<골든부츠>

<미스터 대박 디너 서비스>

설계 산출물

|  |  |
| --- | --- |
| 문서버전 | 2.0 |
| 문서 ID | MDS-2022-002 |
| 최종변경일 | 2022-11-27 |
| 문서상태 | 릴리즈 |

목차

1. 개요
   1. 목적
   2. 참고문헌
2. 아키텍처도
3. 클래스도
   1. Interface
   2. Homepage
   3. Resource
   4. Order
   5. People
   6. DB
4. 교류도
   1. Sign up & Sign in
   2. Place an Order
   3. Check & Modify User Status
   4. Look up Events
   5. View Vouchers
   6. Manage Resources
   7. Check Orders
   8. Manage Website
5. 객체 상태도
   1. Ordermanager

요약

미스터 대박 디너 서비스 시스템의 설계 산출물을 기술.

서브 시스템의 구성과 각 서브 시스템의 구조를 기술.

주요 산출물

* 아키텍처도
* 클래스도
* 교류도
* 객체 상태도

표 1 문서 변경 기록

|  |  |  |  |
| --- | --- | --- | --- |
| 문서이름 | <골든부츠 팀> <Mr.대박 서비스> 설계 산출물 | | |
| 문서 ID | MDS-2022-002 | | |
| 버전 | | 변경일 | 설명 |
| 1 | 0 | 2022-10-15 | 아키텍처도로 전체적인 시스템 구조를, 클래스도를 통해 각 패키지의 내부 객체들의 구조를 자세히 표현하였다. |
| 1 | 2022-10-16 | 아키텍처도의 각 서브시스템의 기능을 기술하였다.  주문하기 유스케이스에 대한 교류도를 작성하여 객체들의 상호작용을 나타내었다.  상태도를 통해 복잡한 객체(Cart List, Stock List)의 상태를 명확히 표현하였다. |
|
| 2 | 0 | 2022-11-19 | 아키텍처도를 수정하였다. |
|  | 1 | 2022-11-20 | 클래스도들을 수정하였다. |
|  | 2 | 2022-11-26 | 교류도와 객체상태도를 수정하였다. |
|  | 3 | 2022-11-27 | 교류도와 객체상태도를 수정하였다. |
| 3 | 0 | 2022-11-28 | Release |

1 개 요

1.1 목 적

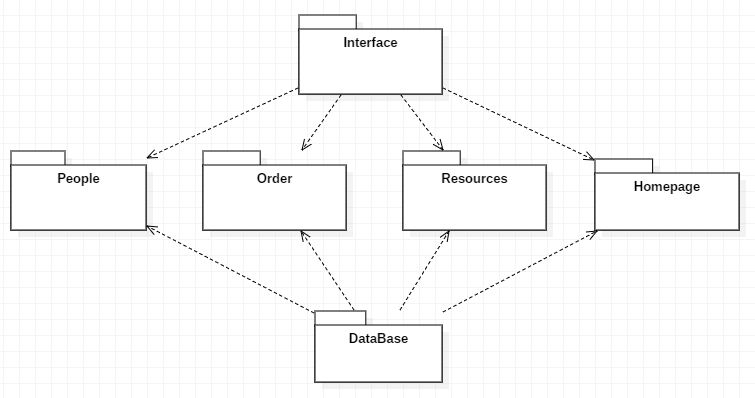
본 문서는 미스터 대박 디너 서비스 시스템의 설계 산출물을 기술한다.

* 전체 시스템이 어떤 패키지들로 구성되어 있는지 “아키텍처도”를 통해 보여준다.
* 서브 시스템이 어떤 클래스로 구성되어 있고 클래스들간에 어떤 관계가 있는지를 “클래스도”를 통해서 보여준다.
* 객체간에 발생하는 동적인 행위를 “교류도”를 통해서 보여준다.
* 한 객체의 상태 변화를 “상태도”를 통해서 보여준다.

1.2 참고 문헌

없음.

2. 아키텍처도



전체 시스템은 아래의 서브 시스템으로 구성된다.

* **Interface**

사용자(고객,직원)가 편리하게 홈페이지를 이용하도록 그래픽 인터페이스를 제공한다.

* **HomePage**

이벤트, 쿠폰, 메뉴 리스트를 인터페이스에 제공한다.

* **People**

고객, 직원의 정보 리스트를 인터페이스에 제공한다.

* **Resource**

재고 현황이 담긴 리스트를 인터페이스, 홈페이지 시스템에 제공한다.

* **Order**

주문 대기 & 승인 리스트와 장바구니 리스트, 이전 주문 목록을 Interface, People 시스템에 제공한다.

* **DataBase**

각 서브시스템에서 메소드에 의해 변경된 데이터를 저장한다.

