

# Design Specification Document



과목명	종합설계과제
담당교수	손영호 교수님
제출마감일	2022.05.17
팀장	김준서 / 21720831
팀원 1	김현욱 / 21720883
팀원 2	여민수 / 21720890
팀원 3	이승민 / 21720843

## <제목 차례>

1. Introduction .....	11
2. Use Case Diagram .....	12
2.1. 회원가입 .....	13
2.2. 로그인 .....	15
2.3. 로그아웃 .....	17
2.4. 아이디 찾기 .....	18
2.5. 비밀번호 찾기 .....	20
2.6. 회원 탈퇴 .....	22
2.7. 매장 등록 .....	24
2.8. 매장 삭제 .....	26
2.9. 매장 리스트 .....	28
2.10. QR 코드를 통한 도장 적립 .....	29
2.11. 전화번호를 통한 도장 적립 .....	31
2.12. 이벤트 등록 .....	33
2.13. 이벤트 삭제 .....	34
2.14. 리워드 등록 .....	35
2.15. 리워드 삭제 .....	36
2.16. 비밀번호 변경 .....	37
2.17. 전화번호 변경 .....	39
2.18. 출석체크 .....	41
2.19. QR 코드를 통한 쿠폰 사용 .....	42
2.20. 쿠폰 리스트 .....	44
2.21. 쿠폰 삭제 .....	45
2.22. 이벤트 상세 보기 .....	46
2.23. 쿠폰 판 스킨 구매 .....	47
2.24. 도장 스킨 구매 .....	48

2.25. 쿠폰 판 스킨 적용 .....	50
2.26. 도장 스킨 적용 .....	51
3. E-R Diagram .....	52
4. Class Diagram .....	53
4.1. Privacy .....	54
4.2. Customer .....	55
4.3. CustomController .....	56
4.4. CustomerRepository .....	57
4.5. CustomerService .....	58
4.6. Coupon .....	59
4.7. CouponController .....	60
4.8. CouponRepository .....	61
4.9. CouponService .....	62
4.10. StampCustomer .....	63
4.11. StampCustomerController .....	64
4.12. StampCustomerService .....	65
4.13. StampCustomerRepository .....	66
4.14. Stamp .....	67
4.15. StampController .....	68
4.16. StampRepository .....	69
4.17. StampService .....	70
4.18. Owner .....	71
4.19. OwnerController .....	72
4.20. OwnerService .....	73
4.21. OwnerRepository .....	74
4.22. Market .....	75
4.23. MarketController .....	76
4.24. MarketService .....	77

4.25. MarketRepository .....	78
4.26. Reward .....	79
4.27. RewardController .....	80
4.28. RewardService .....	81
4.29. RewardRepository .....	82
4.30. BoardCustomer .....	83
4.31. BoardCustomerController .....	84
4.32. BoardCustomerService .....	85
4.33. BoardCustomerRepository .....	86
4.34. Board .....	87
4.35. BoardController .....	88
4.36. BoardService .....	89
4.37. BoardRepository .....	90
5. Sequence Diagram .....	91
5.1. 회원 .....	91
5.1.1. 회원가입 .....	91
5.1.2. 로그인 .....	92
5.1.3. 로그아웃 .....	93
5.1.4. 아이디 찾기 .....	94
5.1.5. 비밀번호 찾기 .....	95
5.1.6. 회원 정보 수정 .....	96
5.1.7. 메인화면 .....	97
5.1.8. 쿠폰 적립 QR 생성 .....	98
5.1.9. 쿠폰 사용 QR 생성 .....	99
5.1.10. 쿠폰 판 커스터마이징 리스트 .....	100
5.1.11. 도장 커스터마이징 리스트 .....	101
5.1.12. 쿠폰 판 변경 .....	102
5.1.13. 도장 변경 .....	103

5.1.14. 쿠폰 리스트 .....	104
5.2. 접주 .....	105
5.2.1. 회원가입 .....	105
5.2.2. 로그인 .....	106
5.2.3. 로그아웃 .....	107
5.2.4. 아이디 찾기 .....	108
5.2.5. 비밀번호 찾기 .....	109
5.2.6. 회원탈퇴 .....	110
5.2.7. 매장 등록 .....	111
5.2.8. 매장 삭제 .....	112
5.2.9. 매장 리스트 .....	113
5.2.10. QR 코드를 통한 도장 적립 .....	114
5.2.11. 전화번호를 통한 도장 적립 .....	115
5.2.12. 이벤트 등록 .....	116
5.2.13. 이벤트 삭제 .....	117
5.2.14. 리워드 추가 .....	118
5.2.15. 리워드 삭제 .....	119
5.2.16. 접주 정보 수정 .....	120
6. State machine Diagram .....	121
7. Mock Up .....	122
7.1. 공통 .....	122
7.1.1. 스플래시 스크린 .....	122
7.1.2. 시작 화면 .....	123
7.2. 회원 .....	124
7.2.1. 로그인 .....	124
7.2.2. 회원가입 .....	125
7.2.3. 메인 화면 .....	126
7.2.4. 쿠폰 리스트 .....	127

7.2.5. 쿠폰 커스터마이징	128
7.2.6. 쿠폰 적립	129
7.2.7. 쿠폰 사용	130
7.2.8. 더보기	131
7.2.9. 포인트 사용 내역	132
7.2.10. 포인트 상점	133
7.2.11. 회원정보 수정	134
7.2.12. 쿠폰 적립 및 사용 내역	135
7.3. 점주	136
7.3.1. 로그인 화면	136
7.3.2. 회원가입 화면	137
7.3.3. 메인 화면	138
7.3.4. 쿠폰 적립/사용 화면	139
7.3.5. 매장 관리 화면	140
7.3.6. 매장 분석 화면	141
7.3.7. 매장 정보 수정 화면	142
7.3.8. 리워드 관리 화면	143
7.3.9. 리워드 등록 화면	144
7.3.10. 리워드 삭제 화면	145
7.3.11. 이벤트 관리 화면	146
7.3.12. 이벤트 등록 화면	147
7.3.13. 이벤트 삭제 화면	148
7.3.14. 쿠폰 적립/사용 내역 화면	149
7.3.15. 더보기 화면	150
7.3.16. 회원정보 수정 화면	151
8. Implementation Requirements	152
8.1. Frontend	152
8.2. Backend & DataBase	152

8.3. Configuration Management .....	152
9. Glossary .....	153
10. Reference .....	154

## <그림 차례>

그림 1 Use Case Diagram .....	12
그림 2 E-R Diagram .....	52
그림 3 Class Diagram .....	53
그림 4 Privacy 클래스 .....	54
그림 5 Customer 클래스 .....	55
그림 6 CustomerController 클래스 .....	56
그림 7 CustomerRepository 클래스 .....	57
그림 8 CustomerService 클래스 .....	58
그림 9 Coupon 클래스 .....	59
그림 10 CouponController 클래스 .....	60
그림 11 CouponRepository 클래스 .....	61
그림 12 CouponService 클래스 .....	62
그림 13 StampCustomer 클래스 .....	63
그림 14 StampCustomerController 클래스 .....	64
그림 15 StampCustomerService 클래스 .....	65
그림 16 StampCustomerRepository 클래스 .....	66
그림 17 Stamp 클래스 .....	67
그림 18 StampController 클래스 .....	68
그림 19 StampRepository 클래스 .....	69
그림 20 StampService 클래스 .....	70
그림 21 Owner 클래스 .....	71
그림 22 OwnerController 클래스 .....	72
그림 23 OwnerService 클래스 .....	73

그림 24 OwnerRepository 클래스	74
그림 25 Market 클래스	75
그림 26 MarketController 클래스	76
그림 27 MarketService 클래스	77
그림 28 MarketRepository 클래스	78
그림 29 Reward 클래스	79
그림 30 RewardController 클래스	80
그림 31 RewardService 클래스	81
그림 32 RewardRepository 클래스	82
그림 33 BoardCustomer 클래스	83
그림 34 BoardCustomerController 클래스	84
그림 35 BoardCustomerService 클래스	85
그림 36 BoardCustomerRepository 클래스	86
그림 37 Board 클래스	87
그림 38 BoardController 클래스	88
그림 39 BoardService 클래스	89
그림 40 BoardRepository 클래스	90
그림 41 회원가입(회원) sequence diagram	91
그림 42 로그인(회원) sequence diagram	92
그림 43 로그아웃(회원) sequence diagram	93
그림 44 아이디 찾기(회원) sequence diagram	94
그림 45 비밀번호 찾기(회원) sequence diagram	95
그림 46 회원 정보 수정(회원) sequence diagram	96
그림 47 메인화면(회원) sequence diagram	97
그림 48 쿠폰 적립 QR 생성(회원) sequence diagram	98
그림 49 쿠폰 사용 QR 생성(회원) sequence diagram	99
그림 50 쿠폰 판 커스터마이징 리스트(회원) sequence diagram	100
그림 51 도장 커스터마이징 리스트(회원) sequence diagram	101

그림 52 쿠폰 판 변경(회원) sequence diagram .....	102
그림 53 도장 변경(회원) sequence diagram .....	103
그림 54 쿠폰 리스트(회원) sequence diagram .....	104
그림 55 회원가입(점주) sequence diagram .....	105
그림 56 로그인(점주) sequence diagram .....	106
그림 57 로그아웃(점주) sequence diagram .....	107
그림 58 아이디 찾기(점주) sequence diagram .....	108
그림 59 비밀번호 찾기(점주) sequence diagram .....	109
그림 60 회원탈퇴(점주) sequence diagram .....	110
그림 61 매장 등록 sequence diagram .....	111
그림 62 매장 삭제 sequence diagram .....	112
그림 63 매장 리스트 sequence diagram .....	113
그림 64 QR 코드를 통한 도장 적립 sequence diagram .....	114
그림 65 전화번호를 통한 도장 적립 sequence diagram .....	115
그림 66 이벤트 등록 sequence diagram .....	116
그림 67 이벤트 삭제 sequence diagram .....	117
그림 68 리워드 추가 sequence diagram .....	118
그림 69 리워드 삭제 sequence diagram .....	119
그림 70 점주 정보 수정 sequence diagram .....	120
그림 71 State Machine Diagram .....	121
그림 72 스플래시 스크린 .....	122
그림 73 시작 화면 .....	123
그림 74 로그인(회원) 화면 .....	124
그림 75 회원가입(회원) 화면 .....	125
그림 76 메인(회원) 화면 .....	126
그림 77 쿠폰리스트 화면 .....	127
그림 78 쿠폰 커스터마이징 화면 .....	128
그림 79 쿠폰 적립(회원) 화면 .....	129

그림 80 쿠폰 사용(회원) 화면	130
그림 81 더보기(회원) 화면	131
그림 82 포인트 사용 내역(회원) 화면	132
그림 83 포인트 상점 화면	133
그림 84 회원정보 수정(회원) 화면	134
그림 85 쿠폰 적립 및 사용(회원) 내역 화면	135
그림 86 로그인(점주) 화면	136
그림 87 회원가입(점주) 화면	137
그림 88 메인(점주) 화면	138
그림 89 쿠폰 적립/사용(점주) 화면	139
그림 90 매장 관리 화면	140
그림 91 매장 분석 화면	141
그림 92 매장 정보 수정 화면	142
그림 93 리워드 관리 화면	143
그림 94 리워드 등록 화면	144
그림 95 리워드 삭제 화면	145
그림 96 이벤트 관리 화면	146
그림 97 이벤트 등록 화면	147
그림 98 이벤트 삭제 화면	148
그림 99 쿠폰 적립/사용 내역 화면	149
그림 100 더보기(점주) 화면	150
그림 101 회원 정보 수정(점주) 화면	151

# 1. Introduction

학교 앞에서 커피를 사 먹거나, 미용실에서 머리를 하면 매장에서 쿠폰에 도장을 찍어주곤 한다. 실물 쿠폰의 도장을 모아 보상을 받기 위해선 항상 쿠폰을 들고 다녀야 하는 불편함이 존재한다. 이러한 점을 보완하기 위해 대기업에서는 스마트폰을 사용해 쿠폰들을 저장 및 관리할 수 있는 서비스를 제공한다. 그러나 동네 카페, 미용실, 배이커리 등의 소상공인들은 이러한 혜택을 누릴 수 없는 불편함이 있다. 또한, 실물 쿠폰을 사용하게 되면 제작하기 위한 비용과 사용하고 난 뒤 쿠폰을 처리할 때 환경적인 문제점이 발생한다. 이에 환경적, 경제적인 문제점을 해소하기 위한 서비스의 필요성을 느끼고 다음과 같은 서비스를 개발하여 사용자의 불편을 개선하고자 한다.

MOCUMOCU은 애플리케이션을 통해 실물 쿠폰을 디지털화함으로써 소상공인이 쿠폰을 더 체계적으로 관리하고 실물 쿠폰 제작으로 인해 발생하는 경제적인 부담을 줄이고자 한다. 소비자는 쿠폰 적립과 사용을 애플리케이션을 통해 실물 쿠폰보다 더욱 편리하게 사용하는 것을 지향한다. 애플리케이션의 주 사용자인 점주와 소비자가 직관적인 UI를 통해 간단히 이용할 수 있게 해야 한다.

본 문서는 실제 구현에 관한 세부적 사항을 명시하고, 이전의 단계들 보다 구체적으로 시스템의 구성에 대해 설명한다. 이를 위해 다음의 내용들이 본 문서에서 기술되어진다.

- Use Case Diagram
  - Use Case Diagram 과 각각의 Use Case Diagram 에 대한 상세 기술
- Class Diagram
  - Class Diagram 과, Class Diagram 들의 attribute, method 들의 상세 기술
- E-R Diagram
  - E-R Diagram 과, 각각의 E-R Diagram 에 대한 상세 기술
- Sequence Diagram
  - Sequence Diagram 과, 각각의 Sequence Diagram 에 대한 상세 기술
- State Machine Diagram
  - State Machine Diagram 과, State Machine Daigram 에 대한 상세 기술
- Mock Up
  - Mock Up 이미지와, Mock Up 에 대한 상세 기술
- Implementation Requirements
  - System 구현을 위한, S/W 에 대한 기술

## 2. Use Case Diagram

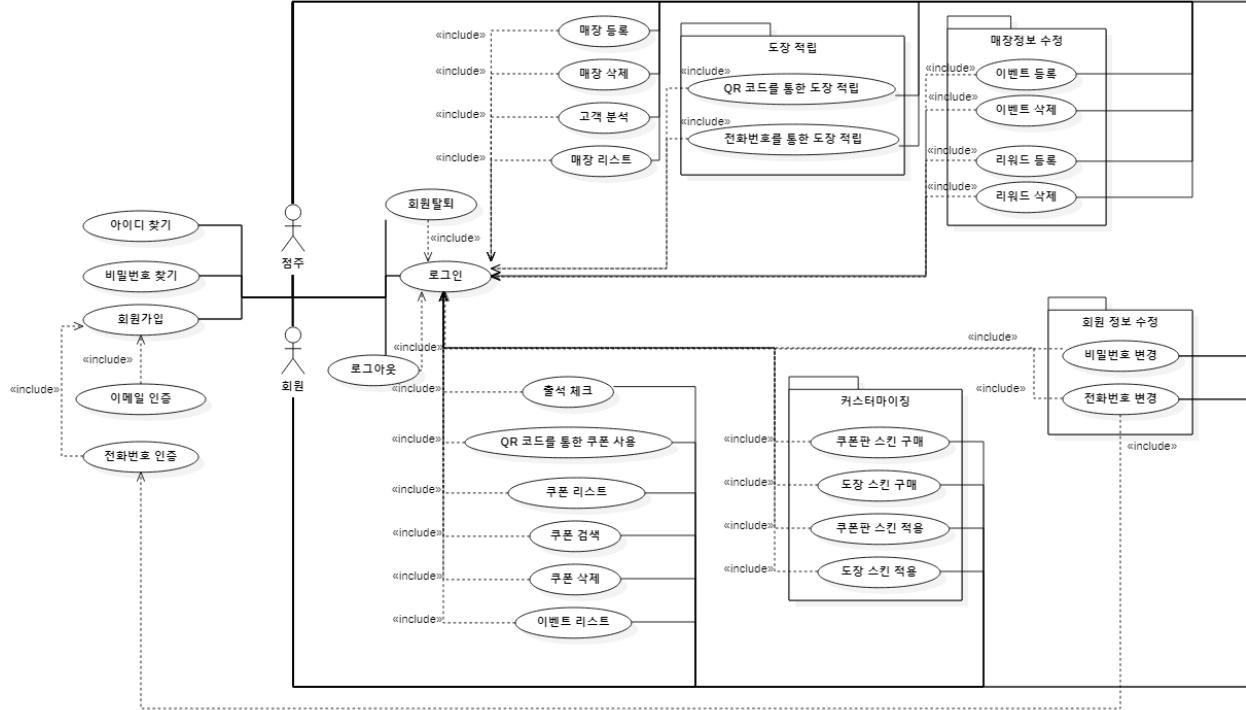


그림 1 Use Case Diagram

위 그림은 MOCUMOCU의 Use Case Diagram이다. Actor는 점주와 회원 두 가지 타입으로 분류되어 있다. 공통 기능으로 로그인, 로그아웃, 아이디 찾기, 비밀번호 찾기, 회원가입, 회원 탈퇴, 비밀번호 변경, 전화번호 변경이 있다. 점주의 기능으로는 매장 등록, 매장 삭제, 고객 분석, 매장 리스트, QR 코드를 통한 도장 적립, 전화번호를 통한 도장 적립, 이벤트 등록, 이벤트 삭제, 리워드 등록, 리워드 삭제가 있으며, 회원은 출석체크, QR 코드를 통한 쿠폰 사용 쿠폰 리스트, 쿠폰 검색, 쿠폰 삭제, 이벤트 리스트, 쿠폰 판 스킨 구매, 도장 스킨 구매, 쿠폰 판 스킨 적용, 도장 스킨 적용 기능이 있다.

## 2.1. 회원가입

Use case #1 : 회원가입	
GENERAL CHARACTERISTICS	
Summary	등록되지 않은 사용자가 시스템을 사용하기 위한 계정을 만들 때 사용한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	로그인 되어있지 않은 상태여야 한다.
Trigger	로그인 화면에서 회원가입 버튼을 클릭한다.
Success Post Condition	회원가입을 위해 정보를 입력하는 화면을 출력한다.
Failed Post Condition	회원가입 정보를 입력하는 화면이 출력되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자는 시작화면에서 점주로 시작하기 혹은 회원으로 시작하기 버튼을 클릭한다.
1	애플리케이션은 이용약관 및 개인정보 수집 동의 화면을 출력한다.
2	사용자는 모두 동의하고 다음으로 버튼을 클릭한다.
3	애플리케이션은 회원가입을 위해 정보를 기입하는 화면을 출력한다.
4	사용자는 정보(이름, 이메일 비밀번호, 비밀번호 확인, 성별(회원만), 생년월일(회원만)) 입력한다.
5	사용자는 이메일 인증 버튼을 클릭한다.
6	애플리케이션은 받은 이메일에 본인 인증을 하는 버튼을 클릭한다.
7	사용자는 전화번호 인증 버튼을 클릭한다.
8	시스템은 입력된 전화번호로 인증 코드를 보낸다.
9	사용자는 받은 인증 코드를 입력한다.
10	사용자는 회원가입 버튼을 클릭한다.
11	애플리케이션은 회원가입이 완료되었다는 알림 창을 출력한다.
12	로그인 창으로 돌아간다.

EXTENSION SCENARIOS	
Step	Branching Action
10	<p>10a. 이메일 형식에 맞지 않는 경우            10a.1 이메일 형식이 맞지 않다는 알림 창을 출력한다.            10a.2 회원가입 화면으로 돌아간다.</p> <p>10b. 비밀번호가 일치하지 않는 경우            10b.1 비밀번호가 일치하지 않는다는 알림 창을 출력한다.            10b.2 회원가입 화면으로 돌아간다.</p> <p>10c. 비밀번호 형식이 맞지 않는 경우            10c.1 비밀번호 형식이 맞지 않다는 알림 창을 출력한다.            10c.2 회원가입 화면으로 돌아간다.</p>

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	최초 애플리케이션 실행 시 한 번
Concurrency	제한 없음
Due Date	2022.05.17

## 2.2. 로그인

Use case #2 : 로그인	
GENERAL CHARACTERISTICS	
Summary	사용자가 애플리케이션을 사용 권한을 얻기 위한 기능이다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	접주, 회원
Preconditions	사용자의 계정이 애플리케이션에 등록되어 있는 상태여야 한다.
Trigger	시작화면에서 사용자 태입에 맞는 시작하기 버튼을 클릭한다.
Success Post Condition	로그인에 성공하고 메인 화면을 출력한다.
Failed Post Condition	로그인 화면으로 돌아간다.

MAIN SUCCESS SCENARIO	
Step	Action
S	애플리케이션은 이메일과 비밀번호를 입력할 수 있는 로그인 화면을 출력한다.
1	사용자는 이메일과 비밀번호를 입력한다.
2	사용자는 로그인 버튼을 클릭한다.
3	애플리케이션은 메인화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
2	2a. 등록되지 않은 이메일인 경우 2a.1 이메일이 일치하지 않는 알림 창을 출력한다. 2a.2 로그인 화면으로 돌아간다.
	2b. 비밀번호가 일치하지 않는 경우 2b.1 비밀번호가 일치하지 않는 알림 창을 출력한다. 2b.2 로그인 화면으로 돌아간다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Virable
Concurrency	제한 없음
Due Date	2022.05.17

### 2.3. 로그아웃

Use case #3 : 로그아웃	
GENERAL CHARACTERISTICS	
Summary	애플리케이션에서 로그아웃 한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어있는 상태여야한다.
Trigger	더보기 화면에서 로그아웃 버튼을 클릭한다.
Success Post Condition	로그아웃에 성공하고 시작 화면을 출력한다.
Failed Post Condition	로그아웃에 실패한다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자가 로그아웃 버튼을 클릭한다.
1	애플리케이션은 로그아웃하고 시작화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	6개월에 한 번
Concurrency	제한 없음
Due Date	2022.05.17

## 2.4. 아이디 찾기

Use case #4 : 아이디 찾기	
GENERAL CHARACTERISTICS	
Summary	애플리케이션에 등록된 사용자의 아이디를 찾는다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어있지 않은 상태여야 한다.
Trigger	로그인 화면에서 아이디 찾기 버튼을 클릭한다.
Success Post Condition	애플리케이션에 등록된 사용자의 아이디를 출력한다.
Failed Post Condition	애플리케이션이 사용자의 아이디를 찾는데 실패한다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자가 아이디 찾기 버튼을 클릭한다.
1	애플리케이션은 이름과 전화번호를 입력할 수 있는 화면을 출력한다.
2	사용자는 이름과 전화번호를 입력하고 전화번호 인증 버튼을 클릭한다.
3	사용자는 전화번호 인증 절차를 거친 후 확인 버튼을 클릭한다.
4	애플리케이션은 사용자의 아이디를 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
3	3a. 등록되지 않은 정보를 기입한 경우 3a.1 등록되지 않은 정보라는 알림 창을 출력한다. 5a.2 아이디 찾기 화면으로 돌아간다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	6개월에 한번
Concurrency	제한 없음
Due Date	2022.05.17

## 2.5. 비밀번호 찾기

Use case #5 : 비밀번호 찾기	
GENERAL CHARACTERISTICS	
Summary	애플리케이션에 등록된 사용자의 아이디에 해당하는 비밀번호를 변경한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어 있지 않은 상태여야 한다.
Trigger	로그인 화면에서 비밀번호 찾기 버튼을 클릭한다.
Success Post Condition	애플리케이션에 등록된 사용자의 아이디에 해당하는 비밀번호를 변경한다.
Failed Post Condition	비밀번호가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자가 비밀번호 찾기 버튼을 클릭한다.
1	애플리케이션은 사용자의 아이디를 입력할 수 있는 화면을 출력한다.
2	사용자는 아이디를 입력한다.
3	사용자는 이메일 인증 버튼을 클릭하고 인증절차를 거친다.
4	사용자는 확인 버튼을 클릭한다.
5	애플리케이션은 새 비밀번호와 새 비밀번호 확인을 입력하는 화면을 출력한다.
6	사용자는 새 비밀번호와 새 비밀번호 확인을 입력하고 확인 버튼을 클릭한다.
7	애플리케이션은 비밀번호가 성공적으로 변경됐다는 알림 창을 출력한다.
8.	애플리케이션은 로그인 화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
4	<p>4a. 등록되지 않은 정보를 기입한 경우</p> <p>4a.1 등록되지 않은 정보라는 알림 창을 출력한다.</p> <p>4a.2 비밀번호 찾기 화면으로 돌아간다.</p>
6	<p>6a. 비밀번호 형식이 올바르지 않은 경우</p> <p>6a.1 비밀번호 형식이 올바르지 않는 알림 창을 출력한다.</p> <p>6a.2 새 비밀번호 및 새 비밀번호 확인을 입력하는 화면으로 돌아간다.</p> <p>6a. 비밀번호가 일치하지 않은 경우</p> <p>6a.1 비밀번호가 일치하지 않는다는 알림 창을 출력한다.</p> <p>6a.2 새 비밀번호 및 새 비밀번호 확인을 입력하는 화면으로 돌아간다.</p>

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	6개월에 한 번
Concurrency	제한 없음
Due Date	2022.05.17

## 2.6. 회원 탈퇴

Use case #6 : 회원 탈퇴	
GENERAL CHARACTERISTICS	
Summary	애플리케이션에 등록된 계정을 삭제한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어있는 상태여야한다.
Trigger	회원정보 수정 화면에서 회원 탈퇴 버튼을 클릭한다.
Success Post Condition	로그인된 계정을 애플리케이션에서 삭제한다.
Failed Post Condition	애플리케이션에 계정이 등록된 채로 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자가 회원 탈퇴 버튼을 클릭한다.
1	애플리케이션은 현재 비밀번호를 입력하는 화면을 출력한다.
2	사용자는 현재 비밀번호를 입력한다.
3	사용자는 확인 버튼을 클릭한다.
4	애플리케이션은 회원 탈퇴가 정상적으로 처리되었다는 알림 창을 출력한다.
5	애플리케이션은 시작화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
3	3a. 현재 비밀번호가 일치하지 않을 경우 3a.1 현재 비밀번호가 일치하지 않는 알림 창을 출력한다. 3a.2 회원 탈퇴 화면으로 돌아간다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	사용자가 어플리케이션을 더 이상 사용하고 싶지 않을 때 한 번
Concurrency	제한 없음
Due Date	2022.05.17

