Controller, Routes, Scope, Module로 이루어져 있다.

View는 사용자에게 직접적으로 보이는 부분으로 HTML, CSS, JAVASCRIPT로 이루어져 있다.

Controller는 View에 출력해줄 변수나 이벤트를 생성, 관리하는 부분이며 기능 단위(유저, 상품, 게시판 등등)로 Controller가 나뉘어져 있다.

Scope는 View와 Controller 사이의 연결을 담당하는 객체이며 변수나 함수 형태로 이루어져 있다. View의 동적인 변화에 맞게 객체를 수정하고 Controller는 이에 대한 반응을 한다.

Routes는 사용자의 페이지 이동 Request에 맞게 출력하는 HTML 파일과 Controller을 Handling 하는 부분으로, 네트워크 기기 중 라우터와 같은 기능을 수행한다.

Module은 sub-system의 메인인 부분으로 큰 기능 단위를 나타낸다.

Node.js는 크롬에서 만든 자바스크립트 엔진 V8을 이용해서 만든 서버 사이드 미들웨어이며, Express는 Node.js에서 쓰이는 웹 프레임워크 이다.

Client에서 페이지 이동에 대한 Request가 오면 Middle ware를 거쳐서 Route로 이동하게 된다. Route는 url에 맞는 view 파일을 res.render()를 이용하여 Client에게 제공한다. 또는 Client에서 데이터에 대한 Request가 오면 Middle ware를 거쳐 Route 내부의 REST API 아키텍처로 이동하게 된다. 그리고 Sequelize를 이용하여 DB와 통신을 한 후 데이터를 res.send()를 이용하여 JSON 형태로 Client에게 제공한다.

Sequelize는 DB의 데이터를 서버가 이해할 수 있는 형태로 호환을 해주는 모듈로 ORM(Object-relational mapping)이라고도 불린다. REST API를 통해 메소드를 요청하면 DB에서 그 상황에 맞는 데이터를 가져온다.

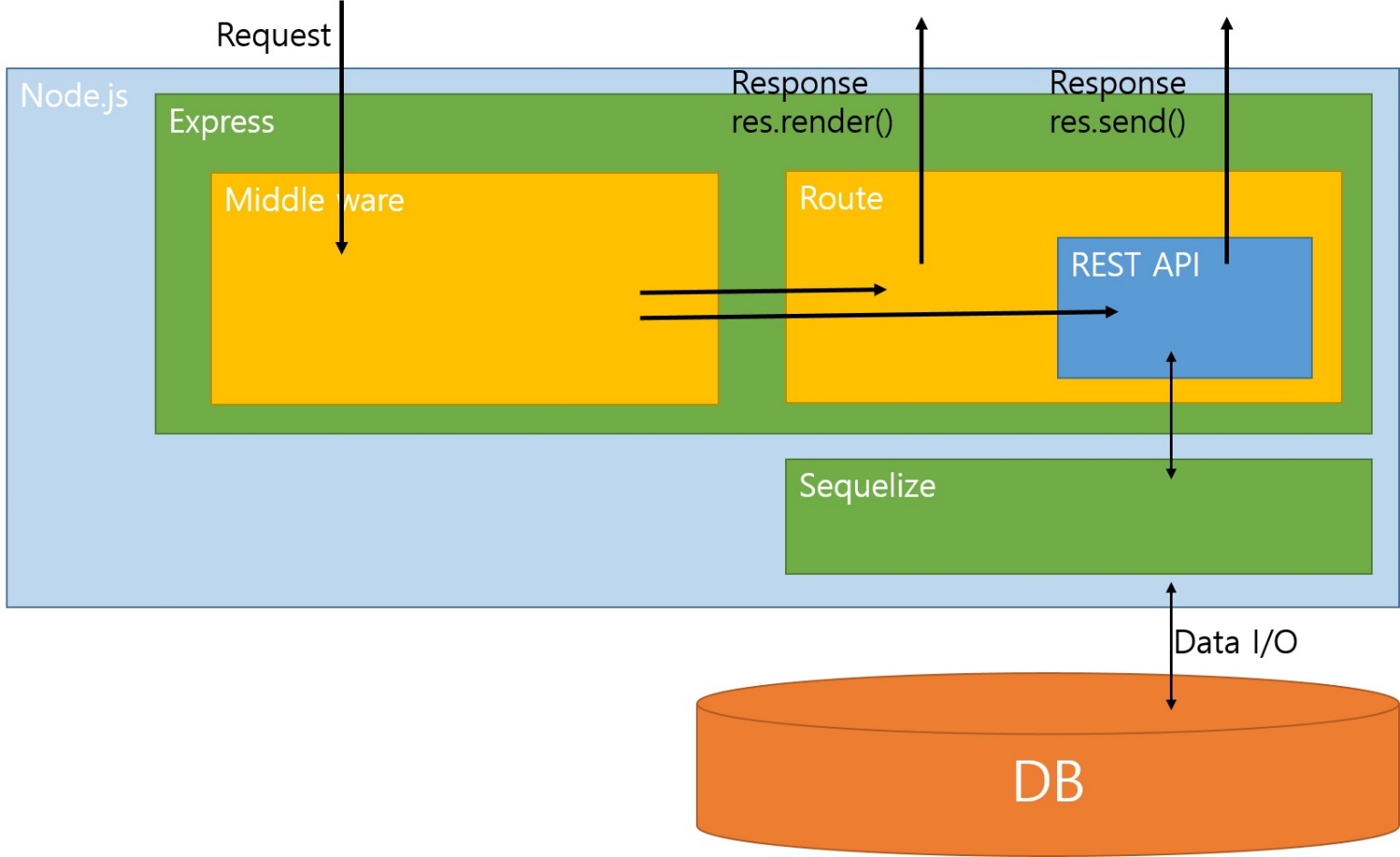


Figure 3.3 서버 부분 시스템 아키텍처

SE Shoes 서비스의 시스템 중 서버 사이드의 구조도를 나타낸다. Node.js는 크롬에서 만든 자바스크립트 엔진 V8을 이용해서 만든 서버 사이드 미들웨어이며, Express는 Node.js에서 쓰이는 웹 프레임워크 이다.

Client에서 페이지 이동에 대한 Request가 오면 Middle ware를 거쳐서 Route로 이동하게 된다. Route는 url에 맞는 view 파일을 res.render()를 이용하여 Client에게 제공한다. 또는 Client에서 데이터에 대한 Request가 오면 Middle ware를 거쳐 Route 내부의 REST API 아키텍처로 이동하게 된다. 그리고 Sequelize를 이용하여 DB와 통신을 한 후 데이터를 res.send()를 이용하여 JSON 형태로 Client에게 제공한다.

Sequelize는 DB의 데이터를 서버가 이해할 수 있는 형태로 호환을 해주는 모듈로 ORM(Object-relational mapping)이라고도 불린다. REST API를 통해 메소드를 요청하면 DB에서 그 상황에 맞는 데이터를 가져온다.

## **Deployment Diagram**

C:\Users\KIM\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Deployment Diagram.png

Diagram 3.3 SE Shoes Deployment Diagram

## **Package Diagram**



Diagram 3.4 SE Shoes Package Diagram

## **Item SubSystem**

이 부분에서는 사용자가 웹 어플리케이션을 통해 상품을 조회하거나 장바구니에 상품을 담아서 구매를 하는 역할을 하는 상품 서브시스템에 대해 서술한다.

## **Class Diagram**

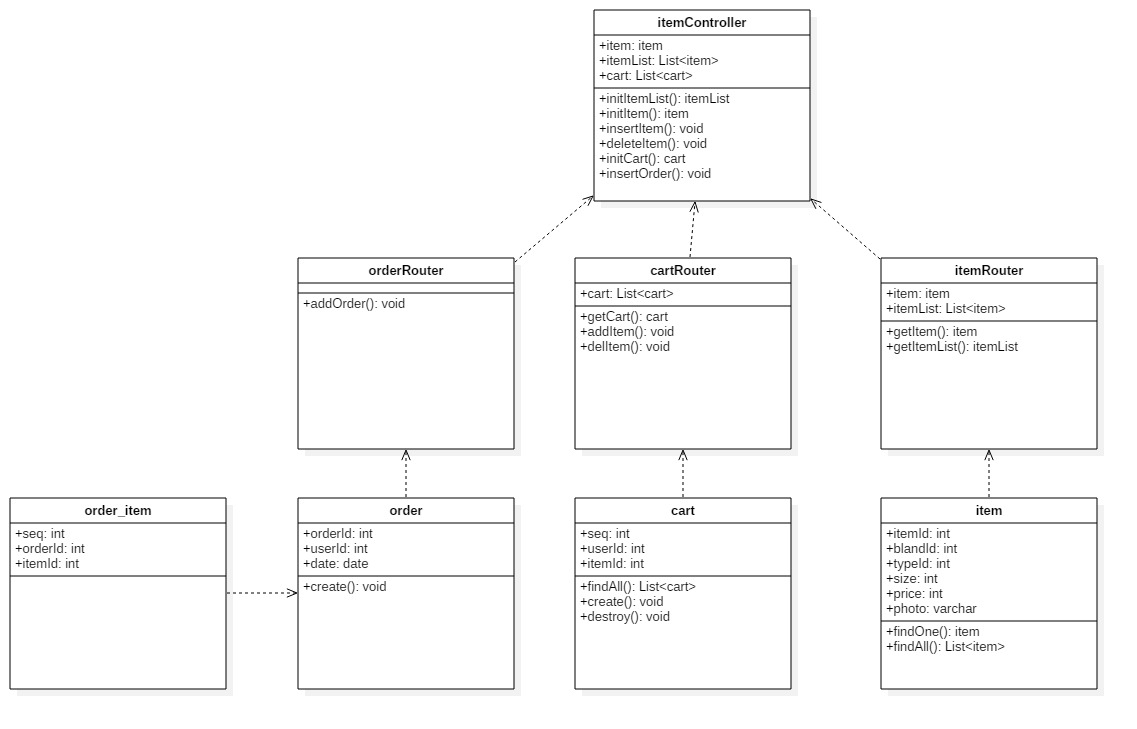


Diagram 4.1 Item SubSystem Class Diagram

## itemController

## Attributes

item : 개별 상품을 조회할 때 이용되는 상품 한 개에 대한 객체이다.

itemList : (조건별)상품 조회에 이용되는 상품에 대한 리스트 객체이다.

cart : 유저가 담아놓을 장바구니에 대한 객체이다.

## Operations

initItemList() : (조건별)상품 조회 페이지 이동시 itemList를 가져오기 위한 함수이다.

initItem() : 개별 상품 조회 페이지 이동시 item을 가져오기 위한 함수이다.

insertItem() : 한 상품을 장바구니에 추가하기 위한 함수이다.

deleteItem() : 장바구니에 있는 상품을 제거하기 위한 함수이다.

initCart() : 장바구니 조회/주문 페이지 이동시 cart를 가져오기 위한 함수이다.

insertOrder() : 주문 성공시 데이터베이스에 주문 내용을 추가하는 함수이다.

## itemRouter

## Attributes

Item : 데이터베이스의 item 테이블에서 한 개의 item 정보를 가져와 담기 위한 객체이다.

itemList : 데이터베이스의 item 테이블에서 여러 개의 item 정보를 가져와 담기 위한 객체이다.

## Operations

getItem() : itemController의 initItem() 함수를 통해 호출되는 함수로 데이터베이스의 item테이블에서 한 개의 item 정보를 조회 요청하기 위한 함수이다.

getItemList() : itemController의 initItemList() 함수를 통해 호출되는 함수로 데이터베이스의 item테이블에서 여러 개의 item 정보를 조회 요청하기 위한 함수이다.

## item

## Attributes

itemId : 상품의 고유번호이다.

blandId : 상품의 브랜드번호이다.

typeId : 상품의 항목번호이다.

size : 상품의 사이즈이다.

price : 상품의 가격이다.

photo : 상품의 이미지 경로이다.

## Operations

findOne() : 데이터베이스의 item테이블에서 한 개의 item 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 item 테이블에서 여러 개의 item 정보를 가져오는 통신을 하는 함수이다.

## cartRouter

## Attributes

cart : 데이터베이스의 cart 테이블에서 한 유저에 대한 item 정보를 가져와 담기 위한 객체이다.

## Operations

getCart() : itemController의 initCart() 함수를 통해 호출되는 함수로 데이터베이스의 cart테이블에서 한 유저에 대한 item 정보를 조회 요청하기 위한 함수이다.

addItem() : itemController의 insertItem() 함수를 통해 호출되는 함수로 데이터베이스의 cart테이블에 한 유저에 대한 한 개의 item 정보를 추가 요청하기 위한 함수이다.

delItem() : itemController의 deleteItem() 함수를 통해 호출되는 함수로 데이터베이스의 cart테이블에서 한 유저에 대한 한 개의 item 정보를 삭제 요청하기 위한 함수이다.

## cart

## Attributes

seq : cart 테이블의 한 row에 대한 고유값이다.

userId : 어떤 사용자의 장바구니인지에 대한 식별값이다.

itemId : 장바구니에 담겨있는 상품에 대한 식별값이다.

## Operations

findAll() : 데이터베이스의 cart 테이블에서 한 개의 cart 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 cart 테이블에 한 개의 cart 정보를 추가하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 cart 테이블에 한 개의 cart 정보를 삭제하는 통신을 하는 함수이다.

## orderRouter

## Attributes

없음

## Operations

addOrder() : itemController의 insertOrder() 함수를 통해 호출되는 함수로 데이터베이스의 order테이블에 한 유저에 대한 order 정보를 추가 요청하고 order\_item 테이블에 주문 item 정보를 추가요청하기 위한 함수이다.

## order

## Attributes

orderId : 주문의 고유번호이다.

userId : 어떤 사용자의 주문인지에 대한 식별값이다.

date : 주문 시간이다.

## Operations

create() : 데이터베이스의 order 테이블에 한 개의 order 정보를 추가하는 통신과 order\_item 테이블에 여러 개의 order\_item 추가하는 통신을 요청하는 함수이다.

## order\_item

## Attributes

seq : order\_item 테이블의 한 row에 대한 고유값이다.

orderId : 어떤 주문에 대한 상품인지에 대한 식별값이다.

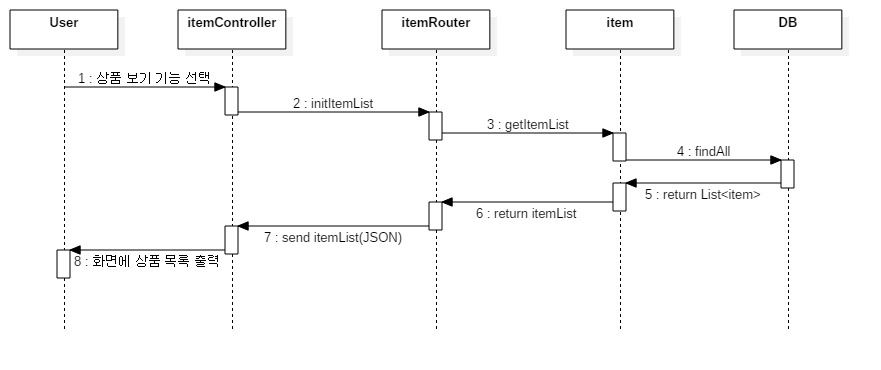
itemId : 주문에 포함되어 있는 상품에 대한 식별값이다.

## Operations

없음

## **Sequence Diagram**

## 상품 보기

 **Diagram 4.2 Sequence Diagram(상품 보기)**

## 장바구니 담기

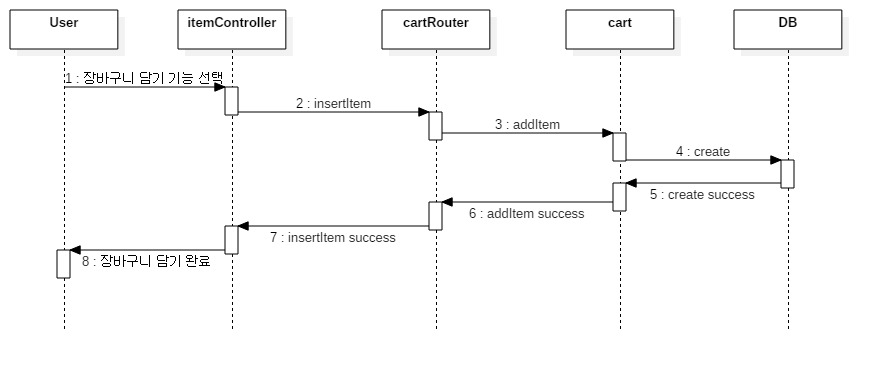


Diagram 4.3 Sequence Diagram(장바구니 담기)

## 장바구니 삭제

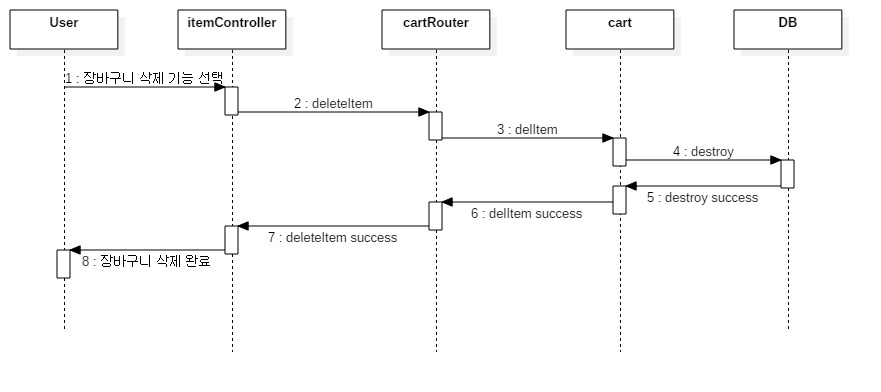


Diagram 4.4 Sequence Diagram(장바구니 삭제)

## 상품 구매

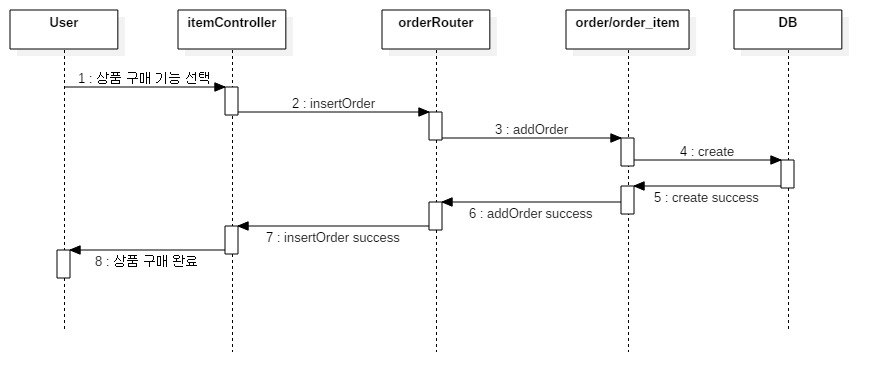


Diagram 4.5 Sequence Diagram(상품 구매)

## **State Diagram**

## 상품 보기

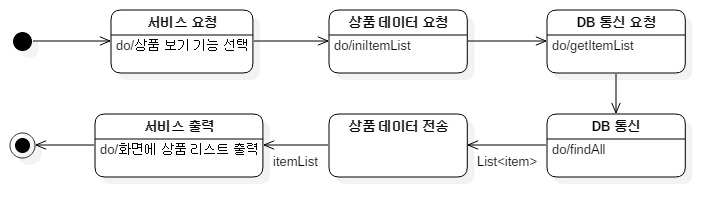


Diagram 4.6 State Diagram(상품 보기)

## 장바구니 담기

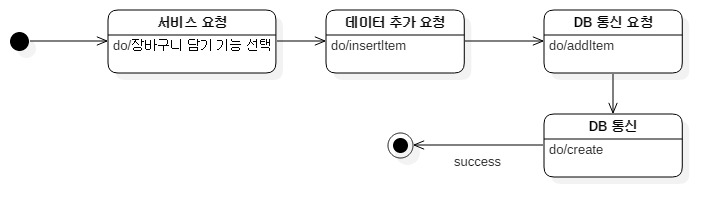


Diagram 4.7 State Diagram (장바구니 담기)

## 장바구니 삭제

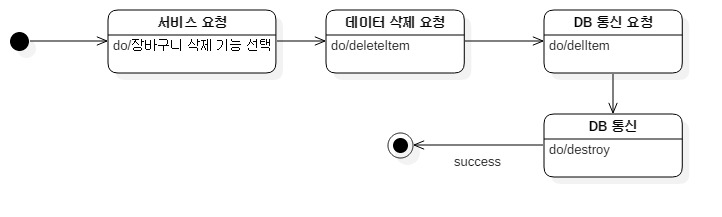


Diagram 4.8 State Diagram(장바구니 삭제)

## 상품 구매

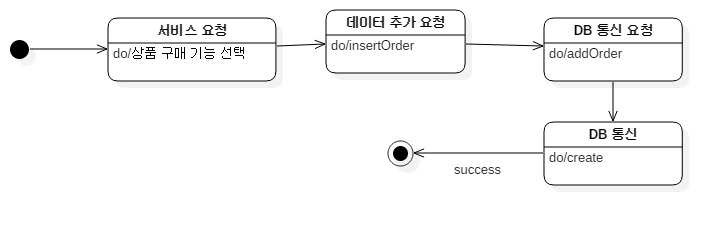


Diagram 4.9 State Diagram(상품 구매)

## **Auction Subsystem**

이 부분에서는 사용자가 웹 어플리케이션을 통해 경매 상품을 조회하거나 입찰을 하여 구매하는 역할을 하는 경매 서브시스템에 대해 서술한다.

## **Class Diagram**

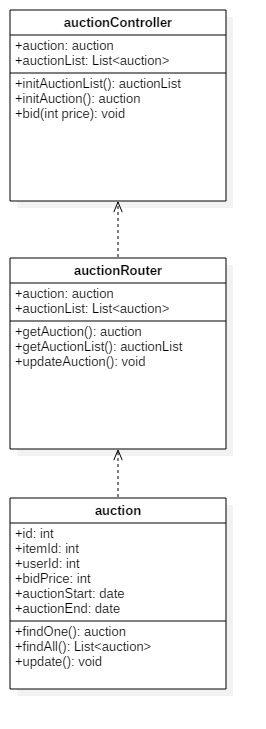


Diagram 5.1 Auction SubSystem Class Diagram

## auctionController

## Attributes

auction : 경매 시스템에 이용되는 한 경매 상품에 대한 객체이다.

auctionList : 경매 목록 조회에 이용되는 경매 상품에 대한 리스트 객체이다.

## Operations

initAuctionList() : 경매 상품 목록 조회 페이지 이동시 auctionList를 가져오기 위한 함수이다.

initAuction() : 경매 상품 조회 페이지 이동시 auction을 가져오기 위한 함수이다.

bid(int price) : 한 경매 상품을 상회입찰하기 위한 함수이다.

## auctionRouter

## Attributes

auction : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 가져와 담기 위한 객체이다.

auctionList : 데이터베이스의 auction 테이블에서 여러 개의 auction 정보를 가져와 담기 위한 객체이다.

## Operations

getAuction() : auctionController의 initAuction() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 조회 요청하기 위한 함수이다.

getAuctionList() : auctionController의 initAuctionList() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 여러 개의 auction 정보를 조회 요청하기 위한 함수이다.

updateAuction() : auctionController의 bid() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 조회 요청하기 위한 함수이다.

## auction

## Attributes

id : 경매의 고유번호이다.

itemId : 어떤 상품의 경매인지에 대한 식별값이다.

userId : 상회입찰자가 어떤 사용자인지에 대한 식별값이다.

bidPrice : 경매의 상회입찰금액이다.

auctionStart : 경매의 시작시간이다.

auctionEnd : 경매의 종료시간이다.