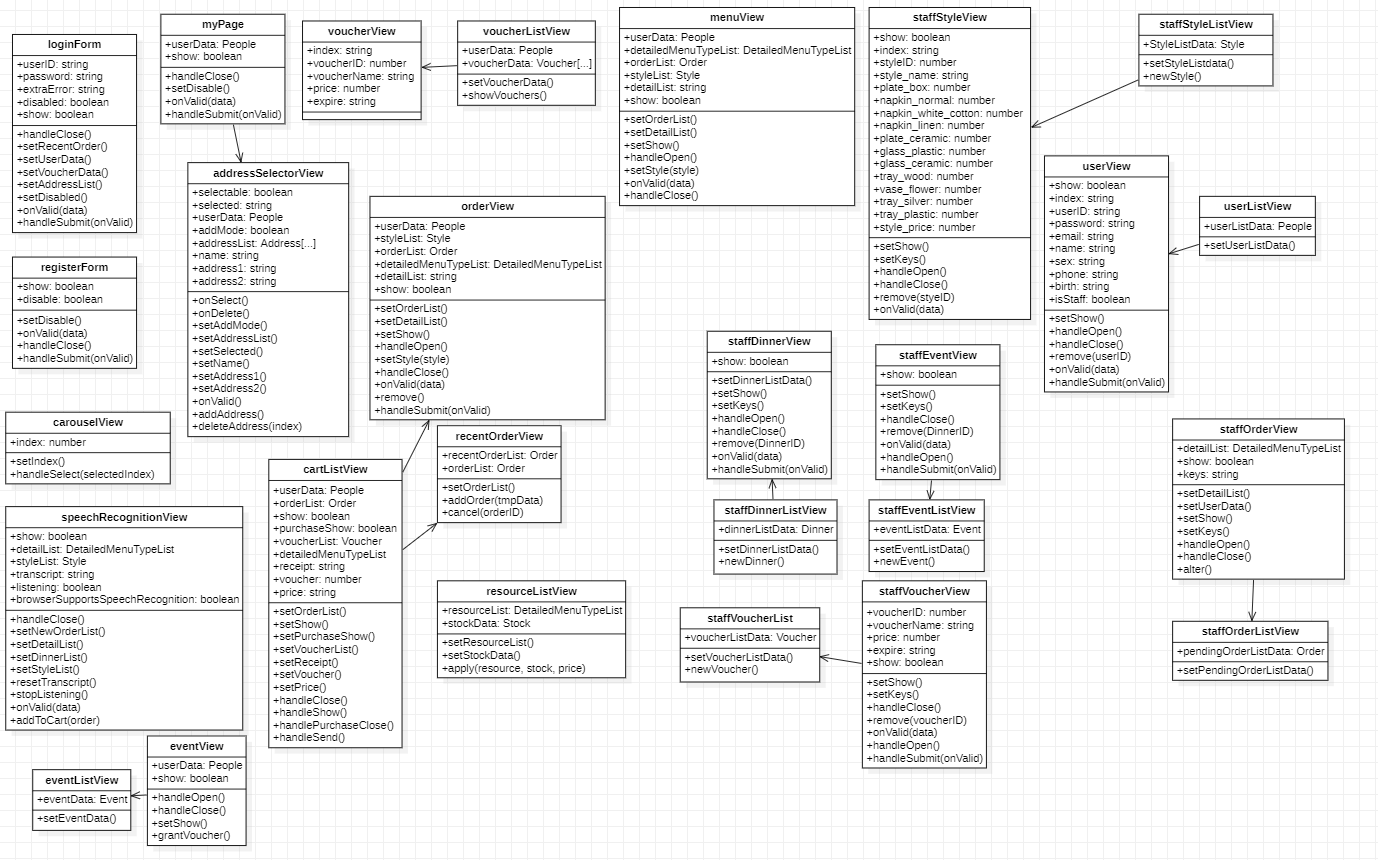
아키텍처도에서 서브 시스템간의 점선은 다음의 경우를 의미한다.

- 시작 서브 시스템의 클래스가 대상 서브 시스템의 클래스로의 의존관계를 가지는 경우 (상속이나 구현의 대상, 변수의 타입)

- 시작 서브 시스템으로부터 대상 서브 시스템으로 메시지가 전달되는 경우

3. 클래스도

3.1. Interface



**addressSelectorView**

주소를 선택하는 뷰이다.

**carouselView**

배너를 보는 뷰이다.

**eventListView**

이벤트들의 리스트를 볼 수 있는 뷰이다.

**eventView**

특정 이벤트를 선택하여 볼 수 있는 뷰이다.

**voucherListView**

상품권들의 리스트를 볼 수 있는 뷰이다.

**voucherView**

특정 상품권을 선택하여 볼 수 있는 뷰이다.

**loginForm**

로그인하기 위한 ID와 패스워드를 입력하는 폼이다. 서버에 로그인을 요청한다.

**registerForm**

회원가입을 하기 위한 정보를 입력하는 폼이다. myPage 내에서도 표시된다.

**myPageView**

개인정보 수정을 위한 뷰이다. 확인과 취소가 가능하다.

**orderView**

고객이 주문을 하는 뷰이다. 디너, 스타일을 선택 후 상세 메뉴를 선택하여 주문을 완료할 수 있다.

**cartListView**

장바구니 목록을 표시한다. 주문을 선택 후 결제가 가능하다.

**menuView**

메뉴 목록을 표시한다.

**recentOrderView**

최근 주문내역을 볼 수 있는 뷰이다.

**resourceListView**

재고를 표시한다.

**speechRecognitionView**

음성주문을 할 수 있는 폼이다.

**staffEventListView**

직원이 이벤트 리스트를 관리하는 뷰이다.

**staffEventView**

직원이 특정 이벤트를 관리하는 뷰이다.

**staffOrderListView**

직원이 주문 리스트를 관리하는 뷰이다.

**staffOrderView**

직원이 특정 주문을 관리하는 뷰이다.

**staffVoucherListView**

직원이 상품권 리스트를 관리하는 뷰이다.

**staffVoucherView**

직원이 특정 상품권을 관리하는 뷰이다.

**styleListView**

직원이 스타일 리스트를 관리하는 뷰이다.

**styleView**

직원이 특정 스타일을 관리하는 뷰이다.

**dinnerListView**

직원이 디너 리스트를 관리하는 뷰이다.

**dinnerView**

직원이 특정 디너를 관리하는 뷰이다.

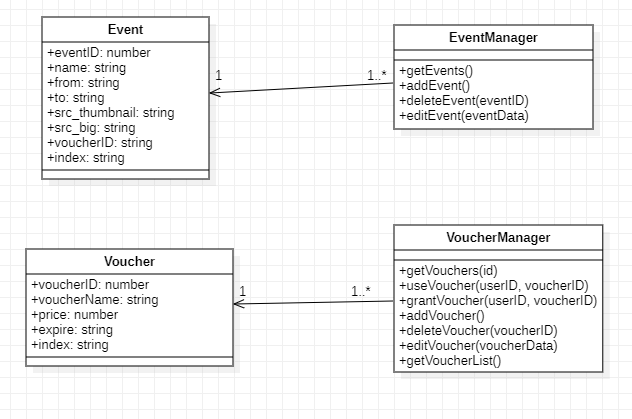
**userListView**

직원이 사용자들의 리스트를 관리하는 뷰이다.