## 2.7. 매장 등록

Use case #7 : 매장 등록	
GENERAL CHARACTERISTICS	
Summary	점주 계정에 새로운 매장을 추가한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	애플리케이션에 로그인 되어있는 상태여야한다.
Trigger	메인 화면에서 매장 등록 버튼을 클릭한다.
Success Post Condition	사용자 계정에 입력한 매장이 등록된다.
Failed Post Condition	사용자 계정의 매장 리스트가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	메인 화면에서 매장 등록 버튼을 클릭한다.
1	애플리케이션은 매장 정보를 입력하는 화면을 출력한다.
2	점주는 매장 정보를 입력한다.
3	점주는 사업자 등록 번호 인증 절차를 거친다.
4	점주는 확인 버튼을 클릭한다.
5	애플리케이션은 정상적으로 등록되었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
4	4a. 사업자 등록 번호 인증이 되지 않은 경우 4a.1 사업자 등록 번호 인증이 완료 되지 않았다는 알림 창을 출력한다. 4a.2 매장 정보를 입력하는 화면으로 돌아간다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.8. 매장 삭제

Use case #8 : 매장 삭제	
GENERAL CHARACTERISTICS	
Summary	점주 계정에 등록된 매장을 삭제한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	메인 화면에서 매장 삭제 버튼을 클릭한다.
Success Post Condition	점주가 선택한 매장이 삭제된다.
Failed Post Condition	점주 계정의 매장 리스트가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	메인 화면에서 매장 삭제 버튼을 클릭한다.
1	애플리케이션은 매장 템 옆에 삭제 버튼을 출력한다.
2	점주는 삭제하고 싶은 매장의 템에서 삭제 버튼을 클릭한다.
3	애플리케이션은 삭제하면 복구할 수 없다는 알림 창을 출력한다.
4	점주는 확인 버튼을 클릭한다.
5	애플리케이션은 해당 매장을 점주 매장 리스트에서 삭제한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	$\leq$ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.9. 매장 리스트

Use case #9 : 매장 리스트	
GENERAL CHARACTERISTICS	
Summary	점주가 등록한 매장의 리스트를 나열한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	메인 화면에 있는 매장 리스트 전체보기를 클릭한다.
Success Post Condition	사용자가 등록한 매장들을 출력한다.
Failed Post Condition	사용자가 등록한 매장들이 출력되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	메인 화면에 있는 매장 리스트 전체보기를 클릭한다.
1	애플리케이션은 점주가 등록한 매장 리스트를 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.10. QR 코드를 통한 도장 적립

Use case #1 : QR 코드를 통한 도장 적립	
GENERAL CHARACTERISTICS	
Summary	점주가 회원에게 도장을 적립한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 회원의 애플리케이션에 있는 QR 코드를 스캔한다.
Success Post Condition	회원이 보유한 쿠폰에 점주가 입력한 개수만큼의 쿠폰이 적립된다.
Failed Post Condition	회원의 쿠폰 정보가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 적립/사용 탭에서 QR코드로 적립 버튼을 클릭한다
1	애플리케이션은 QR 코드 스캐너를 출력한다.
2	점주는 회원의 QR 코드를 스캔한다.
3	애플리케이션은 적립할 개수를 지정할 수 있는 화면을 출력한다.
4	점주는 적립할 개수를 입력하고 확인 버튼을 클릭한다.
5	애플리케이션은 완료되었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	$\leq$ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.11. 전화번호를 통한 도장 적립

Use case #11 : 전화번호를 통한 도장 적립	
GENERAL CHARACTERISTICS	
Summary	점주가 회원에게 도장을 적립한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 회원의 전화번호를 입력한다.
Success Post Condition	회원이 보유한 쿠폰에 점주가 입력한 개수만큼의 쿠폰이 적립된다.
Failed Post Condition	회원의 쿠폰 정보가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 적립/사용 탭에서 전화번호로 적립 버튼을 클릭한다
1	애플리케이션은 회원의 전화번호를 입력할 수 있는 화면을 출력한다.
2	점주는 회원의 전화번호를 입력한다.
3	애플리케이션은 적립할 개수를 지정할 수 있는 화면을 출력한다.
4	점주는 적립할 개수를 입력하고, 확인 버튼을 클릭한다.
5	애플리케이션은 완료되었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.12. 이벤트 등록

Use case #12 : 이벤트 등록	
GENERAL CHARACTERISTICS	
Summary	점주가 회원에게 보여주고 싶은 이벤트를 등록한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 이벤트 등록 버튼을 클릭한다.
Success Post Condition	이벤트 등록에 성공하면, 해당 매장 쿠폰이 있는 회원 애플리케이션의 이벤트 리스트에 추가된다.
Failed Post Condition	점주가 입력한 이벤트가 등록되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 매장 정보 화면에서 이벤트 등록 버튼을 클릭한다.
1	애플리케이션은 배너에 띄울 이미지와 이벤트 상세보기에 띄울 이미지를 업로드할 수 있는 화면을 출력한다.
2	점주는 배너에 띄울 이미자와 이벤트 상세보기에 띄울 이미지를 업로드하고 확인 버튼을 클릭한다.
3	애플리케이션은 등록이 완료되었다는 화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.13. 이벤트 삭제

Use case #13 : 이벤트 삭제	
GENERAL CHARACTERISTICS	
Summary	점주가 이벤트를 삭제한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 이벤트 삭제 버튼을 클릭한다.
Success Post Condition	이벤트 삭제에 성공하면, 해당 매장 쿠폰이 있는 회원 애플리케이션의 이벤트 리스트에서 삭제된다.
Failed Post Condition	점주가 입력한 이벤트가 삭제되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 이벤트 화면에서 이벤트 삭제 버튼을 클릭한다.
1	애플리케이션은 삭제하면 되돌릴 수 없다는 알림 창을 출력한다.
2	점주는 확인 버튼을 클릭한다.
3	애플리케이션은 삭제가 완료되었다는 화면을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.14. 리워드 등록

Use case #14 : 리워드 등록	
GENERAL CHARACTERISTICS	
Summary	점주가 쿠폰 사용 시 얻을 수 있는 리워드를 등록한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 리워드 등록 버튼을 클릭한다.
Success Post Condition	리워드 등록에 성공하면, 해당 매장 쿠폰이 있는 회원 애플리케이션의 쿠폰 리워드가 등록된다.
Failed Post Condition	해당 매장의 리워드가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 리워드 리스트 화면에서 리워드 등록 버튼을 클릭한다.
1	애플리케이션은 리워드와 도장 소모 개수를 설정할 수 있는 화면을 출력한다.
2	점주는 리워드와 도장 소모 개수를 설정하고 확인 버튼을 클릭한다.
3	애플리케이션은 등록 완료되었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.15. 리워드 삭제

Use case #15 : 리워드 삭제	
GENERAL CHARACTERISTICS	
Summary	점주가 등록된 리워드를 삭제한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주
Preconditions	점주 계정에 매장이 등록되어 있는 상태이어야 한다.
Trigger	점주가 리워드 삭제 버튼을 클릭한다.
Success Post Condition	리워드 삭제에 성공하면, 해당 매장 쿠폰이 있는 회원 애플리케이션의 쿠폰 리워드가 삭제된다.
Failed Post Condition	해당 매장의 리워드가 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	점주는 리워드 리스트 화면에서 삭제하고자하는 리워드에서 리워드 삭제 버튼을 클릭한다.
1	애플리케이션은 삭제하면 되돌릴 수 없다는 알림 창을 출력한다.
2	점주는 확인 버튼을 클릭한다.
3	애플리케이션은 삭제가 완료되었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.16. 비밀번호 변경

Use case #1 : 비밀번호 변경	
GENERAL CHARACTERISTICS	
Summary	사용자의 비밀번호를 변경한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어있는 상태여야한다.
Trigger	사용자가 회원 정보 수정 버튼을 클릭한다.
Success Post Condition	사용자의 비밀번호가 변경된다.
Failed Post Condition	사용자의 비밀번호가 변경되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자는 회원 정보 수정 버튼을 클릭한다.
1	애플리케이션은 현재 비밀번호와 새 비밀번호, 새 비밀번호 확인을 입력할 수 있는 화면을 출력한다.
2	사용자는 현재 비밀번호와 새 비밀번호, 새 비밀번호 확인을 입력하고 확인 버튼을 클릭한다.
3	애플리케이션은 비밀번호가 성공적으로 바뀌었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
2	<p>2a. 현재 비밀번호가 일치하지 않을 경우            2a.1 현재 비밀번호가 일치하지 않는 알림 창을 출력한다.            2a.2 회원 정보 수정 화면으로 돌아간다.</p> <p>2b. 새 비밀번호가 일치하지 않을 경우            2b.1 새 비밀번호가 일치하지 않는 알림 창을 출력한다.            2b.2 회원 정보 수정 화면으로 돌아간다.</p> <p>2c. 비밀번호 형식이 맞지 않는 경우            2c.1 비밀번호 형식이 맞지 않다는 알림 창을 출력한다.            2c.2 회원 정보 수정 화면으로 돌아간다.</p>

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.17. 전화번호 변경

Use case #17 : 전화번호 변경	
GENERAL CHARACTERISTICS	
Summary	사용자의 전화번호를 변경한다.
Scope	MOCUMOCU
Author	김준서
Last Update	2022.05.14
Status	Design
Primary Actor	점주, 회원
Preconditions	애플리케이션에 로그인 되어있는 상태여야한다.
Trigger	사용자가 회원 정보 수정 버튼을 클릭한다.
Success Post Condition	사용자의 전화번호가 변경된다.
Failed Post Condition	사용자의 전화번호가 변경되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	사용자가 회원 정보 수정 버튼을 클릭한다.
1	애플리케이션은 새 전화번호를 입력할 수 있는 화면을 출력한다.
2	사용자는 새 전화번호를 입력하고 인증 절차를 거친다
3	사용자는 확인 버튼을 클릭한다.
4	애플리케이션은 전화번호가 성공적으로 바뀌었다는 알림 창을 출력한다.

EXTENSION SCENARIOS	
Step	Branching Action
3	3a. 전화번호 인증이 되지 않은 경우 3a.1 전화번호 인증이 완료 되지 않았다는 알림 창을 출력한다. 3a.2 회원 정보 수정 화면으로 돌아간다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.18. 출석 체크

Use case #18 : 출석 체크	
GENERAL CHARACTERISTICS	
Summary	회원이 로그인 후 출석체크 버튼을 클릭 시 쿠폰 판 및 도장 스킨을 구매할 수 있는 포인트가 적립된다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 로그인이 완료된 상태여야 한다.
Trigger	메인화면에서 출석체크 버튼을 클릭한다.
Success Post Condition	포인트가 성공적으로 적립되었다는 알림이 뜬다.
Failed Post Condition	포인트가 적립되지 않았다는 알림이 뜬다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원은 로그인을 한다.
1	애플리케이션은 메인화면을 보여준다.
2	회원은 메인화면 내 출석체크 버튼을 클릭한다.
3	애플리케이션은 포인트가 성공적으로 적립되었다는 알림 창을 띠운다.
4	애플리케이션은 회원의 적립 포인트를 고정 값 만큼 증가시킨다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	1일 1회
Concurrency	제한 없음
Due Date	2022.05.17

## 2.19. QR 코드를 통한 쿠폰 사용

Use case #19 : QR 코드를 통한 쿠폰 사용	
GENERAL CHARACTERISTICS	
Summary	회원이 QR 코드를 통해 쿠폰을 사용하여 리워드를 얻을 수 있다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 리워드의 도장 개수 만큼 도장을 보유하고 있어야 한다.
Trigger	해당 매장 쿠폰 화면에서 원하는 리워드를 클릭한다.
Success Post Condition	성공적으로 도장 개수를 차감한다.
Failed Post Condition	도장 개수가 부족합니다 라는 알림 창을 보여준다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원은 메인화면 내에서 적립/사용 템을 클릭한다.
1	애플리케이션은 적립, 사용 버튼을 보여준다.
2	회원을 사용 버튼을 클릭한다.
3	애플리케이션은 회원이 보유하고 있는 쿠폰 중 사용 가능한 리스트를 보여준다.
4	회원은 사용하고자 하는 쿠폰을 클릭한다.
5	애플리케이션은 매장에서 지정한 리워드 리스트를 보여준다.
6	회원은 원하는 리워드 버튼을 클릭한다.
7	애플리케이션은 회원 정보 및 차감 할 도장 데이터를 합친 QR 코드를 보여준다.
8	점주는 회원의 QR 코드를 스캔한다.
9	애플리케이션은 회원 쿠폰 판에서 리워드에 할당 된 개수만큼 도장을 차감시키고 사용 완료되었습니다 라는 알림 창을 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
3	<p>3a. 사용 가능한 쿠폰이 없는 경우</p> <p>3a.1. 사용 가능한 쿠폰이 없습니다 라는 알림 창을 보여준다.</p> <p>3a.2. 메인화면으로 돌아간다.</p>
6	<p>6a. 리워드에 지정된 만큼의 도장을 보유하고 있지 않은 경우</p> <p>6a.1. 도장 개수가 부족합니다 라는 알림 창을 보여준다.</p> <p>6a.2. 리워드 리스트 화면으로 돌아간다.</p>

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.20. 쿠폰 리스트

Use case #20 : 쿠폰 리스트	
GENERAL CHARACTERISTICS	
Summary	회원이 메인화면에서 전체+ 버튼 클릭 시 회원이 보유하고 있는 쿠폰 리스트를 보여준다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 로그인이 완료된 상태이어야한다.
Trigger	회원이 메인화면에서 전체+ 버튼을 클릭한다.
Success Post Condition	회원이 보유하고 있는 쿠폰 리스트를 보여준다.
Failed Post Condition	보유하고 있는 쿠폰이 없습니다 라는 알람을 보여준다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원이 메인화면에서 전체+ 버튼을 클릭한다.
1	애플리케이션은 회원이 보유하고 있는 쿠폰 리스트를 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.21. 쿠폰 삭제

Use case #21 : 쿠폰 삭제	
GENERAL CHARACTERISTICS	
Summary	회원이 쿠폰 리스트 화면에서 쿠폰 삭제 버튼을 클릭해 원하는 쿠폰을 삭제한다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 보유 중인 쿠폰이 있어야 한다.
Trigger	회원이 쿠폰 리스트 화면에서 삭제 버튼을 클릭한다.
Success Post Condition	선택한 쿠폰을 삭제한다.
Failed Post Condition	쿠폰이 삭제되지 않고 쿠폰이 유지된다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원이 쿠폰 리스트 화면에서 삭제 버튼을 클릭한다.
1	애플리케이션은 화면에 삭제 버튼을 보여준다.
2	회원은 삭제하고자 하는 쿠폰의 삭제 버튼을 클릭한다.
3	애플리케이션은 회원이 클릭한 쿠폰을 삭제한다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.22. 이벤트 상세 보기

Use case #22 : 이벤트 상세보기	
GENERAL CHARACTERISTICS	
Summary	회원이 메인화면에 보이는 이벤트 배너 중 원하는 이벤트를 클릭해 이벤트 내용을 상세하게 볼 수 있다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 보유 중인 쿠폰이 있어야 한다.
Trigger	회원이 메인화면의 이벤트 배너 중 한 개를 클릭한다.
Success Post Condition	클릭한 이벤트 배너의 상세 내용을 보여준다.
Failed Post Condition	이벤트 배너의 상세 내용을 보여주지 못한다.

MAIN SUCCESS SCENARIO	
Step	Action
S	애플리케이션은 메인화면에 이벤트 내역을 배너 형식으로 보여준다.
1	회원은 메인화면의 이벤트 배너 중 한 개를 클릭한다.
2	애플리케이션은 회원이 클릭한 이벤트의 상세 내용을 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	$\leq$ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.23. 쿠폰 판 스킨 구매

Use case #23 : 쿠폰 판 스킨 구매	
GENERAL CHARACTERISTICS	
Summary	회원이 포인트 상점으로 이동해서 쿠폰 판 스킨을 구매한다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 원하는 쿠폰 판 스킨을 구매하기 위한 포인트가 충분히 있어야 한다.
Trigger	회원이 포인트 상점에서 원하는 쿠폰 판 스킨을 선택해 구매 버튼을 클릭한다.
Success Post Condition	성공적으로 원하는 쿠폰 판 스킨이 구매된다.
Failed Post Condition	원하는 쿠폰 판 스킨이 구매되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원 메인화면에서 포인트 상점 버튼을 클릭한다.
1	애플리케이션은 포인트 상점 화면과 쿠폰 판 스킨 목록을 보여준다.
2	회원은 원하는 쿠폰 판 스킨을 클릭해 구매 버튼을 클릭한다.
3	애플리케이션은 구매완료라는 알림 창을 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
2	2a. 원하는 쿠폰 판 스킨을 구입하기 위한 포인트가 부족할 경우 2a.1. 포인트가 부족합니다라는 알림 창을 보여준다.

RELATED INFORMATION	
Performance	$\leq 2$ seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.24. 도장 스킨 구매

Use case #24 : 도장 스킨 구매	
GENERAL CHARACTERISTICS	
Summary	회원이 포인트 상점으로 이동해서 도장 스킨을 구매한다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 원하는 도장 스킨을 구매하기 위한 포인트가 충분히 있어야 한다.
Trigger	회원이 포인트 상점에서 원하는 도장 스킨을 선택해 구매 버튼을 클릭한다.
Success Post Condition	성공적으로 원하는 도장 스킨이 구매된다.
Failed Post Condition	원하는 도장 스킨이 구매되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원 메인화면에서 포인트 상점 버튼을 클릭한다.
1	애플리케이션은 포인트 상점 화면을 보여준다.
2	회원은 포인트 상점 화면에서 도장 스킨 탭을 클릭한다.
3	애플리케이션은 도장 스킨 목록을 보여준다.
2	회원은 원하는 도장 스킨을 클릭해 구매 버튼을 클릭한다.
3	애플리케이션은 구매완료라는 알림 창을 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
2	2a. 원하는 도장 스킨을 구입하기 위한 포인트가 부족할 경우 2a.1. 포인트가 부족합니다라는 알림 창을 보여준다.

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.25. 쿠폰 판 스킨 적용

Use case #25 : 쿠폰 판 스킨 적용	
GENERAL CHARACTERISTICS	
Summary	회원이 쿠폰을 클릭해 원하는 쿠폰 판 스킨을 적용시킬 수 있다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 구매한 쿠폰 판 스킨이 존재해야 한다.
Trigger	회원이 쿠폰 판 스킨 목록에서 원하는 쿠폰 판 스킨을 선택해 적용 버튼을 클릭한다.
Success Post Condition	해당 쿠폰에 성공적으로 원하는 쿠폰 판 스킨이 적용된다.
Failed Post Condition	원하는 쿠폰 판 스킨이 적용되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원은 쿠폰 리스트 화면에서 스킨 변경을 원하는 쿠폰을 클릭한다.
1	애플리케이션은 회원이 구매한 쿠폰 판 스킨 목록을 보여준다.
2	회원은 원하는 쿠폰 판 스킨을 선택하여 적용 버튼을 클릭한다.
3	애플리케이션은 회원이 선택한 쿠폰 판 스킨을 쿠폰에 적용시켜 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

## 2.26. 도장 스킨 적용

Use case #26 : 도장 스킨 적용	
GENERAL CHARACTERISTICS	
Summary	회원이 쿠폰을 클릭해 원하는 도장 스킨을 적용시킬 수 있다.
Scope	MOCUMOCU
Author	여민수
Last Update	2022.05.14
Status	Design
Primary Actor	회원
Preconditions	회원은 구매한 도장 스킨이 존재해야 한다.
Trigger	회원이 쿠폰 판 스킨 목록에서 원하는 도장 스킨을 선택해 적용 버튼을 클릭한다.
Success Post Condition	해당 쿠폰에 성공적으로 원하는 도장 스킨이 적용된다.
Failed Post Condition	원하는 도장 스킨이 적용되지 않는다.

MAIN SUCCESS SCENARIO	
Step	Action
S	회원은 쿠폰 리스트 화면에서 스킨 변경을 원하는 쿠폰을 클릭한다.
1	애플리케이션은 회원이 구매한 쿠폰 도장 스킨 목록을 보여준다.
2	회원은 원하는 쿠폰 도장 스킨을 선택하여 적용 버튼을 클릭한다.
3	애플리케이션은 회원이 선택한 쿠폰 도장 스킨을 쿠폰에 적용시켜 보여준다.

EXTENSION SCENARIOS	
Step	Branching Action
	None

RELATED INFORMATION	
Performance	≤ 2 seconds
Frequency	Variable
Concurrency	제한 없음
Due Date	2022.05.17

### 3. E-R Diagram

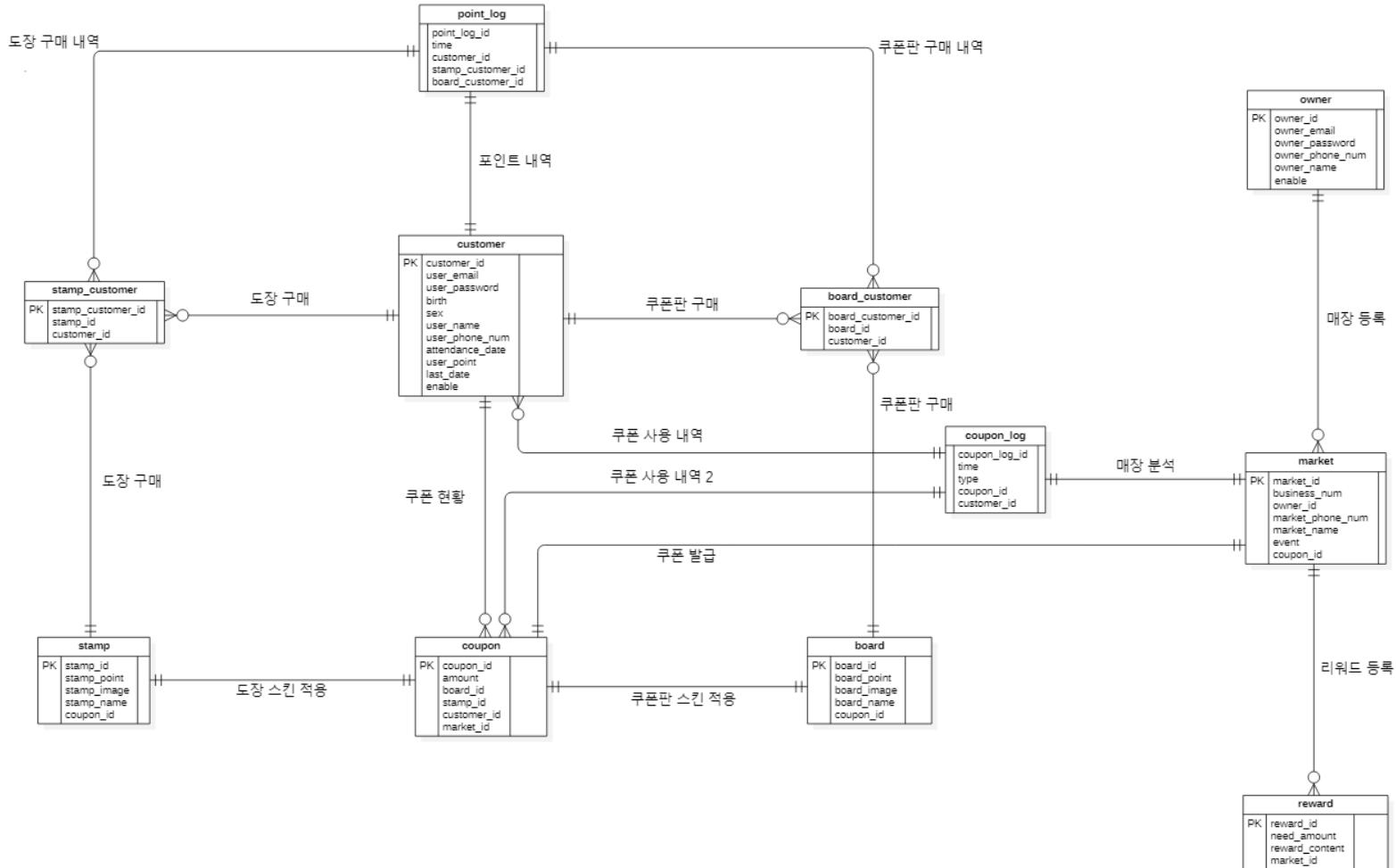


그림 2 E-R Diagram

## 4. Class Diagram

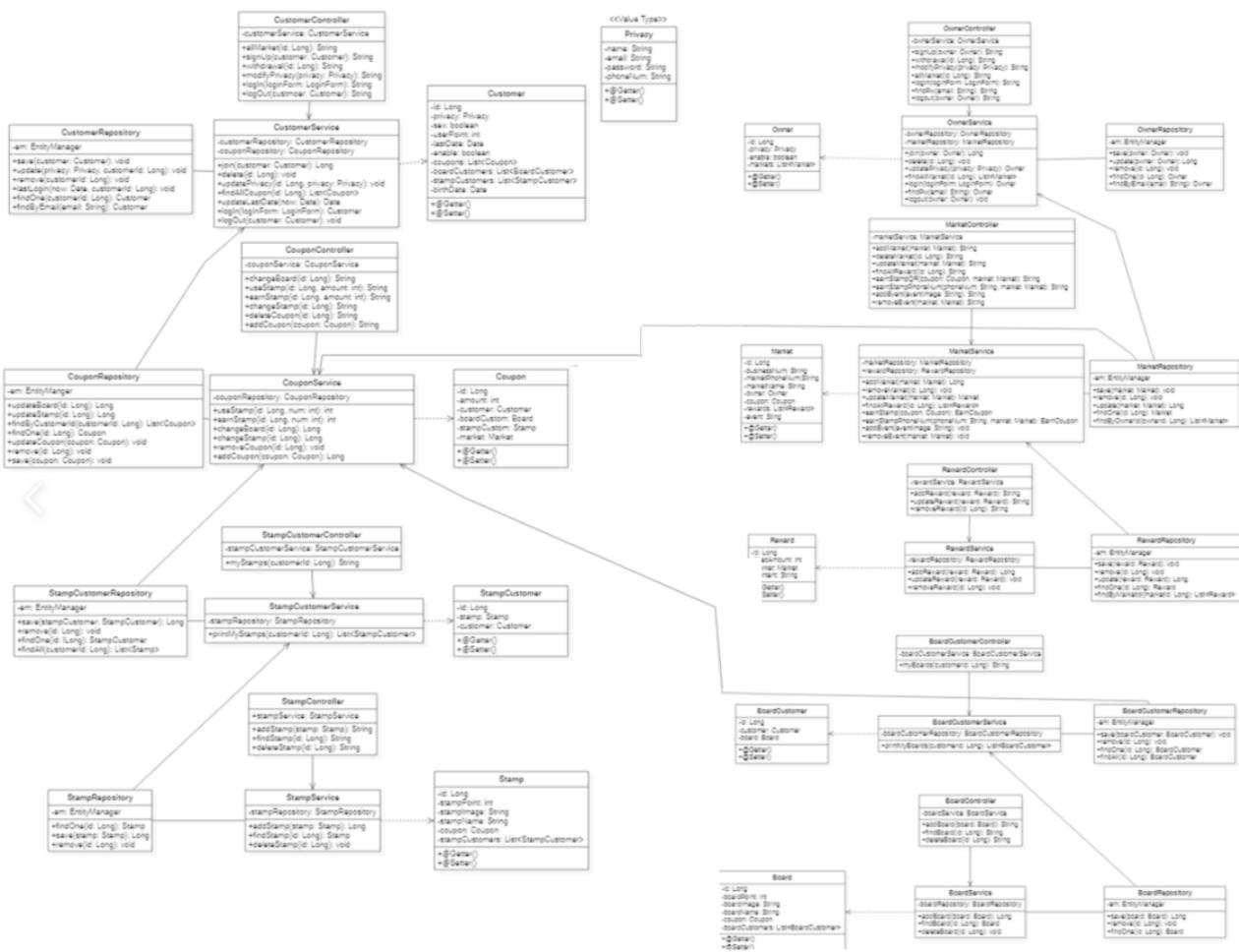


그림 3 Class Diagram

#### 4.1. Privacy

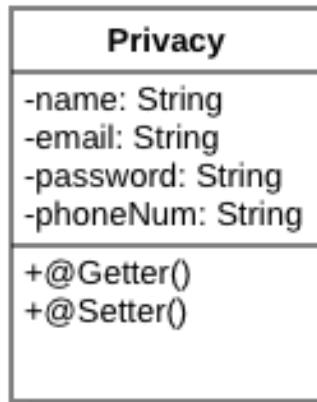


그림 4 Privacy 클래스

점주와 회원의 공통 정보를 갖기 위한 클래스. 회원 가입이나, 점주, 회원 정보를 가져올 때 사용된다.