## Operations

findOne() : 데이터베이스의 auction테이블에서 한 개의 auction 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 auction테이블에서 여러 개의 auction 정보를 가져오는 통신을 하는 함수이다.

update() : 데이터베이스의 auction테이블에서 한 개의 auction 정보의 값을 갱신하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 상품 보기

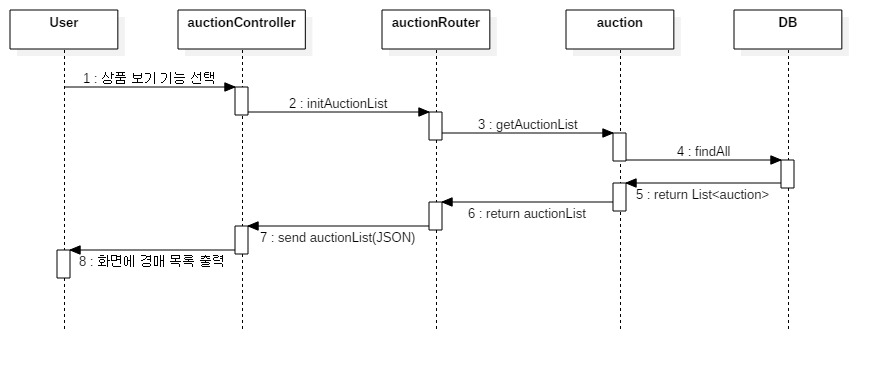


Diagram 5.2 Sequence Diagram(상품보기)

## 입찰하기

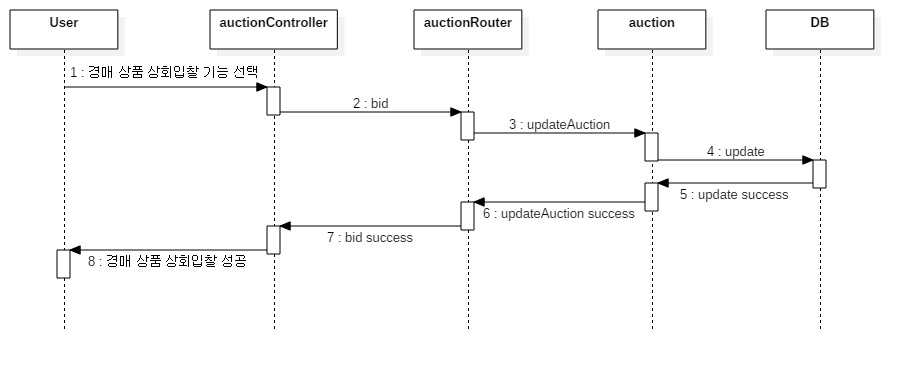


Diagram 5.3 Sequence Diagram(입찰하기)

## **State Diagram**

## 상품 보기

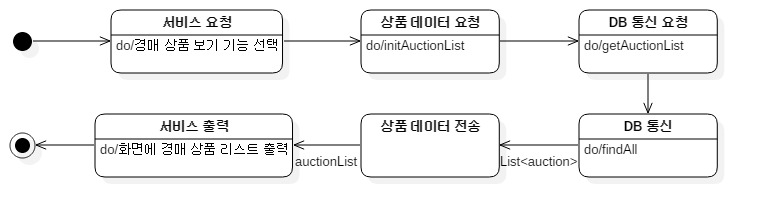


Diagram 5.4 State Diagram(상품 보기)

## 입찰하기

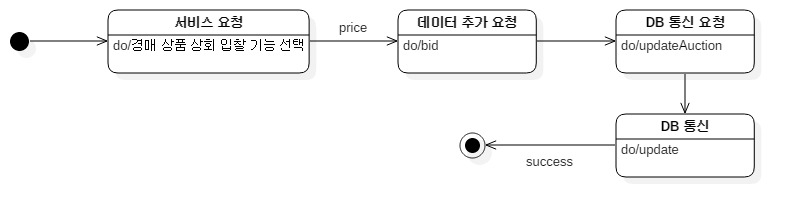


Diagram 5.5 State Diagram(입찰하기)

## **Customer Center Subsystem**

이 부분에서는 사용자가 웹 어플리케이션을 통해 공지사항, FAQ, 교환/환불 게시판의 게시글을 조회하거나 교환/환불 게시판에 문의글을 작성하는 역할을 하는 고객센터 서브시스템에 대해 서술한다.

## **Class Diagram**

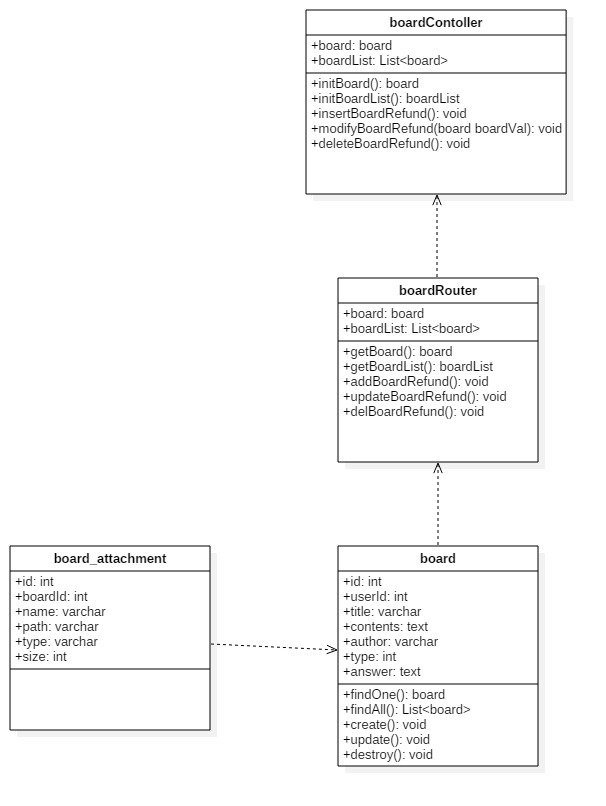


Diagram 6.1 Customer Center SubSystem Class Diagram

## boardController

## Attributes

board : 게시판 시스템에 이용되는 한 게시글에 대한 객체이다.

boardList : 게시판 시스템에 이용되는 여러 게시글에 대한 리스트 객체이다.

## Operations

initBoard() : (FAQ, 공지사항, 환불/교환 문의) 게시글 조회 board를 가져오기 위한 함수이다.

initBoardList() : (FAQ, 공지사항, 환불/교환 문의) 페이지 이동시 boardList를 가져오기 위한 함수이다.

insertBoardRefund() : 사용자가 환불/교환 게시글을 쓰기 위한 함수이다.

modifyBoardRefund(board boardVal) : 사용자가 환불/교환 게시글을 수정하기 위한 함수이다.

deleteBoardRefund() : 사용자가 환불/교환 게시글을 삭제하기 위한 함수이다.

## boardRouter

## Attributes

board : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 가져와 담기 위한 객체이다.

boardList : 데이터베이스의 board/board\_attachment 테이블에서 여러 개의 board 정보를 가져와 담기 위한 객체이다.

## Operations

getBoard() : boardController의 initBoard() 함수를 통해 호출되는 함수로 데이터베이스의 board /board\_attachment테이블에서 한 개의 board 정보를 조회 요청하기 위한 함수이다.

getBoardList() : boardController의 initBoardList() 함수를 통해 호출되는 함수로 데이터베이스의 board /board\_attachment 테이블에서 여러 개의 board 정보를 조회 요청하기 위한 함수이다.

addBoardRefund() : boardController의 insertBoardRefund() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 추가 요청하기 위한 함수이다.

updateBoardRefund() : boardController의 modifyBoardRefund() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 수정 요청하기 위한 함수이다.

delBoardRefund() : boardController의 deleteBoardRefund() 함수를 통해 호출되는 함수로 데이터베이스의 board 테이블에서 한 개의 board 정보를 삭제 요청하기 위한 함수이다.

## board

## Attributes

id : 게시글의 고유번호이다.

userId : 어떤 사용자의 게시글 인지에 대한 식별값이다.

title : 게시글의 제목이다.

contents : 게시글의 내용이다.

author : 게제자의 이름이다.

type : 게시글의 카테고리 타입니다.(공지사항, FAQ, 환불/교환 게시판)

answer : (환불/교환 게시판에서만 사용)관리자의 답변 내용이다.

## Operations

findOne() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 board/board\_attachment 테이블에서 여러 개의 board 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 추가하는 통신을 하는 함수이다.

update() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 수정하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 board 테이블에서 한 개의 board 정보를 삭제하는 통신을 하는 함수이다.

## board\_attachment

## Attributes

Id : 게시글 첨부파일의 고유번호이다.

boardId : 어떤 게시글의 첨부파일인지에 대한 식별값이다.

name : 첨부파일의 이름이다.

path : 첨부파일의 파일경로이다.

type : 첨부파일의 파일종류이다.

size : 첨부파일의 파일크기이다.

## Operations

없음

## **Sequence Diagram**

## 게시글 보기

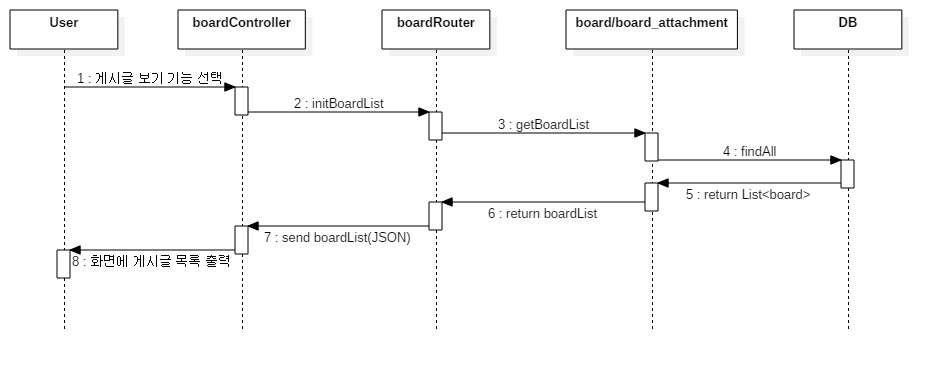


Diagram 6.2 Sequence Diagram(게시글 보기)

## 환불 문의 작성하기

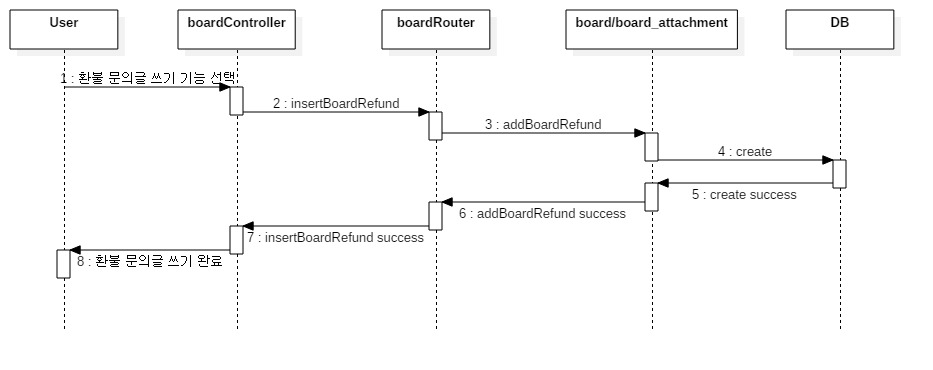


Diagram 6.3 Sequence Diagram(환불 문의 작성하기)

## 환불 문의 수정하기

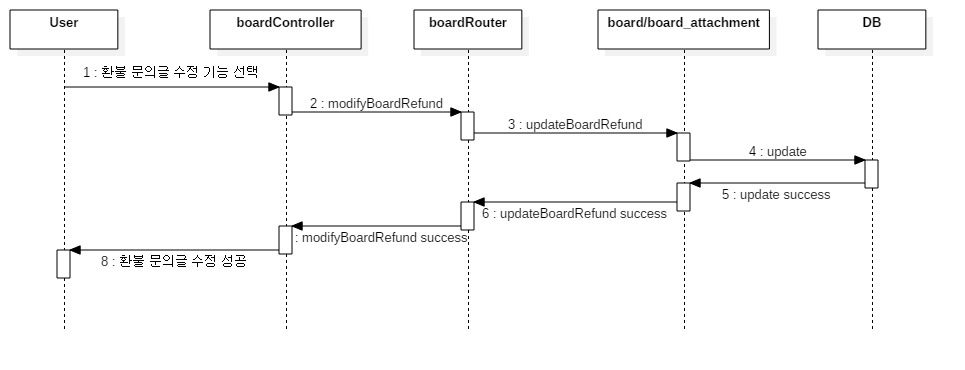


Diagram 6.4 Sequence Diagram(환불 문의 수정하기)

## 환불 문의 삭제하기

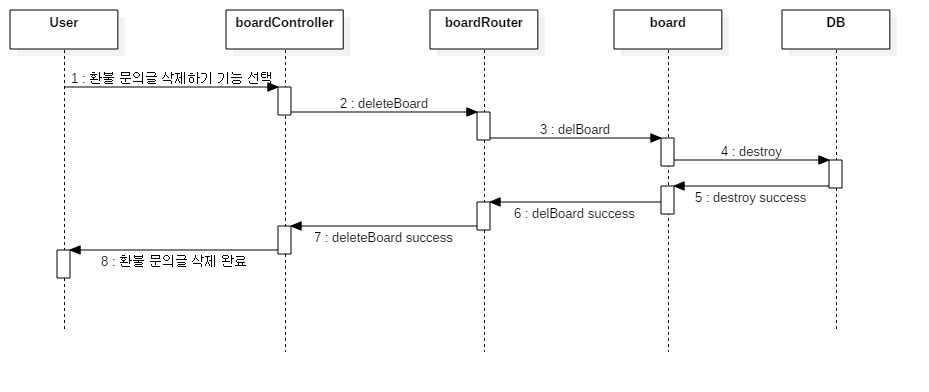


Diagram 6.5 Sequence Diagram(환불 문의 삭제하기)

## **State Diagram**

## 게시글 보기

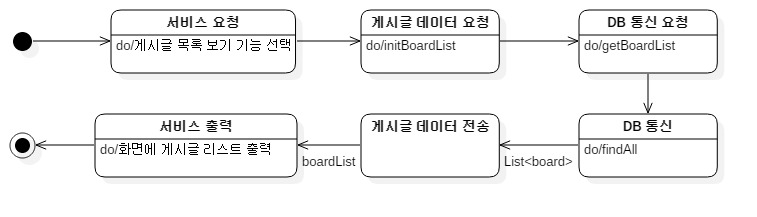


Diagram 6.6 State Diagram(게시글 보기)

## 환불 문의 작성하기

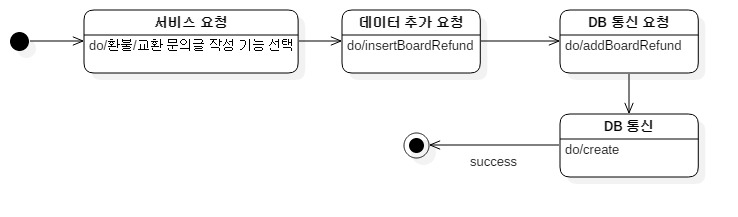


Diagram 6.7 State Diagram(환불 문의 작성하기)

## 환불 문의 수정하기



Diagram 6.8 State Diagram(환불 문의 수정하기)

## 환불 문의 삭제하기

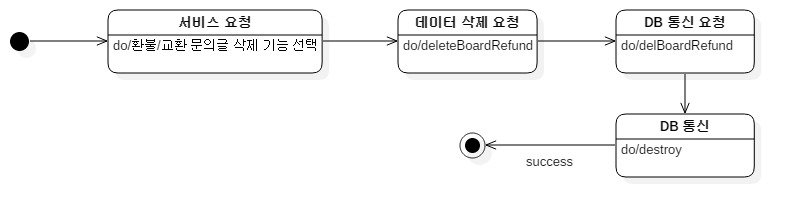


Diagram 6.9 State Diagram(환불 문의 삭제하기)

## **User Information SubSystem**

이 부분에서는 사용자가 웹 어플리케이션을 통해 회원가입 또는 탈퇴를 하거나 로그인/로그아웃을 하거나 ID/PW를 모를 때 찾거나 회원정보를 변경하는 역할을 하는 회원정보 서브시스템에 대해 서술한다.

## **Class Diagram**

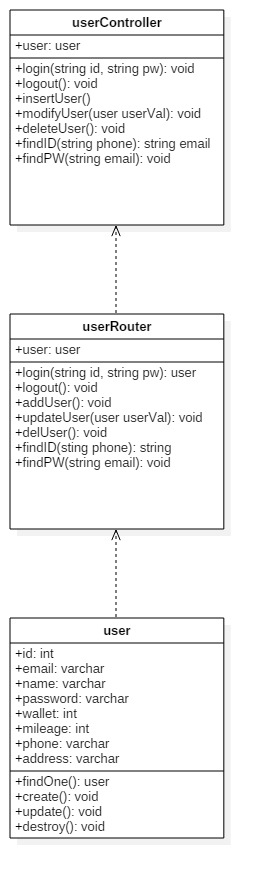


Diagram 7.1 User Information SubSystem Class Diagram

## userController

## Attributes

user : 회원정보 시스템에 이용되는 사용자에 대한 객체이다.

## Operations

login(string id, string pw) : 회원 로그인을 하기 위한 함수이다.

logout() : 회원 로그아웃을 하기 위한 함수이다.

insertUser() : 회원가입을 하기 위한 함수이다.

modifyUser(user userVal) : 회원정보 변경을 하기 위한 함수이다.

deleteUser() : 회원 탈퇴를 하기 위한 함수이다.

findId(string phone) : 사용자 아이디를 찾기 위한 함수이다.

findPW(string email) : 사용자 비밀번호를 찾기 위한 함수이다.

## userRouter

## Attributes

user : 데이터베이스의 user 테이블에서 한 개의 user 정보를 가져와 담기 위한 객체이다.

## Operations

login(string id, string pw) : userController의 login() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 입력 아이디, 비밀번호에 대한 user 정보가 있는지 조회 요청하기 위한 함수이다.

logout() : userController의 logout() 함수를 통해 호출되는 함수로 세션 정보를 제거하기위한 함수이다.

addUser() : userController의 insertUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에 한 개의 user 정보를 추가 요청하기 위한 함수이다.

updateUser(user userVal) : userController의 modifyUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 한 개의 user 정보를 수정 요청하기 위한 함수이다.

delUser() : userController의 deleteUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에 한 개의 user 정보를 삭제 요청하기 위한 함수이다.

findID(string phone) : userController의 findID() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 입력 전화번호에 대한 user의 email조회 요청하기 위한 함수이다.

findPW(string email) : userController의 findPW() 함수를 통해 호출되는 함수로 입력 email에 대한 user 객체에 임시 비밀번호를 발급하기 위한 함수이다.

## User

## Attributes

id : 사용자의 고유번호이다.

email : 사용자의 이메일 주소이다.

name : 사용자의 이름이다.

password : 사용자의 비밀번호이다.

wallet : 사용자의 가상화폐 액수이다.

mileage : 사용자의 마일리지 액수이다.

phone : 사용자의 휴대전화 번호이다.

address : 사용자의 집 주소이다.

## Operations

findOne() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 추가하는 통신을 하는 함수이다.

update() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 수정하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 삭제하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 회원가입

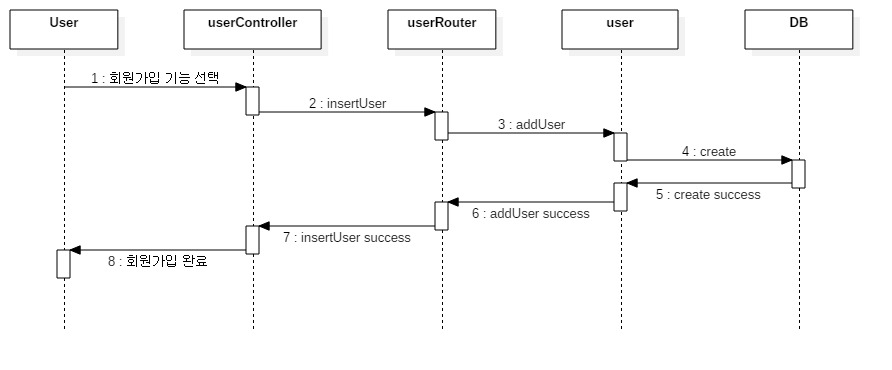


Diagram 7.2 Sequence Diagram(회원가입)

## 로그인

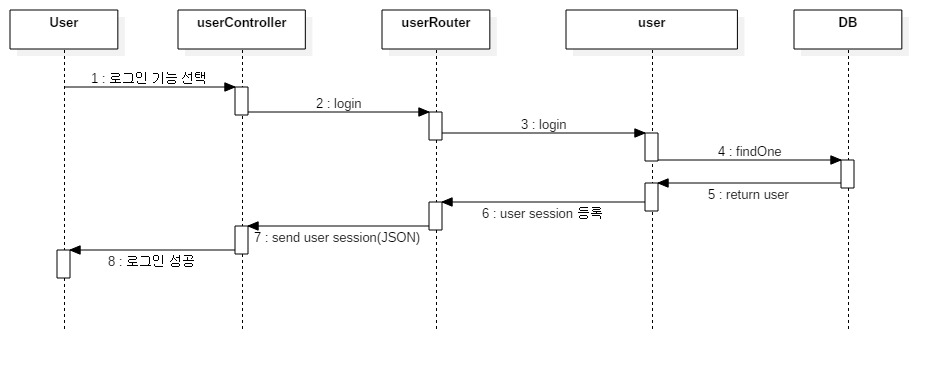


Diagram 7.3 Sequence DIagram(로그인)

## 로그아웃

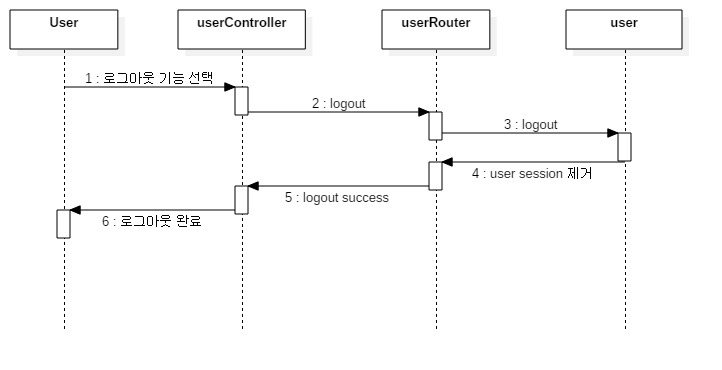


Diagram 7.4 Sequence Diagram(로그아웃)

## 회원정보 변경

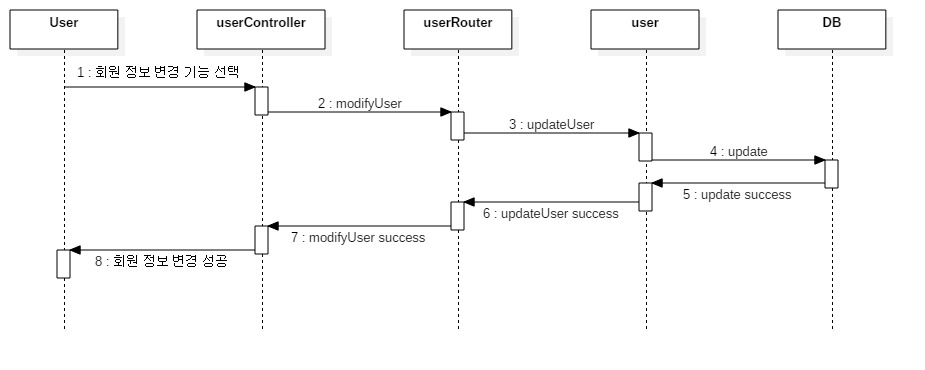


Diagram 7.5 Sequence Diagram(회원정보 변경)

## ID 찾기

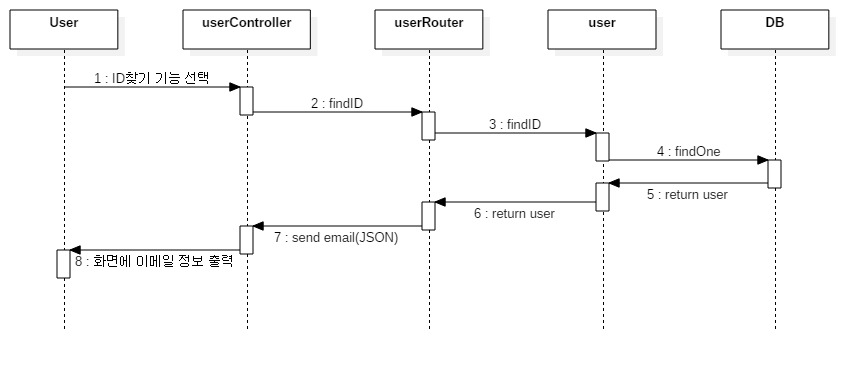


Diagram 7.6 Sequence Diagram(ID찾기)

## PW 찾기



Diagram 7.7 Sequence Diagram(PW찾기)

## 회원탈퇴

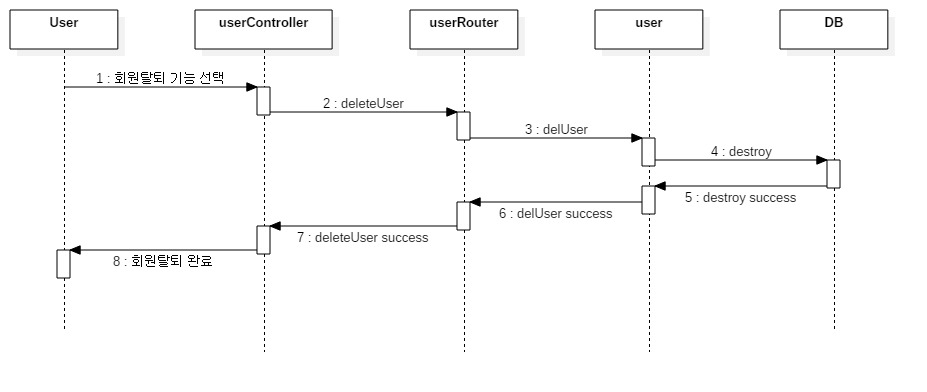


Diagram 7.8 Sequence Diagram(회원탈퇴)

## **State Diagram**

## 회원가입

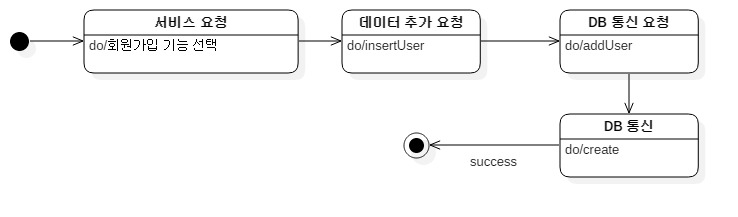


Diagram 7.9 State Diagram(회원가입)

## 로그인

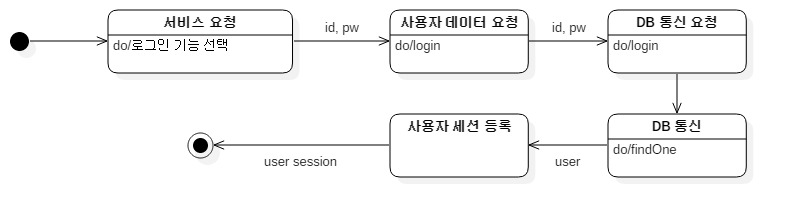


Diagram 7.10 State Diagram(로그인)