**userView**

직원이 특정 사용자를 관리하는 뷰이다.

3.2.Homepage



**Event**

이벤트의 정보들을 저장하는 엔티티 클래스이다.

**EventManager**

이벤트를 관리하는 컨트롤 클래스이다.

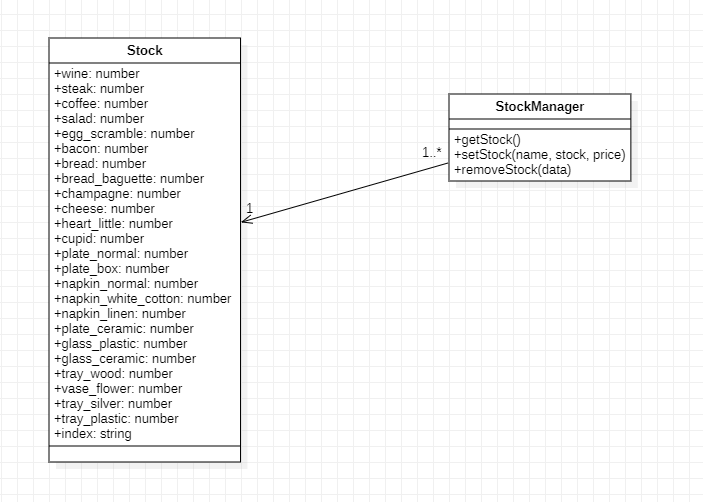
**Voucher**

상품권의 정보들을 저장하는 엔티티 클래스이다.

**VoucherManager**

상품권을 관리하는 컨트롤 클래스이다..

3.3. Resource



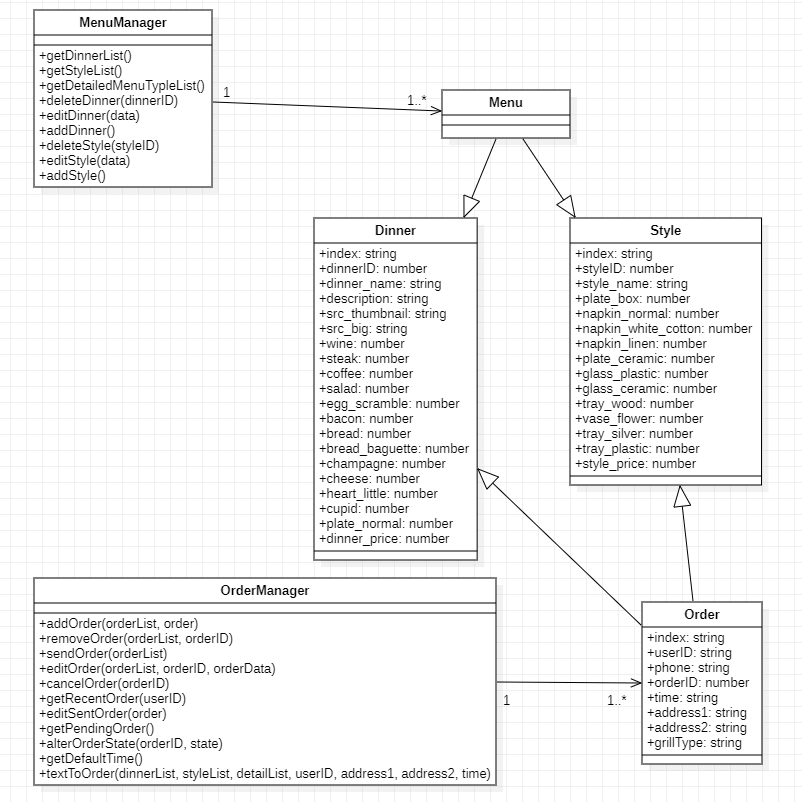
**Stock**

재고들을 저장하는 리스트이다.

**StockManager**

재고를 관리하는 컨트롤 클래스이다.

3.4 Order



**Menu**

Dinner와 Style로부터 데이터를 상속받는다.

**Dinner**

디너의 세부 내역들을 저장하는 엔티티 클래스이다.

**Style**

스타일의 세부 내역들을 저장하는 엔티티 클래스이다.

**MenuManager**

메뉴들을 관리하는 컨트롤 클래스이다.

**Order**

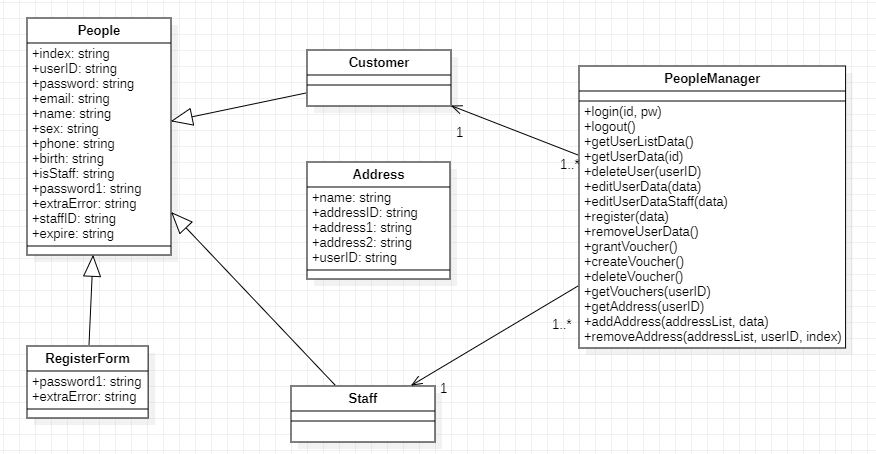
주문에 대한 정보들의 리스트를 저장하는 엔티티 클래스이다.

Dinner와 Style로부터 데이터를 상속받는다.

**OrderManager**

주문들을 관리하는 컨트롤 클래스이다.