Attributes
-name: String ->점주나 회원의 이름을 저장하기 위한 필드
-email: String ->점주나 회원의 이메일을 저장하기 위한 필드
-password: String ->점주나 회원의 비밀번호를 저장하기 위한 필드
-phoneNum: String ->점주나 회원의 전화번호를 저장하기 위한 필드
Annotationss
@Getter, @Setter ->Getter Setter를 쉽게 지정할 수 있도록 하는 lombok 애노테이션

## 4.2. Customer

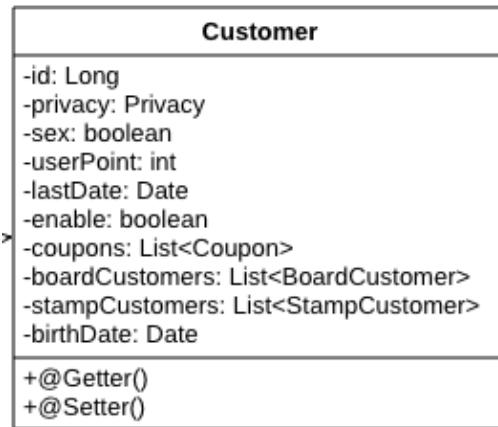


그림 5 Customer 클래스

회원에 관한 정보를 나타내는 클래스이다.

Attributes
-id: Long ->데이터베이스에서 회원의 고유 식별번호로 이용 되는 id 필드
-privacy: Privacy ->회원의 공통 정보를 호출하기 위한 필드
-sex: boolean ->회원의 성별을 저장하기 위한 필드
-userPoint: int ->쿠폰 커스터마이징을 위한 포인트를 저장하기 위한 필드
-lastDate: Date ->최종 접속일을 저장하기 위한 필드
-enable: boolean ->Spring Security 관련 Attribute를 데이터베이스에 생성하기 위한 필드
-coupons: List<Coupon> ->쿠폰 리스트를 받아오기 위한 필드
-boardCustomers: List<BoardCustomer> ->커스터마이징 한 쿠폰 판을 받아오기 위한 필드
-stampCustomers: List<StampCustomer> ->커스터마이징 한 도장을 받아오기 위한 필드
-birthDate: Date ->회원의 생년월일을 저장하기 위한 필드
Annotationss
@Getter, @Setter ->Getter Setter를 쉽게 지정할 수 있도록 하는 lombok 애노테이션

### 4.3. CustomController

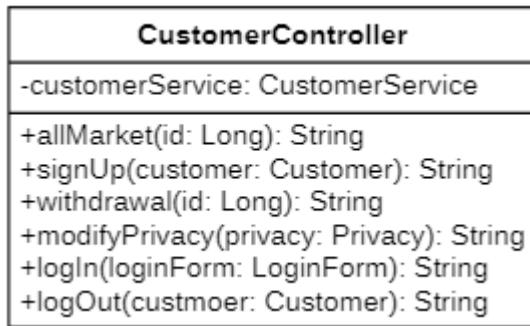


그림 6 CustomerController 클래스

회원 기능을 위한 컨트롤러 클래스이다.

Attributes
<b>-customerService: CustomerService</b> ->회원 서비스를 받아오기 위한 필드
Methods
<b>+allMarket(id: Long): String</b> ->회원이 쿠폰을 발급받은 매장을 보여주기 위한 메소드. HTTP Status를 반환한다. <b>+signUp(customer: Customer): String</b> ->회원 가입 기능을 위한 메소드. HTTP Status를 반환한다. <b>+withdrawal(id: Long): String</b> ->회원 탈퇴 기능을 위한 메소드. HTTP Status를 반환한다. <b>+modifyPrivacy(privacy: Privacy): String</b> ->개인정보 수정 기능을 위한 메소드. HTTP Status를 반환한다. <b>+login(loginForm: LoginForm): String</b> ->로그인 기능을 위한 메소드. HTTP Status를 반환한다. <b>+logOut(customer: Customer): String</b> ->로그아웃 기능을 위한 메소드. HTTP Status를 반환한다.

#### 4.4. CustomerRepository

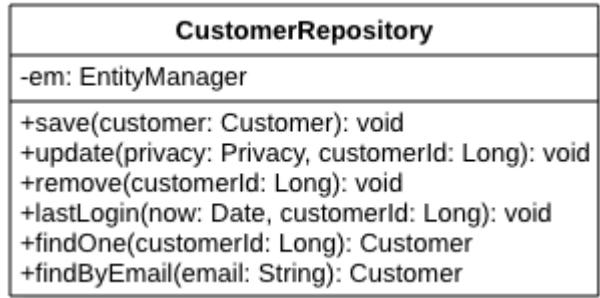


그림 7 CustomerRepository 클래스

회원 저장소를 구현을 위한 클래스이다.

Attributes
<b>-em: EntityManager</b> ->데이터베이스의 Entity를 관리하기 위한 필드
Methods
<b>+save(customer: Customer): void</b> ->회원을 데이터베이스에 저장하기 위한 메소드 <b>+update(privacy: Privacy, customerId: Long): void</b> ->데이터베이스의 해당 회원 정보를 업데이트 하기 위한 메소드 <b>+remove(customerId: Long): void</b> ->데이터베이스에서 해당 회원을 삭제 하기 위한 메소드 <b>+lastLogin(now: Date, customerId: Long): Date</b> ->데이터베이스에서 해당 회원의 최종 접속 일자를 수정하기 위한 메소드 <b>+findOne(customerId: Long): Customer</b> ->데이터베이스에서 해당 회원을 반환 받기 위한 메소드 <b>+findByEmail(email: String): Customer</b> ->데이터베이스에서 이메일로 회원을 찾기 위한 메소드

#### 4.5. CustomerService

CustomerService	
-customerRepository: CustomerRepository	
-couponRepository: CouponRepository	

+join(customer: Customer): Long
+delete(id: Long): void
+updatePrivacy(id: Long, privacy: Privacy): void
+findAllCoupon(id: Long): List<Coupon>
+updateLastDate(now: Date): Date
+login(logInForm: LoginForm): Customer
+logOut(customer: Customer): void

그림 8 CustomerService 클래스

회원 서비스 구현을 위한 클래스이다.

Attributes
-customerRepository: CustomerRepository
->회원 저장소 기능을 이용하기 위한 객체
-couponRepository: CouponRepository
->쿠폰 저장소 기능을 이용하기 위한 객체
Methods
+join(customer: Customer): Long
->회원 가입 기능 구현을 위한 메소드
+delete(id: Long): void
->회원 탈퇴 기능 구현을 위한 메소드
+updatePrivacy(id: Long, privacy: Privacy): void
->회원 수정 기능 구현을 위한 메소드
+findAllCoupon(id: Long): List<Coupon>
->가지고 있는 쿠폰을 모두 받아오기 위한 메소드
+updateLastDate(now: Date): Date
->최종 접속 일자 갱신 기능 구현을 위한 메소드
+login(logInForm: LoginForm): Customer
->클라이언트에서 받아온 json 객체로 로그인 기능 구현을 위한 메소드
+logOut(customer: Customer): void
->로그아웃 기능 구현을 위한 메소드

## 4.6. Coupon

Coupon	
-id: Long	
-amount: int	
-customer: Customer	
-boardCustom: Board	
-stampCustom: Stamp	
-market: Market	
+@Getter()	
+@Setter()	

그림 9 Coupon 클래스

쿠폰에 관한 정보를 나타내는 클래스이다.

Attributes
-id: Long ->데이터베이스에서 쿠폰의 고유 식별번호로 이용 되는 id 필드
-amount: int ->해당 쿠폰에 적립된 도장의 개수를 저장하는 필드
-customer: Customer ->해당 쿠폰을 소유하고 있는 회원을 저장하는 필드
-boardCustom: Board ->커스텀한 쿠폰 판의 정보를 받아오기 위한 필드
-stampCustom: Stamp ->커스텀한 도장의 정보를 받아오기 위한 필드
-market: Market ->해당 쿠폰을 발급하는 매장을 저장하는 필드
Annotationss
@Getter, @Setter ->Getter Setter를 쉽게 지정할 수 있도록 하는 lombok 애노테이션

#### 4.7. CouponController

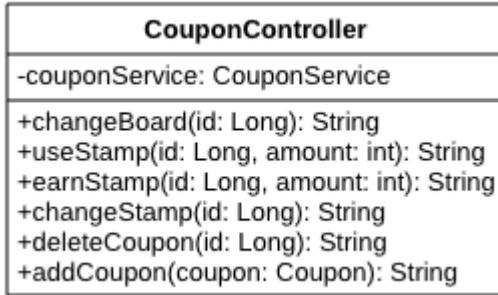


그림 10 CouponController 클래스

쿠폰 기능을 위한 컨트롤러 클래스이다.

Attributes
-couponService: CouponService ->쿠폰 서비스를 받아오기 위한 필드
Methods
+changeBoard(id: Long): String ->커스터마이징 한 쿠폰 판을 변경하기 위한 메소드. HTTP Status를 반환한다. +useStamp(id: Long, amount: int): String ->도장 사용 기능을 위한 메소드, HTTP Status를 반환한다. +earnStamp(id: Long, amount: int): String ->도장 적립 기능을 위한 메소드. HTTP Status를 반환한다. +changeStamp(id: Long): String ->커스터마이징 한 도장을 변경하기 위한 메소드. HTTP Status를 반환한다. +deleteCoupon(id: Long): String ->쿠폰 삭제 기능을 위한 메소드. HTTP Status를 반환한다. +addCoupon(coupon: Coupon): String ->쿠폰 추가 기능을 위한 메소드. HTTP Status를 반환한다.

#### 4.8. CouponRepository



그림 11 CouponRepository 클래스

쿠폰 저장소를 구현을 위한 클래스이다.

Attributes
<p>-em: EntityManager -&gt;데이터베이스의 Entity를 관리하기 위한 필드</p>
Methods
<p>+updateBoard(id: Long): Long -&gt;쿠폰 판의 변경 사항을 저장하기 위한 메소드 +updateStamp(id: Long): Long -&gt;도장의 변경 사항을 저장하기 위한 메소드 +findByCustomerId(customerId: Long): List&lt;Coupon&gt; -&gt;데이터베이스에서 해당 회원이 소유한 쿠폰을 모두 받아오기 위한 메소드 +findOne(id: Long): Coupon -&gt;데이터베이스에서 해당 쿠폰을 반환 받기 위한 메소드 +updateCoupon(coupon: Coupon): void -커스터마이징 한 쿠폰을 데이터베이스 업데이트 하기 위한 메소드 +remove(id: Long): void -&gt;데이터베이스에서 해당 쿠폰을 제거하기 위한 메소드 +save(coupon: Coupon): void -&gt;쿠폰을 데이터베이스에 저장하기 위한 메소드</p>

#### 4.9. CouponService

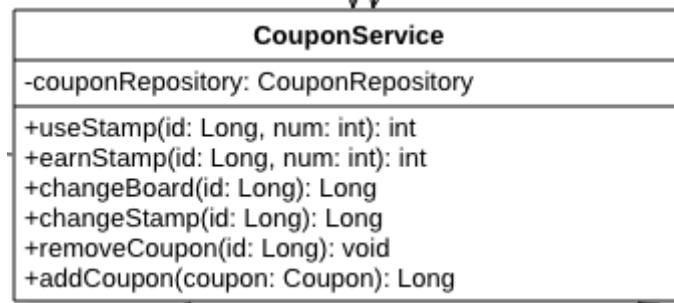


그림 12 CouponService 클래스

쿠폰 서비스를 구현하기 위한 클래스이다.

Attributes
-couponRepository: CouponRepository ->쿠폰 저장소 기능을 이용하기 위한 객체
Methods
+useStamp(id: Long, num: int): int ->도장 사용 기능 구현을 위한 메소드 +earnStamp(id: Long, num: int): int ->도장 적립 기능을 구현을 위한 메소드 +changeBoard(id: Long): Long ->커스터マイ징 한 쿠폰 판을 변경하기 위한 메소드 +changeStamp(id: Long): Long ->커스터マイ징 한 도장을 변경하기 위한 메소드 +removeCoupon(id: Long): void ->해당 쿠폰을 삭제하기 위한 메소드 +addCoupon(coupon: Coupon): Long ->쿠폰 추가 기능을 구현을 위한 메소드

#### 4.10. StampCustomer

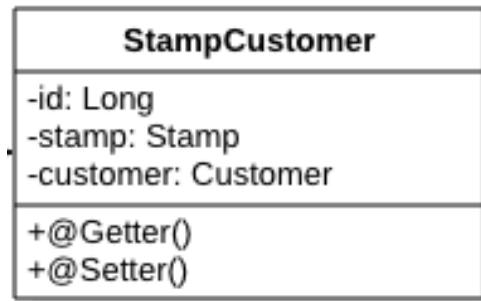


그림 13 StampCustomer 클래스

도장 커스터마이징에 대한 정보를 나타내는 클래스이다.

Attributes
-id: Long ->데이터베이스에서 커스터마이징 한 도장의 고유 식별번호로 이용 되는 id 필드
-stamp: Stamp ->도장의 정보를 받아오기 위한 객체
-customer: Customer ->커스터마이징 한 도장을 소유한 회원을 받아오기 위한 객체
Annotations
@Getter, @Setter ->Getter Setter를 쉽게 지정할 수 있도록 하는 lombok 애노테이션

#### 4.11. StampCustomerController

StampCustomerController
-stampCustomerService: StampCustomerService
+myStamps(customerId: Long): String

그림 14 StampCustomerController 클래스

도장 커스터마이징 기능을 위한 컨트롤러 클래스이다.

Attributes
-stampCustomerService: StampCustomerService ->쿠폰 커스터마이징 서비스를 받아오기 위한 객체
Methods
+myStamps(customerId: Long): String ->해당 회원의 도장을 나타내기 위한 메소드. HTTP Status를 반환한다.

#### 4.12. StampCustomerService

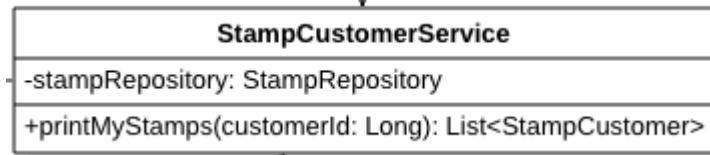


그림 15 StampCustomerService 클래스

도장 커스터마이징을 구현하기 위한 클래스

Attributes
<b>-stampRepository: StampRepository</b> ->도장 저장소 기능을 이용하기 위한 객체
Methods
<b>+printMyStamps(customerId: Long): List&lt;StampCustomer&gt;</b> ->해당 회원의 커스터마이징 한 도장을 모두 출력하기 위한 메소드

#### 4.13. StampCustomerRepository

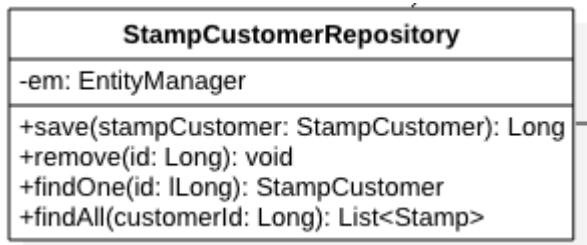


그림 16 StampCustomerRepository 클래스

커스터마이징 한 도장의 저장소 구현을 위한 클래스이다.

Attributes
<b>-em: EntityManager</b> ->데이터베이스의 Entity를 관리하기 위한 필드
Methods
<b>+save(stampCustomer: StampCustomer): Long</b> ->커스터마이징 한 도장을 저장하기 위한 메소드 <b>+remove(id: Long): void</b> ->데이터베이스에서 해당 커스터마이징 도장을 제거하기 위한 메소드 <b>+findOne(id: Long): StampCustomer</b> ->데이터베이스에서 해당 커스터마이징 도장을 반환 받기 위한 메소드 <b>+findAll(customerId: Long): List&lt;Stamp&gt;</b> ->데이터베이스에서 도장을 모두 반환 받기 위한 메소드

#### 4.14. Stamp

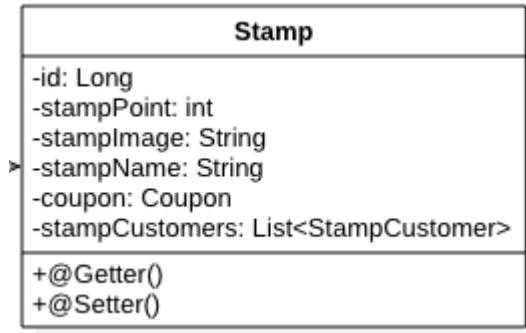


그림 17 Stamp 클래스

쿠폰에 저장되는 도장을 나타내는 클래스이다.

Attributes
<ul style="list-style-type: none"><li>-id: Long<ul style="list-style-type: none"><li>-&gt;데이터 베이스에서 도장의 고유 식별번호로 이용 되는 id 필드</li></ul></li><li>-stampPoint: int<ul style="list-style-type: none"><li>-&gt;도장의 포인트를 저장하기 위한 필드</li></ul></li><li>-stampImage: String<ul style="list-style-type: none"><li>-&gt;도장 이미지의 경로를 받아오기 위한 필드</li></ul></li><li>-stampName: String<ul style="list-style-type: none"><li>-&gt;도장의 이름을 저장하기 위한 필드</li></ul></li><li>-coupon: Coupon<ul style="list-style-type: none"><li>-&gt;쿠폰 판의 정보를 받아오기 위한 필드</li></ul></li><li>-stampCustomers: List&lt;StampCustomer&gt;<ul style="list-style-type: none"><li>-&gt;커스텀 한 도장 리스트를 받아오기 위한 필드</li></ul></li></ul>
Annotationss
<ul style="list-style-type: none"><li>@Getter, @Setter<ul style="list-style-type: none"><li>-&gt;Getter Setter를 쉽게 지정할 수 있도록 하는 lombok 애노테이션</li></ul></li></ul>

#### 4.15. StampController

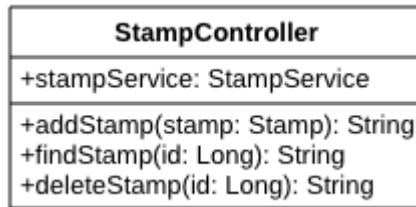


그림 18 StampController 클래스

도장 기능을 위한 컨트롤러 클래스이다.

Attributes
-stampService: StampService ->도장 서비스를 받아오기 위한 필드
Methods
+addStamp(stamp: Stamp): String ->도장을 추가하는 기능을 위한 메소드, HTTP Status를 반환한다. +findStamp(id: Long): String ->도장을 찾는 기능을 위한 메소드, HTTP Status를 반환한다. +deleteStamp(id: Long): String ->도장을 삭제하는 기능을 위한 메소드, HTTP Status를 반환한다.

#### 4.16. StampRepository

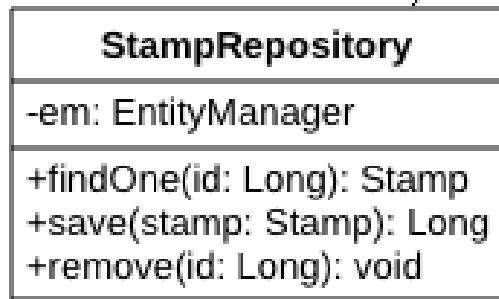


그림 19 StampRepository 클래스

도장 저장소를 구현을 위한 클래스이다.

Attributes
<code>-em: EntityManager</code> ->데이터베이스의 Entity를 관리하기 위한 필드
Methods
<code>+findOne(id: Long): Stamp</code> ->해당 도장을 찾기 위한 메소드 <code>+save(stamp: Stamp): Long</code> ->해당 도장을 저장하기 위한 메소드 <code>+remove(id: Long): void</code> ->해당 도장을 삭제하기 위한 메소드

#### 4.17. StampService

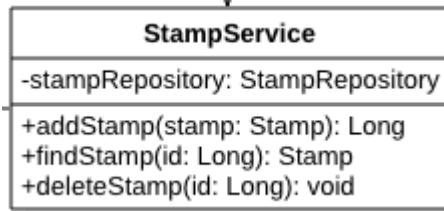


그림 20 StampService 클래스

도장 서비스를 구현하기 위한 클래스이다.

Attributes
<code>-stampRepository: StampRepository</code> ->도장 저장소 기능을 이용하기 위한 객체
Methods
<code>+addStamp(stamp: Stamp): Long</code> ->도장을 추가하는 기능 구현을 위한 메소드 <code>+findStamp(id: Long): Stamp</code> ->해당 도장 찾기 기능 구현을 위한 메소드 <code>+deleteStamp(id: Long): void</code> ->해당 도장 삭제 기능 구현을 위한 메소드

#### 4.18. Owner



그림 21 Owner 클래스

Owner 클래스는 점주에 대한 정보를 나타내는 클래스이다. 각 점주들의 id, Privacy, enable, markets이 선언되어 있다. 또한 이에 대한 Getter, Setter가 있다.

Attributes
-id: Long -> DB에서 점주를 식별하기 위한 Long형 필드
-privacy: Privacy -> 점주의 개인정보는 name, email, password, phoneNum을 저장하기 위한 필드
-enable: boolean -> Spring Security 관련 Attribute를 데이터베이스에 생성하기 위한 필드
-markets: List<Markets> -> 점주가 소유하고 있는 매장 리스트를 저장하기 위한 List형 필드
Annotations
+@Getter -> 각 필드를 불러오기 위한 get 메소드
+@Setter -> 각 필드를 설정하기 위한 set 메소드

#### 4.19. OwnerController

OwnerController	
-ownerService: OwnerService	
+signUp(owner: Owner): String	
+withdrawal(id: Long): String	
+modifyPrivacy(privacy: Privacy): String	
+allMarket(id: Long): String	
+logIn(logInForm: LogInForm): String	
+findPw(email: String): String	
+logout(owner: Owner): String	

그림 22 OwnerController 클래스

클라이언트의 접속 기능 중 해당 요청 url로 요청이 왔을 시 각 메소드로 mapping하여 필요한 기능이 있는 OwnerService 클래스를 호출한다.