## 로그아웃

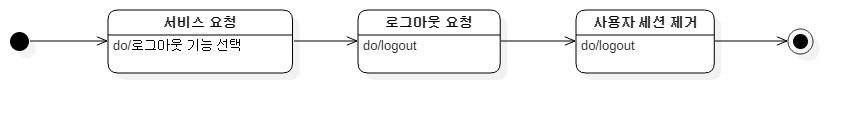


Diagram 7.11 State Diagram(로그아웃)

## 회원정보 변경

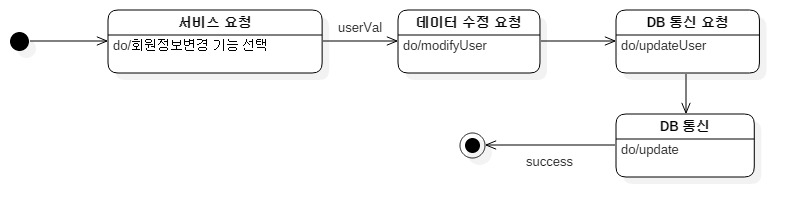


Diagram 7.12 State Diagram(회원정보 변경)

## ID 찾기

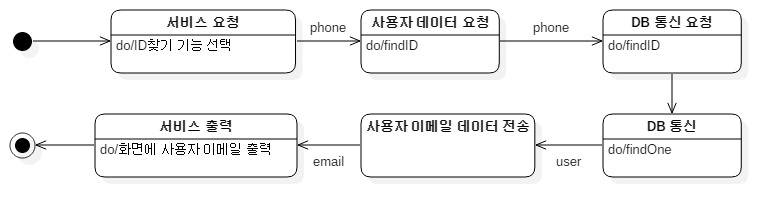


Diagram 7.13 State Diagram(ID찾기)

## PW 찾기

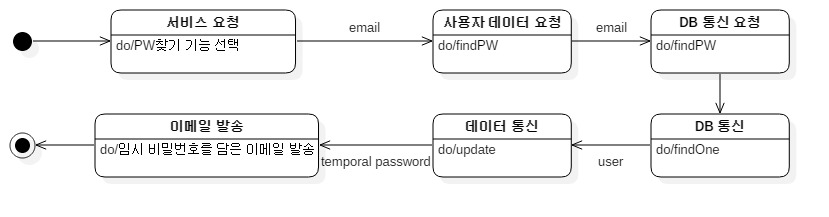


Diagram 7.14 State Diagram(PW 찾기)

## 회원탈퇴

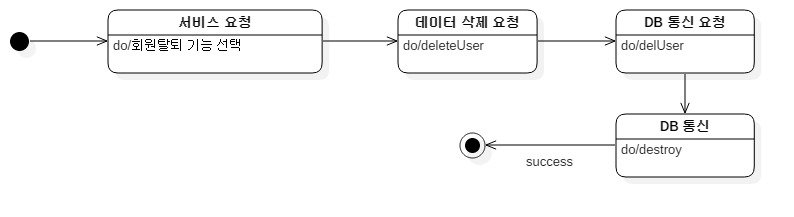
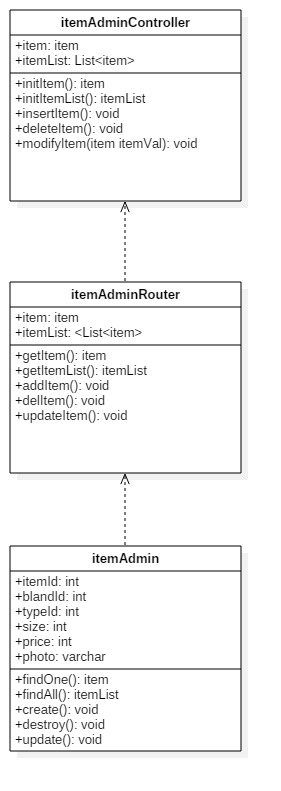


Diagram 7.15 State Diagram(회원탈퇴)

## **Item Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 상품을 조회, 등록, 수정, 삭제하는 역할을 하는 상품 관리 서브시스템에 대해 서술한다.

## **Class Diagram**



**Diagram 8.1 Item Admin SubSystem Class Diagram**

## itemAdminController

## Attributes

item : 상품 관리 시스템에 이용되는 한 개의 상품에 대한 객체이다.

itemList : 상품 관리 시스템에 이용되는 여러 개의 상품에 대한 리스트 객체이다.

## Operations

initItem() : (개별)상품 조회를 하기 위한 함수이다.

initItemList() : 상품 목록 조회를 하기 위한 함수이다.

insertItem() : 상품 추가를 하기 위한 함수이다.

deleteItem() : 상품 삭제를 하기 위한 함수이다.

modifyItem() : 상품 수정을 하기 위한 함수이다.

## itemAdminRouter

## Attributes

Item : 데이터베이스의 item 테이블에서 한 개의 item 정보를 가져와 담기 위한 객체이다.

itemList : 데이터베이스의 item 테이블에서 여러 개의 item 정보를 가져와 담기 위한 객체이다.

## Operations

getItem() : itemAdminController의 initItem() 함수를 통해 호출되는 함수로 데이터베이스의 item 테이블에서 한 개의 item 정보를 조회 요청하기 위한 함수이다.

getItemList() : itemAdminController의 initItemList() 함수를 통해 호출되는 함수로 데이터베이스의 item 테이블에서 여러 개의 item 정보를 조회 요청하기 위한 함수이다.

addItem() : itemAdminController의 insertItem() 함수를 통해 호출되는 함수로 데이터베이스의 item 테이블에서 한 개의 item 정보를 추가 요청하기 위한 함수이다.

updateItem() : itemAdminController의 modifyItem() 함수를 통해 호출되는 함수로 데이터베이스의 item 테이블에서 한 개의 item 정보를 수정 요청하기 위한 함수이다.

## itemAdmin

## Attributes

itemId : 상품의 고유번호이다.

blandId : 상품의 브랜드번호이다.

typeId : 상품의 항목번호이다.

size : 상품의 사이즈이다.

price : 상품의 가격이다.

photo : 상품의 이미지 경로이다.

## Operations

findOne() : 데이터베이스의 item 테이블에서 한 개의 item 정보를 조회하는 통신을 하는 함수이다.

findAll() : 데이터베이스의 item 테이블에서 여러 개의 item 정보를 조회하는 통신을 하는 함수이다.

create() : 데이터베이스의 item 테이블에서 한 개의 item 정보를 추가하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 item 테이블에서 한 개의 item 정보를 삭제하는 통신을 하는 함수이다.

update() : 데이터베이스의 item 테이블에서 한 개의 item 정보를 수정하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 상품 보기

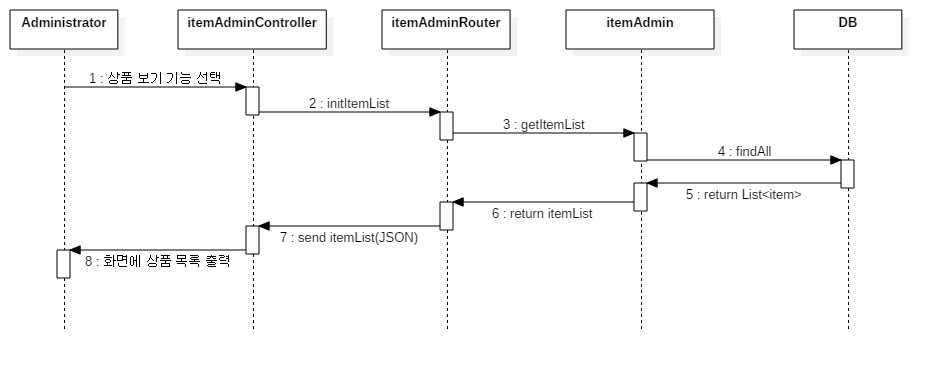


Diagram 8.2 Sequence Diagram(상품 보기)

## 상품 등록

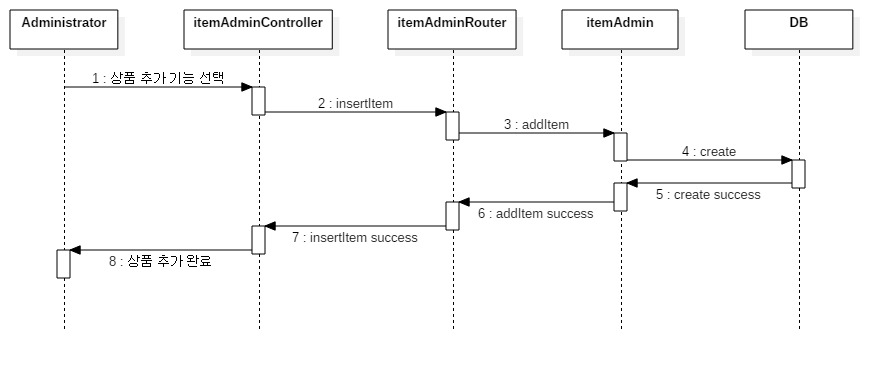


Diagram 8.3 Sequence Diagram(상품 등록)

## 상품 수정

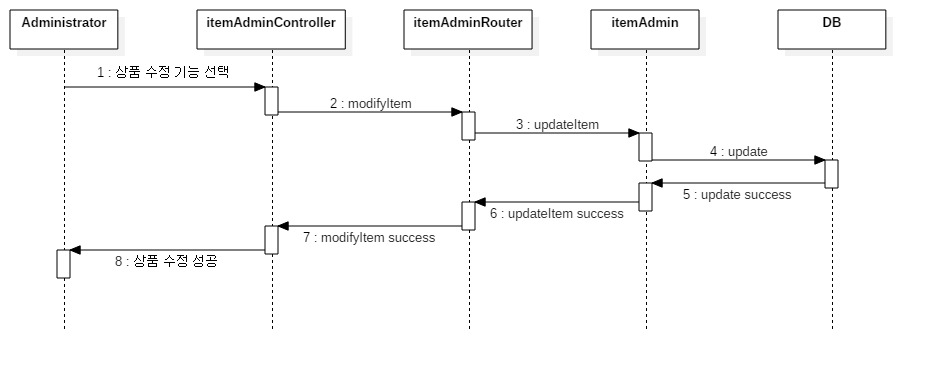


Diagram 8.4 Sequence Diagram(상품 수정)

## 상품 삭제



Diagram 8.5 Sequence Diagram(상품 삭제)

## **State Diagram**

## 상품 보기

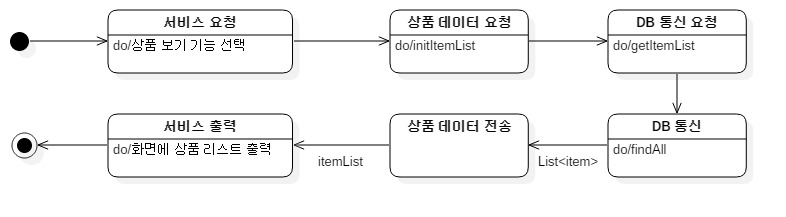


Diagram 8.6 State Diagram(상품 보기)

## 상품 등록

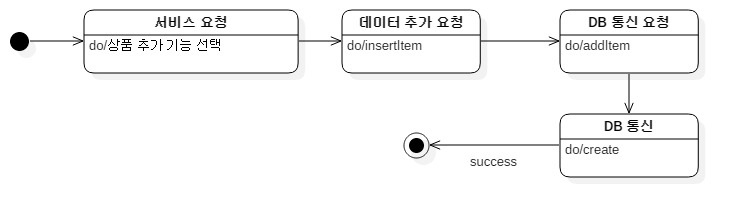


Diagram 8.7 State Diagram(상품 추가)

## 상품 수정

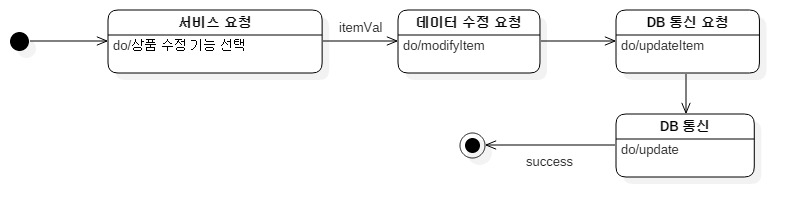


Diagram 8.8 State Diagram(상품 수정)

## 상품 삭제

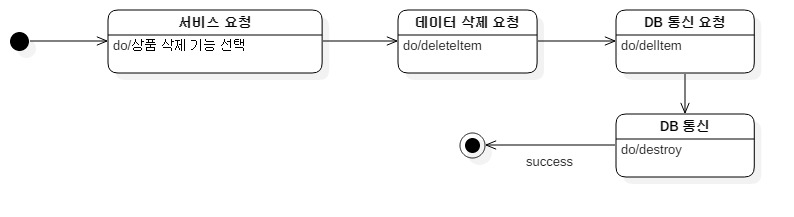


Diagram 8.9 State Diagram(상품 삭제)

## **Auction Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 경매 상품을 조회, 등록, 수정, 삭제하는 역할을 하는 경매 관리 서브시스템에 대해 서술한다.

## **Class Diagram**

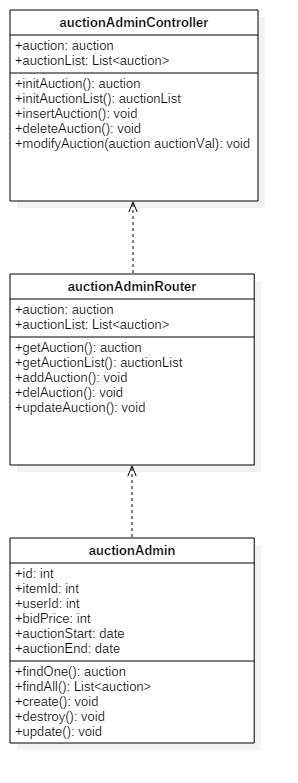


Diagram 9.1 Auction Admin SubSystem Class Diagram

## auctionAdminController

## Attributes

auction : 경매 관리 시스템에 이용되는 한 경매 상품에 대한 객체이다.

auctionList : 경매 관리 시스템에 이용되는 여러 경매 상품에 대한 리스트 객체이다.

## Operations

initAuction() : 경매 상품 조회 페이지 이동시 auction을 가져오기 위한 함수이다.

initAuctionList() : 경매 상품 목록 조회 페이지 이동시 auctionList를 가져오기 위한 함수이다,

insertAuction() : 경매 상품 추가를 하기 위한 함수이다.

deleteAuction() : 경매 상품 삭제를 하기 위한 함수이다.

modifyAuction() : 경매 상품 수정을 하기 위한 함수이다.

## auctionAdminRouter

## Attributes

auction : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 가져와 담기 위한 객체이다.

auctionList : 데이터베이스의 auction 테이블에서 여러 개의 auction 정보를 가져와 담기 위한 객체이다.

## Operations

getAuction() : auctionAdminController의 initAuction() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 조회 요청하기 위한 함수이다.

getAuctionList() : auctionAdminController의 initAuctionList() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 여러 개의 auction 정보를 조회 요청하기 위한 함수이다.

addAuction() : auctionAdminController의 insertAuction() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 추가 요청하기 위한 함수이다.

delAuction() : auctionAdminController의 deleteAuction() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 삭제 요청하기 위한 함수이다.

updateAuction() : auctionAdminController의 modifyAuction() 함수를 통해 호출되는 함수로 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 수정 요청하기 위한 함수이다.

## auctionAdmin

## Attributes

id : 경매의 고유번호이다.

itemId : 어떤 상품의 경매인지에 대한 식별값이다.

userId : 상회입찰자가 어떤 사용자인지에 대한 식별값이다.

bidPrice : 경매의 상회입찰금액이다.

auctionStart : 경매의 시작시간이다.

auctionEnd : 경매의 종료시간이다.

## Operations

findOne() : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 auction 테이블에서 여러 개의 auction 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 추가하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 삭제하는 통신을 하는 함수이다.

update() : 데이터베이스의 auction 테이블에서 한 개의 auction 정보를 수정하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 경매 보기

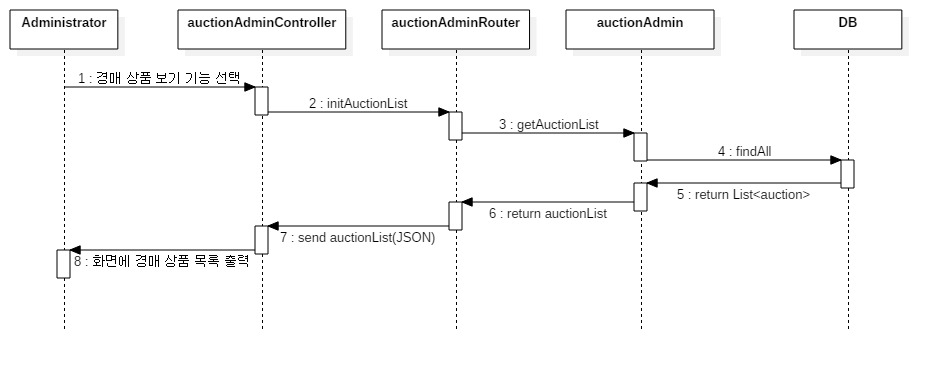


Diagram 9.2 Sequence Diagram(경매 보기)

## 경매 등록

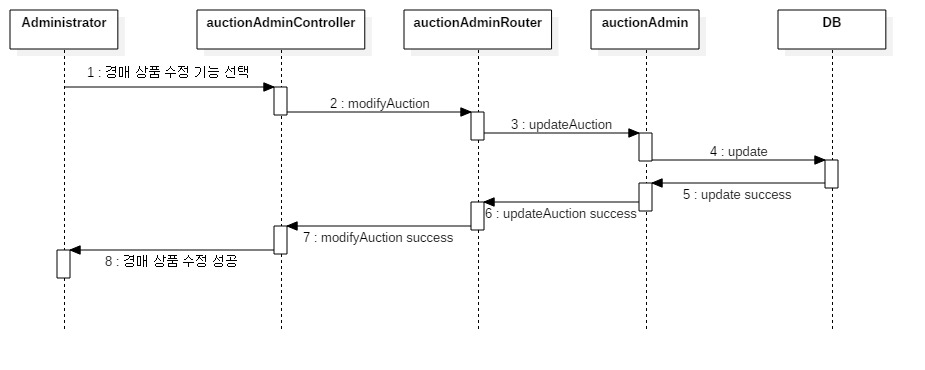


Diagram 9.3 Sequence Diagram(경매 등록)

## 경매 수정

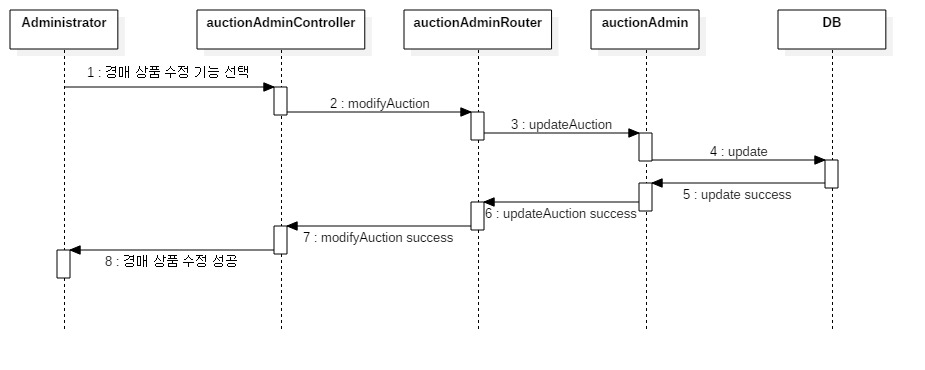


Diagram 9.4 Sequence Diagram(경매 수정)

## 경매 삭제

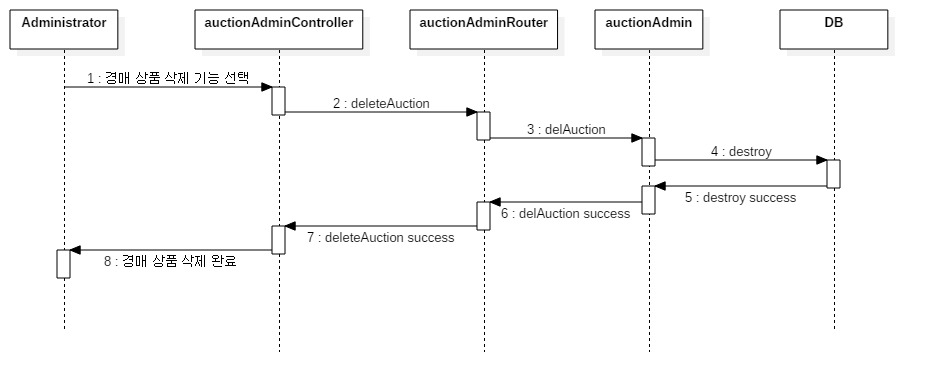


Diagram 9.5 Sequence Diagram(경매 삭제)

## **State Diagram**

## 경매 보기

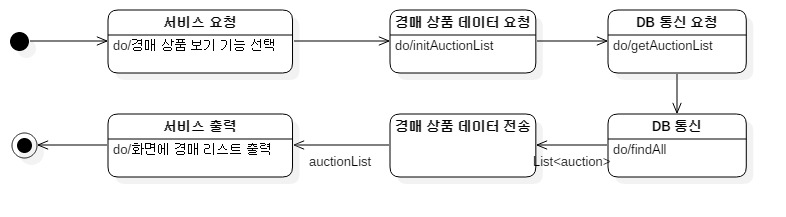


Diagram 9.6 State DIagram(경매 보기)

## 경매 등록

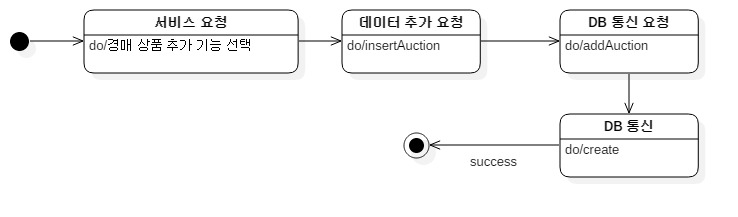


Diagram 9.7 State DIagram(경매 등록)

## 경매 수정

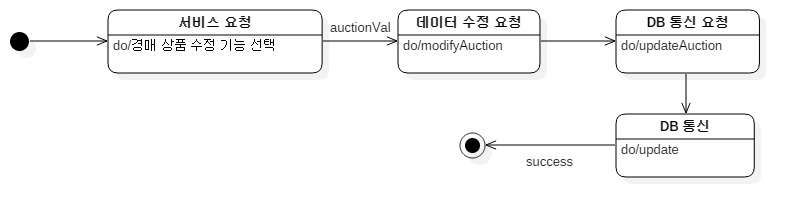


Diagram 9.8 State DIagram(경매 수정)

## 경매 삭제



Diagram 9.9 State DIagram(경매 삭제)

## **ItemType/Bland Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 상품의 분류, 브랜드의 조회, 등록, 수정, 삭제하는 역할을 하는 상품 분류/브랜드 관리 서브시스템에 대해 서술한다.

## **Class Diagram**

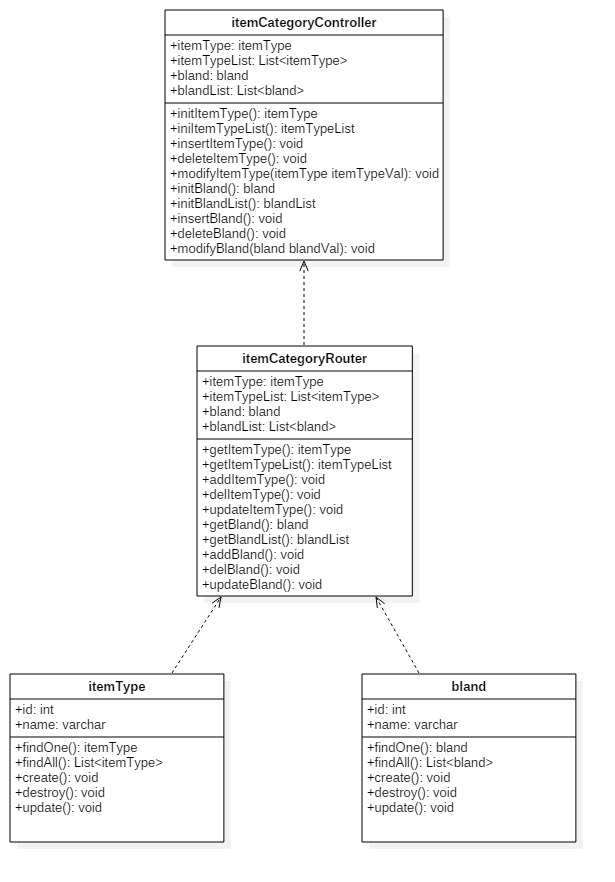


Diagram 10.1 itemType/Bland Admin SubSystem Class Diagram

## itemCategoryController

## Attributes

itemType : 상품 분류 브랜드 관리 시스템에 이용되는 한 분류에 대한 객체이다.

itemTypeList : 상품 분류 브랜드 관리 시스템에 이용되는 여러 분류에 대한 객체이다.

bland : 상품 분류 브랜드 관리 시스템에 이용되는 한 브랜드에 대한 객체이다.

blandList : 상품 분류 브랜드 관리 시스템에 이용되는 여러 브랜드에 대한 객체이다.