3.5. People



**People**

id, pw, 주소, 익명 주소, 이메일, 이름, 성별, 전화번호, 생일, 직원유무를 저장한다.

**Customer**

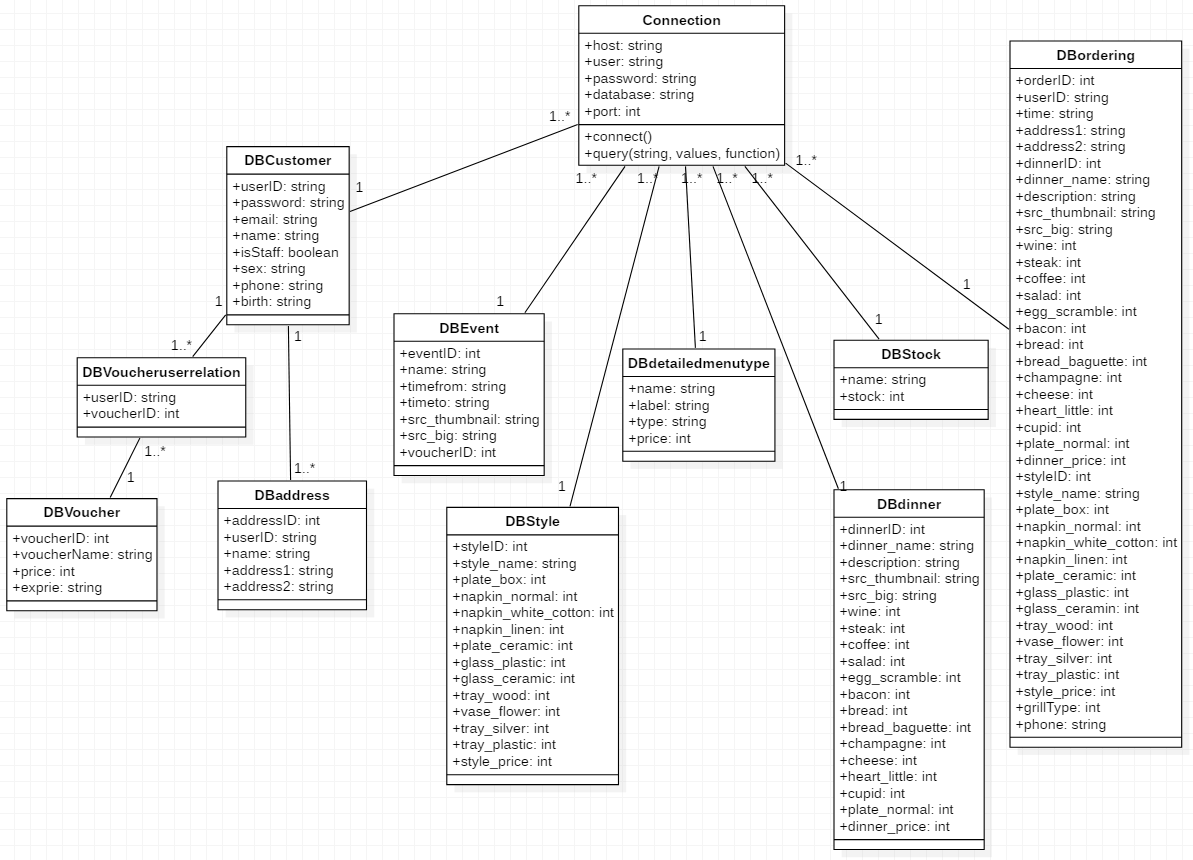
People로부터 데이터를 상속받는다.

**Staff**

People로부터 데이터를 상속받는다.

**PeopleManager**

사용자가 로그인, 로그아웃, 개인정보 수정 등을 할 수 있다.

3.6 DB

DBcustomer: 회원가입, 로그인 등 유스케이스를 시행했을 때 정보를 저장한다.

DBvoucheruserrelation: 상품권 발급/사용 시에 상품권-유저의 관계를 저장한다.

DBvoucher: 상품권 추가/삭제 시에 상품권의 정보를 저장한다.

DBaddress: 주소를 생성할 때 주소-유저의 관계를 저장한다.

DBevent: 이벤트 추가/삭제 시에 이벤트의 정보를 저장한다.

DBstyle: 스타일 추가/삭제/수정 시에 스타일의 정보를 저장한다.

DBdetailedmenutype: 상세 메뉴 리스트 정보를 저장한다.

DBdinner: 디너 추가/삭제/수정 시에 디너의 정보를 저장한다.

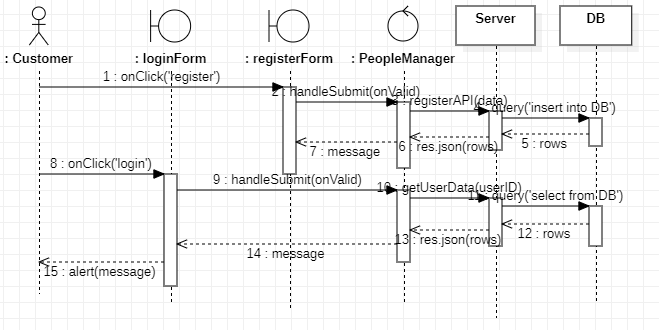
DBordering: 주문하기, 주문 수락&취소 시에 변경된 정보를 저장한다.

DBstock: 재고 조회, 수정 시에 변경된 정보를 저장한다.

Connection: DB에서 SQL을 보내 정보를 서버에 제공한다.