Attributes
-ownerService: OwnerService -> OwnerService를 의존 관계 주입을 위한 필드
Methods
+signUp(owner: Owner): String -> 접속 회원가입을 위한 요청 url을 mapping 하는 메소드
+withdrawal(id: Long): String -> 접속 회원탈퇴를 위한 요청 url을 mapping 하는 메소드
+modifyPrivacy(privacy: Privacy): String -> 접속 회원 정보 수정을 위한 요청 url을 mapping 하는 메소드
+allMarket(id: Long): String -> 접속자가 소유한 매장 리스트 출력을 위한 요청 url을 mapping 하는 메소드
+logIn(logInForm: LogInForm): String -> 접속 로그인을 위한 요청 url을 mapping 하는 메소드
+findPw(email: String): String -> 접속 비밀번호 찾기를 위한 요청 url을 mapping 하는 메소드
+logout(owner: Owner): String -> 접속 로그아웃을 위한 요청 url을 mapping 하는 메소드

#### 4.20. OwnerService

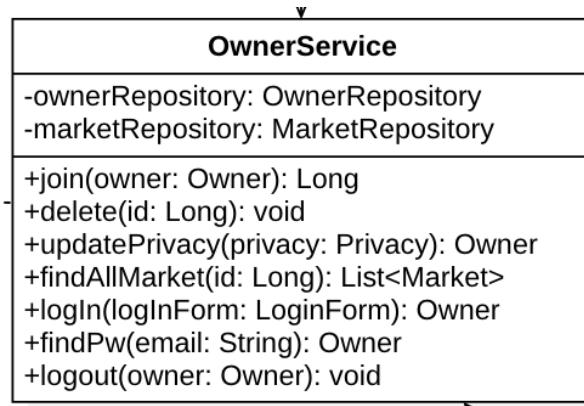


그림 23 OwnerService 클래스

점주에 관한 5가지 기능인 회원가입, 회원탈퇴, 회원 정보 수정, 소유 매장 리스트 출력, 로그인을 구현하는 클래스

Attributes	
<code>-ownerRepository: OwnerRepository</code>	
<code>-&gt; ownerRepository를 의존 관계 주입을 하기 위한 필드</code>	
Methods	
<code>+join(owner: Owner): Long</code>	
<code>-&gt; 점주 회원가입 로직을 실행하기 위한 메소드</code>	
<code>+delete(id: Long): void</code>	
<code>-&gt; 점주 회원탈퇴 로직을 실행하기 위한 메소드</code>	
<code>+updatePrivacy(privacy: Privacy): Owner</code>	
<code>-&gt; 점주 회원정보수정 로직을 실행하기 위한 메소드</code>	
<code>+findAllMarket(id: Long): List&lt;Market&gt;</code>	
<code>-&gt; 점주가 소유한 매장 리스트를 가져오는 로직을 실행하기 위한 메소드</code>	
<code>+logIn(logInForm: LoginForm): Owner</code>	
<code>-&gt; 점주 로그인 로직을 실행하기 위한 메소드</code>	
<code>+findPw(email: String): Owner</code>	
<code>-&gt; 점주 비밀번호 찾기 로직을 실행하기 위한 메소드</code>	
<code>+logout(owner: Owner): void</code>	
<code>-&gt; 점주 로그아웃을 하기 위한 메소드</code>	

#### 4.21. OwnerRepository

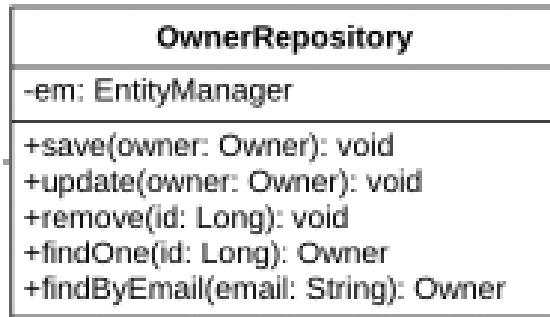


그림 24 OwnerRepository 클래스

JPA를 이용하여 Owner와 관련된 DB를 관리하는 클래스

Attributes
<pre>-em: EntityManager</pre> <p>-&gt; 영속 컨텍스트에 접근하여 엔티티에 대한 DB 작업을 제공하는 필드</p>
Methods
<pre>+save(owner: Owner): void -&gt; 점주 회원가입한 정보를 DB에 저장하는 메소드  +update(owner: Owner): void -&gt; 점주 회원 정보 수정한 정보를 DB에 저장하는 메소드  +remove(id: Long): void -&gt; 점주를 DB에서 삭제하기 위한 메소드  +findOne(id: Long): Owner -&gt; id를 통해 DB에서 해당 점주를 찾아오기 위한 메소드  +findByEmail(email: String): Owner -&gt; email을 통해 DB에서 해당 점주를 찾아오기 위한 메소드</pre>

## 4.22. Market



그림 25 Market 클래스

Market 클래스는 매장에 대한 정보를 나타내는 클래스이다. 각 점주들의 id, businessNum, marketPhoneNum, marketName, owner, coupon, rewards, event가 선언되어 있다. 또한 이에 대한 Getter, Setter가 있다.

Attributes
-id: Long -> DB에서 매장을 식별하기 위한 Long형 필드
-businessNum: String -> 매장의 사업자 등록번호를 저장하기 위한 필드
-marketPhoneNum: String -> 매장의 전화번호를 저장하기 위한 필드
-marketName: String -> 매장의 이름을 저장하기 위한 필드
-owner: Owner -> 매장의 점주 정보를 저장하기 위한 필드
-coupon: Coupon -> 매장의 쿠폰 정보를 저장하기 위한 필드
-rewards: List<Reward> -> 매장의 리워드들을 저장하기 위한 필드
-event: String -> 매장의 이벤트 이미지의 저장 경로를 저장하기 위한 필드
Annotations
+@Getter -> 각 필드를 불러오기 위한 get 메소드
+@Setter -> 각 필드를 설정하기 위한 set 메소드

#### 4.23. MarketController

<b>MarketController</b>
-marketService: MarketService
+addMarket(market: Market): String +deleteMarket(id: Long): String +updateMarket(market: Market): String +findAllReward(id: Long): String +earnStampQR(coupon: Coupon, market: Market): String +earnStampPhoneNum(phoneNum: String, market: Market): String +addEvent(eventImage: String): String +removeEvent(market: Market): String

그림 26 MarketController 클래스

클라이언트의 매장 기능 중 해당 요청 url로 요청이 왔을 시 각 메소드로 mapping 하여 필요한 기능이 있는 MarketService 클래스를 호출한다.

Attributes
-marketService: MarketService -> MarketService를 의존 관계 주입을 위한 필드
Methods
+addMarket(market: Market): String -> 매장 추가를 위한 요청 url을 mapping 하는 메소드 +deleteMarket(id: Long): String -> 매장 삭제를 위한 요청 url을 mapping 하는 메소드 +updateMarket(market: Market): String -> 매장 정보 수정을 위한 요청 url을 mapping 하는 메소드 +findAllReward(id: Long): String -> 매장의 모든 리워드 리스트 출력을 위한 요청 url을 mapping 하는 메소드 +earnStampQR(coupon: Coupon, market: Market): String -> QR을 통한 쿠폰 적립 요청 url을 mapping 하는 메소드 +earnStampPhoneNum(phoneNum: String, market: Market): String -> 전화번호를 통한 쿠폰 적립 요청을 위한 요청 url을 mapping 하는 메소드 +addEvent(eventImage: String): String -> 매장의 이벤트 추가를 위한 요청 url을 mapping 하는 메소드 +removeEvent(market: Market): String -> 매장의 이벤트 삭제를 위한 요청 url을 mapping 하는 메소드

#### 4.24. MarketService



그림 27 MarketService 클래스

매장에 관한 8가지 기능인 매장 등록, 매장 삭제, 매장 정보 수정, 매장의 리워드 출력, 쿠폰 적립, 이벤트 등록 및 삭제를 구현하는 클래스

Attributes
-marketRepository: MarketRepository -> MarketRepository를 의존 관계 주입을 하기 위한 필드
-rewardRepository: RewardRepository -> RewardRepository를 의존 관계 주입을 하기 위한 필드
Methods
+addMarket(market: Market): Long -> 매장 등록 로직을 실행하기 위한 메소드
+removeMarket(id: Long): void -> 매장 삭제 로직을 실행하기 위한 메소드
+updateMarket(market: Market): Market -> 매장 정보 수정 로직을 실행하기 위한 메소드
+findAllReward(id: Long): List<Reward> -> 매장의 리워드 리스트를 가져오는 로직을 실행하기 위한 메소드
+earnStamp(coupon: Coupon): EarnCoupon -> 매장의 쿠폰 적립하기 위한 메소드
+earnStampPhoneNum(phoneNum: String, market: Market): EarnCoupon -> 매장의 쿠폰을 전화번호를 통해 적립을 위한 메소드
+addEvent(eventImage: String): void -> 매장의 이벤트 등록을 실행하기 위한 메소드
+removeEvent(market: Market): void -> 매장의 이벤트 삭제를 실행하기 위한 메소드

#### 4.25. MarketRepository

MarketRepository	
-em: EntityManager	
+save(market: Market): void	
+remove(id: Long): void	
+update(market: Market): Long	
+findOne(id: Long): Market	
+findByOwnerId(ownerId: Long): List<Market>	

그림 28 MarketRepository 클래스

JPA를 이용하여 Owner와 관련된 테이블을 관리하는 클래스

Attributes
-em: EntityManager
-> 영속 컨텍스트에 접근하여 엔티티에 대한 DB 작업을 제공하는 필드
Methods
+save(market: Market): void
-> 매장 등록 정보를 DB에 저장하는 메소드
+remove(id: Long): void
-> 매장을 DB에서 삭제하기 위한 메소드
+update(market: Market): void
-> 매장 정보를 수정한 데이터를 DB에 저장하는 메소드
+findOne(id: Long): Market
-> id를 통해 DB에서 해당 매장을 찾아오기 위한 메소드
+findByOwnerId(ownerId: Long): List<Market>
-> 접수 정보를 통해 해당 접수의 매장 리스트를 가져오기 위한 메소드

#### 4.26. Reward

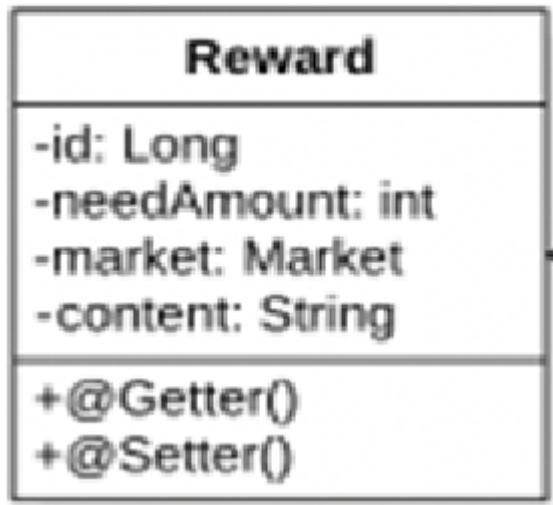


그림 29 Reward 클래스

리워드에 대한 정보를 나타내는 클래스이다. 각 리워드의 id, needAmount, market, content가 선언되어 있다. 또한 이에 대한 Getter, Setter가 있다.

Attributes
-id: Long -> DB에서 리워드를 식별하기 위한 Long형 필드
-needAmount: int -> 해당 리워드를 받기 위해 필요한 도장 개수를 저장하기 위한 int형 필드
-market: Market -> 리워드가 소속된 매장을 저장하기 위한 필드
-content: String -> 리워드 내용을 저장하기 위한 String 필드
Annotations
+@Getter -> 각 필드를 불러오기 위한 get 메소드
+@Setter -> 각 필드를 설정하기 위한 set 메소드

#### 4.27. RewardController

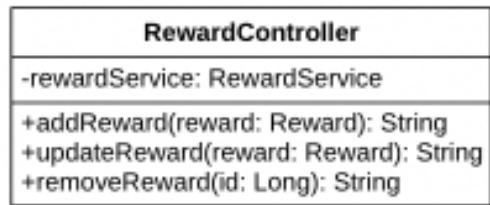


그림 30 RewardController 클래스

클라이언트의 리워드와 관련된 기능 중 해당 요청 url로 요청이 왔을 시 각 메소드로 mapping하여 필요한 기능이 있는 RewardService 클래스를 호출한다.

Attributes
-rewardService: RewardService -> RewardService를 의존 관계 주입을 위한 필드
Methods
+addReward(reward: Reward): String -> 리워드 추가를 위한 요청 url을 mapping 하는 메소드
+updateReward(reward: Reward): String -> 리워드 수정을 위한 요청 url을 mapping 하는 메소드
+removeReward(id: Long): String -> 리워드 삭제를 위한 요청 url을 mapping 하는 메소드

#### 4.28. RewardService

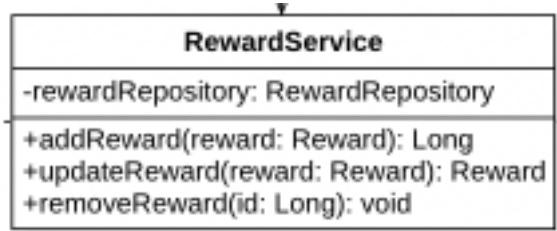


그림 31 RewardService 클래스

리워드와 관련된 3가지 기능인 리워드 등록, 리워드 삭제, 리워드 수정을 구현하는 클래스

Attributes
-rewardRepository: RewardRepository -> RewardRepository를 의존 관계 주입을 하기 위한 필드
Methods
+addReward(reward: Reward): Long -> 리워드 등록 로직을 실행하기 위한 메소드 +updateReward(reward: Reward): Reward -> 리워드 수정 로직을 실행하기 위한 메소드 +removeReward(id: Long): void -> 리워드 삭제 로직을 실행하기 위한 메소드

#### 4.29. RewardRepository

RewardRepository	
-em:	EntityManager
+save(reward: Reward): void	
+remove(id: Long): void	
+update(reward: Reward): Long	
+findOne(id: Long): Reward	
+findByMarketId(marketId: Long): List<Reward>	

그림 32 RewardRepository 클래스

JPA를 이용하여 Reward와 관련된 테이블을 관리하는 클래스

Attributes
-em: EntityManager
-> 영속 컨텍스트에 접근하여 엔티티에 대한 DB 작업을 제공하는 필드
Methods
+save(reward: Reward): void
-> 리워드 등록 정보를 DB에 저장하는 메소드
+remove(id: Long): void
-> 리워드를 DB에서 삭제하기 위한 메소드
+update(reward: Reward): Reward
-> 리워드 정보를 수정한 데이터를 DB에 저장하는 메소드
+findOne(id: Long): Reward
-> id를 통해 DB에서 해당 리워드를 찾아오기 위한 메소드
+findByMarketId(marketId: Long): List<Reward>
-> 매장 정보를 통해 해당 매장의 리워드 리스트를 가져오기 위한 메소드

#### 4.30. BoardCustomer

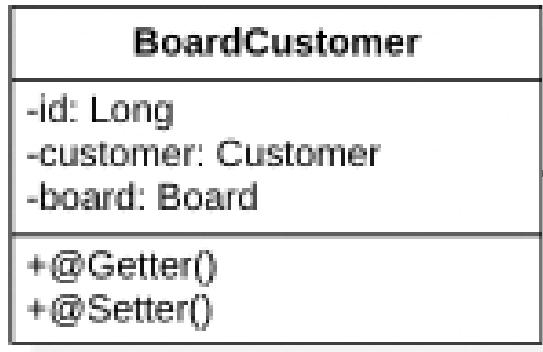


그림 33 BoardCustomer 클래스

소비자가 소유하고 있는 쿠폰 판에 대한 정보를 나타내는 클래스이다. 각 객체들의 id, customer, board가 선언되어 있다. 또한 이에 대한 Getter, Setter가 있다.

Attributes
<b>-id: Long</b> -> DB에서 매장을 식별하기 위한 Long형 필드
<b>-customer: Customer</b> -> 소비자 정보를 저장하기 위한 필드
<b>-board: Board</b> -> 쿠폰 판 정보를 저장하기 위한 필드
Annotations
<b>+@Getter</b> -> 각 필드를 불러오기 위한 get 메소드
<b>+@Setter</b> -> 각 필드를 설정하기 위한 set 메소드

#### 4.31. BoardCustomerController

BoardCustomerController	
-	boardCustomerService: BoardCustomerService
+	myBoards(customerId: Long): String

그림 34 BoardCustomerController 클래스

클라이언트의 소유 쿠폰 판 기능 중 해당 요청 url로 요청이 왔을 시 각 메소드로 mapping하여 필요한 기능이 있는 BoardCustomerService 클래스를 호출한다.

Attributes
-boardCustomerService: BoardCustomerService -> BoardCustomerService를 의존 관계 주입을 위한 필드
Methods
+myBoards(customerId: Long): String -> 소유 쿠폰 판 리스트 출력을 위한 요청 url을 mapping 하는 메소드

#### 4.32. BoardCustomerService

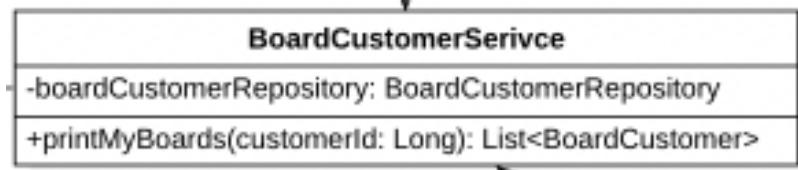


그림 35 BoardCustomerService 클래스

소비자 소유 쿠폰 판에 관한 기능인 쿠폰 판 리스트 출력을 구현하는 클래스

Attributes
<b>-boardCustomerRepository: BoardCustomerRepository</b> -> BoardCustomerRepository를 의존 관계 주입을 하기 위한 필드
Methods
<b>+printMyBoards(customerId: Long): List&lt;BoardCustomer&gt;</b> -> 해당 소비자가 소유한 쿠폰 판 리스트를 가져오기 위한 메소드

#### 4.33. BoardCustomerRepository



그림 36 BoardCustomerRepository 클래스

JPA를 이용하여 BoardCustomer와 관련된 DB를 관리하는 클래스

Attributes
<b>-em: EntityManager</b> -> 영속 컨텍스트에 접근하여 엔티티에 대한 DB 작업을 제공하는 필드
Methods
<b>+save(market: Market): void</b> -> 소유 쿠폰 판 정보를 DB에 저장하는 메소드 <b>+remove(id: Long): void</b> -> 소유 쿠폰 판 정보를 DB에서 삭제하기 위한 메소드 <b>+findOne(id: Long): Market</b> -> id를 통해 DB에서 소유 쿠폰 판 정보를 찾아오기 위한 메소드 <b>+findAll(id: Long): List&lt;BoardCustomer&gt;</b> -> 소비자 정보를 통해 해당 소비자의 소유 쿠폰 판 리스트를 가져오기 위한 메소드

#### 4.34. Board

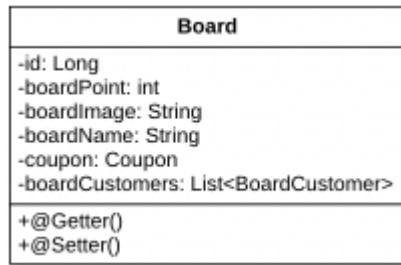


그림 37 Board 클래스

쿠폰 판에 대한 정보를 나타내는 클래스이다. 각 쿠폰 판들의 id, boardPoint, boardImage, boardName, coupon, boardCustomers가 선언되어 있다. 또한 이에 대한 Getter, Setter가 있다.

Attributes
-id: Long -> DB에서 쿠폰 판을 식별하기 위한 Long형 필드
-boardPoint: int -> 쿠폰 판 구매를 위한 포인트를 저장하기 위한 String형 필드
-boardImage: String -> 쿠폰 판 이미지의 저장 경로를 저장하기 위한 String형 필드
-boardName: String -> 쿠폰 판 이름을 저장하기 위한 String형 필드
-coupon: Coupon -> 쿠폰 판의 쿠폰 정보를 저장하기 위한 필드
-boardCustomers: List<BoardCustomer> -> 쿠폰 판을 소유하고 있는 소비자들을 저장하기 위한 필드
Annotations
+@Getter -> 각 필드를 불러오기 위한 get 메소드
+@Setter -> 각 필드를 설정하기 위한 set 메소드

#### 4.35. BoardController

BoardController
-boardService: BoardService
+addBoard(board: Board): String
+findBoard(id: Long): String
+deleteBoard(id: Long): String

그림 38 BoardController 클래스

클라이언트의 쿠폰 판 기능 중 해당 요청 url로 요청이 왔을 시 각 메소드로 mapping하여 필요한 기능이 있는 boardService 클래스를 호출한다.

Attributes
-boardService: BoardService -> BoardService를 의존 관계 주입을 위한 필드
Methods
+addBoard(board: Board): String -> 쿠폰 판 추가를 위한 요청 url을 mapping 하는 메소드
+findBoard(id: Long): String -> 해당 쿠폰을 가져오기 위한 요청 url을 mapping 하는 메소드
+deleteBoard(id: Long): String -> 쿠폰 판 삭제를 위한 요청 url을 mapping 하는 메소드

#### 4.36. BoardService

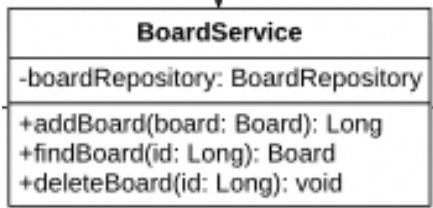


그림 39 BoardService 클래스

쿠폰 판에 관한 3가지 기능인 쿠폰 판 등록, 쿠폰 판 찾기, 쿠폰 판 삭제를 구현하는 클래스

Attributes
-boardRepository: BoardRepository -> BoardRepository를 의존 관계 주입을 하기 위한 필드
Methods
+addBoard(board: Board): Long -> 쿠폰 판 등록 로직을 실행하기 위한 메소드
+findBoard(id: Long): Board -> 해당 쿠폰 판을 가져오는 로직을 실행하기 위한 메소드
+deleteBoard(id: Long): void -> 쿠폰 판 삭제 로직을 실행하기 위한 메소드

#### 4.37. BoardRepository

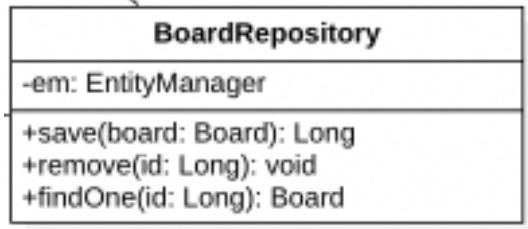


그림 40 BoardRepository 클래스

JPA를 이용하여 Board와 관련된 DB를 관리하는 클래스

Attributes
<code>-em: EntityManager</code> -> 영속 컨텍스트에 접근하여 엔티티에 대한 DB 작업을 제공하는 필드
Methods
<code>+save(market: Market): void</code> -> 쿠폰 판 등록 정보를 DB에 저장하는 메소드 <code>+remove(id: Long): void</code> -> 쿠폰 판을 DB에서 삭제하기 위한 메소드 <code>+findOne(id: Long): Board</code> -> id를 통해 DB에서 해당 쿠폰 판을 찾아오기 위한 메소드

## 5. Sequence Diagram

### 5.1. 회원

#### 5.1.1. 회원가입

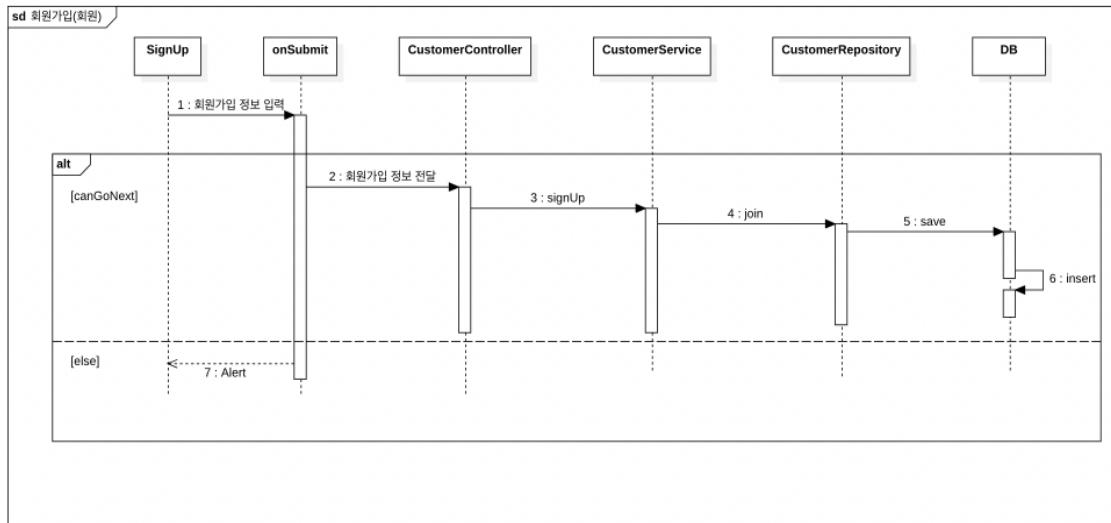


그림 41 회원가입(회원) sequence diagram

회원이 회원가입을 할 때의 절차를 나타낸 Sequence Diagram이다. 회원가입 컴포넌트인 SignUp에서 회원가입 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 회원가입 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 signUp 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 join 메소드가 호출된다. 호출된 join 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 save 메소드를 호출하여 DB에 회원가입 정보를 insert한다.