## Operations

initItemType() : 상품 분류 조회 페이지 이동시 itemType을 가져오기 위한 함수이다.

initItemTypeList() : 상품 분류 목록 페이지 이동시 itemTypeList를 가져오기 위한 함수이다.

insertItemType() : 상품 분류 추가를 하기 위한 함수이다.

deleteItemType() : 상품 분류 삭제를 하기 위한 함수이다.

modifyItemType(itemType itemTypeVal) : 상품 분류 수정을 하기 위한 함수이다.

initBland() : 상품 브랜드 조회 페이지 이동시 bland을 가져오기 위한 함수이다.

initBlandList() : 상품 브랜드 목록 페이지 이동시 blandList를 가져오기 위한 함수이다.

insertBland() : 상품 브랜드 추가를 하기 위한 함수이다.

deleteBland() : 상품 브랜드 삭제를 하기 위한 함수이다.

modifyBland(bland blandVal) : 상품 브랜드 수정을 하기 위한 함수이다.

## itemCategoryRouter

## Attributes

itemType : 데이터베이스의 item\_type 테이블에서 한 개의 itemType 정보를 가져와 담기 위한 객체이다.

itemTypeList : 데이터베이스의 item\_type 테이블에서 여러 개의 itemType 정보를 가져와 담기 위한 객체이다.

bland : 데이터베이스의 bland 테이블에서 한 개의 bland 정보를 가져와 담기 위한 객체이다.

blandList : 데이터베이스의 bland 테이블에서 여러 개의 bland 정보를 가져와 담기 위한 객체이다.

## Operations

getItemType() : itemCategoryController의 initItemType() 함수를 통해 호출되는 함수로 데이터베이스의 item\_type테이블에서 한 개의 itemType 정보를 조회 요청하기 위한 함수이다.

getItemTypeList() : itemCategoryController의 initItemTypeList() 함수를 통해 호출되는 함수로 데이터베이스의 item\_type테이블에서 여러 개의 itemType 정보를 조회 요청하기 위한 함수이다.

addItemType() : itemCategoryController의 insertItemType() 함수를 통해 호출되는 함수로 데이터베이스의 item\_type테이블에서 한 개의 itemType 정보를 추가 요청하기 위한 함수이다.

delItemType() : itemCategoryController의 deleteItemType() 함수를 통해 호출되는 함수로 데이터베이스의 item\_type테이블에서 한 개의 itemType 정보를 삭제 요청하기 위한 함수이다.

updateItemType() : itemCategoryController의 modifyItemType() 함수를 통해 호출되는 함수로 데이터베이스의 item\_type테이블에서 한 개의 itemType 정보를 수정 요청하기 위한 함수이다.

getBland() : itemCategoryController의 initBland() 함수를 통해 호출되는 함수로 데이터베이스의 bland테이블에서 한 개의 bland 정보를 조회 요청하기 위한 함수이다.

getBlandList() : itemCategoryController의 initBlandList() 함수를 통해 호출되는 함수로 데이터베이스의 bland테이블에서 여러 개의 bland 정보를 조회 요청하기 위한 함수이다.

addBland() : itemCategoryController의 insertBland() 함수를 통해 호출되는 함수로 데이터베이스의 bland테이블에서 한 개의 bland 정보를 추가 요청하기 위한 함수이다.

delBland() : itemCategoryController의 deleteBland() 함수를 통해 호출되는 함수로 데이터베이스의 bland테이블에서 한 개의 bland 정보를 삭제 요청하기 위한 함수이다.

updateBland() : itemCategoryController의 modifyBland() 함수를 통해 호출되는 함수로 데이터베이스의 bland테이블에서 한 개의 bland 정보를 수정 요청하기 위한 함수이다.

## itemType

## Attributes

id : 상품 분류의 고유번호이다.

name : 상품 분류의 이름이다.

## Operations

findOne() : 데이터베이스의 item\_type 테이블에서 한 개의 itemType 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 item\_type 테이블에서 여러 개의 itemType 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 item\_type 테이블에서 한 개의 itemType 정보를 추가하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 item\_type 테이블에서 한 개의 itemType 정보를 삭제하는 통신을 하는 함수이다.

update() : 데이터베이스의 item\_type 테이블에서 한 개의 itemType 정보를 수정하는 통신을 하는 함수이다.

## Bland

## Attributes

id : 상품 브랜드의 고유번호이다.

name : 상품 브랜드의 이름이다.

## Operations

findOne() : 데이터베이스의 bland 테이블에서 한 개의 bland 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 bland 테이블에서 여러 개의 bland 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 bland 테이블에서 한 개의 bland 정보를 추가하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 bland 테이블에서 한 개의 bland 정보를 삭제하는 통신을 하는 함수이다.

update() : 데이터베이스의 bland 테이블에서 한 개의 bland 정보를 수정하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 분류/브랜드 보기

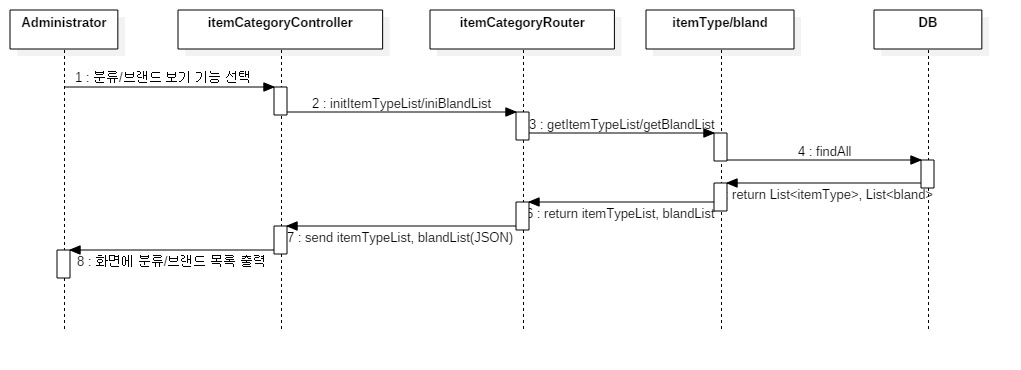


Diagram 10.2 Sequence Diagram (분류/브랜드 보기)

## 분류/브랜드 추가

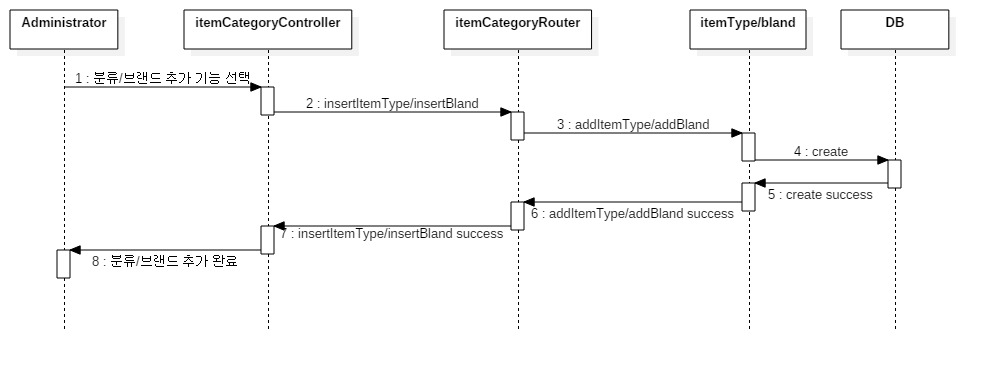


Diagram 10.3 Sequence Diagram(분류/브랜드 추가)

## 분류/브랜드 수정

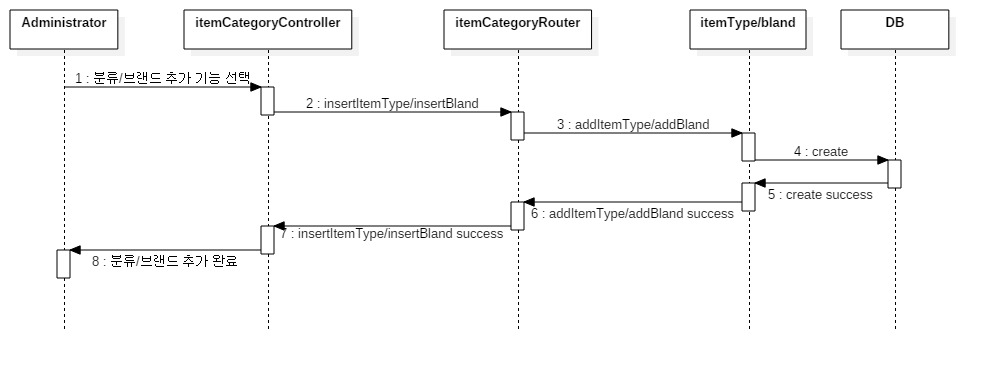


Diagram 10.4 Sequence Diagram(분류/브랜드 수정)

## 분류/브랜드 삭제

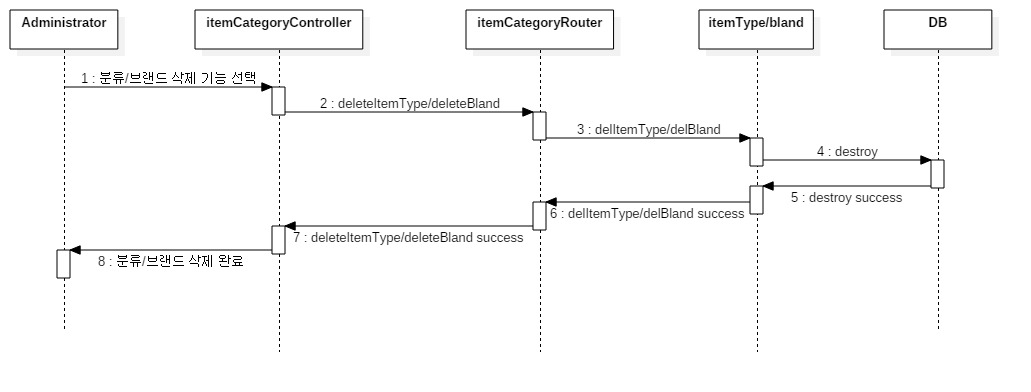


Diagram 10.5 Sequence Diagram(분류/브랜드 삭제)

## **State Diagram**

## 분류/브랜드 보기



Diagram 10.6 State Diagram(분류/브랜드 보기)

## 분류/브랜드 추가

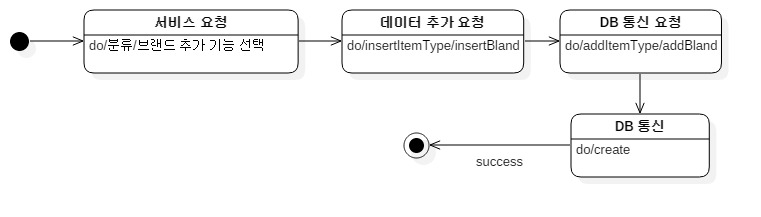


Diagram 10.7 State Diagram(분류/브랜드 추가)

## 분류/브랜드 수정

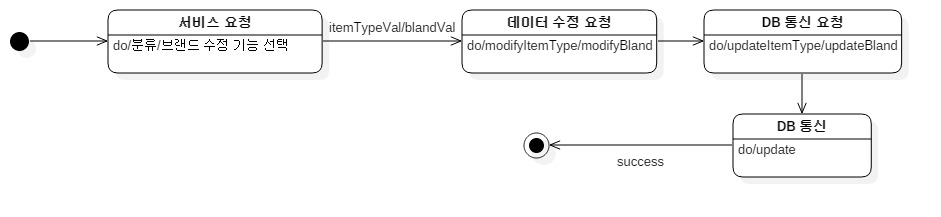


Diagram 10.8 State Diagram(분류/브랜드 수정)

## 분류/브랜드 삭제

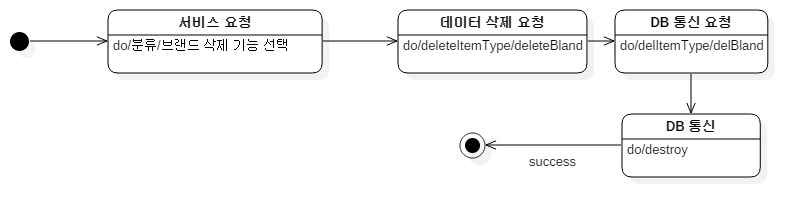


Diagram 10.9 State Diagram(분류/브랜드 삭제)

## **User Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 회원을 조회, 수정, 삭제하는 역할을 하는 회원 관리 서브시스템에 대해 서술한다. 회원가입 기능이 있기 때문에 회원 추가 기능은 없다.

## **Class Diagram**

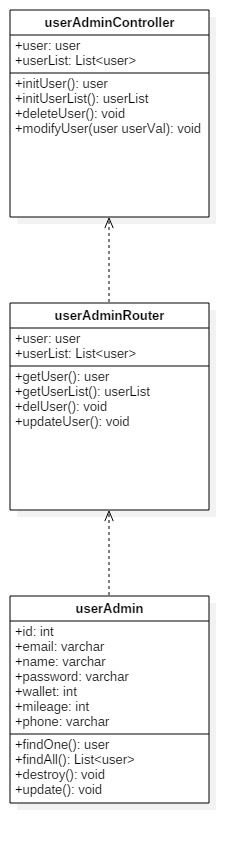


Diagram 11.1 user Admin SubSystem Class Diagram

## userAdminController

## Attributes

user : 회원 관리 시스템에 이용되는 한 회원에 대한 객체이다.

userList : 회원 관리 시스템에 이용되는 여러 회원에 대한 객체이다.

## Operations

initUser() : 회원 정보 조회 페이지 이동시 user를 가져오기 위한 함수이다.

initUserList() : 회원 목록 조회 페이지 이동시 userList를 가져오기 위한 함수이다.

deleteUser() : 회원 삭제를 하기 위한 함수이다.

modifyUser(user userVal) : 회원 정보 수정을 하기 위한 함수이다.

## userAdminRouter

## Attributes

user : 데이터베이스의 user 테이블에서 한 개의 user 정보를 가져와 담기 위한 객체이다.

userList : 데이터베이스의 user 테이블에서 여러 개의 user 정보를 가져와 담기 위한 객체이다.

## Operations

getUser() : userAdminController의 initUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 한 개의 user 정보를 조회 요청하기 위한 함수이다.

getUserList() : userAdminController의 initUserList() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 여러 개의 user 정보를 조회 요청하기 위한 함수이다.

delUser() : userAdminController의 deleteUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 한 개의 user 정보를 삭제 요청하기 위한 함수이다.

updateUser() : userAdminController의 modifyUser() 함수를 통해 호출되는 함수로 데이터베이스의 user 테이블에서 한 개의 user 정보를 수정 요청하기 위한 함수이다.

## userAdmin

## Attributes

id : 사용자의 고유번호이다.

email : 사용자의 이메일 주소이다.

name : 사용자의 이름이다.

password : 사용자의 비밀번호이다.

wallet : 사용자의 가상화폐 액수이다.

mileage : 사용자의 마일리지 액수이다.

phone : 사용자의 휴대전화 번호이다.

address : 사용자의 집 주소이다.

## Operations

findOne() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 user 테이블에서 여러 개의 user 정보를 가져오는 통신을 하는 함수이다.

destroy() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 삭제하는 통신을 하는 함수이다.

update() : 데이터베이스의 user 테이블에서 한 개의 user 정보를 수정하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 회원 목록 보기



Diagram 11.2 Sequence Diagram(회원 목록 보기)

## 회원 정보 수정

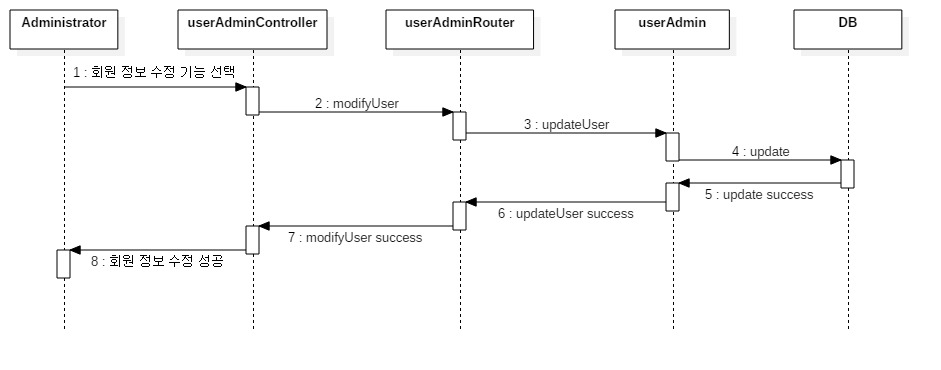


Diagram 11.3 Sequence Diagram(회원 정보 수정)

## 회원 삭제

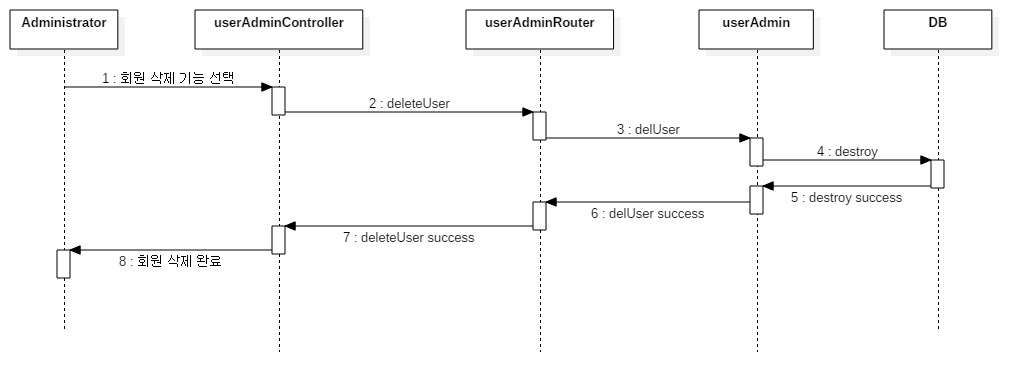


Diagram 11.4 Sequence Diagram(회원 삭제)

## **State Diagram**

## 회원 목록 보기

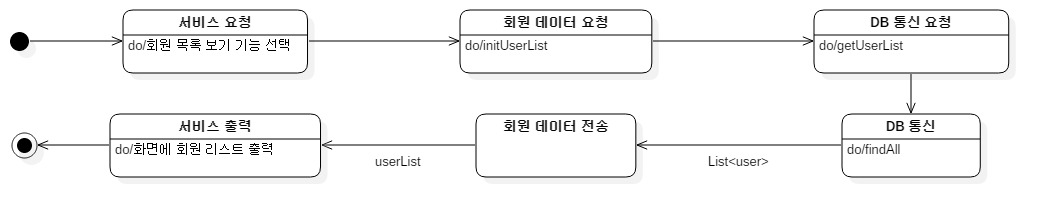


Diagram 11.5 State Diagram(회원 목록 보기)

## 회원 정보 수정

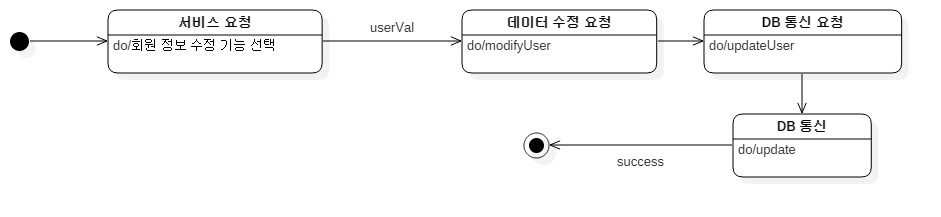


Diagram 11.6 State Diagram (회원 정보 수정)

## 회원 삭제

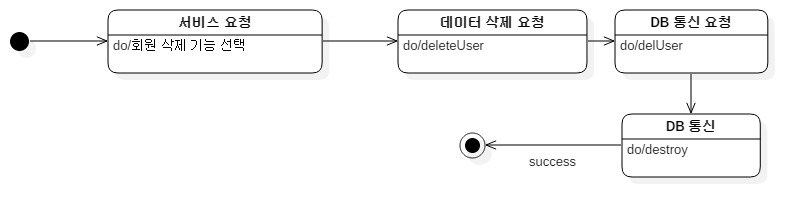


Diagram 11.7 State Diagram(회원 삭제)

## **Order Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 회원들을 주문한 정보에 대해 조회, 통계화 하는 역할을 하는 주문 관리 서브시스템에 대해 서술한다.

## **Class Diagram**

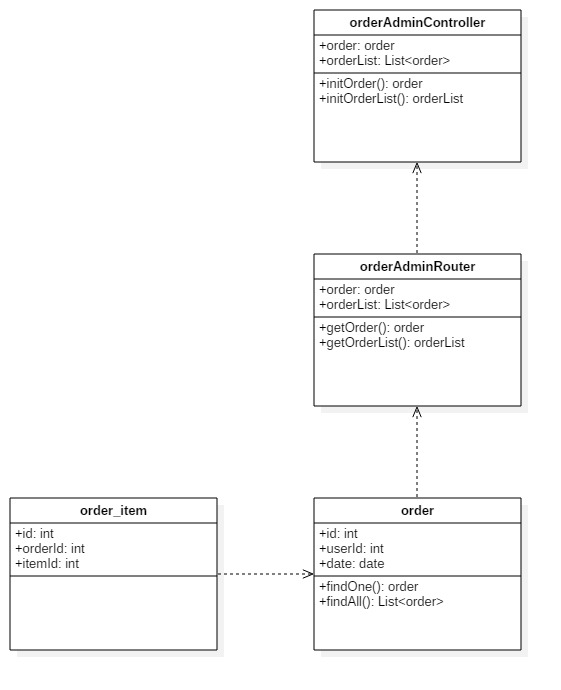


Diagram 12.1 Order Admin SubSystem Class Diagram

## orderAdminController

## Attributes

order : 주문 관리 시스템에 이용되는 한 주문에 대한 객체이다.

orderList : 주문 관리 시스템에 이용되는 여러 주문에 대한 객체이다.

## Operations

initOrder() : 주문 조회 페이지 이동시 order를 가져오기 위한 함수이다.

initOrderList() : 주문 목록 조회 페이지/판매 내역 통계 페이지 이동시 orderList를 가져오기 위한 함수이다.

## orderAdminRouter

## Attributes

order : 데이터베이스의 order/order\_item 테이블에서 한개의 주문에 대한 order\_item 정보를 가져와 담기 위한 객체이다.

orderList : 데이터베이스의 order/order\_item 테이블에서 여러개의 주문에 대한 order\_item 정보를 가져와 담기 위한 객체이다.

## Operations

getOrder() : orderAdminController의 initOrder() 함수를 통해 호출되는 함수로 데이터베이스의 order/order\_item 테이블에서 한 개의 주문에 대한 order\_item 정보를 조회 요청하기 위한 함수이다.

getOrderList() : orderAdminController의 initOrderList() 함수를 통해 호출되는 함수로 데이터베이스의 order/order\_item 테이블에서 여러 개의 주문에 대한 order\_item 정보를 조회 요청하기 위한 함수이다.

## order

## Attributes

id : 주문의 고유번호이다.

userId : 어떤 사용자의 주문인지에 대한 식별값이다.

date : 주문 시간이다.

## Operations

findOne() : 데이터베이스의 order 테이블에서 한 개의 order 정보를 가져오는 통신을 하는 함수이다.[

findAll() : 데이터베이스의 order 테이블에서 여러 개의 order 정보를 가져오는 통신을 하는 함수이다.

## order\_item

## Attributes

Id : order\_item 테이블의 한 row에 대한 고유값이다.

orderId : 어떤 주문에 대한 상품인지에 대한 식별값이다.

itemId : 주문에 포함되어 있는 상품에 대한 식별값이다.