4. 교류도( usecase : Input Experiment Plan)

4.1.Sign up & Sign in



**Sign up:** 사용자가 registerForm에 작성한 내용을 PeopleManager가 Server에게 보내주고 이를 DB에 저장하고 사용자에게 이를 알려준다.

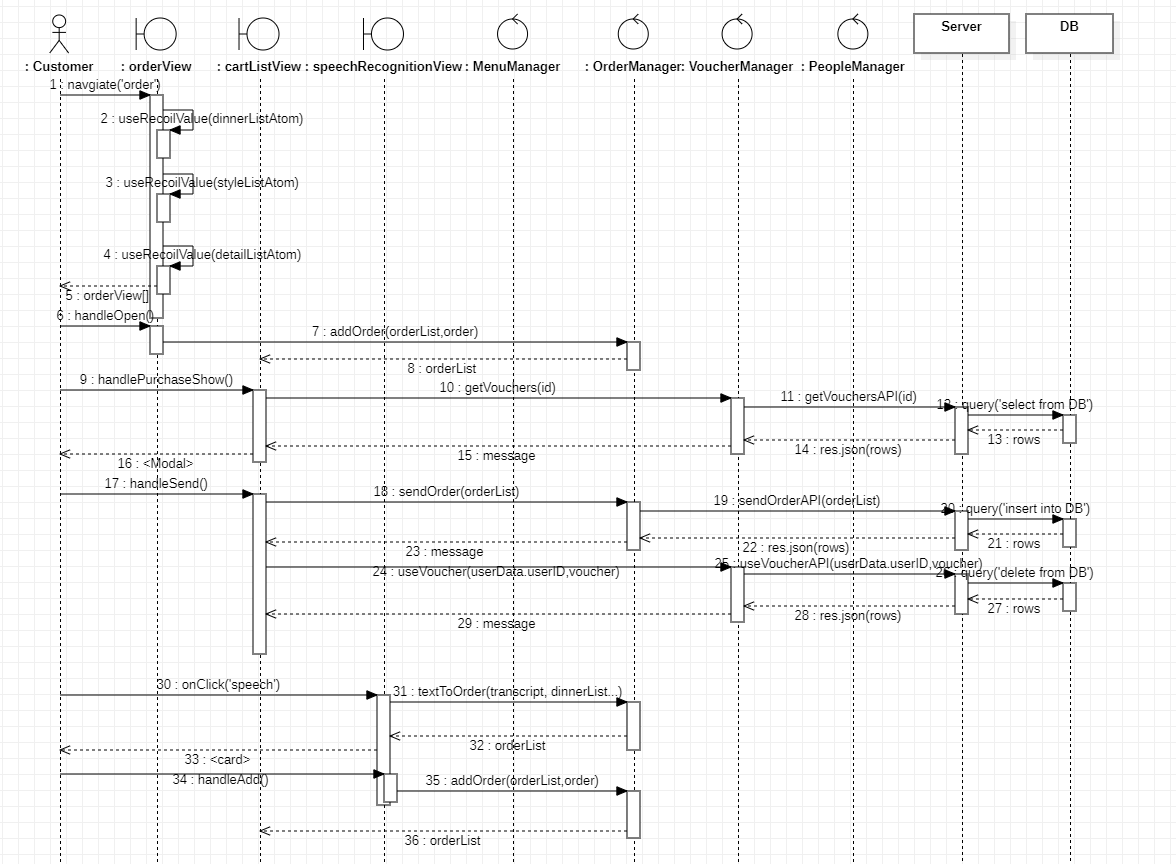
**Sign in :** Sign up할 때 사용한 id,pw를 loginForm에 작성하고 이를 PeopleManager가 Server에 전송 후 DB의 내용과 비교하여 그 결과를 알려준다.

**2.** **handleSubmit(onValid) :** registerForm에 작성한 내용을 PeopleManager에게 전송한다.

**4.** **query(‘insert into DB’) :** 새로운 정보를 DB에 저장한다.

**10.** **getUserData(userID) :** 사용자의 ID를 이용하여 DB로부터 user의 id와 pw를 가져온다.

4.2. Place an Order



사용자가 주문하기 버튼을 클릭하여 orderView로 이동하면 사용자는 orderView를 통해 로그인할 때 가져온 dinner, style, detailList에 대한 정보를 볼 수 있다.

사용자가 디너, 스타일, 상세 주문을 모두 작성하면 OrderManager를 통해 cartList에 order를 추가한다.

사용자가 handlePurchaseShow()버튼을 클릭하면 cartListView에서는 VoucherManager를 통해 상품권의 정보를 가져오고, 결제창을 사용자에게 보여준다.

사용자가 handleSend() 버튼을 클릭하면 cartList의 주문을 전송하고 상품권을 사용하여 결제하고 이를 사용자에게 알려준다.

음성주문

사용자가 마이크 버튼을 클릭하여 speechRecognitionView로 이동하면 사용자는 OrderManager의 textToOrder함수를 통하여 음성주문을 할 수 있다.

음성주문의 결과는 speechRecognitionView에 표현되고 사용자가 이를 주문에 추가하길 원하면 orderManager의 addOrder를 통하여 주문을 장바구니 추가할 수 있다.

1.navigate(‘order’) : 주문하기 버튼을 클릭하여 orderView로 이동한다.

2~4 : dinnerList, styleList, detailList를 받아온다.

7. addOrder(orderList,order) : 사용자가 OrderManager를 통하여 장바구니에 주문을 추가한다.

9. handlePurchaseShow() : 사용자가 결제하기 버튼을 클릭한다.

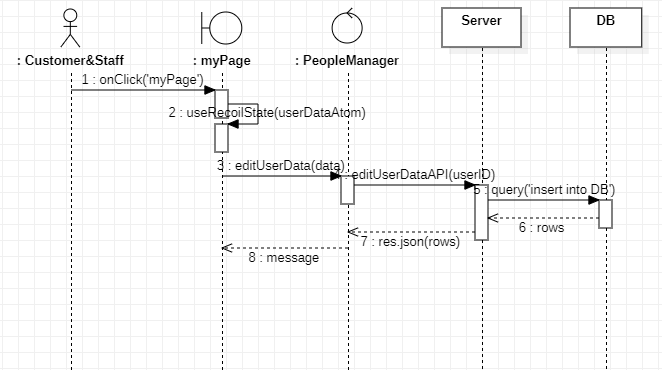
10. getVouchers(id) : 상품권의 id를 통하여 상품권에 대한 정보를 가져온다.