### 5.1.2. 로그인

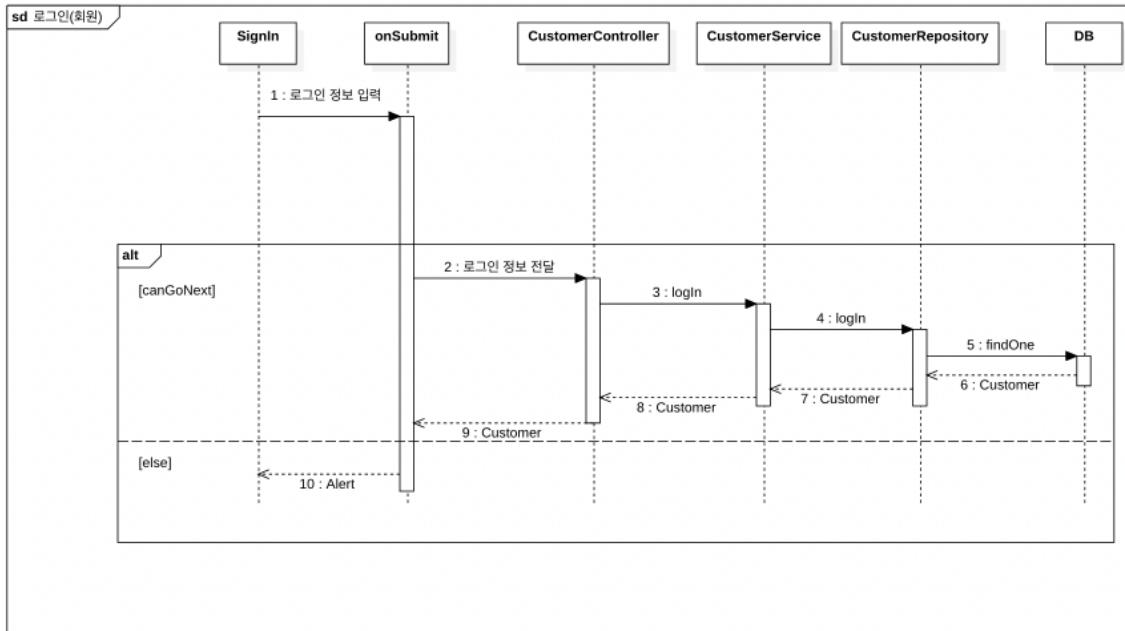


그림 42 로그인(회원) sequence diagram

회원이 로그인을 할 때의 절차를 나타낸 Sequence Diagram 이다. 로그인 컴포넌트인 SignIn에서 로그인 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 로그인 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 logIn 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 logIn 메소드가 호출된다. 호출된 logIn 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 findOne 메소드를 호출한다. 호출된 findOne 메소드를 통해 DB에서 해당 회원의 정보를 찾아 CustomerService 클래스에서 로그인 정보를 비교하여 결과 값을 CustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.3. 로그아웃

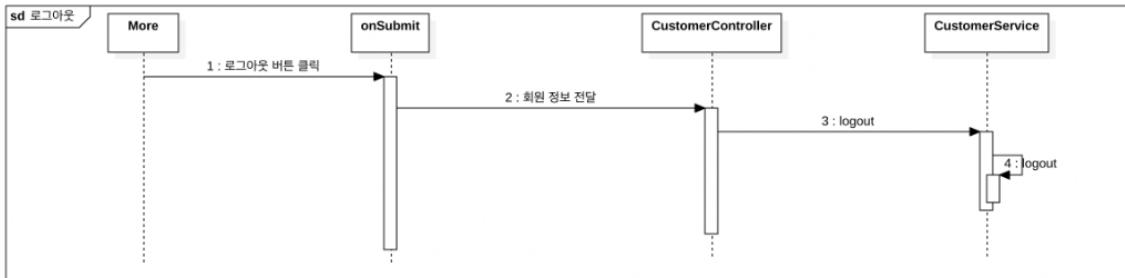


그림 43 로그아웃(회원) sequence diagram

회원이 로그아웃을 할 때의 절차를 나타낸 Sequence Diagram 이다. 로그아웃 컴포넌트인 More에서 로그아웃 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 로그아웃하는 회원의 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 logout 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 logout 메소드가 호출된다. 호출된 logIn 메소드에서 로그아웃 로직이 실행된다.

#### 5.1.4. 아이디 찾기

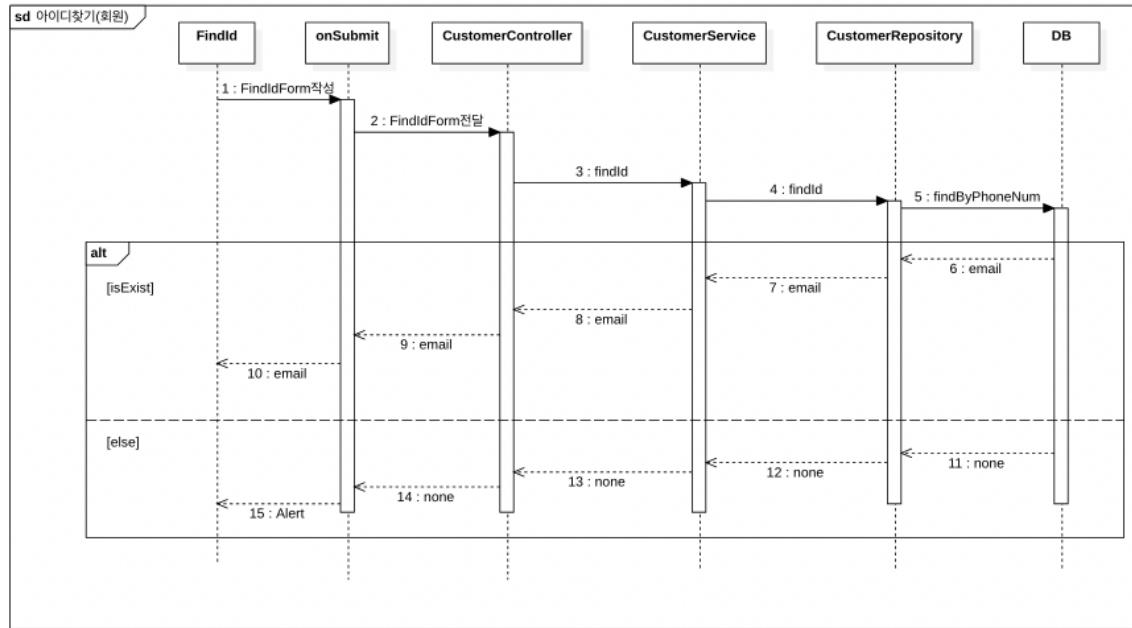


그림 44 아이디 찾기(회원) sequence diagram

회원이 아이디 찾기를 할 때의 절차를 나타낸 Sequence Diagram 이다. 아이디 찾기 컴포넌트인 FindId 에서 아이디 찾기에 필요한 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 on Submit 함수를 이용하여 서버로 아이디 찾기 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 findId 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 findId 메소드가 호출된다. 호출된 findId 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 findByPhoneNum 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원의 휴대폰 번호 통해 회원을 찾는다. 해당 데이터가 없는 경우 none을 반환하고 있는 경우 회원이 CustomerService 클래스에 반환되어 CustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.5. 비밀번호 찾기

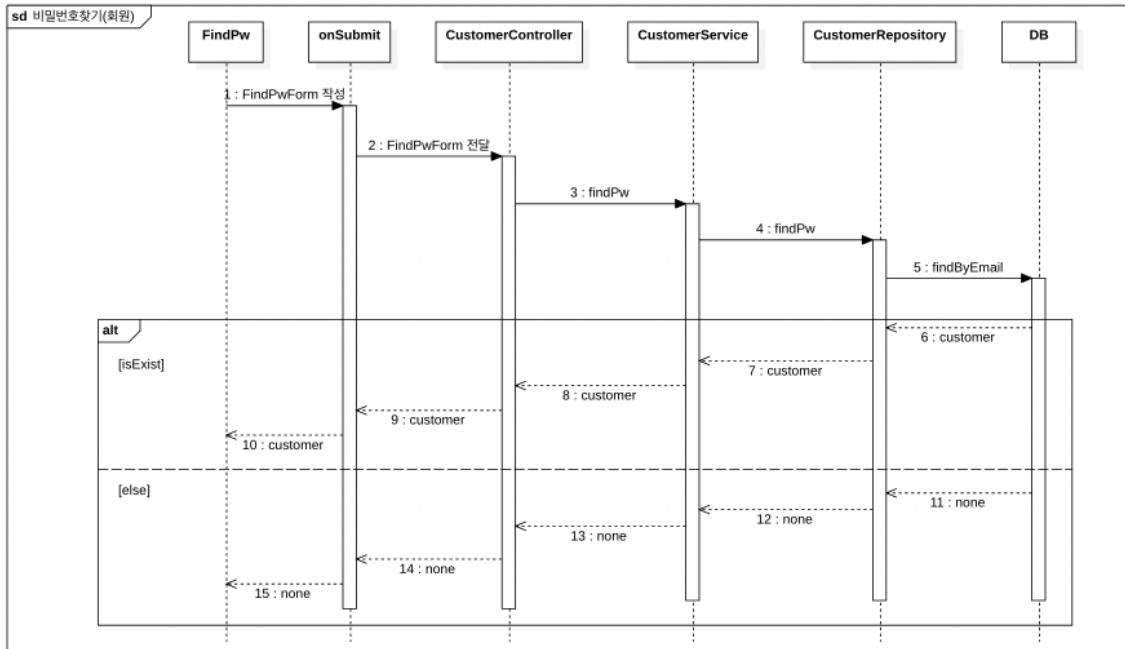


그림 45 비밀번호 찾기(회원) sequence diagram

회원이 비밀번호 찾기를 할 때의 절차를 나타낸 Sequence Diagram 이다. 비밀번호 찾기 컴포넌트인 FindPw 에서 비밀번호 찾기에 필요한 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 비밀번호 찾기 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 findPw 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 findPw 메소드가 호출된다. 호출된 findPw 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 findByEmail 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원의 이메일을 통해 회원을 찾는다. 해당 데이터가 없는 경우 none을 반환하고 있는 경우 회원이 CustomerService 클래스에 반환되어 CustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.6. 회원 정보 수정

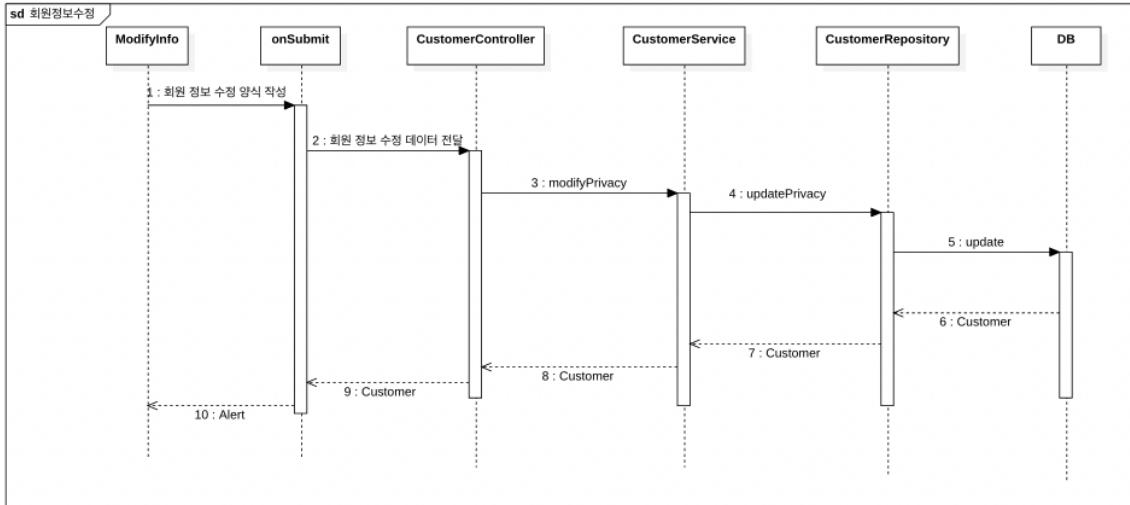


그림 46 회원 정보 수정(회원) sequence diagram

회원이 회원 정보 수정을 할 때의 절차를 나타낸 Sequence Diagram 이다. 회원 정보 수정 컴포넌트인 ModifyInfo 에서 수정할 회원 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 수정한 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 modifyPrivacy 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 updatePrivacy 메소드가 호출된다. 호출된 updatePrivacy 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에서 해당 회원을 찾아 회원 정보를 수정한다.

### 5.1.7. 메인화면

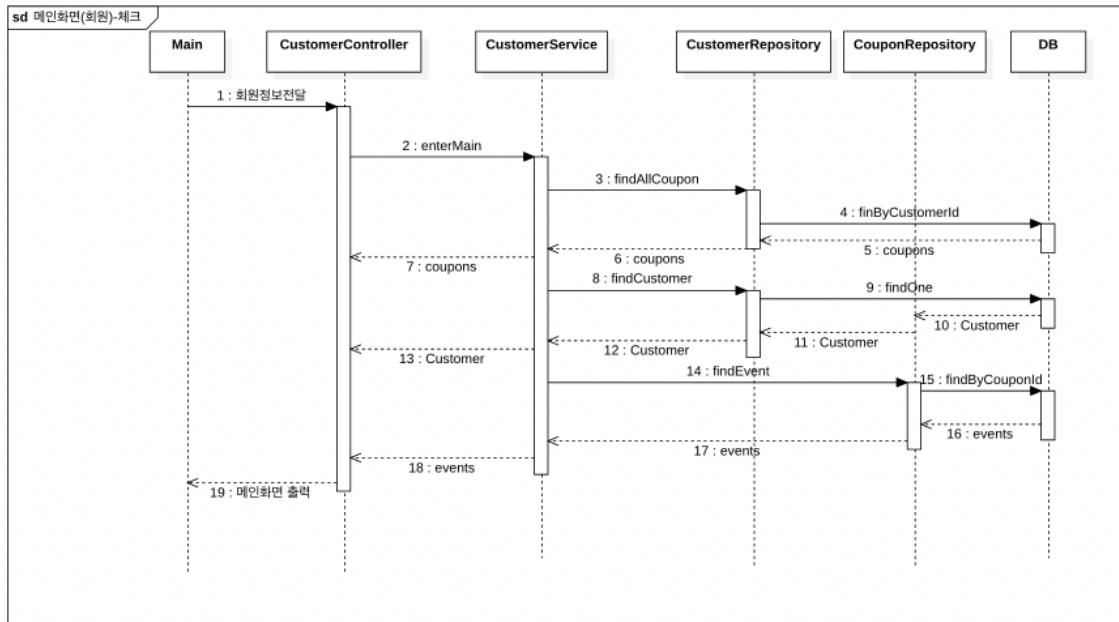


그림 47 메인화면(회원) sequence diagram

회원이 메인화면에 들어갈 때 절차를 나타낸 Sequence Diagram 이다. 메인화면 컴포넌트인 Main에서 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 Customer Controller 클래스를 이용해 요청 url에 mapping 되어 enterMain 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 findAllCoupon, findCustomer, findEvent 메소드가 호출된다. 호출된 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스에서 각각 findByCustomerId, findOne을 호출하고 CouponRepository에서 findByCouponId 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원이 가진 쿠폰 리스트, 회원 보정, 쿠폰이 등록된 매장의 이벤트 리스트가 CustomerService 클래스에 반환되어 CustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.8. 쿠폰 적립 QR 생성

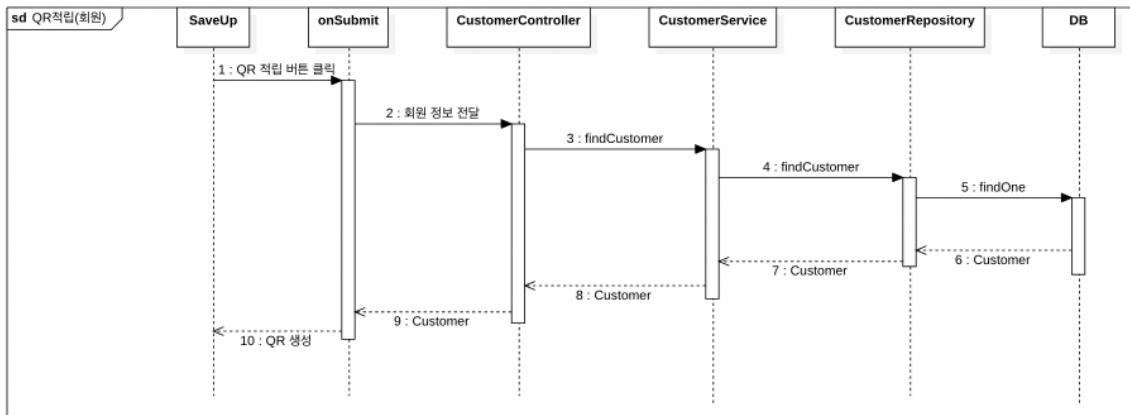


그림 48 쿠폰 적립 QR 생성(회원) sequence diagram

회원이 쿠폰 적립 QR 생성을 할 때의 절차를 나타낸 Sequence Diagram 이다. 회원 쿠폰 QR 컴포넌트인 SaveUp에서 QR 적립 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 적립하는 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 findCustomer 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 findCustomer 메소드가 호출된다. 호출된 findCustomer 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 findOne 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원의 id를 통해 회원을 찾는다. 해당 회원은 CustomerService 클래스에 반환되어 CustomerController 클래스를 거쳐 view로 값을 리턴한다. view에서는 리턴된 값으로 적립 QR을 생성한다.

### 5.1.9. 쿠폰 사용 QR 생성

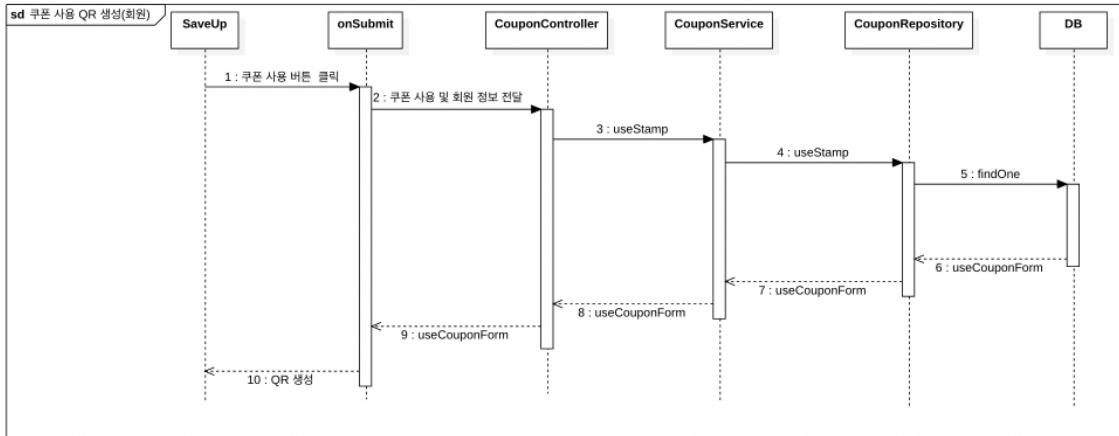


그림 49 쿠폰 사용 QR 생성(회원) sequence diagram

회원이 쿠폰 사용 QR 생성을 할 때의 절차를 나타낸 Sequence Diagram 이다. 회원 쿠폰 QR 컴포넌트인 SaveUp 에서 쿠폰 사용 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 쿠폰 사용 정보와 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 CouponController 클래스를 이용해 요청 url에 mapping 되어 useStamp 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CouponService 클래스의 useStamp 메소드가 호출된다. 호출된 useStamp 메소드를 이용해서 DB에 접근하는 Coupon Repository 클래스의 findOne 메소드를 호출한다. 호출된 메소드를 통해 DB에서 쿠폰의 id를 통해 쿠폰 정보를 찾아 useCouponForm을 생성한다. 해당 useCouponForm은 CouponService 클래스에 반환되어 CouponController 클래스를 거쳐 view로 값을 리턴한다. view에서는 리턴된 값으로 쿠폰 사용 QR을 생성한다.

### 5.1.10. 쿠폰 판 커스터마이징 리스트

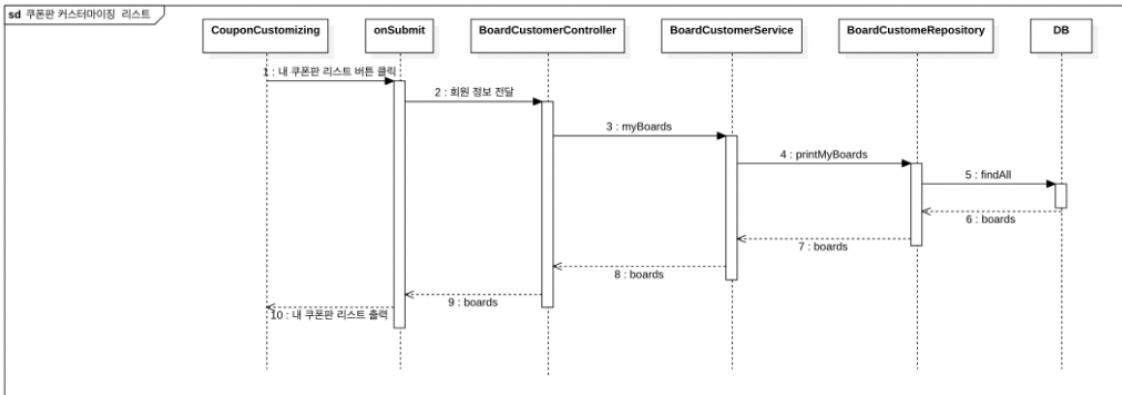


그림 50 쿠폰 판 커스터마이징 리스트(회원) sequence diagram

회원이 소유한 쿠폰 판 커스터마이징 리스트 출력을 할 때의 절차를 나타낸 Sequence Diagram 이다. 커스터마이징 컴포넌트인 CouponCustomizing 에서 내 쿠폰 판 리스트 사용 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 BoardCustomerController 클래스를 이용해 요청 url에 mapping 되어 myBoards 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 BoardCustomerService 클래스의 printMyBoards 메소드가 호출된다. 호출된 printMyBoards 메소드를 이용해서 DB에 접근하는 BoardCustomerRepository 클래스의 findAll 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원의 id를 통해 쿠폰 판 정보를 찾아 회원이 소유한 쿠폰 판 리스트를 생성한다. 해당 리스트 BoardCustomerService 클래스에 반환되어 BoardCustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.11. 도장 커스터마이징 리스트

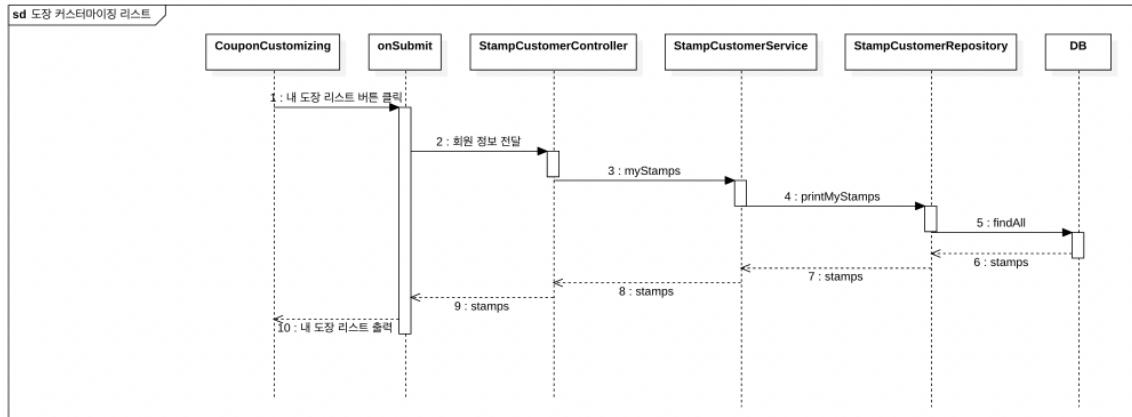


그림 51 도장 커스터마이징 리스트(회원) sequence diagram

회원이 소유한 도장 커스터마이징 리스트 출력을 할 때의 절차를 나타낸 Sequence Diagram 이다. 커스터마이징 컴포넌트인 CouponCustomizing 에서 내 쿠폰 판 리스트 사용 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 회원 정보를 서버로 전달한다. 서버로 전달된 데이터는 StampCustomerController 클래스를 이용해 요청 url에 mapping 되어 myStamps 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 StampCustomerService 클래스의 printMyStamps 메소드가 호출된다. 호출된 printMyStamps 메소드를 이용해서 DB에 접근하는 StampCustomerRepository 클래스의 findAll 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원의 id를 통해 도장 정보를 찾아 회원이 소유한 도장 리스트를 생성한다. 해당 리스트는 StampCustomerService 클래스에 반환되어 StampCustomerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.12. 쿠폰 판 변경

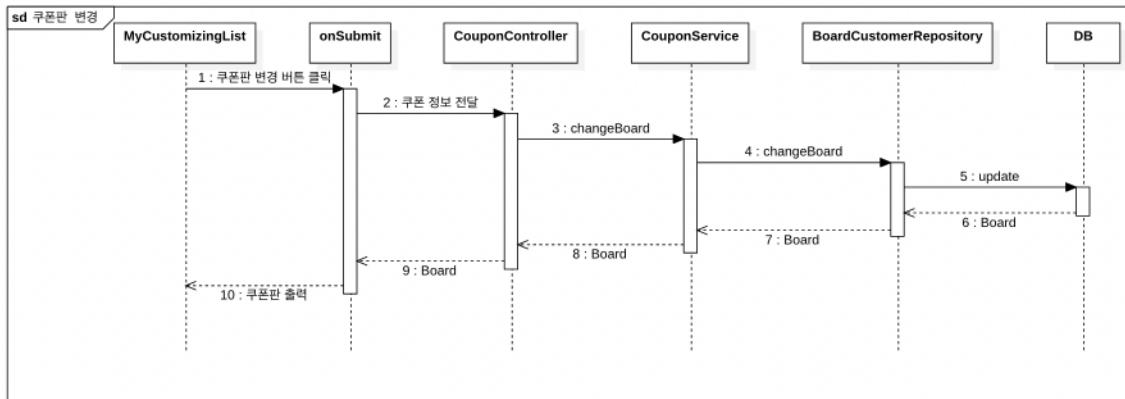


그림 52 쿠폰 판 변경(회원) sequence diagram

회원이 쿠폰의 쿠폰 판 변경을 할 때의 절차를 나타낸 Sequence Diagram 이다. 커스터마이징 컴포넌트인 MyCustomizingList 에서 쿠폰 판 변경 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 쿠폰 정보를 서버로 전달한다. 서버로 전달된 데이터는 CouponController 클래스를 이용해 요청 url에 mapping 되어 change Board 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CouponService 클래스의 changeBoard 메소드가 호출된다. 호출된 changeBoard 메소드를 이용해서 DB에 접근하는 BoardCustomerRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에서 선택된 쿠폰 판 정보를 찾는다. 해당 정보는 CouponService 클래스에 반환되어 CouponController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.13. 도장 변경

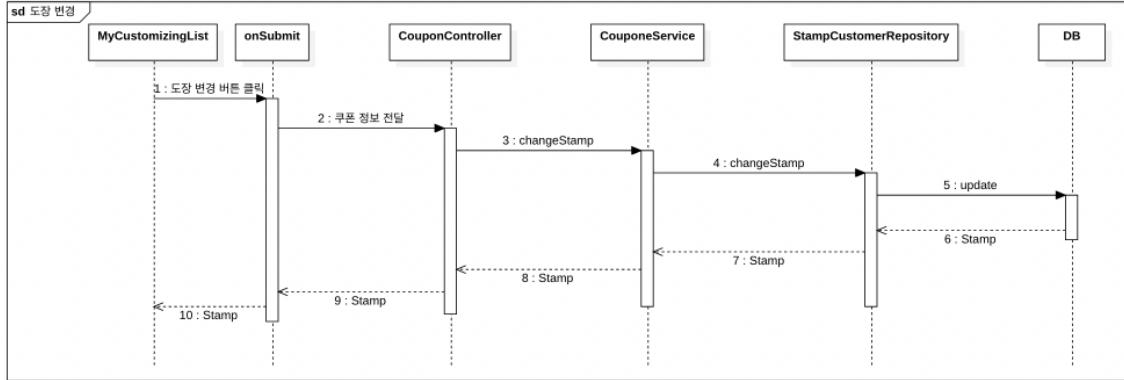


그림 53 도장 변경(회원) sequence diagram

회원이 쿠폰의 도장 변경을 할 때의 절차를 나타낸 Sequence Diagram 이다. 커스터マイ징 변경 컴포넌트인 MyCustomizingList 에서 도장 변경 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 쿠폰 정보를 서버로 전달한다. 서버로 전달된 데이터는 CouponController 클래스를 이용해 요청 url에 mapping 되어 changeStamp 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CouponService 클래스의 changeStamp 메소드가 호출된다. 호출된 changeStamp 메소드를 이용해서 DB에 접근하는 StampCustomerRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에서 선택된 도장 정보를 찾는다. 해당 정보 CouponService 클래스에 반환되어 CouponController 클래스를 거쳐 view로 값을 리턴한다.