## Operations

없음

## **Sequence Diagram**

## 판매 내역 보기

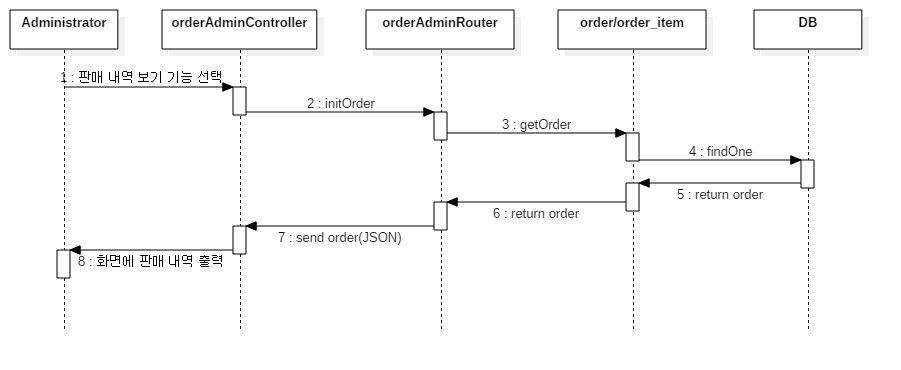


Diagram 12.2 Sequence Diagram (판매 내역 보기)

## 판매 내역 통계

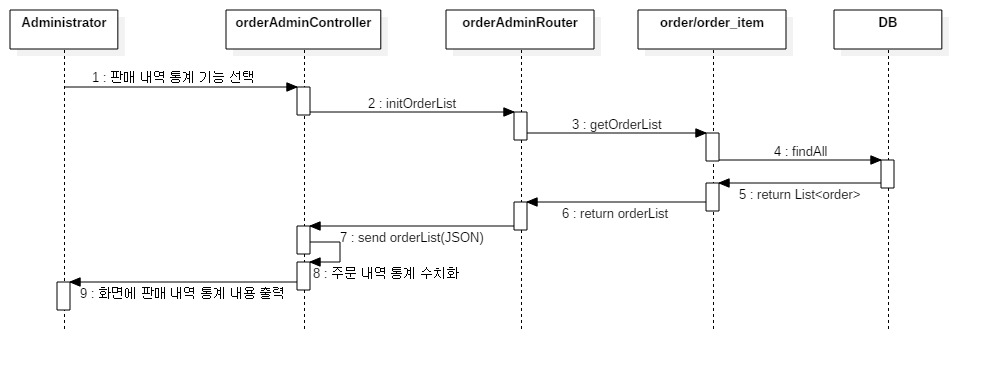


Diagram 12.3 Sequence Diagram (판매 내역 통계)

## **State Diagram**

## 판매 내역 보기

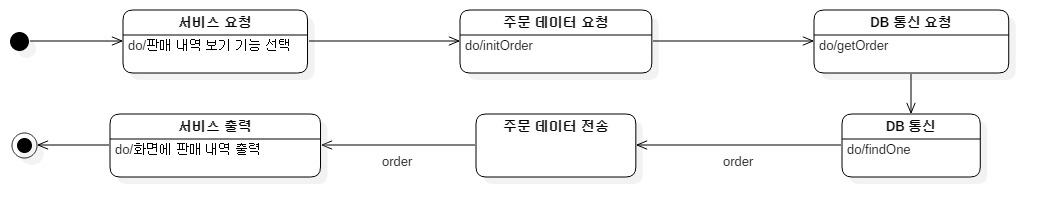


Diagram 12.4 State Diagram (판매 내역 보기)

## 판매 내역 통계

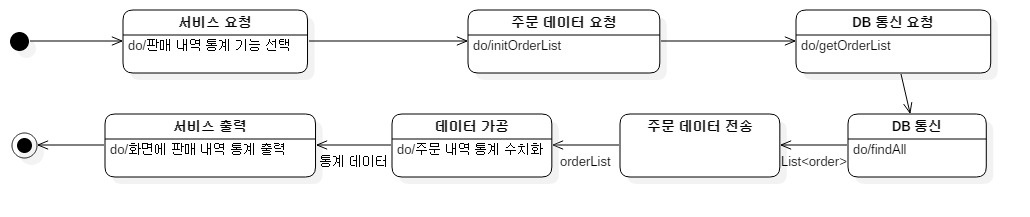


Diagram 12.5 State Diagram (판매 내역 통계)

## **Customer Center Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 게시글(공지사항, FAQ, 환불 문의 답변)을 조회, 쓰기, 수정, 삭제를 하는 역할을 하는 게시판 관리 서브시스템에 대해 서술한다.

## **Class Diagram**

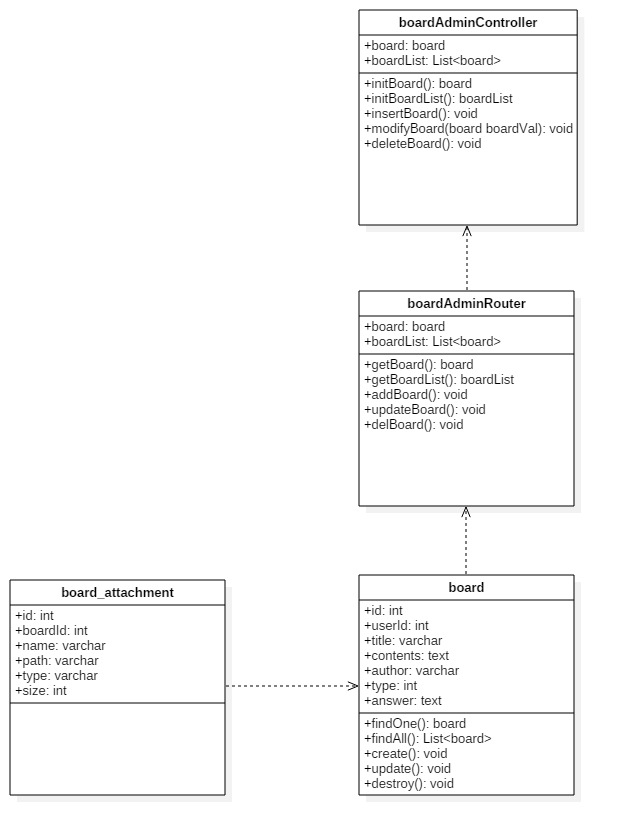
****

Diagram 13.1 Customer Center Admin SubSystem Class Diagram

## boardAdminController

## Attributes

board : 게시판 관리 시스템에 이용되는 한 게시글에 대한 객체이다.

boardList : 게시판 관리 시스템에 이용되는 여러 게시글에 대한 객체이다.

## Operations

initBoard() : 게시글 조회 페이지 이동시 board 를 가져오기 위한 함수이다.

initBoardList() : 게시글 목록 페이지 이동시 board 를 가져오기 위한 함수이다.

insertBoard() : 게시글을 추가하기 위한 함수이다.

modifyBoard() : 게시글을 수정하기 위한 함수이다.

deleteBoard() : 게시글을 삭제하기 위한 함수이다.

## boardAdminRouter

## Attributes

board : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 게시글에 대한 board 정보를 가져와 담기 위한 객체이다.

boardList : 데이터베이스의 board/board\_attachment 테이블에서 여러 개의 게시글에 대한 board 정보를 가져와 담기 위한 객체이다.

## Operations

getBoard() : boardAdminController의 initBoard() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 한 개의 게시글에 대한 board 정보를 조회 요청하기 위한 함수이다.

getBoardList() : boardAdminController의 initBoardList() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 여러 개의 게시글에 대한 board 정보를 조회 요청하기 위한 함수이다.

addBoard() : boardAdminController의 insertBoard() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 한 개의 게시글에 대한 board 정보를 추가 요청하기 위한 함수이다.

updateBoard() : boardAdminController의 modifyBoard() 함수를 통해 호출되는 함수로 데이터베이스의 board/board\_attachment 테이블에서 한 개의 게시글에 대한 board 정보를 수정 요청하기 위한 함수이다.

delBoard() : boardAdminController의 deleteBoard() 함수를 통해 호출되는 함수로 데이터베이스의 board 테이블에서 한 개의 게시글에 대한 board 정보를 조회 요청하기 위한 함수이다.

## board

## Attributes

id : 게시글의 고유번호이다.

userId : 어떤 사용자의 게시글 인지에 대한 식별값이다.

title : 게시글의 제목이다.

contents : 게시글의 내용이다.

author : 게제자의 이름이다.

type : 게시글의 카테고리 타입니다.(공지사항, FAQ, 환불/교환 게시판)

answer : (환불/교환 게시판에서만 사용)관리자의 답변 내용이다.

## Operations

findOne() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 board/board\_attachment 테이블에서 여러 개의 board 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 추가오는 통신을 하는 함수이다.

update() : 데이터베이스의 board/board\_attachment 테이블에서 한 개의 board 정보를 수정하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 board 테이블에서 한 개의 board 정보를 삭제하는 통신을 하는 함수이다.

## board\_attachment

## Attributes

Id : 게시글 첨부파일의 고유번호이다.

boardId : 어떤 게시글의 첨부파일인지에 대한 식별값이다.

name : 첨부파일의 이름이다.

path : 첨부파일의 파일경로이다.

type : 첨부파일의 파일종류이다.

size : 첨부파일의 파일크기이다.

## Operations

없음

## **Sequence Diagram**

## 게시글 보기

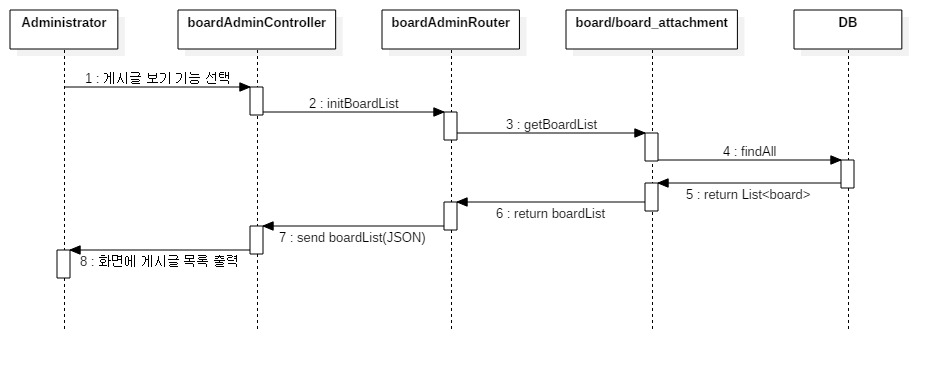


Diagram 13.2 Sequence Diagram (게시글 보기)

## 게시글 등록

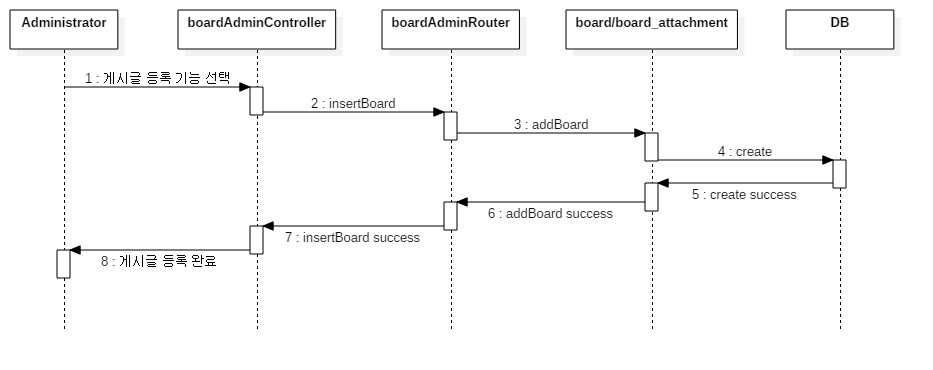


Diagram 13.3 Sequence Diagram(게시글 등록)

## 게시글 수정



Diagram 13.4 Sequence Diagram(게시글 수정)

## 게시글 삭제

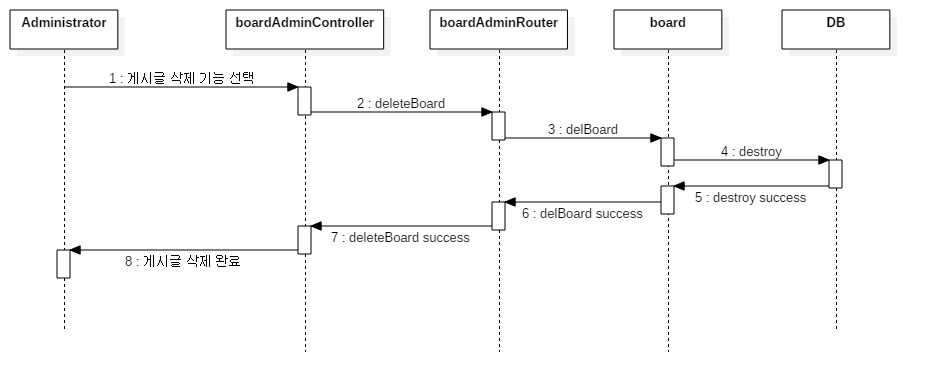


Diagram 13.5 Sequence Diagram(게시글 삭제)

## 환불/교환 문의 답변하기

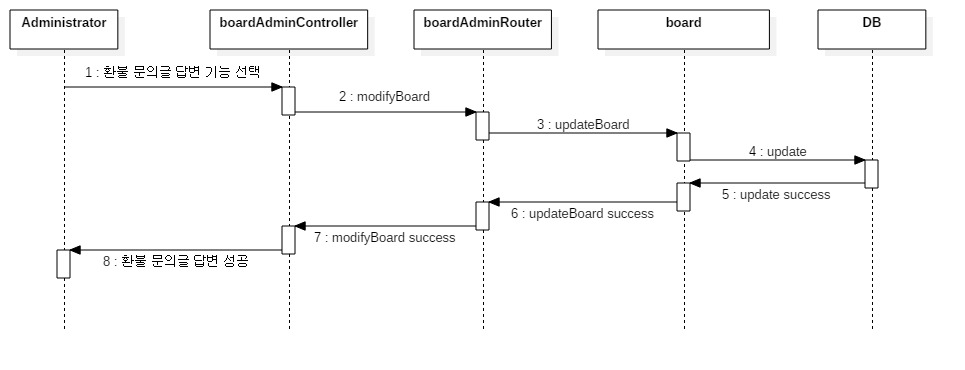


Diagram 13.6 Sequence Diagram(환불/교환 문의 답변하기)

## **State Diagram**

## 게시글 보기

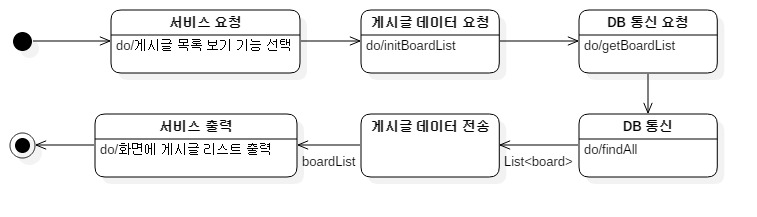


Diagram 13.7 State Diagram(게시글 보기)

## 게시글 등록

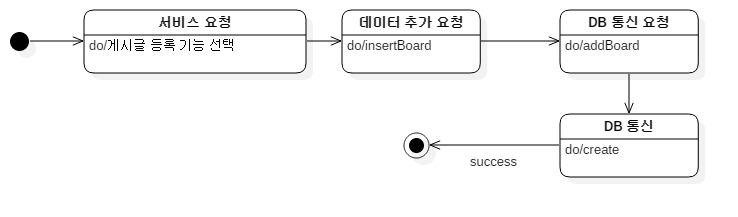


Diagram 13.8 State Diagram(게시글 등록)

## 게시글 수정

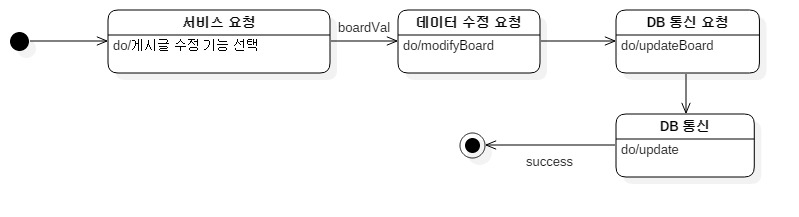


Diagram 13.9 State Diagram(게시글 수정)

## 게시글 삭제

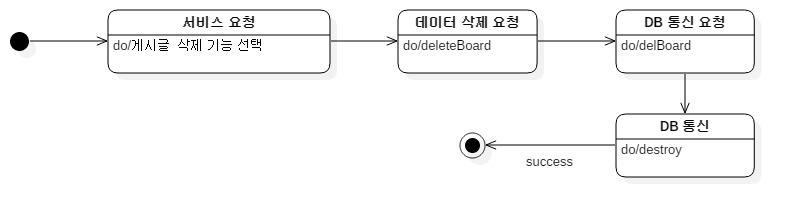


Diagram 13.10 State Diagram(게시글 삭제)

## 환불/교환 문의 답변하기

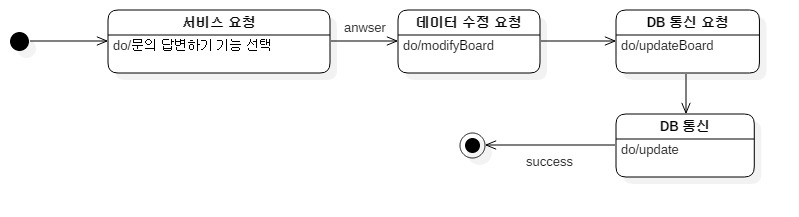


Diagram 13.11 State Diagram(환불/교환 문의 답변하기)

## **Popup Admin SubSystem**

이 부분에서는 관리자가 웹 어플리케이션을 통해 팝업을 조회, 쓰기, 수정, 삭제를 하는 역할을 하는 팝업 관리 서브시스템에 대해 서술한다.

## **Class Diagram**

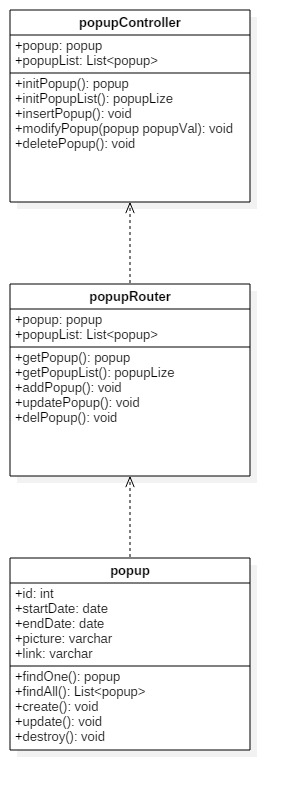


Diagram 14.1 Popup Admin SubSystem Class Diagram

## popupController

## Attributes

popup : 팝업 관리 시스템에 이용되는 한 팝업에 대한 객체이다.

popupList : 팝업 관리 시스템에 이용되는 여러 팝업에 대한 객체이다.

## Operations

initPopup() : 팝업 조회 페이지 이동시 popup을 가져오기 위한 함수이다.

initPopupList() : 팝업 목록 조회 페이지 이동시 popupList을 가져오기 위한 함수이다.

insertPopup() : 팝업 추가를 하기 위한 함수이다.

modifyPopup(popup popupVal) : 팝업 수정을 하기 위한 함수이다.

deletePopup() : 팝업 삭제를 하기 위한 함수이다.

## popupRouter

## Attributes

popup : 데이터베이스의 popup 테이블에서 한 개의 팝업에 대한 popup 정보를 가져와 담기 위한 객체이다.

popupList : 데이터베이스의 popup 테이블에서 여러 개의 팝업에 대한 popup 정보를 가져와 담기 위한 객체이다.

## Operations

getPopup() : popupController의 initPopup() 함수를 통해 호출되는 함수로 데이터베이스의 popup 테이블에서 한 개의 팝업에 대한 popup정보를 조회 요청하기 위한 함수이다.

getPopupList() : popupController의 initPopup() 함수를 통해 호출되는 함수로 데이터베이스의 popup 테이블에서 여러 개의 팝업에 대한 popup정보를 조회 요청하기 위한 함수이다.

## popup

## Attributes

id : 팝업의 고유번호이다.

startDate : 팝업의 게시 시작 날짜이다.

endDate : 팝업의 게시 종료 날짜이다.

picture : 팝업 이미지파일 경로이다.

link : 팝업 클릭시 이동할 url이다.

## Operations

findOne() : 데이터베이스의 popup 테이블에서 한 개의 popup 정보를 가져오는 통신을 하는 함수이다.

findAll() : 데이터베이스의 popup 테이블에서 여러 개의 popup 정보를 가져오는 통신을 하는 함수이다.

create() : 데이터베이스의 popup 테이블에서 한 개의 popup 정보를 추가하는 통신을 하는 함수이다.

update() : 데이터베이스의 popup 테이블에서 한 개의 popup 정보를 수정하는 통신을 하는 함수이다.

destroy() : 데이터베이스의 popup 테이블에서 한 개의 popup 정보를 삭제하는 통신을 하는 함수이다.

## **Sequence Diagram**

## 팝업 보기

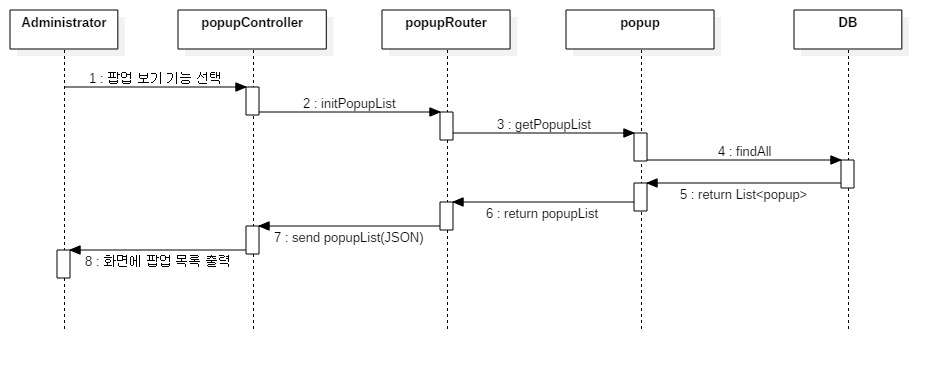


Diagram 14.2 Sequence Diagram (팝업 보기)

## 팝업 추가

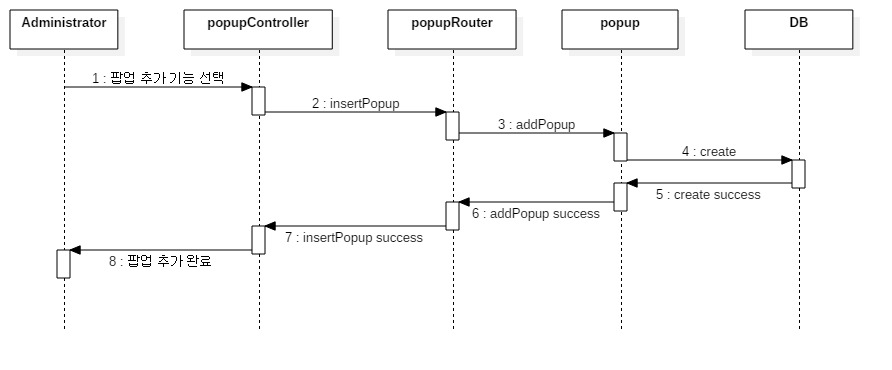


Diagram 14.3 Sequence Diagram(팝업 추가)

## 팝업 수정

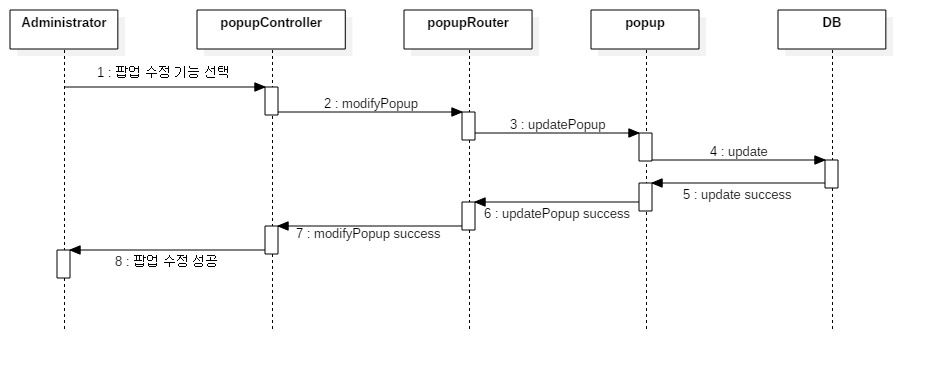


Diagram 14.4 Sequence Diagram(팝업 수정)

## 팝업 삭제

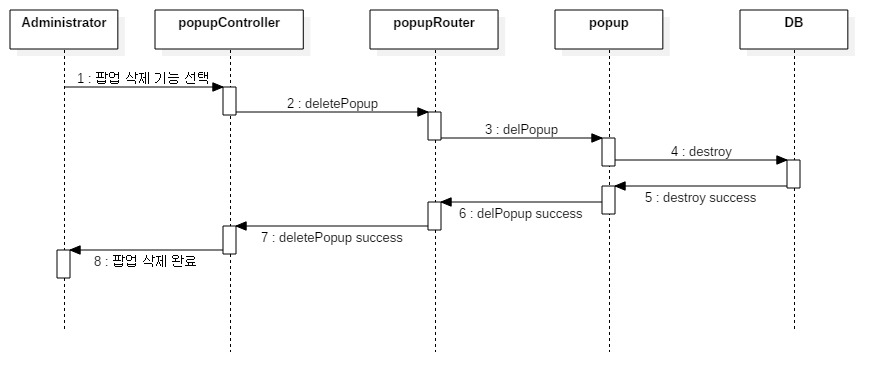


Diagram 14.5 Sequence Diagram(팝업 삭제)

## **State Diagram**

## 팝업 보기

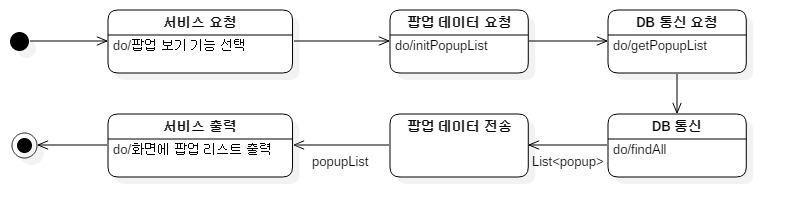


Diagram 14.6 State Diagram(팝업 보기)

## 팝업 추가

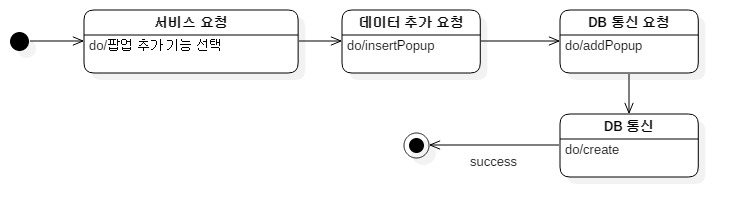


Diagram 14.7 State Diagram(팝업 추가)

## 팝업 수정

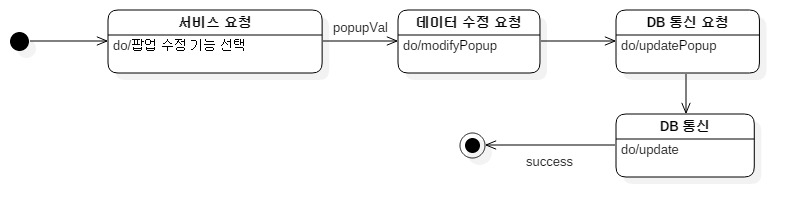


Diagram 14.8 State Diagram(팝업 수정)

## 팝업 삭제

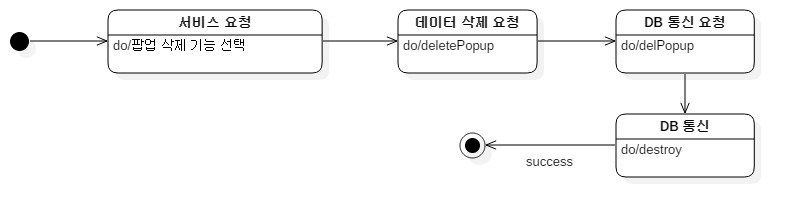


Diagram 14.9 State Diagram(팝업 삭제)

## **Communication Protocol**

Communication Protocol 에서는 서브 시스템들이 상호 통신하기 위하여 필수적으로 준수해야 하는 프로토콜에 대하여 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지의 형식과 용도, 의미를 설명한다.