30. onClick(‘speech’) : 마이크 버튼을 클릭하여 speechRecognitionView로 이동한다.

31. textToOrder(transcript, dinnerList…) : OrderManager의 textToOrder함수를 이용하여 음성주문한다.

34. addOrder(orderList,order) : 음성으로 주문한 내역들을 OrderManager를 통하여 장바구니에 추가한다.

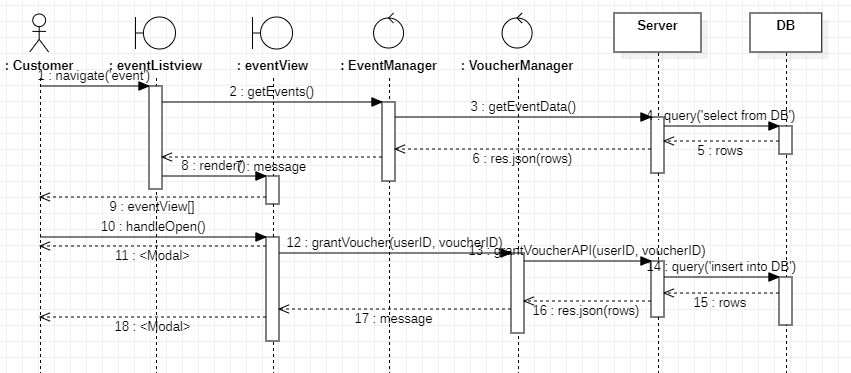
4.3. Check & Modify User Status



사용자가 마이페이지 버튼을 myPage에 접속하면 로그인 할 때 받은 정보를 통해 마이페이지의 정보를 표현하고 정보를 수정할 경우 이를 server를 통해 DB에 수정된 내용을 저장한다.

2. useRecoilState(userDataAtom) : 로그인할 때 가져온 사용자의 정보를 저장하는 userDataAtom을 useRecoilState를 통하여 마이페이지 폼에 보여준다.

5. query(‘insert into customer’) : 수정된 개인 정보를 DB에 추가한다.

4.4. Look up Events**

고객이 이벤트 버튼을 통해 eventListView에 접속하면 EventManager를 통해서 서버로부터 Event들의 data를 가져오고 이를 render해서 여러 개의 eventView로 만들어준다.

사용자가 handleOpen을 통해 특정 event를 선택하면 해당 event의 정보를 grantVoucher를 통해 가져오고 이를 사용자에게 보여준다.

1. navigate(‘event’) : 고객이 이벤트 버튼을 클릭하여 eventListView에 접속한다.

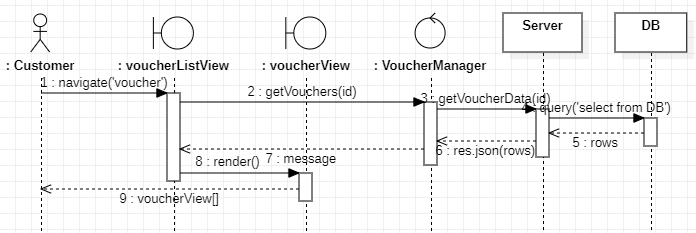
8. render() : 가져온 eventList를 eventView로 render해준다.

10. handleOpen() : 사용자가 특정 이벤트를 선택한다.

12. grantVoucher(userID, voucherID) : 이벤트에 당첨되었을 경우 상품권을 부여한다.

14. query(‘insert into DB’) : 새로운 상품권에 대한 정보를 DB에 추가한다.

4.5. View Vouchers



고객이 상품권 버튼을 클릭하여 voucherListView에 접속하면 VoucherManager를 통해 상품권들에 대한 정보를 받아오고 이를 render하여 사용자에게 voucherView를 통해 보여준다.

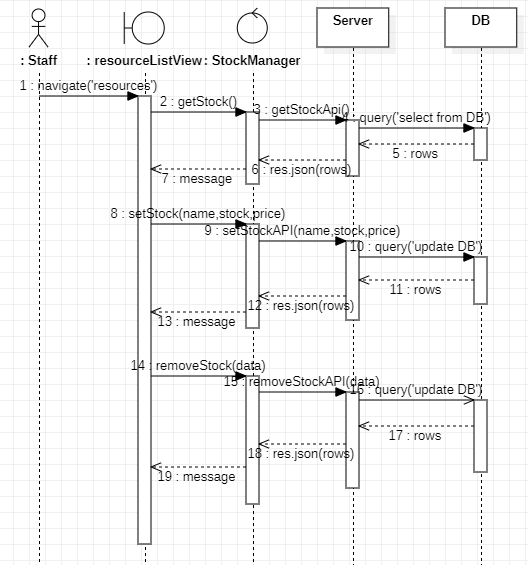
1. navigate(‘voucher’) : 사용자가 상품권 버튼을 클릭하여 voucherListView에 접속한다.

2. getVouchers(id) : 상품권들에 대한 정보를 서버에 요청한다.

4. query(‘select form DB’) : DB로부터 상품권들에 대한 정보를 가져온다.

8. render() : 가져온 상품권들에 대한 정보를 render 한다.

4.6 Manage Resources



직원이 자원 관리 버튼을 클릭하여 resourceListView에 접속하면 StockManager를 통해 재고에 대한 정보를 받아와서 사용자에게 보여준다.

직원이 재고를 수정하면 수정된 재고에 대한 정보를 DB에 저장한다.

직원이 재고를 삭제하면 삭제된 재고에 대한 정보를 DB에서 삭제한다.

1. navigate(‘resources’) : 직원이 자원 관리 버튼을 클릭하여 resourceListView에 접속한다.

2. getStock() : 재고에 대한 정보를 서버에 요청한다.

4. query(‘select form DB’) : DB로부터 재고에 대한 정보를 가져온다.