### 5.1.14. 쿠폰 리스트

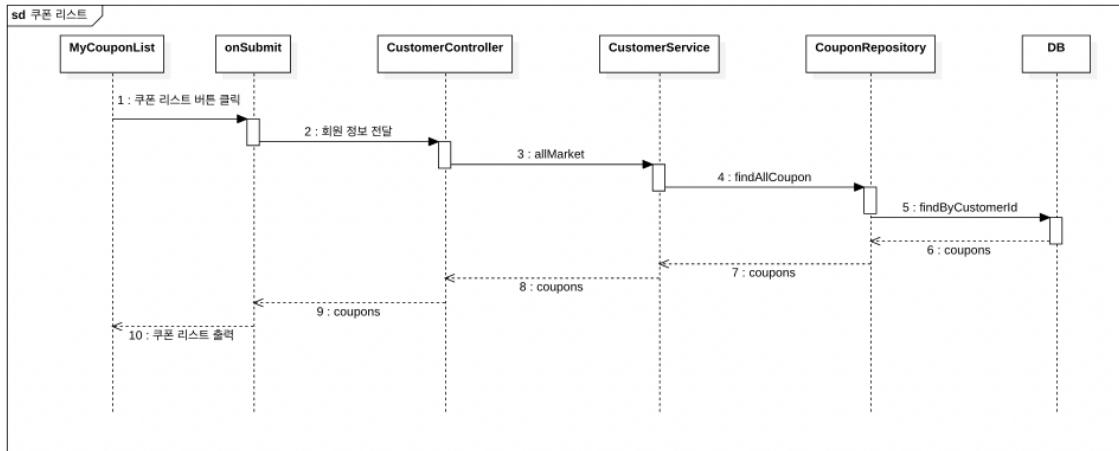


그림 54 쿠폰 리스트(회원) sequence diagram

회원이 쿠폰 리스트를 볼 때의 절차를 나타낸 Sequence Diagram이다. 쿠폰 리스트 출력 컴포넌트인 MyCouponList에서 쿠폰 리스트 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 회원 정보를 서버로 전달한다. 서버로 전달된 데 이터는 CustomerController 클래스를 이용해 요청 url에 mapping 되어 allMarket 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 CustomerService 클래스의 findAllCoupon 메소드가 호출된다. 호출된 findAllCoupon 메소드를 이용해서 DB에 접근하는 Couponrepository 클래스의 findByCustomerId 메소드를 호출한다. 호출된 메소드를 통해 DB에서 회원 소유의 쿠폰들을 찾는다. 해당 쿠폰들을 리스트로 CustomerService 클래스에 반환되어 CouponRepository 클래스를 거쳐 view로 값을 리턴한다.

## 5.2. 점주

### 5.2.1. 회원가입

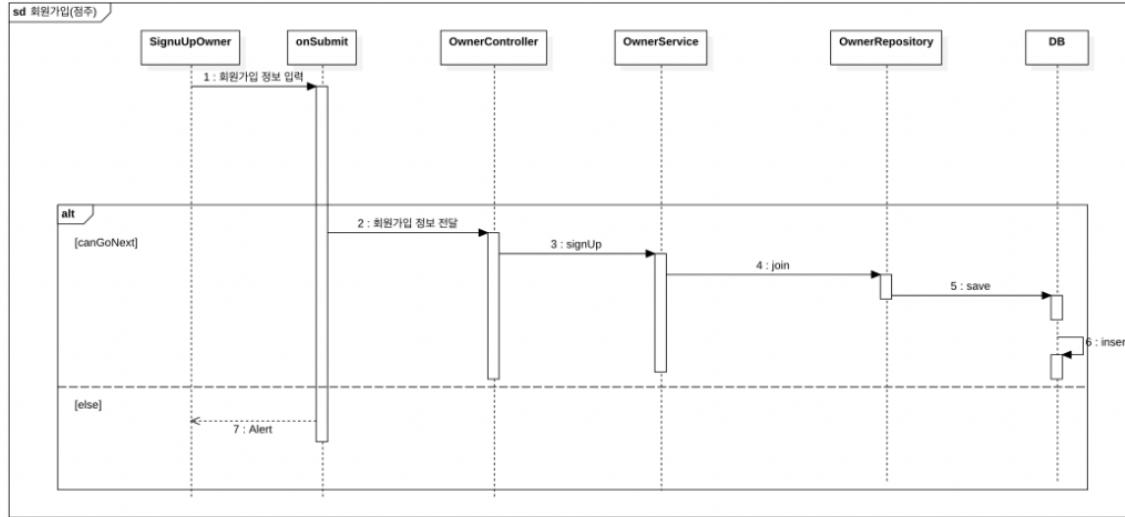


그림 55 회원가입(점주) sequence diagram

점주가 회원가입을 할 때의 절차를 나타낸 Sequence Diagram 이다. 점주 회원가입 컴포넌트인 SignUpOwner 에서 회원가입 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 회원가입 정보를 서버로 전달한다. 서버로 전달 된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 signUp 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 join 메소드가 호출된다. 호출된 join 메소드를 이용해서 DB에 접근하는 OwnerRepository 클래스의 save 메소드를 호출하여 DB에 회원가입 정보를 insert한다.

### 5.2.2. 로그인

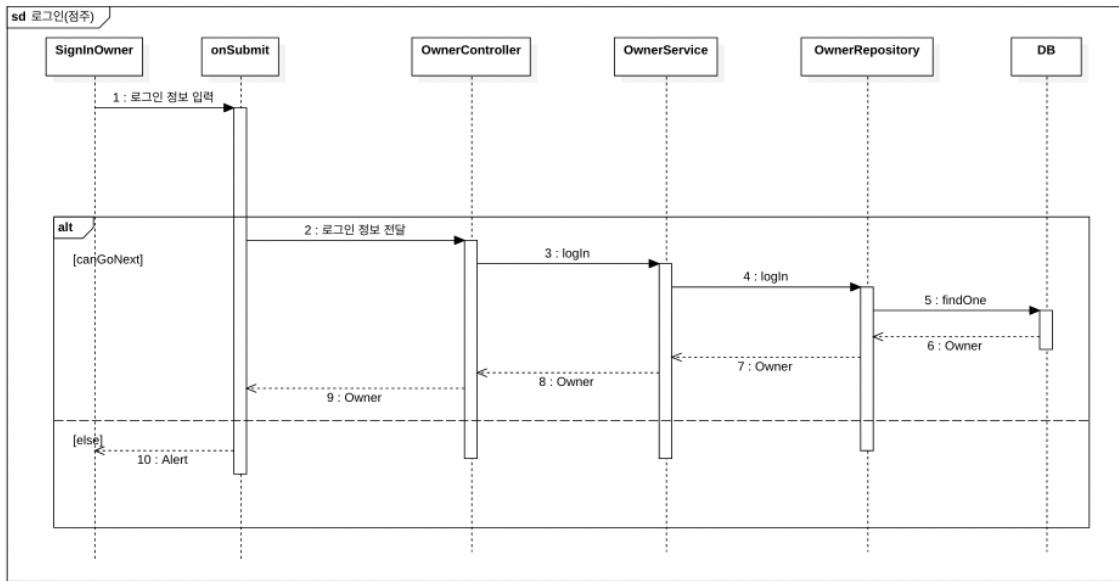


그림 56 로그인(점주) sequence diagram

점주가 로그인을 할 때의 절차를 나타낸 Sequence Diagram 이다. 로그인 컴포넌트인 SignInOwner 에서 로그인 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 로그인 정보를 서버로 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 logIn 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 logIn 메소드가 호출된다. 호출된 logIn 메소드를 이용해서 DB에 접근하는 OwnerRepository 클래스의 findOne 메소드를 호출한다. 호출된 findOne 메소드를 통해 DB에서 해당 회원의 정보를 찾아 OwnerService 클래스에서 로그인 정보를 비교하여 결과 값을 OwnerController 클래스를 거쳐 view로 값을 리턴한다.

### 5.2.3. 로그아웃

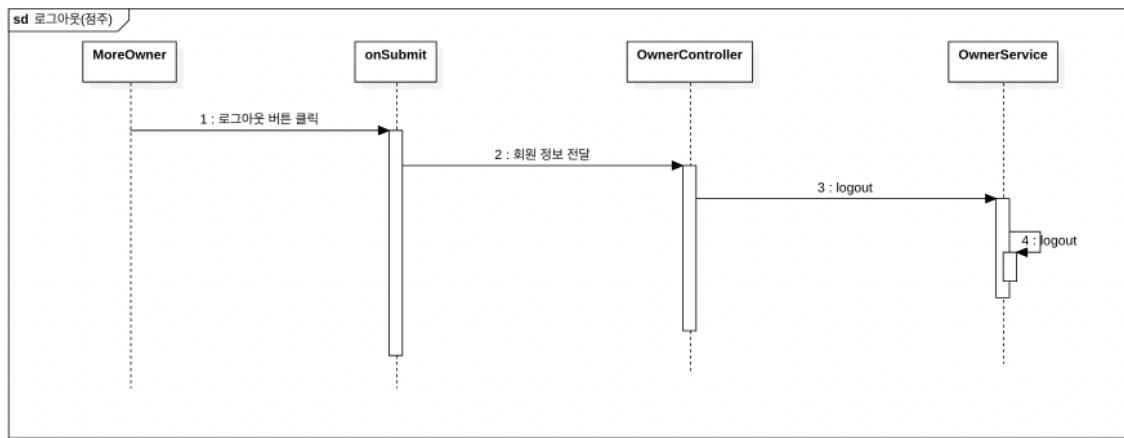


그림 57 로그아웃(점주) sequence diagram

점주가 로그아웃을 할 때의 절차를 나타낸 Sequence Diagram 이다. 로그아웃 컴포넌트인 MoreOwner에서 로그아웃 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 로그아웃하는 점주의 정보를 서버로 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 logout 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 logout 메소드가 호출된다. 호출된 logout 메소드에서 로그아웃 로직이 실행된다.

#### 5.2.4. 아이디 찾기

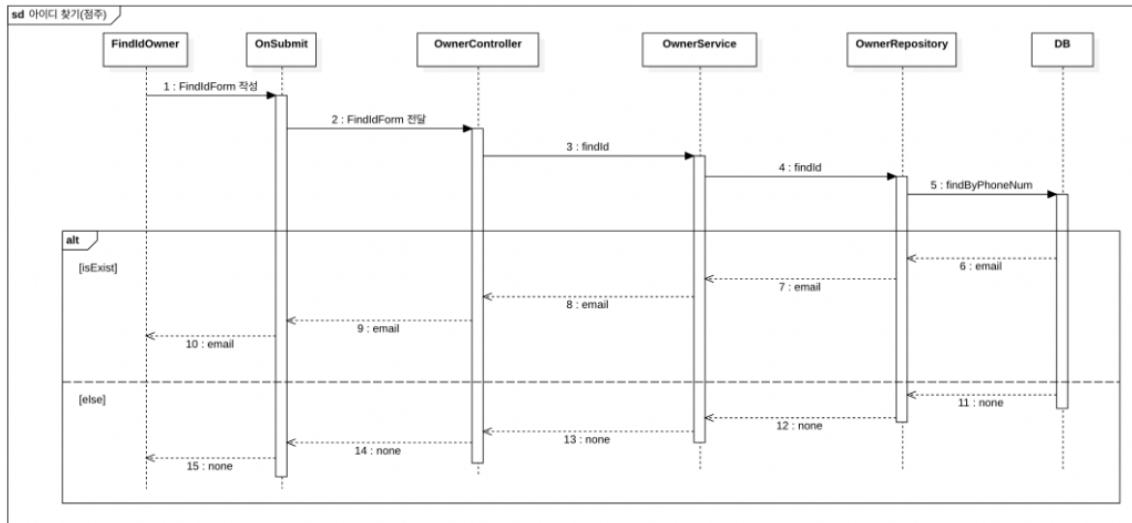


그림 58 아이디 찾기(점주) sequence diagram

점주가 아이디 찾기를 할 때의 절차를 나타낸 Sequence Diagram이다. 아이디 찾기 컴포넌트인 **FindIdOwner**에서 아이디 찾기에 필요한 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 **onSubmit** 함수를 이용하여 서버로 아이디 찾기 정보를 서버로 전달한다. 서버로 전달된 데이터는 **OwnerController** 클래스를 이용해 요청 url에 mapping 되어 **findId** 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 **OwnerService** 클래스의 **findId** 메소드가 호출된다. 호출된 **findId** 메소드를 이용해서 DB에 접근하는 **OwnerRepository** 클래스의 **findByPhoneNum** 메소드를 호출한다. 호출된 메소드를 통해 DB에서 점주의 휴대폰 번호 통해 점주를 찾는다. 해당 데이터가 없는 경우 **none**을 반환하고 있는 경우 점주가 **OwnerService** 클래스에 반환되어 **OwnerController** 클래스를 거쳐 view로 값을 리턴한다.

### 5.2.5. 비밀번호 찾기

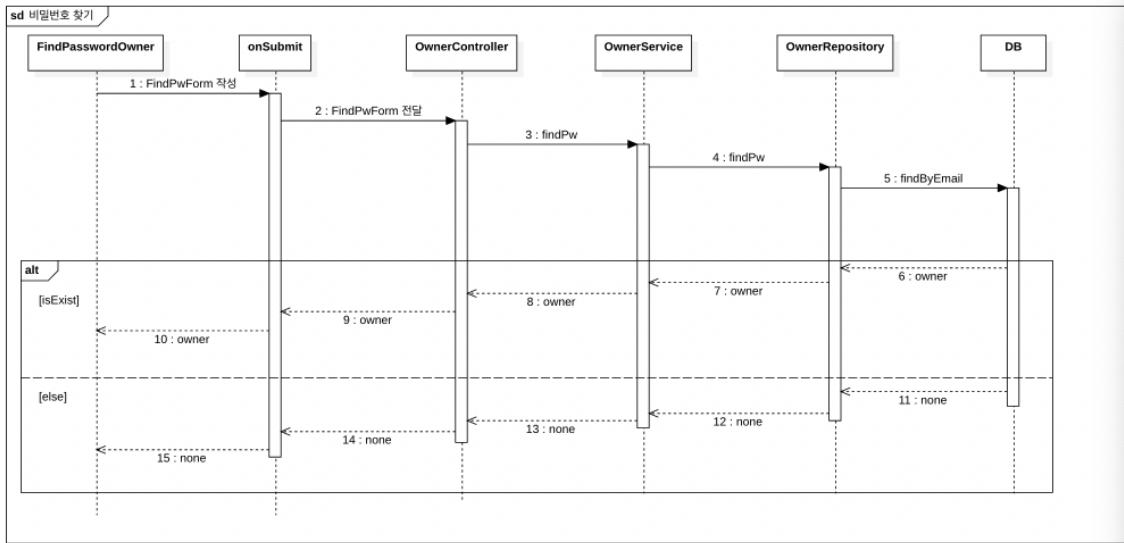


그림 59 비밀번호 찾기(점주) sequence diagram

점주가 비밀번호 찾기를 할 때의 절차를 나타낸 Sequence Diagram 이다. 비밀번호 찾기 컴포넌트인 FindPwOwner 에서 비밀번호 찾기에 필요한 정보를 입력한다. 이 과정에서 입력 정보가 원하는 양식에 맞지 않는 경우 알림 창을 보여준다. 원하는 양식에 맞는 경우 onSubmit 함수를 이용하여 서버로 비밀번호 찾기 정보를 서버로 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url 에 mapping 되어 findPw 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 findPw 메소드가 호출된다. 호출된 findPw 메소드를 이용해서 DB에 접근하는 OwnerRepository 클래스의 findByEmail 메소드를 호출한다. 호출된 메소드를 통해 DB에서 점주의 이메일을 통해 회원을 찾는다. 해당 데이터가 없는 경우 none을 반환하고 있는 경우 점주가 OwnerService 클래스에 반환되어 Owner Controller 클래스를 거쳐 view로 값을 리턴한다.

### 5.2.6. 회원탈퇴

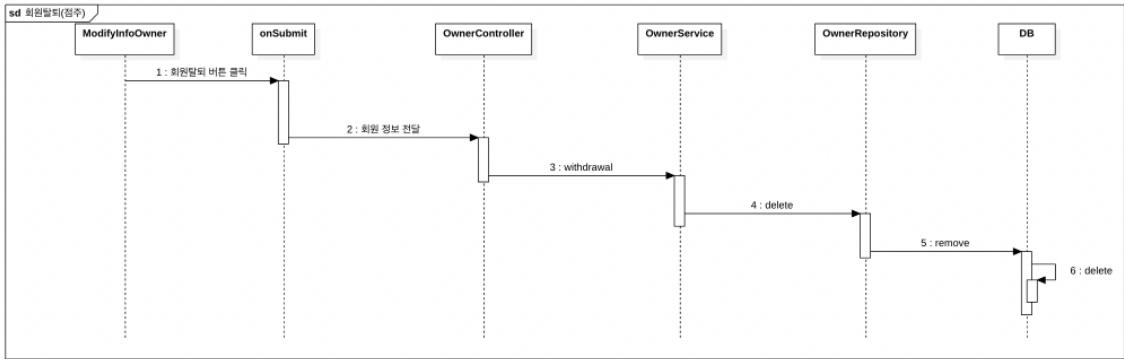


그림 60 회원탈퇴(점주) sequence diagram

점주가 회원탈퇴를 할 때의 절차를 나타낸 Sequence Diagram 이다. 점주 회원탈퇴 컴포넌트인 ModifyInfoOwner 에서 회원탈퇴 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 해당 점주 정보를 서버로 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 withdrawal 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 withdrawal 메소드가 호출된다. 호출된 withdrawal 메소드를 이용해서 DB에 접근하는 OwnerRepository 클래스의 remove 메소드를 호출한다. 호출된 메소드를 통해 DB에서 해당 점주 정보를 삭제한다.

### 5.2.7. 매장 등록

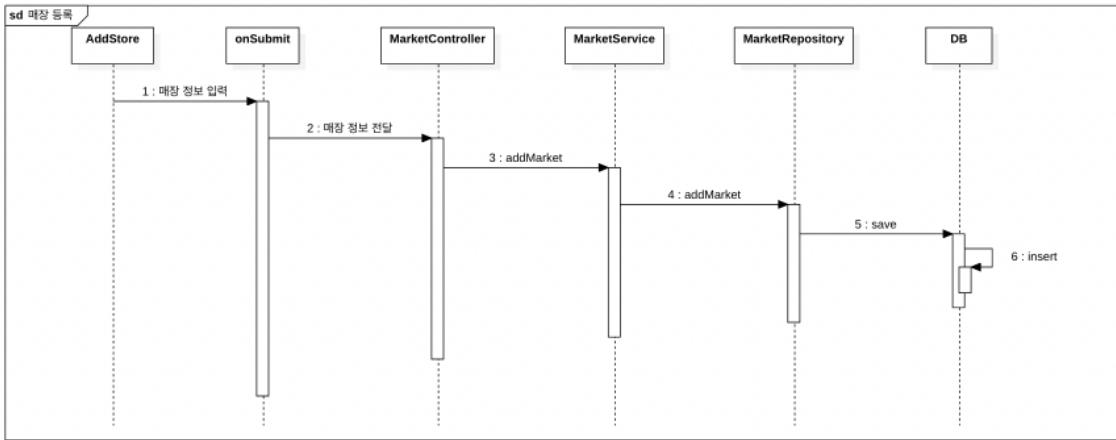


그림 61 매장 등록 sequence diagram

점주가 매장 등록을 할 때의 절차를 나타낸 Sequence Diagram 이다. 매장등록 컴포넌트인 AddStore 에서 매장 등록 정보를 입력한다. onSubmit 함수를 이용하여 서버로 수정한 매장 정보를 서버로 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 addMarket 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 addMarket 메소드가 호출된다. 호출된 addMarket 메소드를 이용해서 DB에 접근하는 MarketRepository 클래스의 save 메소드를 호출한다. 호출된 메소드를 통해 DB에서 해당 매장 정보를 insert한다.

### 5.2.8. 매장 삭제

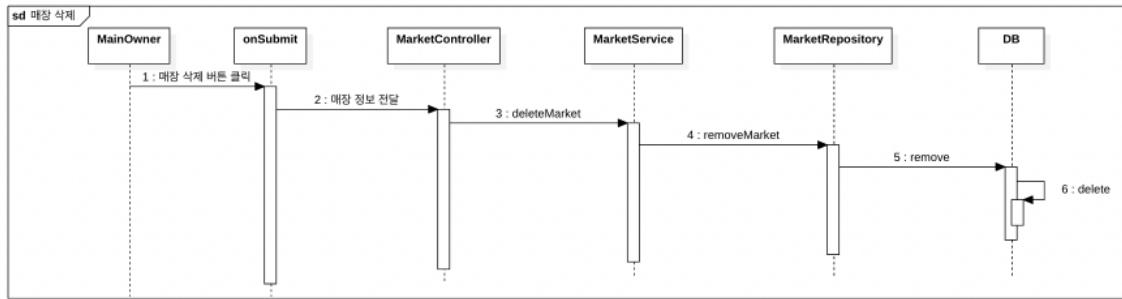


그림 62 매장 삭제 sequence diagram

점주가 매장 삭제를 할 때의 절차를 나타낸 Sequence Diagram 이다. 매장등록 컴포넌트인 MainOwner 에서 매장 삭제 버튼을 클릭한다. onSubmit 함수를 이용하여 서버로 삭제할 매장 정보를 서버로 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 deleteMarket 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 deleteMarket 메소드가 호출된다. 호출된 deleteMarket 메소드를 이용해서 DB에 접근하는 MarketRepository 클래스의 remove 메소드를 호출한다. 호출된 remove 메소드를 통해 DB에서 해당 매장 정보를 delete한다.

### 5.2.9. 매장 리스트

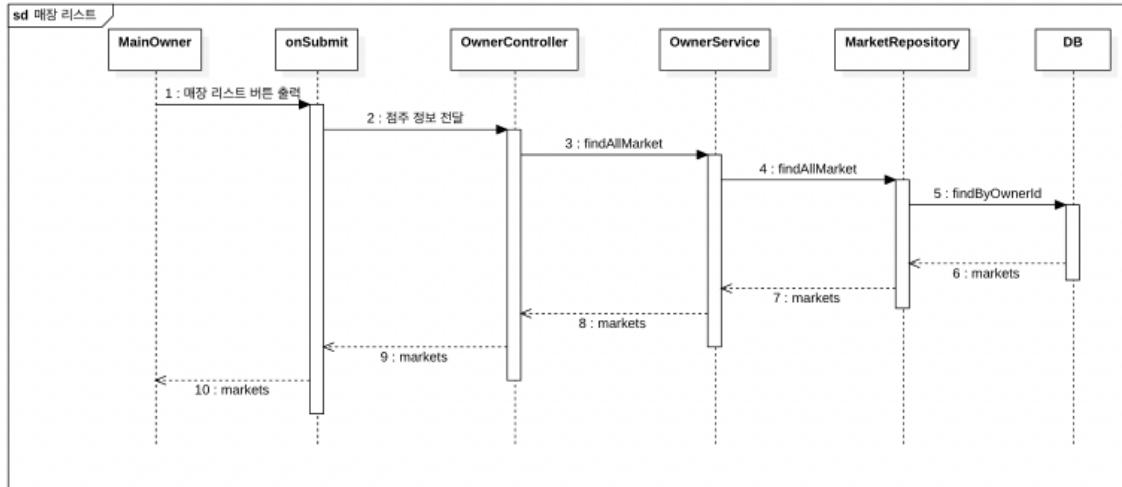


그림 63 매장 리스트 sequence diagram

점주가 매장 리스트를 출력할 때의 절차를 나타낸 Sequence Diagram 이다. 매장 리스트 출력 컴포넌트인 MainOwner 에서 매장 리스트 버튼을 클릭한다. onSubmit 함수를 이용하여 서버로 점주의 정보를 서버로 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 findAllMarket 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 OwnerService 클래스의 findAllMarket 메소드가 호출된다. 호출된 findAllMarket 메소드를 이용해서 DB에 접근하는 MarketRepository 클래스의 findByOwnerId 메소드를 호출한다. 호출된 메소드를 통해 DB에서 해당 매장 리스트를 반환한다.