## **AJAX**

AJAX(Asynchronous Javascript And XML)은 비동기 웹 어플리케이션에서 클라이언트-서버 모델 개발에 사용되는 기술 묶음이다. AJAX를 통해 웹 어플리케이션은 백그라운드에서 서버로 데이터를 전송하고 서버로부터 데이터를 요청할 수 있다.

## **JSON**

JSON(JavaScript Object Notation)은 객체의 형태로 자료를 표현하기 위하여 적용된 데이터 교환 형식이다. 간단한 데이터를 XML 보다 좀 더 쉽게 표현할 수 있으며, 기능도 적어서 파싱(parsing)이 빠르다는 장점을 가지고 있다. 클라이언트의 입장에서 매우 유용하며, 모바일을 기반으로 하는 데이터 전송에 사용하기에 적절한 형식이다. AJAX를 사용하여 데이터를 주고 받을 때의 포맷으로써 사용된다.

## **REST API**

REST(REpresentational State Transfer) API은 HTTP URL(리소스) + HTTP Method(메소드) 조합으로 행위(메시지)를 지정하는 방식의 웹 서비스 아키텍처이다. 메시지는 JSON 문서를 이용해 표현된다.

보통 HTTP의 여러 메소드 중 CRUD(Create, Read, Update, Delete)에 해당하는 4가지(Get, Post, Put, Delete)만 사용한다.

## **상품 시스템 프로토콜**

## 상품 조회 시

**AJAX**

GET, /item/:itemId

**JSON**

{

id : id,

blandId : blandId,

typeId : typeId,

size : size,

price : price,

photo : photo

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| itemId | integer | 상품의 고유번호 |  |

Table 15.1 상품 시스템 프로토콜 상품 조회 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| id | integer | 상품의 고유번호 |  |
| blandId | integer | 상품의 브랜드번호 |  |
| typeId | integer | 상품의 항목번호 |  |
| size | integer | 상품의 사이즈 |  |
| price | integer | 상품의 가격 |  |
| photo | string | 상품의 이미지 경로 |  |

Table 15.2 상품 시스템 프로토콜 상품 조회 응답 변수

## 상품을 장바구니에 추가했을 시

**AJAX**

POST, /cart/:itemId

**JSON**

{

error : (false,true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| itemId | integer | 상품의 고유번호 |  |

Table 15.3 상품 시스템 프로토콜 상품 장바구니 추가 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.4 상품 시스템 프로토콜 상품 장바구니 추가 응답 변수

## 장바구니의 상품을 구매했을 시

**AJAX**

POST, /order/

**JSON**

{

error : (false,true)

}

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.5 상품 시스템 프로토콜 상품 구매 응답 변수

## 장바구니의 상품을 삭제했을 시

**AJAX**

DELETE, /cart/:itemId

**JSON**

{

error : (false,true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| itemId | integer | 상품의 고유번호 |  |

Table 15.6 상품 시스템 프로토콜 상품 장바구니 삭제 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.7 상품 시스템 프로토콜 상품 장바구니 삭제 응답 변수

## **게시판 시스템 프로토콜**

## 게시글 조회 시

**AJAX**

GET, /board/:boardId

**JSON**

{

id : id,

userId : userId,

title : title,

contents : contents,

author : author,

type : type,

answer : answer,

file : {

id : id,

name : name,

path : path,

type : type,

size : size

}

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| boardId | integer | 게시글의 고유번호 |  |

Table 15.8 게시글 시스템 프로토콜 게시글 조회 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| id | integer | 게시글의 고유번호 |  |
| userId | integer | 게시글 글쓴이의 식별키 |  |
| title | integer | 게시글의 제목 |  |
| contents | integer | 게시글의 내용 |  |
| author | integer | 게시글의 글쓴이 |  |
| type | string | 게시판 타입 | 1:공지사항 2:FAQ 3:환불/교환 |
| answer | String | 환불/교환 게시글의 답변 | 공지사항,FAQ의 경우 항상 null |
| file.id | integer | 파일의 고유번호 |  |
| file.name | string | 파일의 이름 |  |
| file.path | string | 파일의 경로 |  |
| file.type | string | 파일의 종류 |  |
| file.size | integer | 파일의 크기 |  |

Table 15.9 게시글 시스템 프로토콜 게시글 조회 응답 변수

## 게시글 추가 시

**AJAX**

POST, /board/

**JSON**

{

error : (false, true)

}

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.10 게시글 시스템 프로토콜 게시글 추가 응답 변수

## 게시글 수정 시

**AJAX**

PUT, /board/:boardId

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| boardId | integer | 게시글의 고유번호 |  |

Table 15.11 게시글 시스템 프로토콜 게시글 수정 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.12 게시글 시스템 프로토콜 게시글 수정 응답 변수

## 게시글 삭제 시

**AJAX**

DELETE, /board/:boardId

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| boardId | integer | 게시글의 고유번호 |  |

Table 15.13 게시글 시스템 프로토콜 게시글 삭제 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.14 게시글 시스템 프로토콜 게시글 삭제 응답 변수

## **회원 정보 시스템 프로토콜**

## 회원가입 시

**AJAX**

POST, /user/

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| email | string | 사용자의 이메일 주소 |  |
| name | string | 사용자의 이름 |  |
| password | string | 사용자의 비밀번호 | sha256 암호화 |
| phone | string | 사용자의 휴대전화 번호 |  |
| address | String | 사용자의 집 주소 |  |

Table 15.15 회워 시스템 프로토콜 회원 가입 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.16 회원 시스템 프로토콜 회원 가입 응답 변수

## 로그인 시

**AJAX**

POST, /user/login

**JSON**

{

session : {

id : id,

email : email,

name : name,

wallet : wallet,

mileage : mileage,

phone : phone,

address : address

}

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| email | string | 사용자의 이메일 주소 |  |
| password | string | 사용자의 비밀번호 | sha256 암호화 |

Table 15.17 회워 시스템 프로토콜 로그인 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| session.id | integer | 사용자의 고유번호 |  |
| session.email | string | 사용자의 이메일 주소 |  |
| session.name | string | 사용자의 이름 |  |
| session.wallet | integer | 사용자의 가상화폐 액수 |  |
| session.mileage | integer | 사용자의 마일리지 액수 |  |
| session.phone | string | 사용자의 휴대전화 번호 |  |
| session,address | string | 사용자의 집 주소 |  |

Table 15.18 회원 시스템 프로토콜 로그인 응답 변수

## 로그아웃 시

**AJAX**

GET, /user/logout

**JSON**

{

error : (false, true)

}

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.19 회원 시스템 프로토콜 로그아웃 응답 변수

## 회원정보 변경 시

**AJAX**

PUT, /user/

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| password | string | 사용자의 비밀번호 | sha256 암호화 |
| phone | string | 사용자의 휴대전화 번호 |  |
| address | string | 사용자의 집 주소 |  |

Table 15.20 회워 시스템 프로토콜 회원정보 변경 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.21 회원 시스템 프로토콜 회원정보 변경 응답 변수

## **팝업 시스템 프로토콜**

## 팝업 조회 시

**AJAX**

GET, /popup/

**JSON**

[{

id : id,

startDate : startDate,

endDate : endDate,

picture : picture,

link : link

},{

id : id,

startDate : startDate,

endDate : endDate,

picture : picture,

link : link

},

….

]

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| id | integer | 팝업의 고유번호 |  |
| startDate | string | 팝업의 게시 시작 날짜 |  |
| endDate | string | 팝업의 게시 종료 날짜 |  |
| picture | string | 팝업 이미지 |  |
| link | string | 팝업 클릭시 링크 주소 |  |

Table 15.22 팝업 시스템 프로토콜 팝업 조회 응답 변수

## 팝업 등록 시

**AJAX**

POST, /popup/

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| startDate | string | 팝업의 게시 시작 날짜 |  |
| endDate | string | 팝업의 게시 종료 날짜 |  |
| picture | string | 팝업 이미지 |  |
| link | string | 팝업 클릭시 링크 주소 |  |

Table 15.23 팝업 시스템 프로토콜 팝업 등록 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.24 팝업 시스템 프로토콜 팝업 등록 응답 변수

## 팝업 수정 시

**AJAX**

PUT, /popup/:popupId

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| popupId | integer | 팝업의 고유 번호 |  |
| startDate | string | 팝업의 게시 시작 날짜 |  |
| endDate | string | 팝업의 게시 종료 날짜 |  |
| picture | string | 팝업 이미지 |  |
| link | string | 팝업 클릭시 링크 주소 |  |

Table 15.25 팝업 시스템 프로토콜 팝업 수정 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.26 팝업 시스템 프로토콜 팝업 수정 응답 변수

## 팝업 삭제 시

**AJAX**

DELETE, /popup/:popupId

**JSON**

{

error : (false, true)

}

**요청 변수**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| popupId | integer | 팝업의 고유번호 |  |

Table 15.27 팝업 시스템 프로토콜 팝업 삭제 요청 변수

**서버 응답**

|  |  |  |  |
| --- | --- | --- | --- |
| 변수명 | 타입 | 설명 | 비고 |
| error | bool | 에러 발생 여부 | 에러 발생시 true 아니면 false |

Table 15.28 팝업 시스템 프로토콜 팝업 삭제 응답 변수

## **Database Design**

Database design에서는 요구사항 명세서에 기술하였던 요구사항들을 바탕으로 데이터베이스를 설계하고 설명한다. 데이터베이스에 대한 ER diagram을 제시하고 각 Entity에 대해서 설명한다. 데이터베이스 관리 시스템으로는 MySQL을 사용한다.

## **ER Diagram**

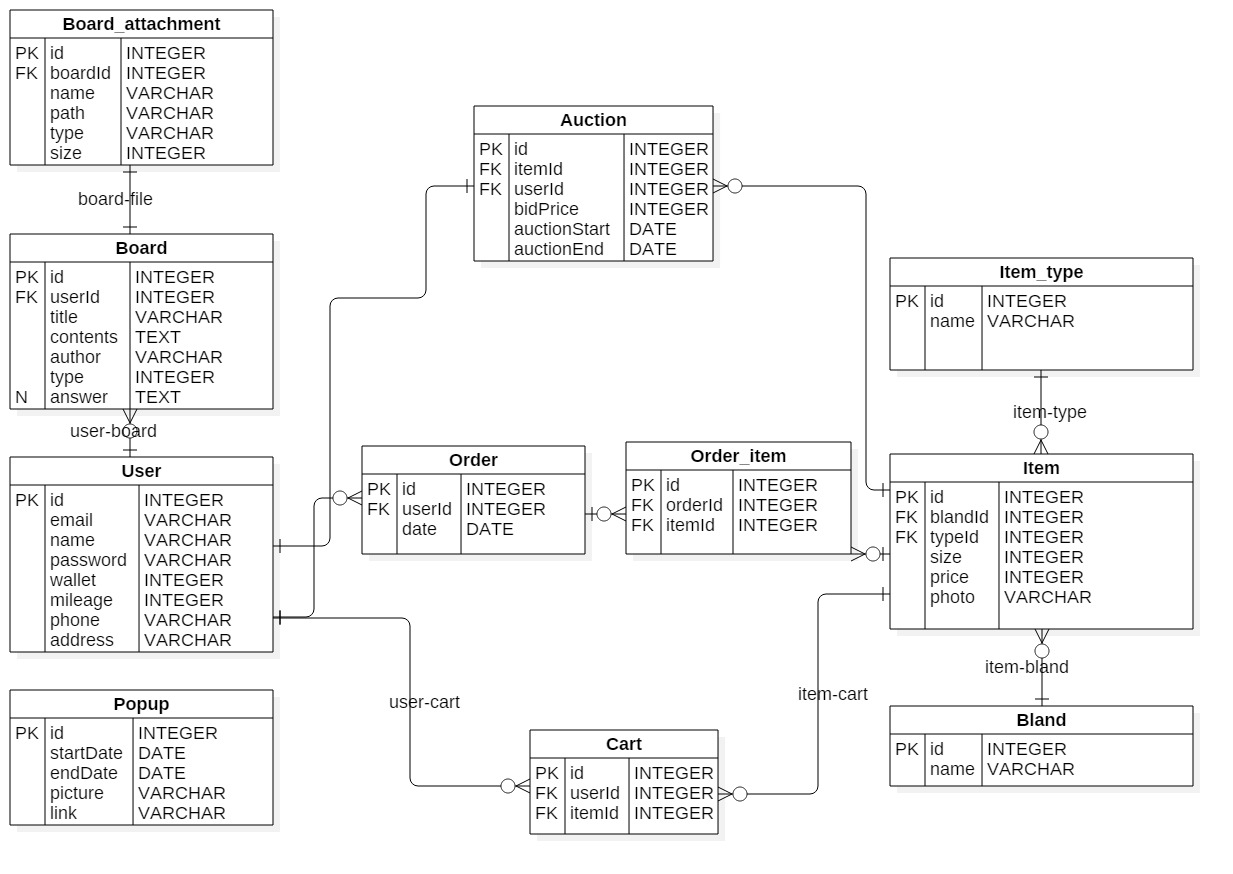


Diagram 16.1 SE Shoes ER Diagram

## **Entities**

## User

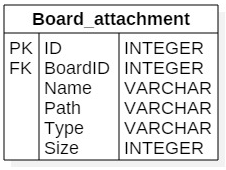


Diagram 16.2 User ER Diagram

User는 시스템을 이용하는 사용자를 나타낸다. ID는 사용자가 생성한 계정의 고유 ID로 Primary Key이다. Email은 사용자의 이메일을 주소이다. Name은 사용자의 이름이다. Password는 로그인하기 위한 비밀번호이다. Wallet은 사용자의 전자 지갑의 잔액이다. Mileage는 구매로 얻은 마일리지를 의미한다. Phone은 사용자의 휴대전화번호이다.

## Cart

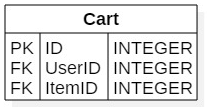


Diagram 16.3 Cart ER Diagram

Cart는 사용자의 장바구니이다. ID는 장바구니 고유 번호로 Primary Key이다. UserId는 해당 장바구니를 사용하는 User의 ID로 User로부터 참조하는 Foreign Key이다. ItemId는 장바구니에 담긴 상품의 ID로 하나의 장바구니에 여러 개의 ID가 담길 수 있다.

## Order

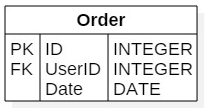


Diagram 16.4 Orde ER Diagram

Order는 사용자가 구매한 구매내역이다. ID는 구매 내역의 고유 번호로 Primary Key이다. UserID는 해당 구매내역을 소유한 User의 ID로 User로부터 참조하는 Foreign Key이다. Date는 해당 구매내역이 발생한 날짜이다.

## Order\_item

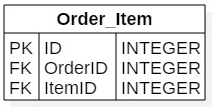


Diagram 16.5 Order\_item ER Diagram

Order\_Item은 Order의 부수적인 Entity로 해당 구매내역에서 구매했던 상품들을 의미한다. ID는 Order\_Item의 고유번호로 Primary Key이다. OrderID는 해당 Order\_Item에 대응하는 Order의 ID로 Order로부터 참조하는 Foreign Key이다. ItemID는 해당 Order\_Item에서의 구매상품들로 Item으로부터 참조하는 Foreign Key이며 여러 개의 ItemID가 담길 수 있다.

## Item

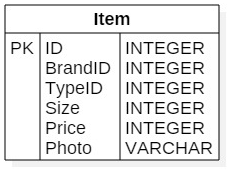


Diagram 16.6 Item ER Diagram

상품의 정보를 저장하는 Entity이다. ID는 해당 상품에 대한 고유번호로 Primary Key이다. BrandID는 해당 상품의 Brand의 ID로 Brand Entity로부터 참조하는 Foreign Key이다. TypeID는 해당 상품의 분류를 나타내는 속성으로 해당하는 분류의 ID이며 Item\_type으로부터 참조하는 Foreign Key이다. Size는 해당 상품의 크기를 나타내는 것으로 INTEGER값을 갖는다. Price는 상품의 가격으로 INTEGER값을 갖는다. Photo는 사용자에게 보일 상품의 이미지이다.

## Bland

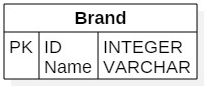


Diagram 16.7 Bland ER Diagram

Brand는 상품의 브랜드들을 저장하는 Entity이다. ID는 해당 브랜드의 고유번호로 Primary Key이다. Name은 해당 브랜드의 이름으로 VARCHAR값을 갖는다.

## Item\_type

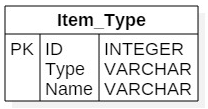


Diagram 16.8 Item\_type ER Diagram

Item\_Type은 상품의 분류를 저장하는 Entity이다. ID는 해당 분류의 고유번호로 Primary Key이다. Type은 분류 자체 속성으로 VARCHAR값을 갖는다. Name은 해당 분류의 이름으로 VARCHAR값을 갖는다.

## Board

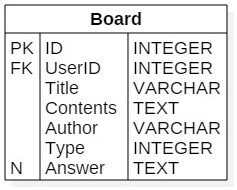


Diagram 16.9 Board ER Diagram

Board는 사용자가 이용할 수 있는 고객센터를 의미한다. 고객센터는 게시판 형식으로 운영되어 공지사항, FAQ, 반품 문의를 할 수 있다. ID는 공지사항, FAQ, 반품 문의 등 작성된 글의 고유 번호로 Primary Key이다. User는 반품 문의를 작성한 User의 ID로 User를 참조하는 Foreign Key이다. Title은 해당 글의 제목이다. Content는 해당 글의 내용으로 TEXT로 이루어져 있다. Author는 해당 Entity의 작성자로 VARCHAR값을 갖는다. Type은 해당 글의 종류를 구분하는 것으로 Type을 통해 공지사항인지, FAQ인지, 반품 문의인지 구별할 수 있다. Answer는 고객의 반품 문의에 대한 답변으로 반품 문의가 아닌 경우 내용이 비여 있을 수 있다.

## Board\_attachment

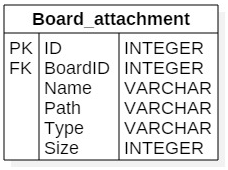


Diagram 16.10 Board\_attachment 바람

Board\_Attachment는 Board의 글에 첨부될 첨부파일이다. ID는 첨부파일의 고유 번호로 Primary Key이다. BoardID는 해당 파일이 첨부될 글의 ID로 Board를 참조하는 Foreign Key이다. Name은 첨부파일의 이름, Path는 첨부파일의 경로이며, Type, Size는 각각 첨부파일의 형식, 크기이다.

## Popup

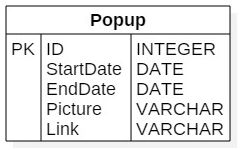


Diagram 16.11 Popup ER Diagram

Popup은 사용자에게 보일 팝업을 관리한다. ID는 팝업의 고유 번호로 Primary Key이다. StartDate는 팝업이 보이기 시작하는 날짜이다. EndDate는 팝업이 중단되는 날짜이다. 팝업은 현재 날짜가 StartDate와 EndDate사이일 때만 팝업을 띄운다. Picture는 팝업창에 들어갈 이미지이다. Link는 팝업을 통해 이동할 수 있는 웹 페이지의 링크이다.

## Auction

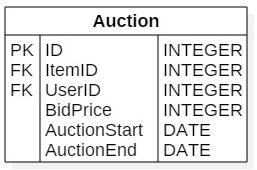


Diagram 16.12 Auction ER Diagram

Auction은 경매를 관리한다. ID는 경매의 고유 번호로 Primary Key이다. ItemID는 경매에 나온 상품의 ID로 Item을 참조하는 Foreign Key이다. UserID는 해당 상품에 입찰한 User의 ID로 User를 참조하는 Foreign Key이다. BidPrice는 상품의 현재 입찰가격이다. AuctionStart는 경매의 시작날짜, AuctionEnd는 경매의 종료날짜이다.

## **Relationship**

## Cart

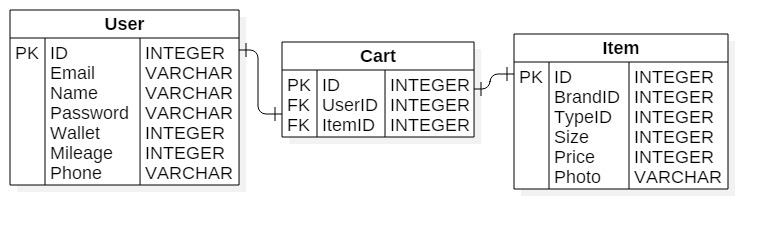


Diagram 16.13 Cart Relationship

Cart는 장바구니를 의미한다. Cart에는 해당 Cart의 사용자인 User의 ID를 참조하여 UserID로 저장한다. Cart의 ItemID는 장바구니에 담긴 상품의 ID로 Item Entity를 참조하는 값이다. 여러개의 ItemID가 Cart에 저장될 수 있다.

## Item

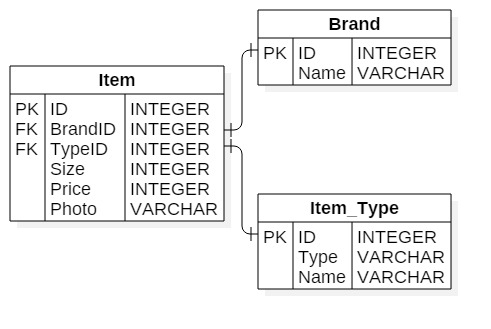


Diagram 16.14 Item Relationship

Item은 상품을 의미한다. Item에서 BrandID는 Brand Entity를 참조하는 값으로 상품의 브랜드의 ID값이다. TypeID는 Item\_Type를 참조하는 값으로 해당 type의 ID값이다.

## Order

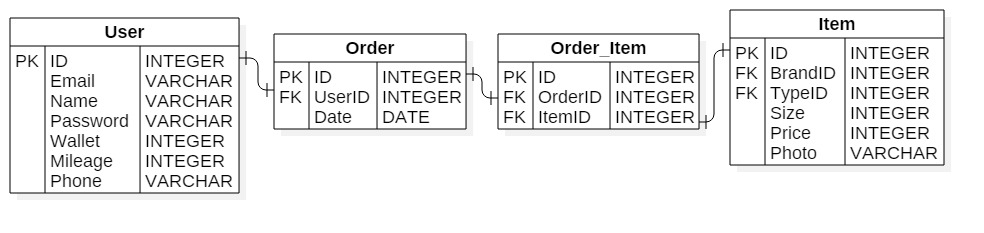


Diagram 16.15 Order Relationship

Order는 상품의 구매내역을 의미한다. 구매 내역인 Order는 해당 구매내역을 구매한 사용자의 ID를 User Entity로부터 참조한다. Order\_Item은 구매내역안의 아이템의 목록으로 구매 내역의 ID를 참조하여 OrderID에 저장하고 해당 구매 내역에 있는 상품들의 ID를 참조하여 ItemID로 저장한다. ItemID는 여러 개가 있을 수 있다.

## Board

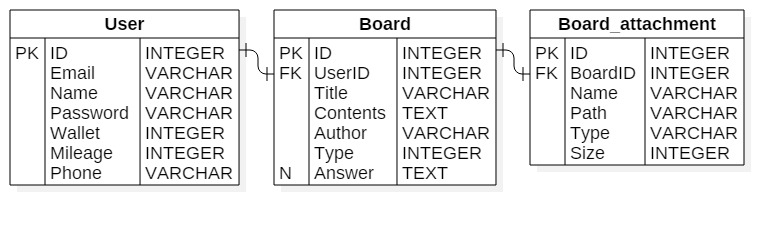


Diagram 16.16 Board Relationship

Board는 고객센터의 게시판을 의미한다. 고객센터의 게시판에는 공지사항, FAQ, 반품 문의가 있다. 반품 문의의 경우 해당 문의를 작성한 사용자의 ID를 참조하여 UserID로 저장한다. Board\_Attachment는 게시판에 작성된 글의 첨부파일로 해당 글의 ID를 참조하여 BoardID에 저장한다.