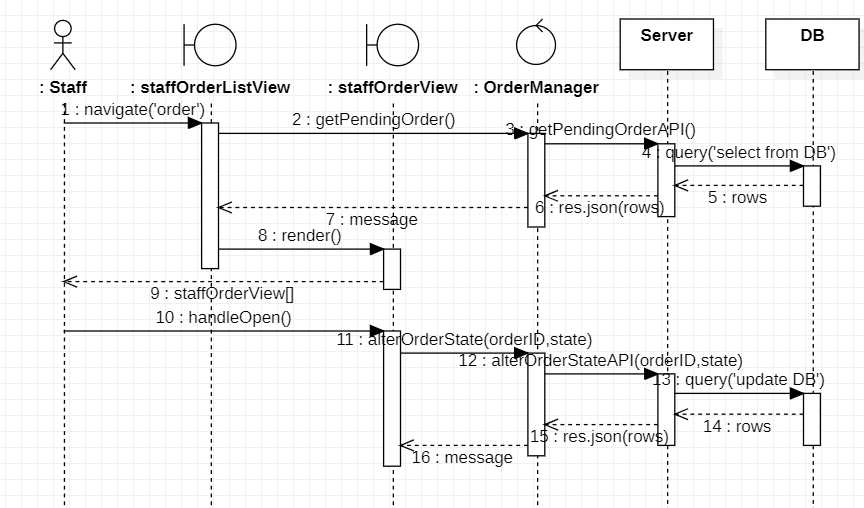
8. setStock(name,stock,price) : 재고에 대한 정보를 변경한다.

10. query(‘update DB’) : 수정된 정보를 토대로 DB를 update한다.

14. removeStock(data) : 재고를 제거한다.

15. query(‘update DB’) : 수정된 정보를 토대로 DB를 update한다.

4.7 Check Orders



직원이 주문 관리 버튼을 클릭하여 staffOrderView로 이동하면 OrderManager를 통하여 주문 리스트를 가져오고 이를 render하여 직원이 주문을 선택할 수 있다.

특정 주문을 선택하여 주문의 상태를 바꾸면 바뀐 주문의 상태를 OrderManager를 통하여 DB에 변경하고 직원에게 변경됐음을 알려준다.

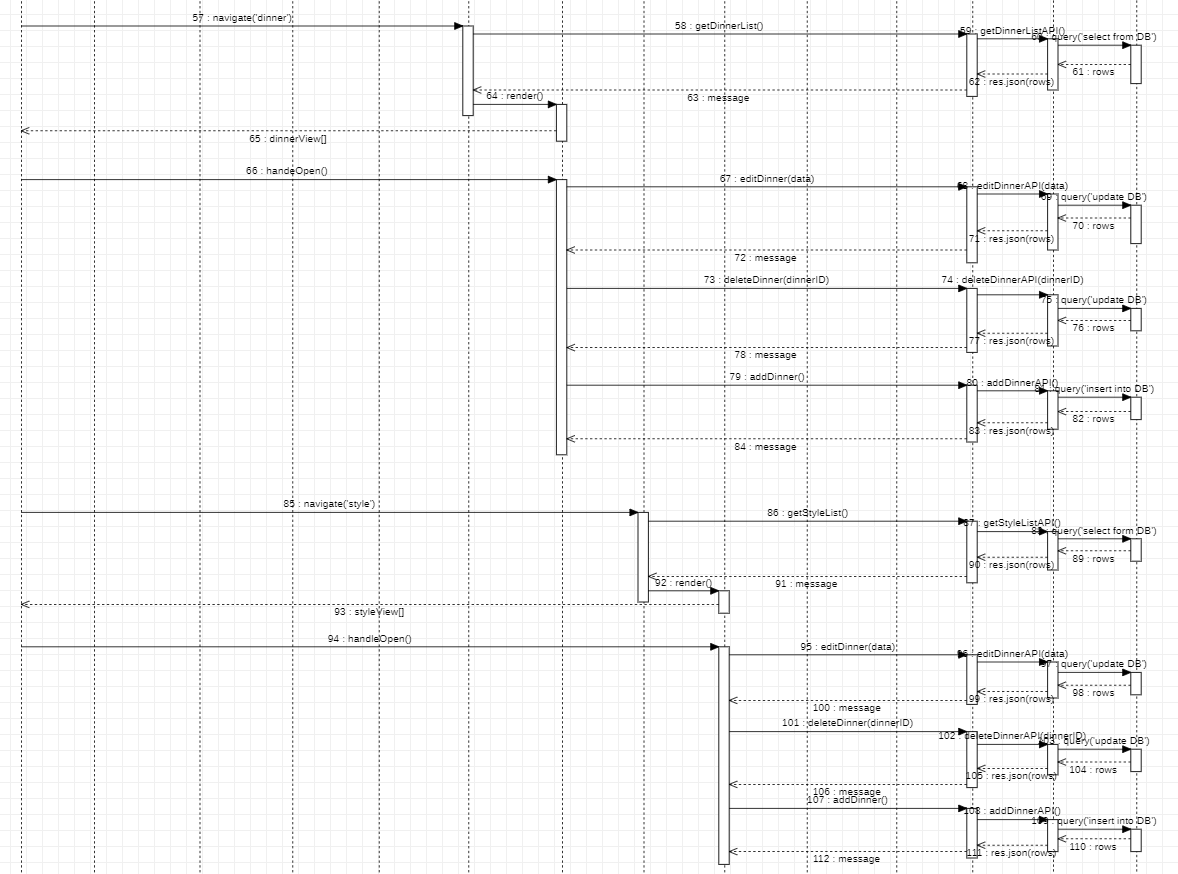
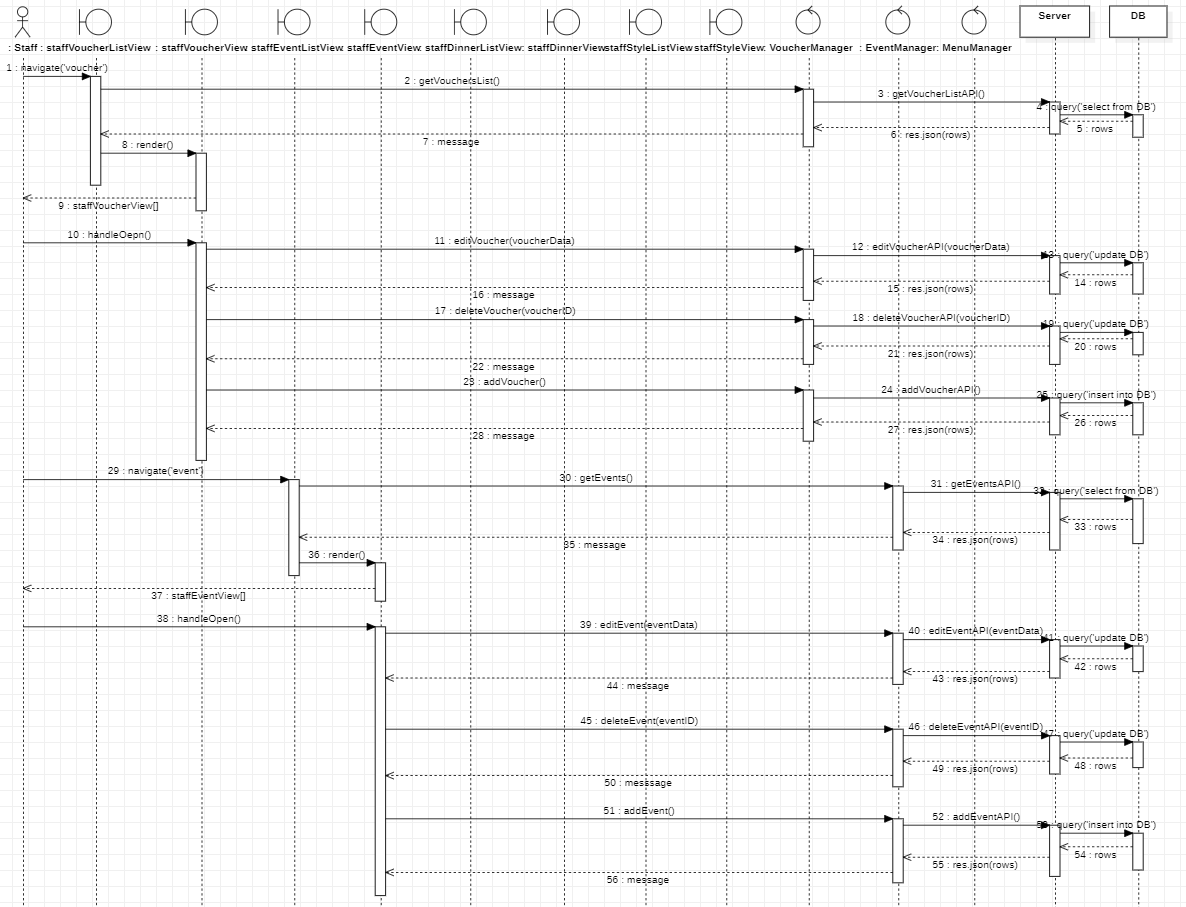
1. navigate(‘order’) : 직원이 주문 관리 버튼을 클릭해 staffOrderView로 이동한다.
2. getPendingOrder() : OrderManager를 통해 주문 리스트를 가져온다.