### 5.2.10. QR 코드를 통한 도장 적립

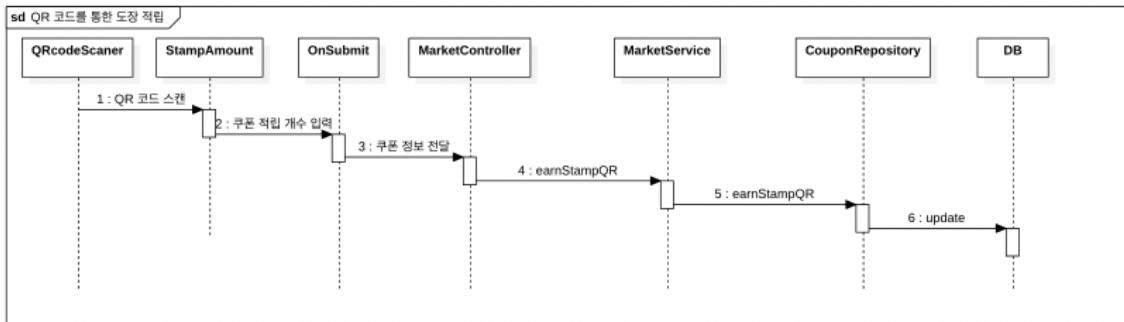


그림 64 QR 코드를 통한 도장 적립 sequence diagram

점주가 회원이 QR을 통해 도장 적립을 할 때의 절차를 나타낸 Sequence Diagram 이다. QR 코드를 통한 도장 적립 컴포넌트인 QRcodeScanner에서 QR 코드 스캔을 하고 StampAmount 컴포넌트에서 쿠폰 적립 개수 입력을 한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 적립 정보를 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 earnStampQR 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 earnStampQR 메소드가 호출된다. 호출된 earnStampQR 메소드를 이용해서 DB에 접근하는 CouponRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에서 쿠폰의 id를 통해 쿠폰 정보를 찾아 쿠폰 정보를 update한다.

### 5.2.11. 전화번호를 통한 도장 적립

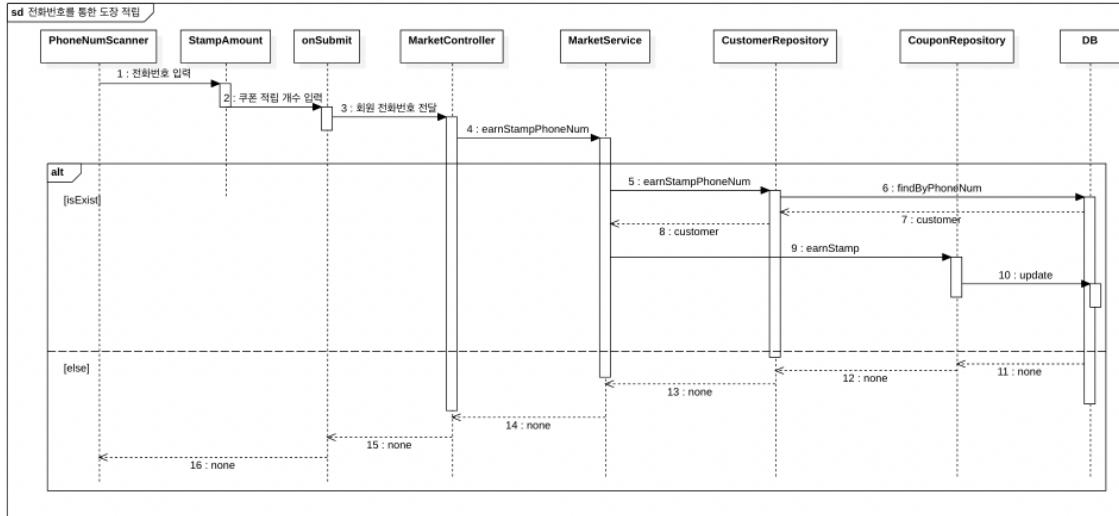


그림 65 전화번호를 통한 도장 적립 sequence diagram

점주가 회원이 전화번호를 통해 도장 적립을 할 때의 절차를 나타낸 Sequence Diagram 이다. 전화번호를 통한 도장 적립 컴포넌트인 PhoneNumScanner에서 전화 번호를 입력 받고 StampAmount 컴포넌트에서 쿠폰 적립 개수 입력을 한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 적립 정보를 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 earnStampPhoneNum 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 earnStampPhoneNum 메소드가 호출된다. 호출된 earnStampPhoneNum 메소드를 이용해서 DB에 접근하는 CustomerRepository 클래스의 findByPhoneNum 메소드를 호출한다. 호출된 메소드를 통해 DB에서 전화번호를 통해 회원 정보를 찾아 반환하고 회원 정보가 없는 경우 none이 반환된다. 반환된 회원정보를 파라미터로둔 earnStamp 메소드가 호출되고 CouponRepository 클래스의 update 메소드로 쿠폰 정보가 생성된다.

### 5.2.12. 이벤트 등록

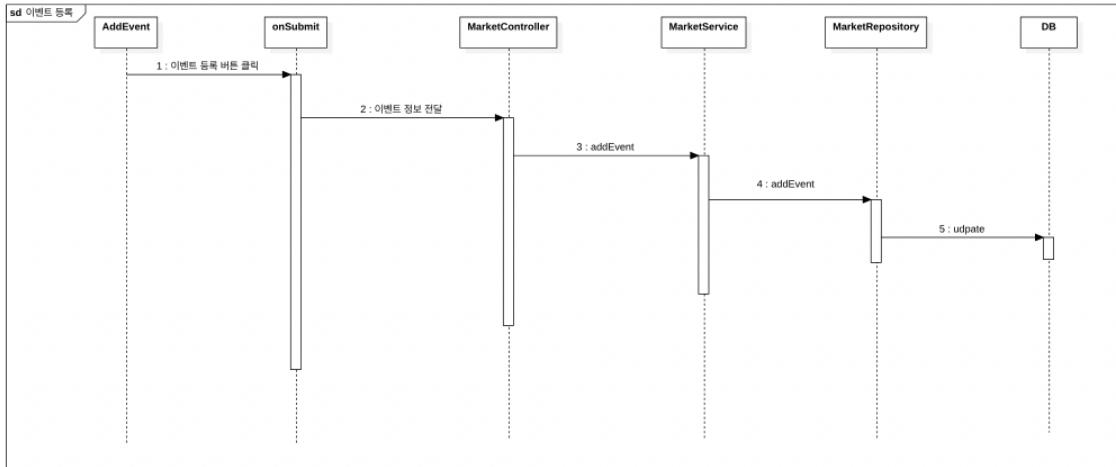


그림 66 이벤트 등록 sequence diagram

점주가 이벤트 등록할 때의 절차를 나타낸 Sequence Diagram 이다. 이벤트 등록 컴포넌트인 AddEvent에서 이벤트 등록 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 등록할 이벤트 정보를 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 addEvent 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 addEvent 메소드가 호출된다. 호출된 addEvent 메소드를 이용해서 DB에 접근하는 MarketRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에 매장의 이벤트 정보를 갱신한다.

### 5.2.13. 이벤트 삭제

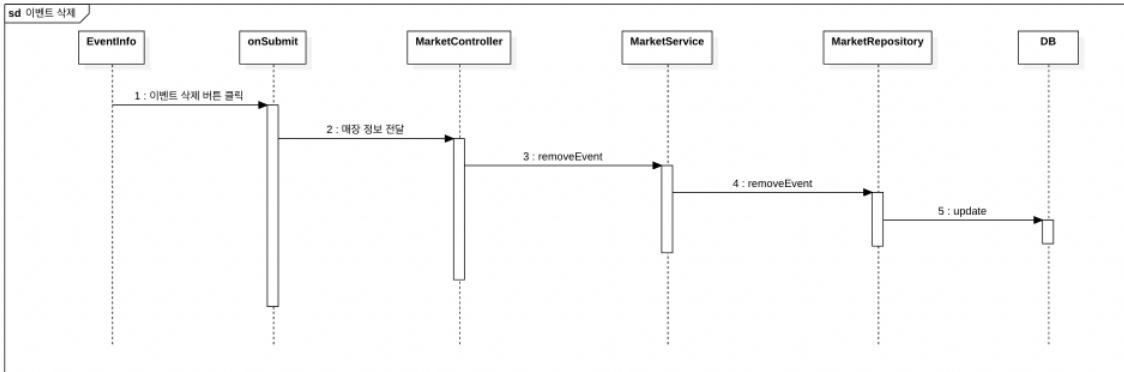


그림 67 이벤트 삭제 sequence diagram

점주가 이벤트 삭제할 때의 절차를 나타낸 Sequence Diagram 이다. 이벤트 삭제 컴포넌트인 EventInfo 에서 이벤트 삭제 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 삭제할 이벤트 정보를 전달한다. 서버로 전달된 데이터는 MarketController 클래스를 이용해 요청 url에 mapping 되어 removeEvent 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 MarketService 클래스의 removeEvent 메소드가 호출된다. 호출된 removeEvent 메소드를 이용해서 DB에 접근하는 MarketRepository 클래스의 update 메소드를 호출한다. 호출된 메소드를 통해 DB에 매장의 이벤트 정보 갱신하고 반환한다. 반환된 점주정보를 통해 CouponRepository 클래스의 update 메소드 쿠폰 정보가 갱신된다.

### 5.2.14. 리워드 추가

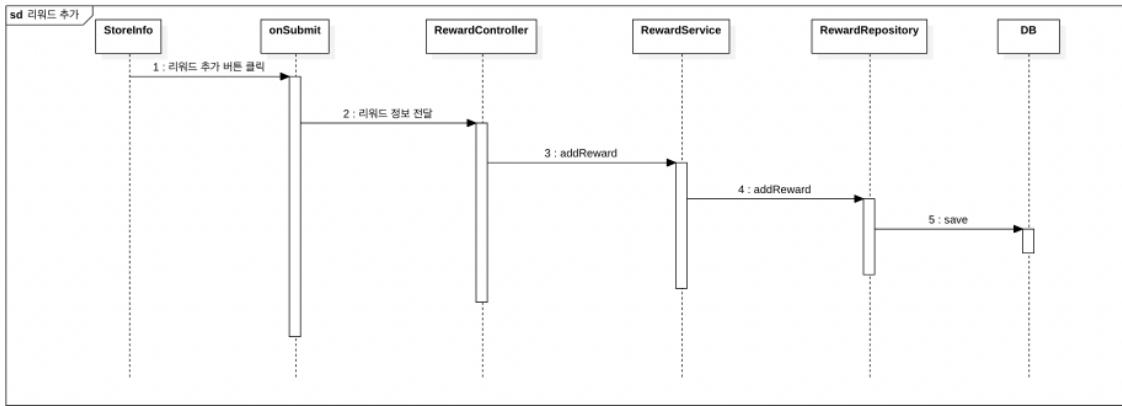


그림 68 리워드 추가 sequence diagram

리워드를 추가할 때의 절차를 나타낸 Sequence Diagram 이다. 리워드 컴포넌트인 RewardInfo 에서 리워드 추가 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 추가할 리워드 정보를 전달한다. 서버로 전달된 데이터는 RewardController 클래스를 이용해 요청 url에 mapping 되어 addReward 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 RewardService 클래스의 addReward 메소드가 호출된다. 호출된 addReward 메소드를 이용해서 DB에 접근하는 RewardRepository 클래스의 save 메소드를 호출한다. 호출된 메소드를 통해 DB에 리워드를 저장한다.

### 5.2.15. 리워드 삭제

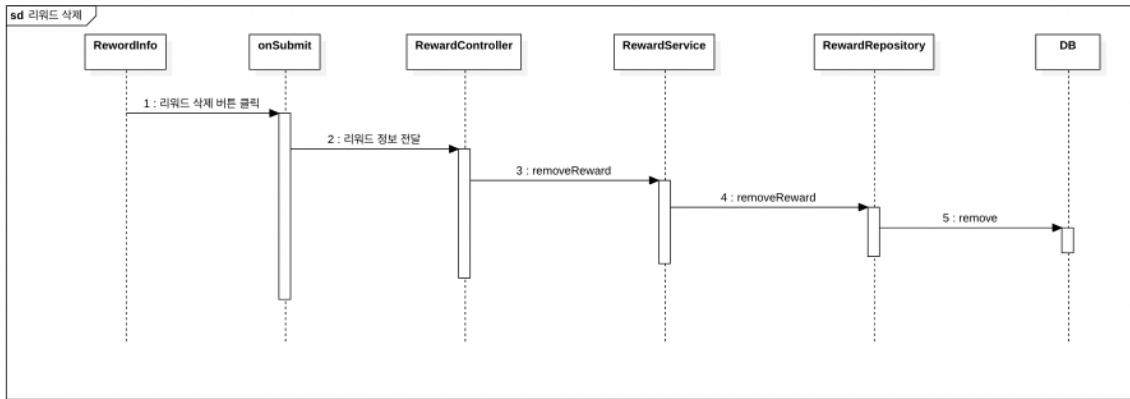


그림 69 리워드 삭제 sequence diagram

리워드를 삭제할 때의 절차를 나타낸 Sequence Diagram 이다. 리워드 컴포넌트인 RewardInfo 에서 리워드 추가 버튼을 클릭한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 삭제할 리워드 정보를 전달한다. 서버로 전달된 데이터는 RewardController 클래스를 이용해 요청 url에 mapping 되어 removeReward 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 RewardService 클래스의 removeReward 메소드가 호출된다. 호출된 removeReward 메소드를 이용해서 DB에 접근하는 RewardRepository 클래스의 remove 메소드를 호출한다. 호출된 메소드를 통해 DB에 리워드를 삭제한다.

### 5.2.16. 점주 정보 수정

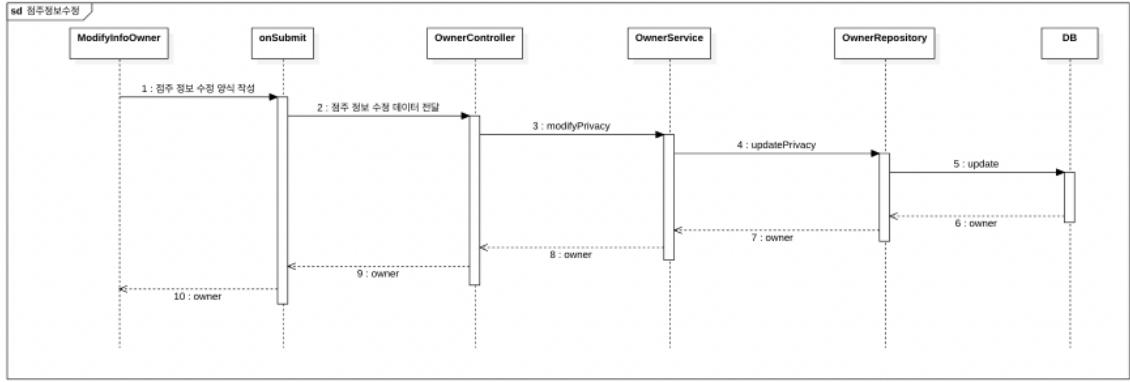


그림 70 점주 정보 수정 sequence diagram

점주 정보 수정할 때의 절차를 나타낸 Sequence Diagram 이다. 점주 정보 수정 컴포넌트인 ModifInfoOwner 에서 점주 정보 수정 양식을 작성한다. 이 과정에서 onSubmit 함수를 이용하여 서버로 점주 수정 정보를 전달한다. 서버로 전달된 데이터는 OwnerController 클래스를 이용해 요청 url에 mapping 되어 modifyPrivacy 메소드를 호출한다. 해당 메소드를 통해 로직이 있는 RewardService 클래스의 updatePrivacy 메소드가 호출된다. 호출된 updatePrivacy 메소드를 이용해서 DB에 접근하는 RewardRepository 클래스의 remove 메소드를 호출한다. 호출된 메소드를 통해 DB에 점주 정보를 수정한다. 수정된 점주가 OwnerService 클래스에 반환되어 OwnerController 클래스를 거쳐 view로 값을 리턴한다.

## 6. State machine Diagram

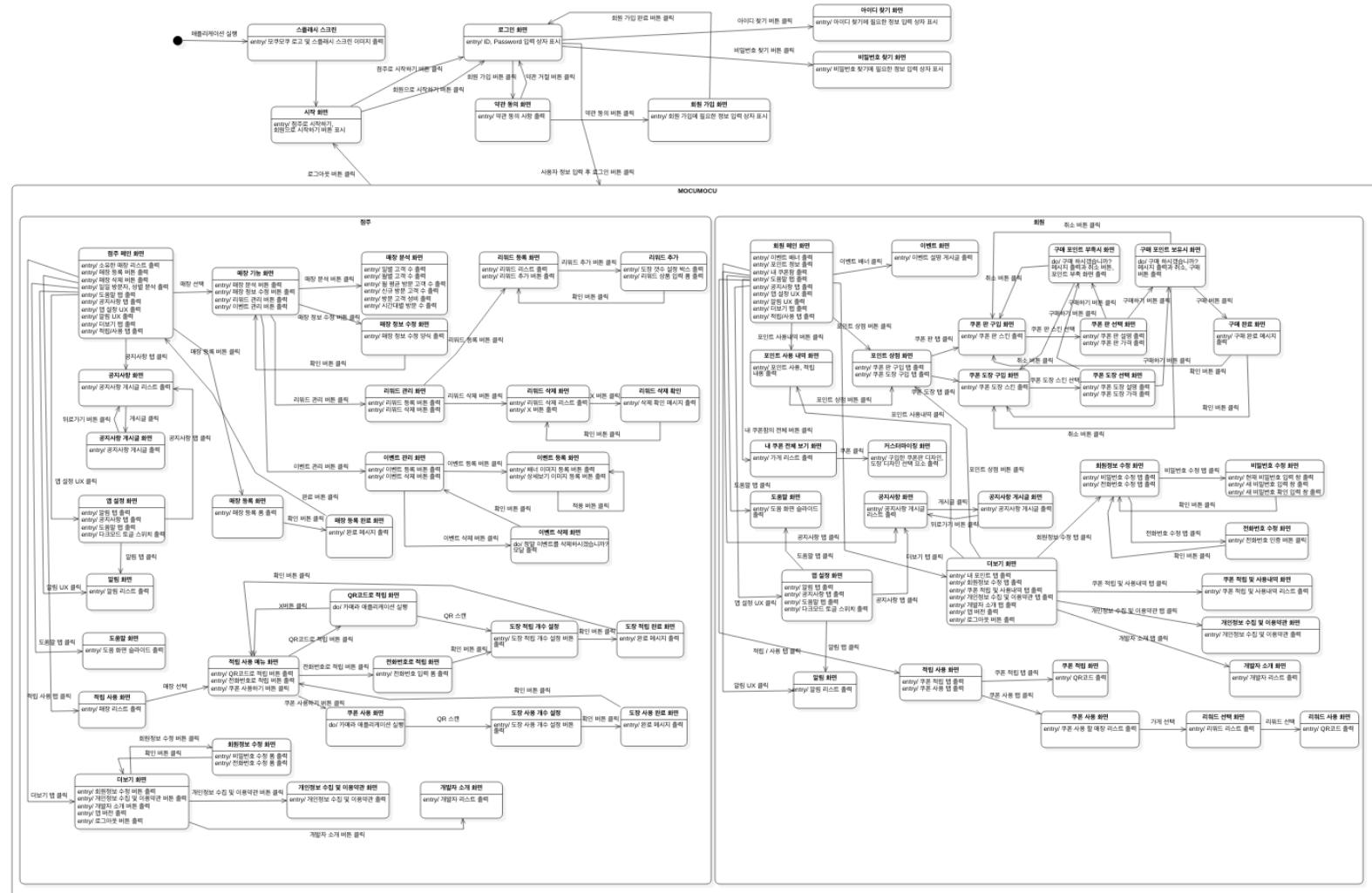


그림 71 State Machine Diagram

## 7. Mock Up

### 7.1. 공통

#### 7.1.1. 스플래시 스크린



그림 72 스플래시 스크린

애플리케이션 실행 시 보여지는 스플래시 스크린 화면이다.

### 7.1.2. 시작 화면

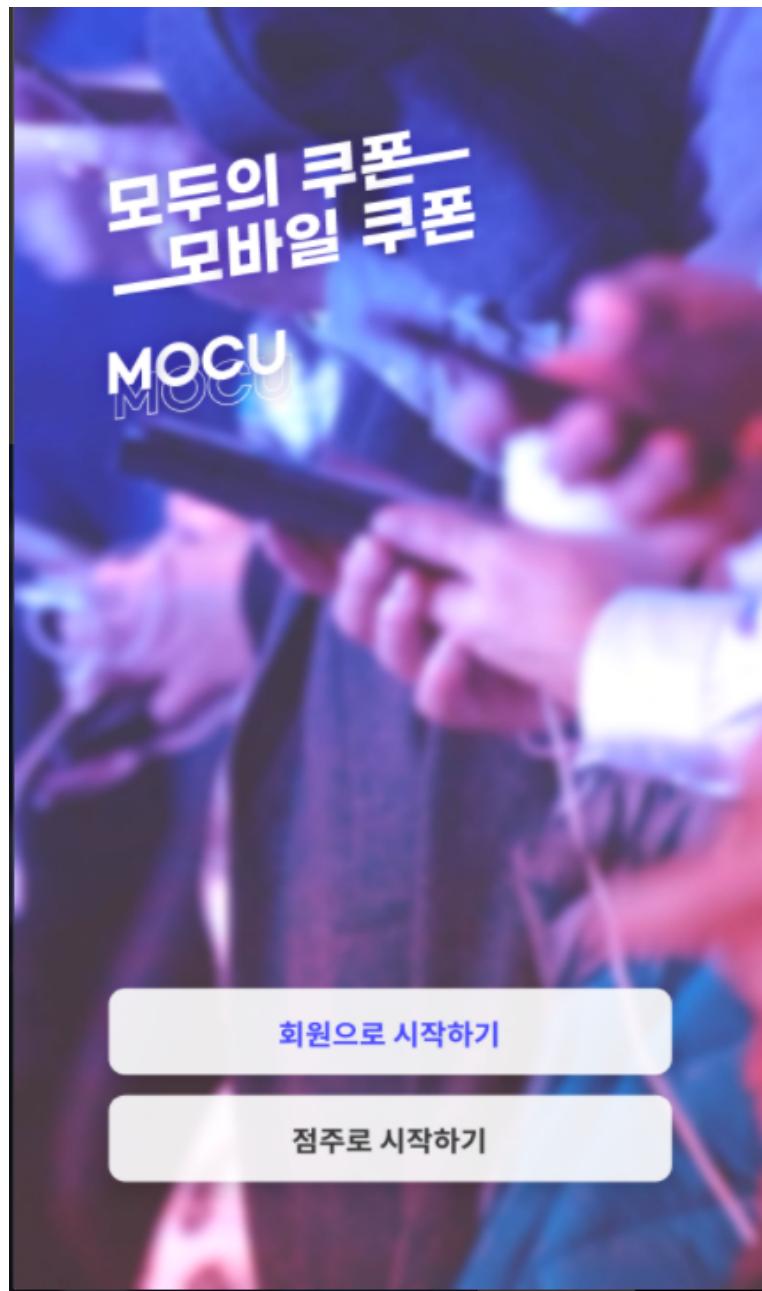


그림 73 시작 화면

스플래시 스크린 이후 사용자 타입을 선택하기 위한 화면이다.  
회원으로 시작하기 위한 버튼과 점주로 시작하기 위한 버튼이 존재한다.

## 7.2. 회원

### 7.2.1. 로그인

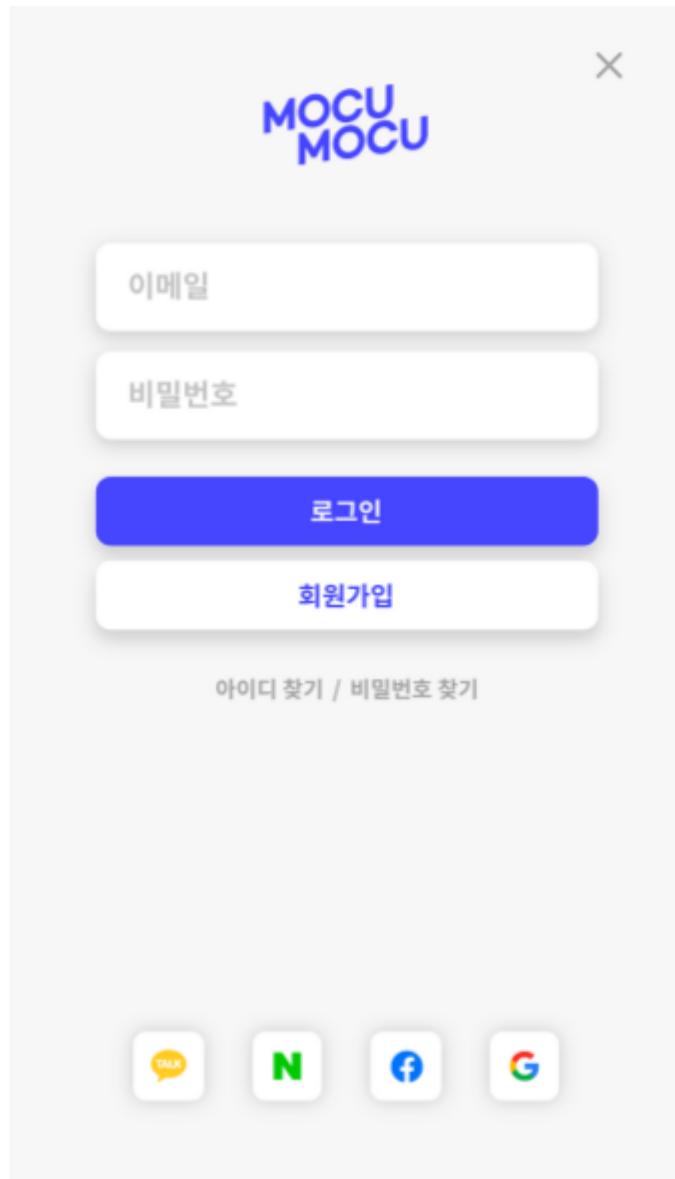


그림 74 로그인(회원) 화면

애플리케이션을 사용하기 위해 로그인을 요구하는 화면이다.