## Auction

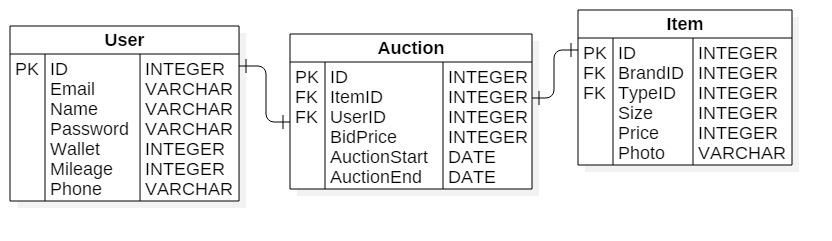


Diagram 16.17 Auction Relationship

Auction은 경매를 관리하는 Entity이다. 경매에 나온 상품의 ID를 참조하여 ItemID로 저장하며 해당 상품을 입찰한 사용자의 ID를 참조하여 UserID로 저장한다. 현재 가격인 BidPrice보다 높은 가격으로 입찰하는 사용자가 나온 경우, 해당 가격으로 BidPrice를 변경하고 그 가격으로 입찰한 사용자의 ID를 참조하여 UserID로 저장한다.

## **Relational Schema**

## User

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | | | | | | |
| id | email | name | password | wallet | mileage | phone |

Table 16.1 User Relational Schema

-Primary Key : id

-Functional Dependencies : id -> {email, name, password, wallet, mileage, phone}

사용자를 표현하는 테이블로 사용자의 이름, Email주소, 비밀번호, 전자지갑, 마일리지, 휴대전화번호 데이터를 담고 있는 테이블이다. Primary Key는 id로id가 {email, name, password, wallet, mileage, phone}를 결정한다.

## Cart

|  |  |  |
| --- | --- | --- |
| cart | | |
| id | userID | ItemID |

Table 16.2 Cart Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {userId, itemId}

사용자의 장바구니를 표현하는 테이블로 사용자의 ID, 상품의 ID 데이터를 담고 있는 테이블이다. userId를 조회하면 해당 장바구니의 사용자를 알 수 있고 itemId를 조회하면 장바구니에 담긴 상품을 알 수 있다. Primary Key는 ID로 ID가 {userId, itemId}를 결정한다.

## Order

|  |  |  |
| --- | --- | --- |
| cart | | |
| id | userId | date |

Table 16.3 Cart Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {userId, date}

사용자의 구매내역을 표현하는 테이블로 사용자의 id, 구매한 날짜 데이터를 담고있는 테이블이다. userId를 조회하면 해당 구매내역을 구매한 사용자를 알 수 있다. Primary Key는 id로 id가 {userId, date}를 결정한다.

## Order\_item

|  |  |  |
| --- | --- | --- |
| cart | | |
| id | orderId | itemId |

Table 16.4 Order\_item Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {orderId, itemId}

구매 내역에 포함된 상품 목록을 표현하는 테이블로 구매 내역의 id, 상품의 id 데이터를 담고있는 테이블이다. orderId를 조회하면 해당 상품 목록이 어떤 구매 내역의 목록인지 확인할 수 있다. Primary Key는 id로 id가 {orderId, itemId}를 결정한다.

## Item

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| item | | | | | |
| id | brandId | typeId | size | price | photo |

Table 16.5 Item Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {brandId, typeId, size, price, photo}

상품을 나타내는 테이블로 상품의 브랜드의 id, 상품의 분류의 id, 상품의 크기, 가격, 이미지 데이터를 담고 있는 테이블이다. brandId를 조회하면 해당 상품이 어느 브랜드의 상품인지 알 수 있고, typeId를 조사하면 해당 상품의 분류를 알 수 있다. Primary Key는 id로 id가 {brandId, typeId, size, price, photo}를 결정한다.

## Brand

|  |  |
| --- | --- |
| brand | |
| id | name |

Table 16.6 Brand Relational Schema

- Primary Key : id

- Functional Dependencies : id -> name

상품의 브랜드를 나타내는 테이블로 브랜드의 이름 데이터를 담고 있는 테이블이다. Primary Key는 id로 id가 name을 결정한다.

## Item\_Type

|  |  |
| --- | --- |
| item\_type | |
| id | name |

Table 16.7 Item\_type Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {type, name}

상품의 분류를 나타내는 테이블로 상품의 종류, 분류 이름 데이터를 담고있는 테이블이다. Primary Key는 id로 id가 {type, name}를 결정한다.

## Board

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Board | | | | | | |
| id | userId | title | contents | author | type | answer |

Table 16.8 Board Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {userId, title, contents, author, type, answer}

고객센터의 게시판을 나타내는 테이블로 글을 쓴 사용자의 ID, 제목, 내용, 작성자, 글의 종류, 답변 데이터를 담고 있는 테이블이다. userId를 조회하면 해당 글을 작성한 사용자가 누구인지 알 수 있다. answer는 사용자의 환불 문의에 대한 답변으로 환불 문의가 아닌 종류의 글에 대해서는 Null값이 들어갈 수 있다. Primary Key는 id로 id가 {userId, title, contents, author, type, answer}를 결정한다.

## Board\_Attachment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Board\_attachment | | | | | |
| id | boardId | name | path | type | size |

Table 16.9 Board\_attachment Schema

- Primary Key : id

- Functional Dependencies : id -> {boardId, name, path, type, size}

게시글의 첨부파일을 나타내는 테이블로 첨부된 글의 ID, 파일명, 파일 경로, 파일 종류, 파일 크기 데이터를 담고있는 테이블이다. boardId를 조회하면 해당 파일이 어떤 글의 첨부파일인지 알 수 있다. Primary Key는 id로 id가 {boardId, name, path, type, size}를 결정한다.

## Popup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| popup | | | | |
| id | startDate | endDate | picture | link |

Table 16.10 Popup Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {startDate, endDate, picture, link}

사용자에게 보일 팝업을 나타내는 테이블로 팝업 시작 날짜, 종료 날짜, 팝업 이미지, 팝업의 링크 데이터를 담고 있는 테이블이다. Primary Key는 id로 id가 {startDate, endDate, picture, link}를 결정한다.

## Auction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Board\_attachment | | | | | |
| id | itemId | userId | bidPrice | auctionStart | auctionEnd |

Table 16.11 Auction Relational Schema

- Primary Key : id

- Functional Dependencies : id -> {itemId, userID, bidPrice, auctionStart, auctionEnd}

경매를 나타내는 테이블로 경매에 나온 상품의 id, 입찰한 사용자의 id, 입찰 가격, 경매 시작 날짜, 경매 종료 날짜 데이터를 담고 있는 테이블이다. itemId를 조회하면 경매에 나온 상품의 정보를 알 수 있고 userId를 조회하면 현재 입찰 중인 사용자는 누구인지 알 수 있다. Primary Key는 id로 id가 {itemId, userId, bidPrice, auctionStart, auctionEnd}를 결정한다.

## Normalization

정규화는 관계형 데이터베이스의 설계에서 중복을 최소화하도록 데이터를 구조화하는 작업이다. 일반적으로 정규화는 이상이 있는 크고 잘 조직되지 않은 테이블들의 관계를 작고 잘 조직된 관계의 테이블로 나누는 것을 의미한다. 잘 조직되지 않은 테이블이 불러오는 이상에는 다음과 같은 것들이 있다.

Redundancy: 테이블에서 필요 이상의 정보가 여러 번 반복되는 것을 의미한다.

Insert Anomaly: 새로운 튜플을 삽입할 수 없다.

Delete Anomaly: 어떤 정보를 삭제할 때, 원치 않는 정보를 삭제하게 된다.

Update Anomaly: 정보를 업데이트 하고자 할 때, 여러 번 업데이트 해야 한다.

이를 해결하는 정규화 방법에는 여러가지가 있지만 현재 대다수의 사용 데이터베이스의 설계에서 고려하는 BCNF(Boyce-Codd Normal Form)을 목표로 하여 BCNF를 만족하도록 설계한다.

BCNF란 주어진 테이블에 대해서 모든 Functional Dependency X->A에 대하여 X가 Super key인 것을 의미한다. 따라서 각각의 테이블에서 모든 Functional Dependency X->A에 대하여 X가 Super key인지 확인한다.

## User

Functional Dependencies : ID -> {Email, Name, Password, Wallet, Mileage, Phone}

→ ID가 Super key이므로 BCNF를 만족한다.

## Cart

Functional Dependencies : ID -> {UserID, ItemID}

→ ID가 Super key이므로 BCNF를 만족한다.

## Order

Functional Dependencies : ID -> {UserID, Date}

→ ID가 Super key이므로 BCNF를 만족한다.

## Order\_item

Order\_Item

Functional Dependencies : ID -> {OrderID, ItemID}

→ ID가 Super key이므로 BCNF를 만족한다.

## Item

Functional Dependencies : ID -> {BrandID, TypeID, Size, Price, Photo}

→ ID가 Super key이므로 BCNF를 만족한다.

## Brand

Functional Dependencies : ID -> Name

→ ID가 Super key이므로 BCNF를 만족한다.

## Item\_type

Functional Dependencies : ID -> {Name}

→ ID가 Super key이므로 BCNF를 만족한다.

## Board

Functional Dependencies : ID -> {UserID, Title, Contents, Author, Type, Answer}

→ ID가 Super key이므로 BCNF를 만족한다.

## Board\_Attachment

Functional Dependencies : ID -> {BoardID, Name, Path, Type, Size}

→ ID가 Super key이므로 BCNF를 만족한다.

## Popup

Functional Dependencies : ID -> {StartDate, EndDate, Picture, Link}

→ ID가 Super key이므로 BCNF를 만족한다.

## Auction

Functional Dependencies : ID -> {ItemID, UserID, BidPrice, AuctionStart, AuctionEnd}

→ ID가 Super key이므로 BCNF를 만족한다.

## **SQL DDL**

## User

|  |
| --- |
| CART |
| CREATE TABLE USER {  ID INTEGER NOT NULL,  Email VARCHAR(20) NOT NULL,  Name VARCHAR(20) NOT NULL,  Password VARCHAR(20) NOT NULL,  Wallet INTEGER NOT NULL,  Mileage INTEGER NOT NULL,  Phone VARCHAR(20) NOT NULL,  PRIMARY KEY(ID)  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. Email은 로그인이나 아이디/비밀번호 찾기에 사용하기 위해 NOT NULL로 설정한다. Name과 Password는 반드시 필요한 정보이며 Wallet과 Mileage는 처음 가입시 0에서 시작하기에 NOT NULL이다. 휴대전화번호는 본인확인과 아이디/비밀번호 찾기에 사용하기 위해 NOT NULL로 설정한다. |

Table 16.12 User SQL DDL

## Cart

|  |
| --- |
| Cart |
| CREATE TABLE CART {  ID INTEGER NOT NULL,  UserID INTEGER NOT NULL,  ItemID INTEGER NOT NULL.  PRIMARY KEY(ID),  FOREIGN KEY(UserID) REPERENCES USER(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  FOREIGN KEY(ItemID) REPERENCES ITEM(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. UserID와 ItemID는 각 테이블의 Primary Key이기 때문에 NOT NULL로 설정한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.13 Cart SQL DDL

## Order

|  |
| --- |
| order |
| CREATE TABLE ORDER {  ID INTEGER NOT NULL,  UserID INTEGER NOT NULL,  Date DATE NOT NULL.  PRIMARY KEY(ID),  FOREIGN KEY(UserID) REPERENCES User(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. Date는 주문내역에 필수적인 요소이기 때문에 NOT NULL로 설정한다. UserID는 User테이블의 Primary Key이기 때문에 NOT NULL이다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.14 Order SQL DDL

## Order\_item

|  |
| --- |
| ORDER\_Item |
| CREATE TABLE ORDER\_ITEM {  ID INTEGER NOT NULL,  OrderID INTEGER NOT NULL,  ItemID INTEGER NOT NULL.  PRIMARY KEY(ID),  FOREIGN KEY(OrderID) REPERENCES ORDER (ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  FOREIGN KEY(ItemID) REPERENCES ITEM(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. OrderID와 ItemID는 각 테이블의 Primary Key이기 때문에 NOT NULL로 설정한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.15 Order\_item SQL DDL

## Item

|  |
| --- |
| Item |
| CREATE TABLE ITEM {  ID INTEGER NOT NULL,  BrandID INTEGER NOT NULL,  TypeID INTEGER NOT NULL.  Size INTEGER NOT NULL,  Price INTEGER NOT NULL,  Photo VARCHAR(20) NOTNULL,  PRIMARY KEY(ID),  FOREIGN KEY(BrandID) REPERENCES BRAND(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  FOREIGN KEY(TypeID) REPERENCES TYPE(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. BrandID와 TypeID는 각 테이블의 Primary Key이기 때문에 NOT NULL로 설정한다. Size, Price, Photo는 상품을 등록하는데 필수적인 요소이기 때문에 NOT NULL로 설정한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.16 Item SQL DDL

## Brand

|  |
| --- |
| brand |
| CREATE TABLE BRAND {  ID INTEGER NOT NULL,  Name VARCHAR(20) NOT NULL,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. Name은 브랜드를 등록하는데 필수적인 요소이기 때문에 NOT NULL로 설정한다. |

Figure 52 Brand SQL DDL

## Item\_type

|  |
| --- |
| ITEM\_TYPE |
| CREATE TABLE ITEM\_TYPE {  ID INTEGER NOT NULL,  Type VARCHAR(20) NOT NULL.  Name VARCHAR(20) NOTNULL,  PRIMARY KEY(ID),  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. Name과 Type은 분류를 등록하는데 필수적인 요소이기 때문에 NOT NULL로 설정한다. |

Table 16.17 Item\_type SQL DDL

## Board

|  |
| --- |
| BOARD |
| CREATE TABLE BOARD {  ID INTEGER NOT NULL,  UserID INTEGER NOT NULL,  Title VARCHAR(20) NOT NULL,  Contents TEXT NOT NULL,  Author VARCHAR(20) NOT NULL,  Type INTEGER NOT NULL,  Answer TEXT,  PRIMARY KEY(ID),  FOREIGN KEY(UserID) REPERENCES USER(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. UserID는 User테이블의 Primary Key이므로 NOT NULL로 설정한다. Title, Contents, Author, Type은 글을 등록하는데 필수적인 요소이므로 NOT NULL로 설정한다. Answer는 환불 문의에만 해당하는 항목이기에 환불 문의가 아닌 항목에서는 NULL을 허용한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.18 Board SQL DDL

## Board\_attachment

|  |
| --- |
| BOARD\_ATTACHMENT |
| CREATE TABLE BOARD {  ID INTEGER NOT NULL,  BoardID INTEGER NOT NULL,  Name VARCHAR(20) NOT NULL,  Path VARCHAR(20) NOT NULL,  Type VARCHAR(20) NOT NULL,  Size INTEGER NOT NULL,  PRIMARY KEY(ID),  FOREIGN KEY(BoardID) REPERENCES BOARD(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. BoardID는 Board테이블의 Primary Key이므로 NOT NULL로 설정한다. Name, Path, Type, Size는 첨부파일의 필수적인 정보이므로 NOT NULL로 설정한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.19 Board\_Attachment SQL DDL

## Popup

|  |
| --- |
| POPUP |
| CREATE TABLE POPUP {  ID INTEGER NOT NULL,  StartDate DATE NOT NULL,  EndDate DATE NOT NULL,  Picture VARCHAR(20) NOT NULL,  Link VARCHAR(20) NOT NULL,  PRIMARY KEY(ID),  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. StartDate와 EndDate는 팝업을 띄우는데 반드시 필요하며 Picture, Link는 팝업의 내용이므로 반드시 필요하기 떄문에 NOT NULL로 설정한다. |

Table 16.20 Popup SQL DDL

## Auction

|  |
| --- |
| AUCTION |
| CREATE TABLE AUCTION {  ID INTEGER NOT NULL,  ItemID INTEGER NOT NULL,  UserID INTEGER NOT NULL,  BidPrice INTEGER NOT NULL,  AuctionStart DATE NOT NULL,  AuctionEnd DATE NOT NULL,  PRIMARY KEY(ID),  FOREIGN KEY(ItemID) REPERENCES Item(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  FOREIGN KEY(UserID) REPERENCES User(ID)  CASCADE ON UPDATE  CASCADE ON DELETE,  } |
| ID는 Primary Key이므로 NOT NULL로 설정한다. ItemID와 UserID는 각 테이블의 Primary Key이므로 NOT NULL로 설정한다. AuctionStart, AuctionEnd는 경매의 시작 시간과 종료 시간이기 때문에 NOT NULL로 설정한다. BidPrice는 최초 경매가격이 반드시 설정되어야 하기 때문에 NOT NULL로 설정한다. 또한, Foreign key들이 수정, 삭제되면 같이 수정, 삭제되도록 CASCADE 설정하였다. |

Table 16.21 Auction SQL DDL

## **Testing Plan**

이 섹션에서는 Testing Plan과 테스트 케이스에 대해 설명한다.

## **Testing Policy**

시스템의 개발에서는 크게 세 단계로 테스트를 한다. Development Testing, Release testing, User testing으로 나뉘어지며, Development testing은 다시 component testing, integrating testing, system testing, acceptance testing의 네 단계로 나뉘어 진다.

1) Development testing은 개발을 하며 시스템의 문제를 찾아내는 작업이다.

- Component testing은 시스템을 component단위로 테스트하는 것으로, 각 구성요소들을 개발한 후에 제대로 작동하는지 확인하는 것이다. Integrating testing 은 시스템의 구성요소들을 개발하고 테스트 한 이후에 구성요소들을 하나씩 점진적으로 합치면서 하는 테스트이다. 각 구성요소 간의 interface와, 문제가 어디서 일어나는지 확인할 수 있다. System testing은 모든 sub-system을 하나로 합친 후에 시스템이 잘 동작하는지 테스트하는 것이다. 마지막으로 Acceptance testing은 사용자의 정보를 이용하여 시스템에 대한 사용자의 요구사항을 테스트하는 것이다.

2) Release testing은 사용자에게 출시하는 최종 시스템을 출시 전에 테스트하는 것이다.

3) User testing은 사용자가 사용자의 환경에서 시스템을 테스트 하는 것을 말한다.

## **Test Case**

## 관리자 시스템

## 로그인

관리자 : ID, PW를 입력하여, 로그인을 시도한다.

시스템 동작: DB의 관리자 기록과 입력 받은 ID, PW의 일치 여부를 확인한다.

(로그인 성공) 시스템 동작 : 관리자 페이지 목록으로 이동

(로그인 실패) 시스템 알림 : “ID나 PW가 일치하지 않습니다.

시스템 동작 : 로그인 화면으로 돌아간다.

## 상품 관리 – 보기

관리자 : 관리자 페이지목록에서 ‘상품보기’ 버튼을 클릭한다.

시스템 동작 : 관리자가 등록한 상품의 정보가 표시된다.

## 상품 관리 – 등록

관리자 : 관리자 페이지목록에서 ‘상품등록’ 버튼을 클릭한다.

시스템 동작 : 상품이 추가되었음을 확인하고 상품 목록을 갱신한다.

## 상품 관리 – 수정

관리자 : 관리자 페이지목록에서 ‘상품보기’ 버튼을 클릭한다.

시스템 동작 : 상품정보가 수정되었음을 확인하고 상품목록을 갱신한다.

## 상품 관리 – 삭제

관리자 : 상품의 정보를 삭제하고자 관리자 시스템을 실행한다.

시스템 알림 : “ 해당 상품을 삭제하시겠습니까? ”

시스템 동작 : 버튼을 클릭한다.

시스템 알림 : “ 상품이 목록에서 삭제되었습니다. “

시스템 동작 : 저장된 상품목록에서 해당 상품을 삭제한다.

## 회원 관리 – 보기

관리자 : 관리자 페이지목록에서 ‘가입된 회원 보기’ 버튼을 클릭한다.

시스템 동작 : 관리자가 웹사이트에 가입한 회원의 정보가 표시된다.

## 회원 관리 – 수정

관리자 : 관리자 페이지목록에서 ‘회원정보 수정’ 버튼을 클릭한다.

시스템 동작 : 관리자가 웹사이트에 가입한 회원의 정보가 수정된다.

## 회원 관리 – 삭제

관리자 : 관리자 페이지목록에서 ‘회원정보 삭제’ 버튼을 클릭한다.

시스템 동작 : 관리자가 웹사이트에 가입한 회원의 정보가 삭제된다.

## 판매 내역 관리 – 보기

관리자 : 관리자 페이지목록에서 ‘판매내역 확인’ 버튼을 클릭한다.

시스템 동작 : 관리자가 등록한 상품들의 판매내역이 표시된다.

## 판매 내역 관리 – 통계

관리자 : 관리자 페이지목록에서 ‘판매통계 보기’ 버튼을 클릭한다.

시스템 동작 : 관리자가 등록한 상품들의 판매내역에 대한 통계치가 표시된다.

## 게시판 관리 – 공지사항

관리자 : 관리자 페이지목록에서 ‘공지사항’ 버튼을 클릭한다.

시스템 동작 : 등록되어 있는 공지사항목록이 출력된다.

관리자 : 공지사항 목록에 ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 텍스트 편집창이 실행된다.

관리자 : 텍스트 편집을 완료한 후 ‘등록’버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : 공지사항목록에 작성한 게시물이 등록된다.

## 게시판 관리 – FAQ

관리자 : 관리자 페이지목록에서 ‘FAQ’ 버튼을 클릭한다.

시스템 동작 : 등록되어 있는 FAQ 목록이 출력된다.

관리자 : 공지사항 목록에 ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 텍스트 편집창이 실행된다.

관리자 : 텍스트 편집을 완료한 후 ‘등록’버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : FAQ 목록에 작성한 게시물이 등록된다.

## 게시판 관리 – 환불 / 반품 문의

관리자 : 관리자 페이지목록에서 ‘환불 / 반품 문의’ 버튼을 클릭한다.

시스템 동작 : 등록되어 있는 환불 / 반품 문의 목록이 출력된다.

관리자 : 공지사항 목록에 ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 텍스트 편집창이 실행된다.

관리자 : 텍스트 편집을 완료한 후 ‘등록’버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : 환불 / 반품 문의 목록에 작성한 게시물이 등록된다.

## 경매 관리 – 수정

관리자 : 관리자 페이지목록에서 ‘경매상품 수정’ 버튼을 클릭한다.

시스템 동작 : 판매자가 등록한 경매상품들의 정보를 수정한다.

## 경매 관리 – 삭제

관리자 : 관리자 페이지목록에서 ‘경매상품 삭제’ 버튼을 클릭한다.

시스템 동작 : 판매자가 등록한 경매상품들의 정보를 삭제한다.

## 사용자 시스템

## 로그인

사용자 : ID, PW를 입력하여, 로그인을 시도한다.

시스템 동작 : DB에 사용자 기록과 입력 받은 ID, PW의 일치 여부를 확인한다.

(로그인 성공) 시스템 동작 : 사용자 페이지 목록으로 이동

(로그인 실패) 시스템 알림 : “ ID나 PW가 일치하지 않습니다. “

시스템 동작 : 로그인 화면으로 돌아간다.

## 로그아웃

사용자 : 웹사이트 상단에 ‘ 로그아웃 ‘ 버튼을 클릭한다.

시스템 동작 : 회원의 상태가 로그아웃 상태로 변경된다.

## 상품 관리 – 조건 별 검색

사용자 : 상품 검색 시 ‘조건 별 검색’ 버튼을 클릭한다.

시스템 동작 : 회원이 입력한 조건에 대한 상품이 검색 된다.

## 상품 관리 – 장바구니 추가

사용자 : 몇가지 상품에 대한 체크박스를 선택하고 ‘장바구니 추가’ 버튼을 클릭한다.

시스템 동작 : 장바구니페이지가 표시되면서 선택한 상품이 추가 되어있는 것을 볼 수 있다.

## 상품 관리 – 최근 본 상품

사용자 : 사용자는 적어도 한가지의 상품의 상세정보를 확인한다.

시스템 동작 : 사용자가 클릭하였던 상품의 순서대로 Widget 형태로 화면에 표시

## 회원 정보 관리 – 회원 가입

사용자 : ‘회원가입’ 버튼을 클릭한다.

시스템 동작 : 회원가입 화면으로 이동한다.

사용자 : 개인 정보를 입력하고 저장한다.

(회원가입 성공) 시스템 알림 : “회원가입에 성공하였습니다. “

시스템 동작 : 개인정보를 DB에 저장하고, 로그인 화면으로 돌아간다.

(회원가입 실패) 시스템 알림 : “동일한 사용자 정보가 이미 있습니다.” Or “입력한 정보가 틀렸습니다.”

시스템 동작 : 회원가입 화면으로 돌아간다.

## 회원 탈퇴

사용자 : 마이페이지에서 ‘회원 탈퇴’ 버튼을 클릭한다.

시스템 알림 : “정말 탈퇴하시겠습니까?”

시스템 동작 : 회원정보가 DB에서 삭제된다.

## ID / PW 찾기

사용자 : 로그인 화면에서 ‘ID / PW 찾기’ 버튼을 클릭한다.

시스템 동작 : ID / PW 찾기 페이지가 실행된다.

사용자 : ID / PW를 찾기 위해 개인정보를 입력하고 ‘찾기’ 버튼을 클릭한다.

시스템 동작 : 입력한 정보가 일치하면 ID / PW를 사용자에게 알려준다.

## 회원정보 수정

사용자 : 마이페이지에서 ‘회원정보 수정’버튼을 클릭한다.

시스템 동작 : 회원정보 수정 페이지가 표시된다.

사용자 : 수정할 정보를 입력하고 ‘완료’ 버튼을 클릭한다.

시스템 알림 : “정보를 수정하시겠습니까?”

시스템 동작 : 마이페이지로 이동

## 마이페이지 – 구매 내역 보기

사용자 : 마이페이지에서 ‘구매 내역 보기’ 버튼을 클릭한다.