8. render : 가져온 주문 리스트를 render한다.

11. alterOrderState(orderID,state) : orderID와 일치하는 주문의 상태를 state로 바꾼다.

13. query(‘update DB’) : 바뀐 주문의 상태를 DB에 update한다.

4.8 Manage Website

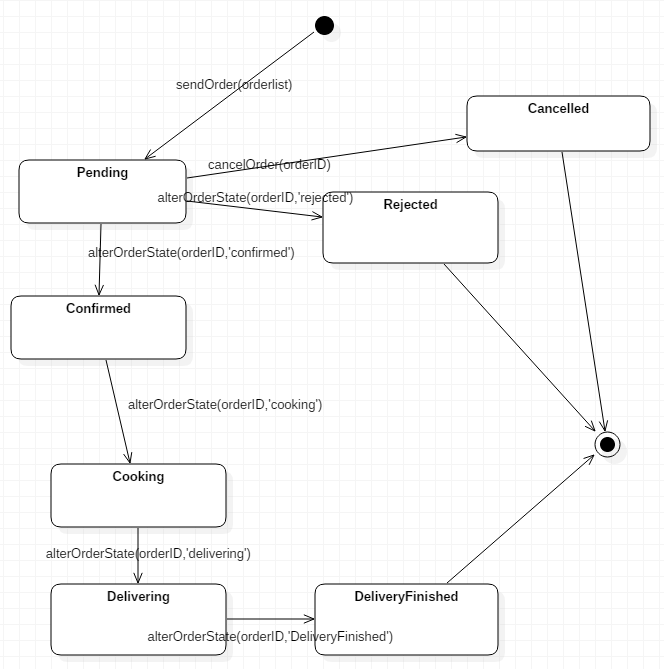


직원은 Dinner, Style, Voucher, Event를 수정, 추가, 삭제할 수 있다.

각각 menuManager, styleManager, VoucherManager, EventManager를 통해 수정, 추가, 삭제를 할 수 있다.

5. 객체 상태도

5.1. OrderManager 객체 상태도



**Pending**

직원의 승인을 기다리는 상태이다. 재고가 없어서 직원이 주문을 거절하면 Rejected 상태가 되고 주문을 승인하면 Confirmed 상태가 된다.

**Confirmed**

직원이 주문을 승인한 상태이다.

**Cancelled**

고객이 주문을 취소한 상태이다.

**Rejected**

직원이 주문을 거절한 상태이다.

**Cooking**

주문이 승인되고 요리사가 요리 중인 상태이다.

**Delivering**

완성된 음식을 배달 중인 상태이다.

**DeliveryFinished**

배달이 완료된 상태이다. 완료된 주문은 주문 내역에 저장된다.