시스템 동작 : 사용자의 구매 내역이 표시된다.

## 마이페이지 – 지갑

사용자 : 마이페이지에서 ‘가상 지갑’ 버튼을 클릭한다.

(잔액이 없다면) 시스템 동작 : 잔액 추가페이지로 이동한다.

시스템 동작 : 결제정보를 입력하고 일치하면 잔액이 추가된다.

(잔액이 있다면) 시스템 동작 : 남은 잔액을 표시해준다.

## 장바구니 – 보기

사용자 : 사용자가 웹사이트에 ‘장바구니 확인’ 버튼을 클릭한다.

시스템 동작 : 상품이 담겨있는 장바구니 페이지가 표시된다.

## 장바구니 – 삭제

사용자 : 장바구니 페이지에서 삭제할 상품의 체크박스를 클릭하고 ‘삭제’

버튼을 클릭한다.

시스템 알림 : “상품을 삭제 하시겠습니까?”

시스템 동작 : 장바구니 페이지에서 해당 상품이 삭제된다.

## 장바구니 – 구매

사용자 : 장바구니 페이지에 담긴 상품들을 ‘일괄 구매’ 버튼을 클릭한다.

시스템 동작 : 결제정보 입력페이지로 이동한다.

시스템 동작 : 정보를 입력하고 ‘결제’ 버튼을 클릭한다.

시스템 알림 : “ 결제를 진행하시겠습니까? “

시스템 동작 : 가상지갑의 포인트가 삭감되면서 상품의 구매가 완료되어진다.

## 고객센터 – 공지사항

사용자 : ‘고객센터’ 버튼을 클릭하고 ‘공지사항’ 버튼을 클릭한다.

시스템 동작 : 공지사항 페이지로 이동한다.

(게시물을 작성한다면) 시스템 동작 : ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 게시물에 대한 내용을 입력한다.

사용자 : ‘완료’ 버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : 공지사항 페이지로 이동한다.

(게시물을 작성하지 않는다면) 시스템 동작 : 스크롤을 이용해 등록된 게시물을 확인할 수 있다.

## 고객센터 – FAQ

사용자 : ‘고객센터’ 버튼을 클릭하고 ‘FAQ’ 버튼을 클릭한다.

시스템 동작 : FAQ 페이지로 이동한다.

(게시물을 작성한다면) 시스템 동작 : ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 게시물에 대한 내용을 입력한다.

사용자 : ‘완료’ 버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : FAQ 페이지로 이동한다.

(게시물을 작성하지 않는다면) 시스템 동작 : 스크롤을 이용해 등록된 게시물을 확인할 수 있다.

## 고객센터 – 환불 / 반품 문의

사용자 : ‘고객센터’ 버튼을 클릭하고 ‘환불 / 반품 문의’ 버튼을 클릭한다.

시스템 동작 : 환불 / 반품 문의 페이지로 이동한다.

(게시물을 작성한다면) 시스템 동작 : ‘글쓰기’ 버튼을 클릭한다.

시스템 동작 : 게시물에 대한 내용을 입력한다.

사용자 : ‘완료’ 버튼을 클릭한다.

시스템 알림 : “게시물을 등록하시겠습니까?”

시스템 동작 : 환불 / 반품 문의 페이지로 이동한다.

(게시물을 작성하지 않는다면) 시스템 동작 : 스크롤을 이용해 등록된 게시물을 확인할 수 있다.

## **Development Environment**

Development Environment 에서는 실제 시스템 개발을 위하여 필요한 개발 환경을 설명한다. 개발 과정에서의 버전 관리 도구를 제시하고, 프로그래밍을 더 쉽게 해주는 프레임워크에 대하여 서술한다.

## **Operating Systems**



Figure 18.1 Window 10

개발 환경 운영 체제로는 기존 팀원들이 개발 환경을 구축하고 사용해 오던 운영체제 Window 10을 채택하였다. 웹 어플리케이션이 구동되는 서버는 리눅스 ubuntu 이므로 호환성에 초점을 맞추어 추후 유지 보수에 개발 환경 운영 체제에 대한 의존도를 제거하였다.

## **Running Environment**

## WEB



Figure 18.2 www

SE Shoes는 웹 어플리케이션으로 웹 환경에서 서비스된다. 웹 어플리케이션은 운영체제에 대한 의존성이 없기 때문에 어떤 운영체제로든 접속할 수 있다. 그리고 웹 브라우저에 대한 의존성을 제거하여 Chrome, Internet Explorer, Safari 등 어떤 웹 브라우저로 접속하든 같은 결과물을 보이게 개발하였다.

## ubuntu



Figure 18.3 ubuntu

우분투는 서버PC에 많이 사용되는 리눅스에 기반한 운영체제이다. 그리고 오픈소스 운영체제이므로 추가 비용이 들지 않고 필요 사양이 높지 않기 때문에 웹 서버가 구동되는 운영체제로 채택하였다.

## Digital Ocean



Figure 18.4 Digital Ocean

Digital Ocean은 VPS(Virtual Private Server) 서비스를 제공하는 호스팅 업체이다. 서버의 사양을 상황에 맞게 자유롭게 조절할 수 있으므로 유지비용 측면에서 효율적이고, 개인 서버가 없는 개발자로서 특별한 조작 없이 서버를 사용할 수 있다는 점에서 장점을 가지고 있다. VPS 업체중 Digital Ocean을 채택한 이유는 국내 호스팅 없체의 경우 서버에 패키지 설치가 비교적 자유롭지 않은데 Digital Ocean은 Console 환경을 제공하여 사용자가 원하는 패키지를 자유롭게 설치할 수 있고 효율적으로 웹 서버를 싼 가격에 운영할 수 있다.

## **IDE**

## Sublime Text 2.0



Figure 18.5 Sublime

서브라임 텍스트(Sublime Text)는 텍스트 에디터 프로그램으로, 프로그래밍에 유용한 플러그인을 제공한다. 웹 개발은 별도의 컴파일러를 필요로 하지 않기 때문에 서브라임 텍스트는 웹 개발에 특히 강력한 기능을 제공한다.

## **Programming Language**

## HTML5+CSS3+JavaScript



Figure 18.6 HTML5, CSS3, JavaScript

HTML(Hyper Text Markup Language)는 웹 환경에서 웹 페이지를 구성하는 구조를 나타내는 언어다. 꺾쇠 괄호에 둘러싸인 태그로 되어있는 HTML 요소 형태로 작성한다. HTML5는 HTML의 5번째 버전으로 좀더 쉽게 웹 표준에 맞는 웹 페이지를 구현을 하게 도와준다.

CSS(Cascading Style Sheets)는 HTML의 외관과 배치를 정의하는 스크립트 언어이다. HTML은 구조를 담당하고 CSS가 외관을 담당한다고 할 수 있다. CSS3는 CSS의 최신버전으로 기존 CSS문법에 JavaScript 없이도 동적인 표현을 하는 기능이 포함 되어있다.

JavaScript는 웹 페이지의 구성을 좀 더 동적으로 보여주기 위해 사용되는 스크립트 언어이며 SE Shoes의 서버 사이드와 프론트 사이드의 프레임워크는 자바스크립트로 이루어져 동적인 표현 외로도 서버 개발에도 이용된다.

## Node.js(+Express)

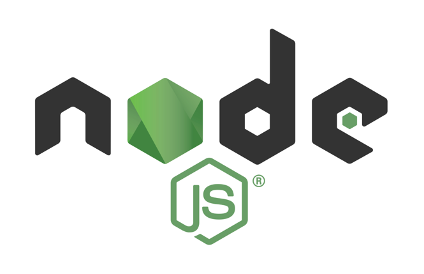


Figure 18.7 Node.js

서버 개발 언어로 Node.js를 사용하기로 하였다. Node.js는 크롬에서 만든 자바스크립트 엔진 V8을 이용해서 만든 서버 사이드 미들웨어로 처리 속도가 매우 빠르다. 그리고 HTTP 메소드에 대한 처리를 쉽게 할 수 있으므로 RESTful한 API를 쉽게 만들 수 있다. 그리고 Express라는 웹 프레임워크를 함께 사용하면 개발자 입장에서 더욱 쉽게 웹 어플리케이션을 개발할 수 있다.

## AngularJS



Figure 18.8 AngularJS

AngularJS는 웹 프론트 엔드에서 파일을 구조화 하여 사용하는 프레임워크이다. 특히 SPA(Single Page Application) 개발에 특화되어 있으며 JQuery 없이 웹 페이지를 동적으로 표현할 수 있다. 웹 개발에 MVC(Model-View-Controller) 구조로 개발을 하는게 보통인데 AngularJS의 경우 MVW(Model-View-Whatever)라고 불린다.

## **Version Control**

## Github



Figure 18.9 Github

본 프로젝트에서는 SE Shoes 어플리케이션 소스 관리 및 협업을 위해 Github를 이용하기로 하였다. Github는 전세계적으로 가장 많이 사용되는 웹 기반 형상관리 서비스이다. 인기가 많은 서비스인 만큼 신뢰할 수 있으므로 코드를 안전하게 관리할 수 있다.

## **Open Source**

|  |  |  |
| --- | --- | --- |
| FIgure no. | title | no. |
| jquery | MIT | 클라이언트 사이드 스크립트 언어를 단순화 한 자바스크립트 라이브러리 |
| angular | MIT | 클라이언트 사이드 뷰 템플릿 프레임워크 |
| angular-animate | MIT | angular에서 css와 디렉티브를 제공 |
| angular-cookies | MIT | Browser cookies 기능을 제공 |
| angular-route | MIT | $route를 통한 URL controller 역할을 제공 |
| angular-sanitize | MIT | 안전하지 않은 입력을 문자열로 반환한다. |
| angular-touch | MIT | 터치를 통한 이벤트를 관리하는 모듈 제공 |
| angular-bootstrap | MIT | angular에 작성된 bootstrap |
| angular-ui-tree | MIT | 정렬이나 drag&drop과 같은 UI 제공 |
| ng-file-upload | MIT | 파일 업로드 기능 제공 |
| bootstrap | MIT | 반응형 웹을 위한 기능을 제공하는 HTML, CSS, JavaScript 프레임워크 |
| async | MIT | 비동기식 작동을 지원하는 모듈 |
| body-parser | MIT | Node.js에서 body parsing을 제공하는 미들웨어 |
| connect-multiparty | MIT | multiparty API 기능 제공 |
| connect-timeout | MIT | Request의 times out 기능을 제공 |
| cookie-parser | MIT | cookie 기능을 제공 |
| debug | MIT | 디버깅 기능을 제공 |
| ejs | Apache-2.0 | Embedded JavaScript templates |
| express | MIT | Node.js를 위한 서버 사이드 웹 프레임워크 |
| express-session | MIT | express에서 세션 기능을 제공 |
| mkdirp | MIT | 리눅스의 mkdir -p 기능을 제공 |
| moment | MIT | date parsing을 위한 JavaScript 라이브러리 |
| morgan | MIT | HTTP request logger |
| multer | MIT | Node.js에서 Handling 기능을 제공하는 미들웨어 |
| mysql | MIT | Node.js를 위한 mysql driver |
| nodemailer | MIT | Node.js에서 이메일 전송 기능을 제공 |
| sequelize | MIT | 객체 관계 매핑을 제공하는 데이터베이스 프레임워크 |
| serve-favicon | MIT | favicon 기능을 제공 |
| sha256 | MIT | SHA256 암호화 기능을 제공 |
| grunt | MIT | JavaScript Task Runner |

## **Index**

## **Figure Index**

|  |  |  |
| --- | --- | --- |
| FIgure no. | title | no. |
| Figure 3.1 | 전체 시스템 아키텍쳐 | 25 |
| Figure 3.2 | 클라이언트 부분 시스템 아키텍쳐 | 26 |
| Figure 3.3 | 서버부분 시스템 아키텍쳐 | 27 |
| Figure 18.1 | Window 10 | 163 |
| Figure 18.2 | www | 164 |
| Figure 18.3 | Ubuntu | 164 |
| Figure 18.4 | Digital Ocean | 165 |
| Figure 18.5 | Sublime Text | 165 |
| Figure 18.6 | HTML5, CSS3, JavaScript | 166 |
| Figure 18.7 | Node.js | 167 |
| Figure 18.8 | AngularJS | 168 |
| Figure 18.9 | Github | 168 |

## **Diagram Index**

|  |  |  |
| --- | --- | --- |
| diagram no. | title | Page. |
| Diagram 3.1 | 사용자 관점 System Organization Block Diagram | 24 |
| Diagram 3.2 | 관리자 관점 System Organization Block Diagram | 24 |
| Diagram 3.3 | SE Shoes Deployment Diagram | 29 |
| Diagram 3.4 | SE Shoes Package Diagram | 30 |
| Diagram 4.1 | Item SubSystem Class Diagram | 31 |
| Diagram 4.2 | Sequence Diagram (상품 보기) | 36 |
| Diagram 4.3 | Sequence Diagram (장바구니 담기) | 36 |
| Diagram 4.4 | Sequence Diagram (장바구니 삭제) | 37 |
| Diagram 4.5 | Sequence Diagram (상품 구매) | 37 |
| Diagram 4.6 | State Diagram (상품 보기) | 38 |
| Diagram 4.7 | State Diagram (장바구니 담기) | 38 |
| Diagram 4.8 | State Diagram (장바구니 삭제) | 38 |
| Diagram 4.9 | State Diagram(상품 구매) | 39 |
| Diagram 5.1 | Auction SubSystem Class Diagram | 40 |
| Diagram 5.2 | Sequence Diagram(상품보기) | 43 |
| Diagram 5.3 | Sequence Diagram(입찰하기) | 43 |
| Diagram 5.4 | State Diagram(상품 보기) | 44 |
| Diagram 5.5 | State Diagram(입찰하기) | 44 |
| Diagram 6.1 | Customer Center SubSystem Class Diagram | 45 |
| Diagram 6.2 | Sequence Diagram(게시글 보기) | 49 |
| Diagram 6.3 | Sequence Diagram(환불 문의 작성하기) | 49 |
| Diagram 6.4 | Sequence Diagram(환불 문의 수정하기) | 50 |
| Diagram 6.5 | Sequence Diagram(환불 문의 삭제하기) | 50 |
| Diagram 6.6 | State Diagram(게시글 보기) | 51 |
| Diagram 6.7 | State Diagram(환불 문의 작성하기) | 51 |
| Diagram 6.8 | State Diagram(환불 문의 수정하기) | 51 |
| Diagram 6.9 | State Diagram(환불 문의 삭제하기) | 52 |
| Diagram 7.1 | User Information SubSystem Class Diagram | 53 |
| Diagram 7.2 | Sequence Diagram(회원가입) | 56 |
| Diagram 7.3 | Sequence Diagram (로그인) | 57 |
| Diagram 7.4 | Sequence Diagram(로그아웃) | 57 |
| Diagram 7.5 | Sequence Diagram(회원정보 변경) | 58 |
| Diagram 7.6 | Sequence Diagram(ID찾기) | 58 |
| Diagram 7.7 | Sequence Diagram(PW찾기) | 59 |
| Diagram 7.8 | Sequence Diagram(회원탈퇴) | 59 |
| Diagram 7.9 | State Diagram(회원가입) | 60 |
| Diagram 7.10 | State Diagram(로그인) | 60 |
| Diagram 7.11 | State Diagram(로그아웃) | 60 |
| Diagram 7.12 | State Diagram(회원정보 변경) | 61 |
| Diagram 7.13 | State Diagram(ID찾기) | 61 |
| Diagram 7.14 | State Diagram(PW 찾기) | 61 |
| Diagram 7.15 | State Diagram(회원탈퇴) | 62 |
| Diagram 8.1 | Item Admin SubSystem Class Diagram | 63 |
| Diagram 8.2 | Sequence Diagram(상품 보기) | 66 |
| Diagram 8.3 | Sequence Diagram(상품 등록) | 66 |
| Diagram 8.4 | Sequence Diagram(상품 수정) | 67 |
| Diagram 8.5 | Sequence Diagram(상품 삭제) | 67 |
| Diagram 8.6 | State Diagram(상품 보기) | 68 |
| Diagram 8.7 | State Diagram(상품 추가) | 68 |
| Diagram 8.8 | State Diagram(상품 수정) | 68 |
| Diagram 8.9 | State Diagram(상품 삭제) | 69 |
| Diagram 9.1 | Auction Admin SubSystem Class Diagram | 70 |
| Diagram 9.2 | Sequence Diagram(경매 보기) | 73 |
| Diagram 9.3 | Sequence Diagram(경매 등록) | 74 |
| Diagram 9.4 | Sequence Diagram(경매 수정) | 74 |
| Diagram 9.5 | Sequence Diagram(경매 삭제) | 75 |
| Diagram 9.6 | State Diagram(경매 보기) | 75 |
| Diagram 9.7 | State Diagram(경매 등록) | 76 |
| Diagram 9.8 | State Diagram(경매 수정) | 76 |
| Diagram 9.9 | State Diagram(경매 삭제) | 76 |
| Diagram 10.1 | itemType/Bland Admin SubSystem Class Diagram | 78 |
| Diagram 10.2 | Sequence Diagram (분류/브랜드 보기) | 83 |
| Diagram 10.3 | Sequence Diagram(분류/브랜드 추가) | 83 |
| Diagram 10.4 | Sequence Diagram(분류/브랜드 수정) | 84 |
| Diagram 10.5 | Sequence Diagram(분류/브랜드 삭제) | 84 |
| Diagram 10.6 | State Diagram(분류/브랜드 보기) | 85 |
| Diagram 10.7 | State Diagram(분류/브랜드 추가) | 85 |
| Diagram 10.8 | State Diagram(분류/브랜드 수정) | 85 |
| Diagram 10.9 | State Diagram(분류/브랜드 삭제) | 86 |
| Diagram 11.1 | User Admin SubSystem Class Diagram | 87 |
| Diagram 11.2 | Sequence Diagram(회원 목록 보기) | 90 |
| Diagram 11.3 | Sequence Diagram(회원 정보 수정) | 90 |
| Diagram 11.4 | Sequence Diagram(회원 삭제) | 91 |
| Diagram 11.5 | State Diagram(회원 목록 보기) | 91 |
| Diagram 11.6 | State Diagram(회원 정보 수정) | 91 |
| Diagram 11.7 | State Diagram(회원 삭제) | 92 |
| Diagram 12.1 | Order Admin SubSystem Class Diagram | 93 |
| Diagram 12.2 | Sequence Diagram (판매 내역 보기) | 96 |
| Diagram 12.3 | Sequence Diagram (판매 내역 통계) | 96 |
| Diagram 12.4 | State Diagram (판매 내역 보기) | 97 |
| Diagram 12.5 | State Diagram (판매 내역 통계) | 97 |
| Diagram 13.1 | Customer Center Admin SubSystem Class Diagram | 98 |
| Diagram 13.2 | Sequence Diagram (게시글 보기) | 102 |
| Diagram 13.3 | Sequence Diagram(게시글 등록) | 102 |
| Diagram 13.4 | Sequence Diagram(게시글 수정) | 103 |
| Diagram 13.5 | Sequence Diagram(게시글 삭제) | 103 |
| Diagram 13.6 | Sequence Diagram(환불/교환 문의 답변하기) | 104 |
| Diagram 13.7 | State Diagram(게시글 보기) | 104 |
| Diagram 13.8 | State Diagram(게시글 등록) | 105 |
| Diagram 13.9 | State Diagram(게시글 수정) | 105 |
| Diagram 13.10 | State Diagram(게시글 삭제) | 105 |
| Diagram 13.11 | State Diagram(환불/교환 문의 답변하기) | 106 |
| Diagram 14.1 | Popup Admin SubSystem Class Diagram | 107 |
| Diagram 14.2 | Sequence Diagram (팝업 보기) | 110 |
| Diagram 14.3 | Sequence Diagram(팝업 추가) | 110 |
| Diagram 14.4 | Sequence Diagram(팝업 수정) | 111 |
| Diagram 14.5 | Sequence Diagram(팝업 삭제) | 111 |
| Diagram 14.6 | State Diagram(팝업 보기) | 112 |
| Diagram 14.7 | State Diagram(팝업 추가) | 112 |
| Diagram 14.8 | State Diagram(팝업 수정) | 112 |
| Diagram 14.9 | State Diagram(팝업 삭제) | 113 |
| Diagram 16.1 | SE Shoes ER Diagram | 130 |
| Diagram 16.2 | User ER Diagram | 130 |
| Diagram 16.3 | Cart ER Diagram | 131 |
| Diagram 16.4 | Order ER Diagram | 131 |
| Diagram 16.5 | Order\_item ER Diagram | 132 |
| Diagram 16.6 | Item ER Diagram | 132 |
| Diagram 16.7 | Brand ER Diagram | 133 |
| Diagram 16.8 | Item\_type ER Diagram | 133 |
| Diagram 16.9 | Board ER Diagram | 133 |
| Diagram 16.10 | Board\_attachment | 134 |
| Diagram 16.11 | Popup ER Diagram | 134 |
| Diagram 16.12 | Auction ER Diagram | 135 |
| Diagram 16.13 | Cart Relationship | 135 |
| Diagram 16.14 | Item Relationship | 136 |
| Diagram 16.15 | Order Relationship | 136 |
| Diagram 16.16 | Board Relationship | 137 |
| Diagram 16.17 | Auction Relationship | 137 |

## **Table Index**

|  |  |  |
| --- | --- | --- |
| table no. | title | Page. |
| Table 15.1 | 상품 시스템 프로토콜 상품 조회 요청 변수 | 115 |
| Table 15.2 | 상품 시스템 프로토콜 상품 조회 응답 변수 | 115 |
| Table 15.3 | 상품 시스템 프로토콜 상품 장바구니 추가 요청 변수 | 116 |
| Table 15.4 | 상품 시스템 프로토콜 상품 장바구니 추가 응답 변수 | 116 |
| Table 15.5 | 상품 시스템 프로토콜 상품 구매 응답 변수 | 116 |
| Table 15.6 | 상품 시스템 프로토콜 상품 장바구니 삭제 요청 변수 | 117 |
| Table 15.7 | 상품 시스템 프로토콜 상품 장바구니 삭제 응답 변수 | 117 |
| Table 15.8 | 게시글 시스템 프로토콜 게시글 조회 요청 변수 | 118 |
| Table 15.9 | 게시글 시스템 프로토콜 게시글 조회 응답 변수 | 119 |
| Table 15.10 | 게시글 시스템 프로토콜 게시글 추가 응답 변수 | 120 |
| Table 15.11 | 게시글 시스템 프로토콜 게시글 수정 요청 변수 | 120 |
| Table 15.12 | 게시글 시스템 프로토콜 게시글 수정 응답 변수 | 121 |
| Table 15.13 | 게시글 시스템 프로토콜 게시글 삭제 요청 변수 | 121 |
| Table 15.14 | 게시글 시스템 프로토콜 게시글 삭제 응답 변수 | 121 |
| Table 15.15 | 회워 시스템 프로토콜 회원 가입 요청 변수 | 122 |
| Table 15.16 | 회원 시스템 프로토콜 회원 가입 응답 변수 | 122 |
| Table 15.17 | 회원 시스템 프로토콜 로그인 요청 변수 | 123 |
| Table 15.18 | 회원 시스템 프로토콜 로그인 응답 변수 | 124 |
| Table 15.19 | 회원 시스템 프로토콜 로그아웃 응답 변수 | 124 |
| Table 15.20 | 회워 시스템 프로토콜 회원정보 변경 요청 변수 | 125 |
| Table 15.21 | 회원 시스템 프로토콜 회원정보 변경 응답 변수 | 125 |
| Table 15.22 | 팝업 시스템 프로토콜 팝업 조회 응답 변수 | 127 |
| Table 15.23 | 팝업 시스템 프로토콜 팝업 등록 요청 변수 | 127 |
| Table 15.24 | 팝업 시스템 프로토콜 팝업 등록 응답 변수 | 127 |
| Table 15.25 | 팝업 시스템 프로토콜 팝업 수정 요청 변수 | 128 |
| Table 15.26 | 팝업 시스템 프로토콜 팝업 수정 응답 변수 | 128 |
| Table 15.27 | 팝업 시스템 프로토콜 팝업 삭제 요청 변수 | 129 |
| Table 15.28 | 팝업 시스템 프로토콜 팝업 삭제 응답 변수 | 129 |
| Table 16.1 | User Relational Schema | 138 |
| Table 16.2 | Cart Relational Schema | 138 |
| Table 16.3 | Order Relational Schema | 139 |
| Table 16.4 | Order\_item Relational Schema | 139 |
| Table 16.5 | Item Relational Schema | 139 |
| Table 16.6 | Brand Relational Schema | 140 |
| Table 16.7 | Item\_type Relational Schema | 140 |
| Table 16.8 | Board Relational Schema | 140 |
| Table 16.9 | Board\_attachment Schema | 141 |
| Table 16.10 | Popup Relational Schema | 141 |
| Table 16.11 | Auction Relational Schema | 142 |
| Table 16.12 | User SQL DDL | 145 |
| Table 16.13 | Cart SQL DDL | 146 |
| Table 16.14 | Order SQL DDL | 147 |
| Table 16.15 | Order\_item SQL DDL | 148 |
| Table 16.16 | Item SQL DDL | 149 |
| Table 16.17 | Item\_type SQL DDL | 150 |
| Table 16.18 | Board SQL DDL | 151 |
| Table 16.19 | Board\_Attachment SQL DDL | 152 |
| Table 16.20 | Popup SQL DDL | 153 |
| Table 16.21 | Auction SQL DDL | 154 |