이메일과 비밀번호를 입력하는 텍스트 입력 창과

로그인 요청을 보내기 위한 버튼, 회원가입을 하기 위한

버튼이 존재한다. 만약 가입 시 입력한 아이디, 비밀번호를 잊어버렸다면  
아이디/비밀번호 찾기 버튼을 통해 회원가입 정보를 찾을 수 있으며, SNS 연동을  
통해 소셜 로그인도 가능하다.

## 7.2.2. 회원가입

**개인정보 수집 및 이용 동의**

가. 개인정보의 수집·이용에 관한 사항  
개인정보의 수집·이용 목적  
귀하의 개인정보는 연수3동 주민자치센터 프로그램강사 채용을 위한 목적으로 수집·이용됩니다.  
수집·이용할 개인정보의 항목  
수집·이용되는 귀하의 개인정보는 다음과 같습니다.

<필수 정보>

- 사진 및 연락사항(성별, 생년월일, 유대전화번호, 주소, 이메일주소 등)
- 응시 자격조건에 필요한 학력 및 경력사항 등

<선택 정보>

- 자기소개 및 유대요건과 관련된 학력 및 경력사항 등
- ※ 상기 선택정보를 미작성하더라도 응시는 가능하며, 수집한 개인정보는 서류전형·면접시험 시
- 직무수행 적합성 등의 판단을 위해 활용됩니다.

개인정보의 보유·이용기간  
개인정보는 원칙적으로 개인정보의 수집 및 이용목적이 종료되면 폐기됩니다.

- 강사 채용 해지에 따라 동의자가 요구할 경우

동의를 거부할 권리 및 동의를 거부할 경우의 불이익  
위 개인정보의 수집·이용에 동의를 거부할 수 있습니다.  
다만, 거부할 경우 채용심사 대상에서 제외됩니다.

모두 동의하고 다음으로

나. 제공에 관한 사항  
제공하는 자

그림 75 회원가입(회원) 화면

회원이 애플리케이션을 사용하기 위해  
계정을 등록하기 위한 회원가입 화면이다.

개인정보 수집 및 이용 동의 화면에서 동의 버튼을 눌러야만  
회원가입을 할 수 있는 화면이 출력된다. 회원가입을 위해서는 이름, 이메일,  
비밀번호, 비밀번호 확인, 전화번호, 성별, 생년월일 텍스트 박스를 모두 입력해야  
하며 사전에 설정된 입력 양식에 맞지 않은 경우 알림 창이 나타난다.

### 7.2.3. 메인 화면

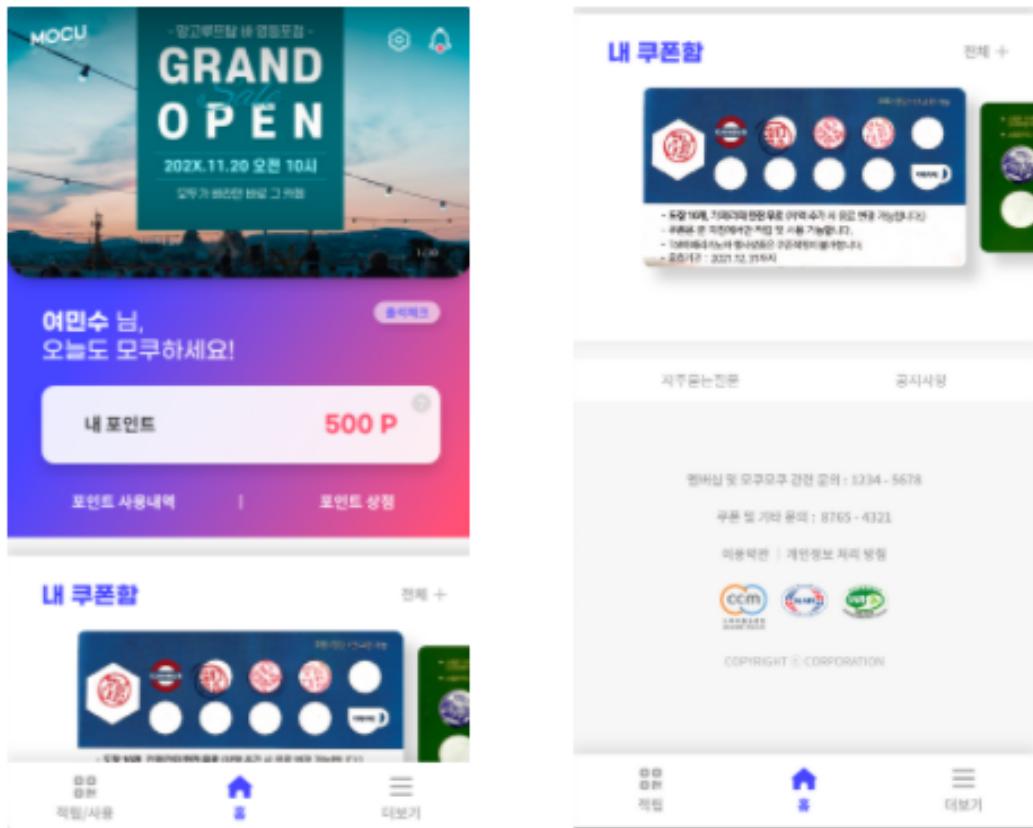


그림 76 메인(회원) 화면

회원이 로그인 했을 때 가장 먼저 볼 수 있는 메인 홈 화면이다. 홈 화면은 스크롤을 통해 위, 아래로 스크롤 할 수 있다. 화면 상단에는 점주가 등록해놓은 이벤트 배너가 나타나고 좌, 우 슬라이드를 하게되면 여러 이벤트 배너를 볼 수 있다. 또한, 앱 내 설정 버튼, 푸시 알림 버튼이 화면 좌측 상단에 배치된다. 화면 중앙에는 쿠폰을 커스터마이징 할 수 있는 포인트 현황과 포인트 사용 내역 버튼, 보유한 포인트를 통해 여러 쿠폰 판 및 도장 스킨을 구매할 수 있는 포인트 상점 버튼이 존재한다. 해당 영역 아래에는 현재 회원이 보유하고 있는 쿠폰 현황을 볼 수 있는 내 쿠폰함이 존재하며, 전체+ 버튼을 클릭 시 회원이 보유 중인 쿠폰 현황을 리스트 형식으로 볼 수 있다. 화면 하단에는 공지사항 버튼 및 애플리케이션 도움말 버튼이 보이게 되고 저작권 정보나 개인정보 처리 방침 등을 볼 수 있는 푸터 영역이 있다.

#### 7.2.4. 쿠폰 리스트

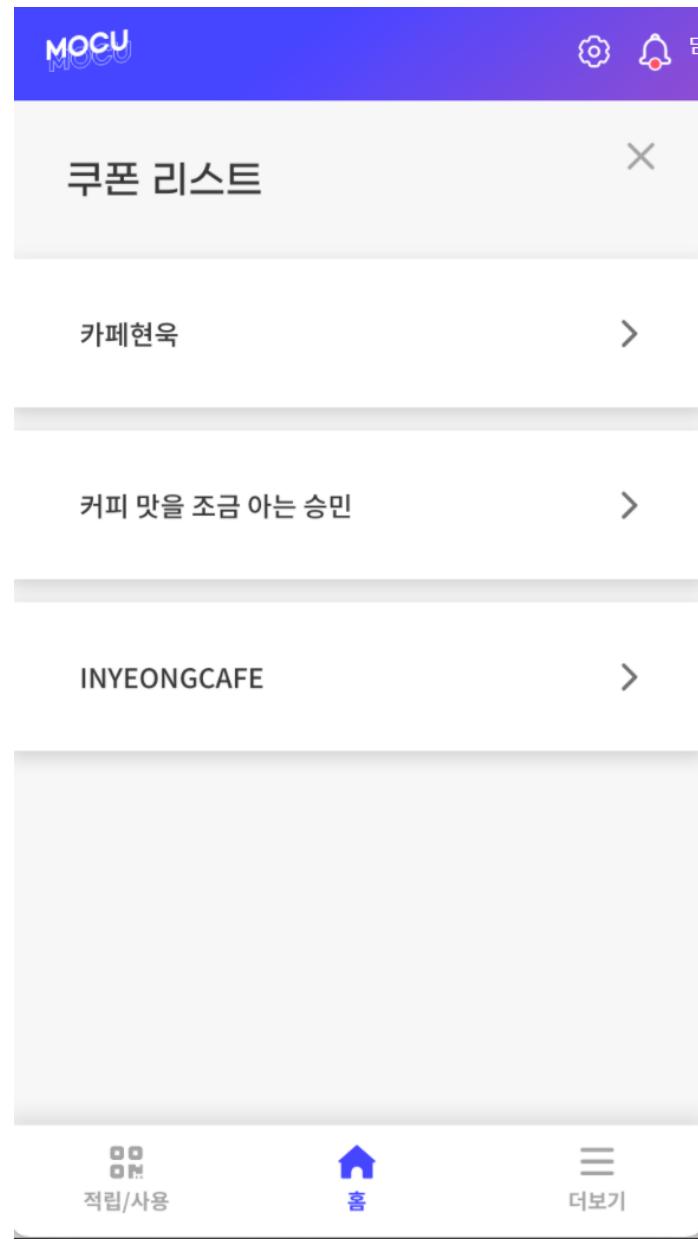


그림 77 쿠폰리스트 화면

회원이 홈 화면에서 전체+ 버튼을 클릭 시 나타나는 화면이다.

회원이 보유하고 있는 쿠폰 리스트가 화면에 나타난다.

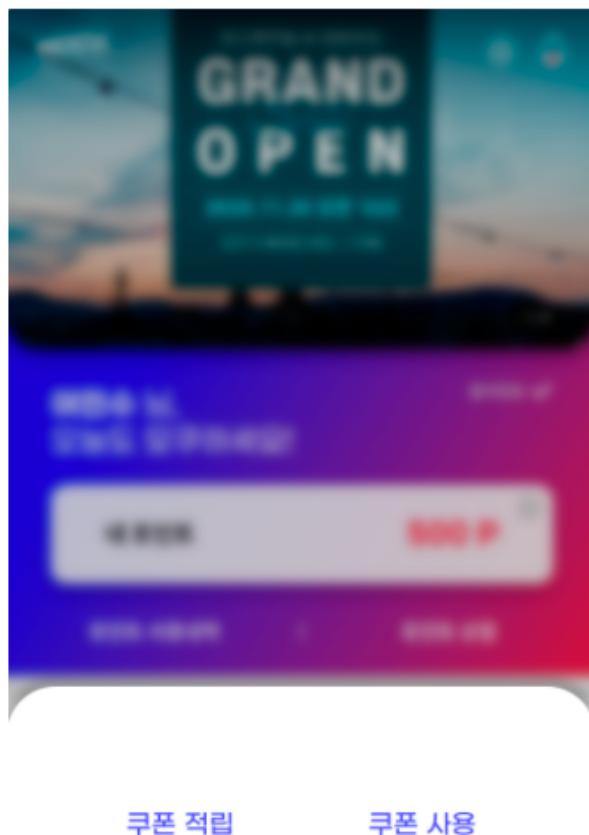
### 7.2.5. 쿠폰 커스터마이징



그림 78 쿠폰 커스터마이징 화면

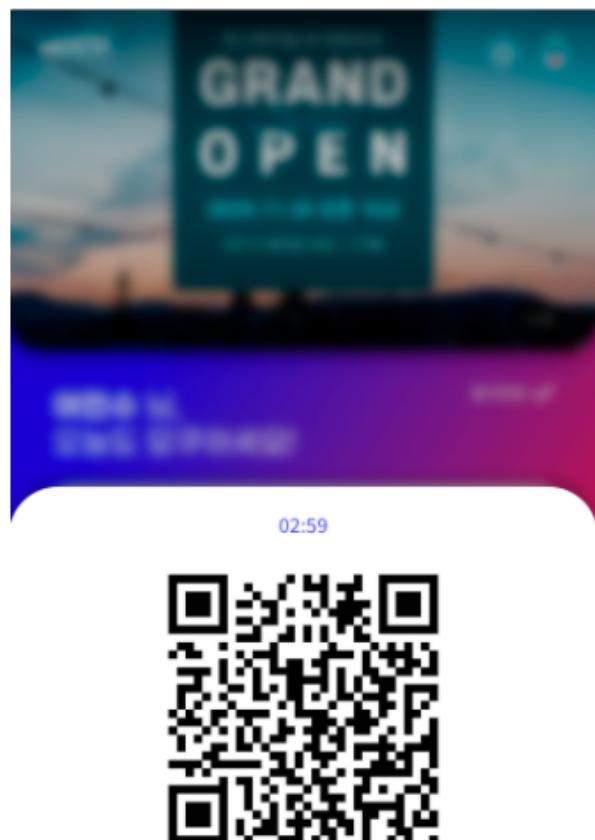
쿠폰 리스트 화면에서 커스터마이징을 원하는 쿠폰을 선택하면 보이는 커스터마이징 화면이다. 회원은 보유한 스킨 중 원하는 쿠폰 판 혹은 도장 스킨을 선택할 수 있고 적용 버튼을 누르면 해당 스킨이 적용된다.

### 7.2.6. 쿠폰 적립



쿠폰 적립

쿠폰 사용



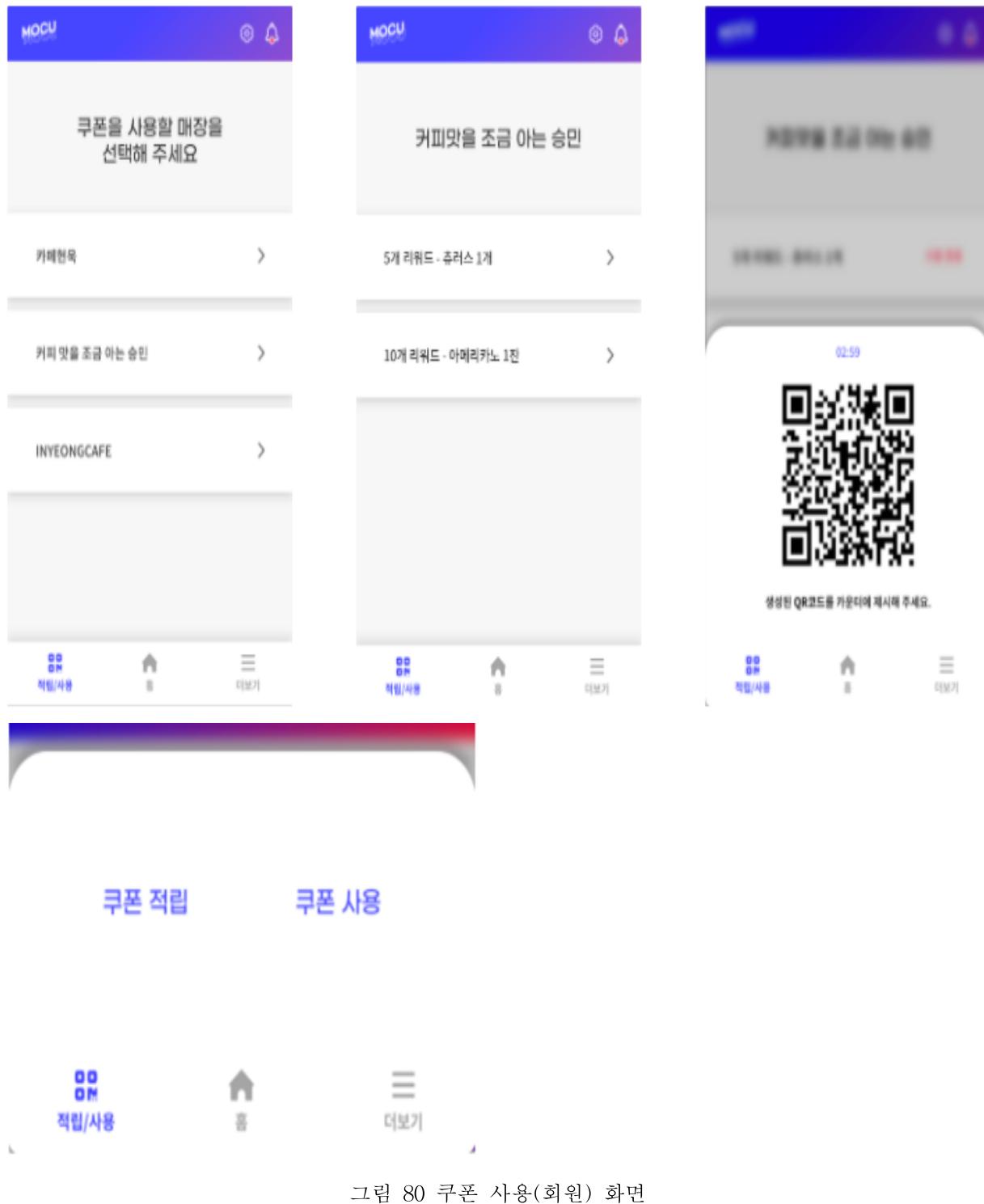
생성된 QR코드를 카운터에 제시해 주세요.



그림 79 쿠폰 적립(회원) 화면

적립/사용 탭을 클릭시 쿠폰을 적립 및 사용할 수 있는 버튼이 화면에 나타나게 되고 쿠폰 적립 버튼을 클릭 시 화면에 QR 코드가 생성된다.

### 7.2.7. 쿠폰 사용



쿠폰 사용 버튼을 클릭 시 쿠폰을 사용하려는 매장을 선택할 수 있는 화면이 보이게 되고 회원은 원하는 매장을 선택해 클릭할 수 있다. 클릭 후 사전에 점주가 설정해놓은 해당 매장의 리워드 목록이 화면에 보이게 되며, 회원은 원하는 리워드를 클릭할 수 있다. 리워드 클릭 시 화면에 QR 코드가 생성된다.

### 7.2.8. 더보기



그림 81 더보기(회원) 화면

더보기 탭을 클릭 시 나타나는 화면이다. 해당 화면 상단에는 회원이 보유하고 있는 포인트 현황, 포인트 사용내역 및 포인트 상점 버튼이 배치된다. 그 밑에는 회원정보 수정, 쿠폰 적립 및 사용내역, 개인정보 수집 및 이용약관, 개발자 소개 버튼과 애플리케이션 버전이 배치된다.

### 7.2.9. 포인트 사용 내역

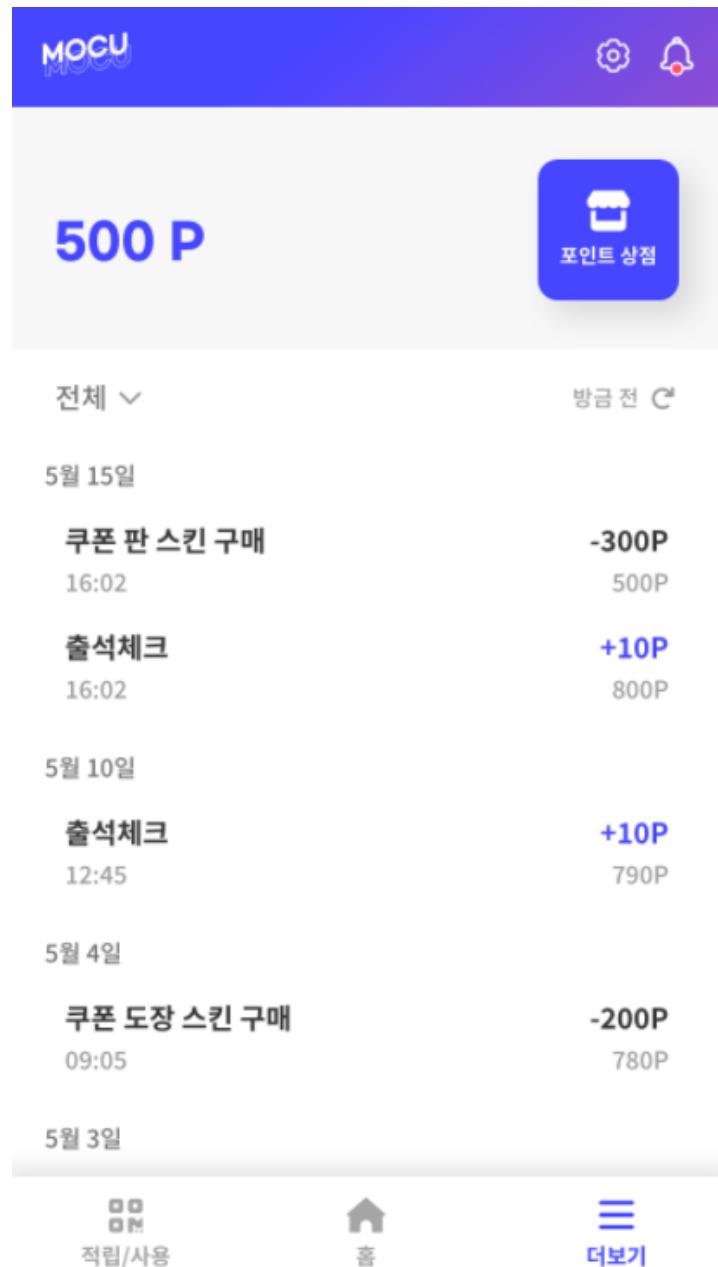


그림 82 포인트 사용 내역(회원) 화면

포인트 사용 내역 버튼 클릭시 나타나는 화면이다. 화면 상단에는 현재 회원이 보유 중인 포인트 현황과 포인트 상점 버튼이 나타나며, 나머지 영역은 회원이 사용 및 적립한 포인트 내역이 보여진다.

### 7.2.10. 포인트 상점



그림 83 포인트 상점 화면

포인트 상점 버튼 클릭 시 나타나는 화면이다. 화면 상단에는 쿠폰 이미지가 보여지고 아래에는 쿠폰 판, 쿠폰 도장 을 선택할 수 있는 탭이 나타난다. 회원이 원하는 쿠폰 판 스킨 혹은 쿠폰 도장 스킨을 선택하면 상단의 쿠폰 이미지가 선택된 이미지로 변경되며, 구매하기 버튼을 클릭 시 해당 스킨이 구매된다.

### 7.2.11. 회원정보 수정

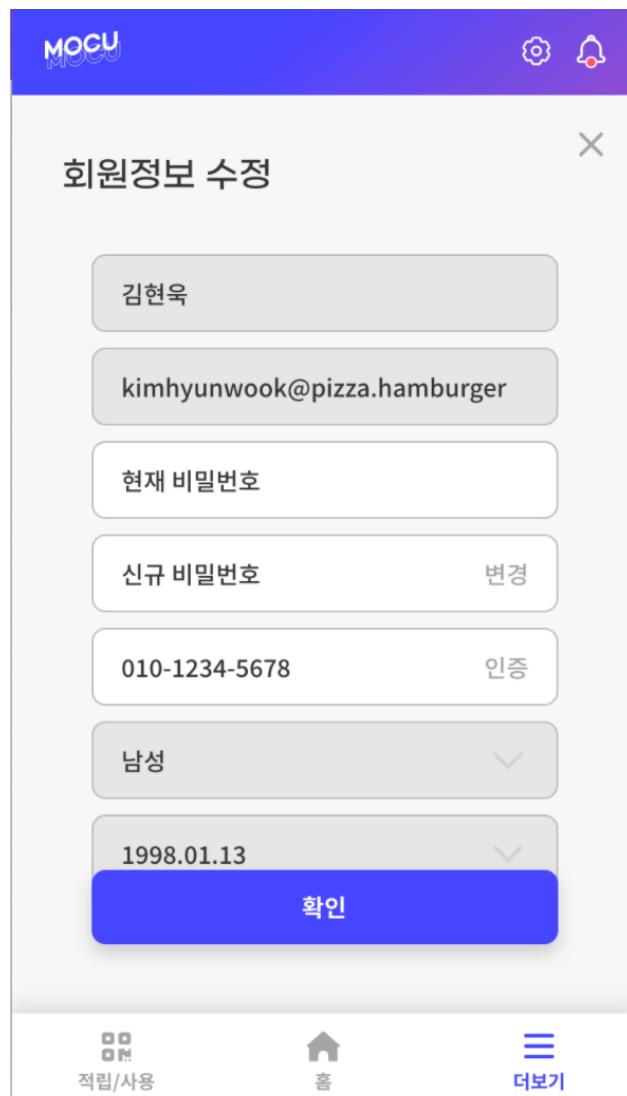


그림 84 회원정보 수정(회원) 화면

회원정보 수정 버튼 클릭 시 나타나는 화면이다. 회원은 해당 화면에서 비밀번호 변경 및 휴대폰 번호 변경이 가능하다.

### 7.2.12. 쿠폰 적립 및 사용 내역



그림 85 쿠폰 적립 및 사용(회원) 내역 화면

쿠폰 적립 및 사용 내역 버튼을 클릭 시 나타나는 화면이다. 회원이 사용하고 적립한 도장 내역을 해당 화면을 통해 볼 수 있다.

### 7.3. 접속

#### 7.3.1. 로그인 화면

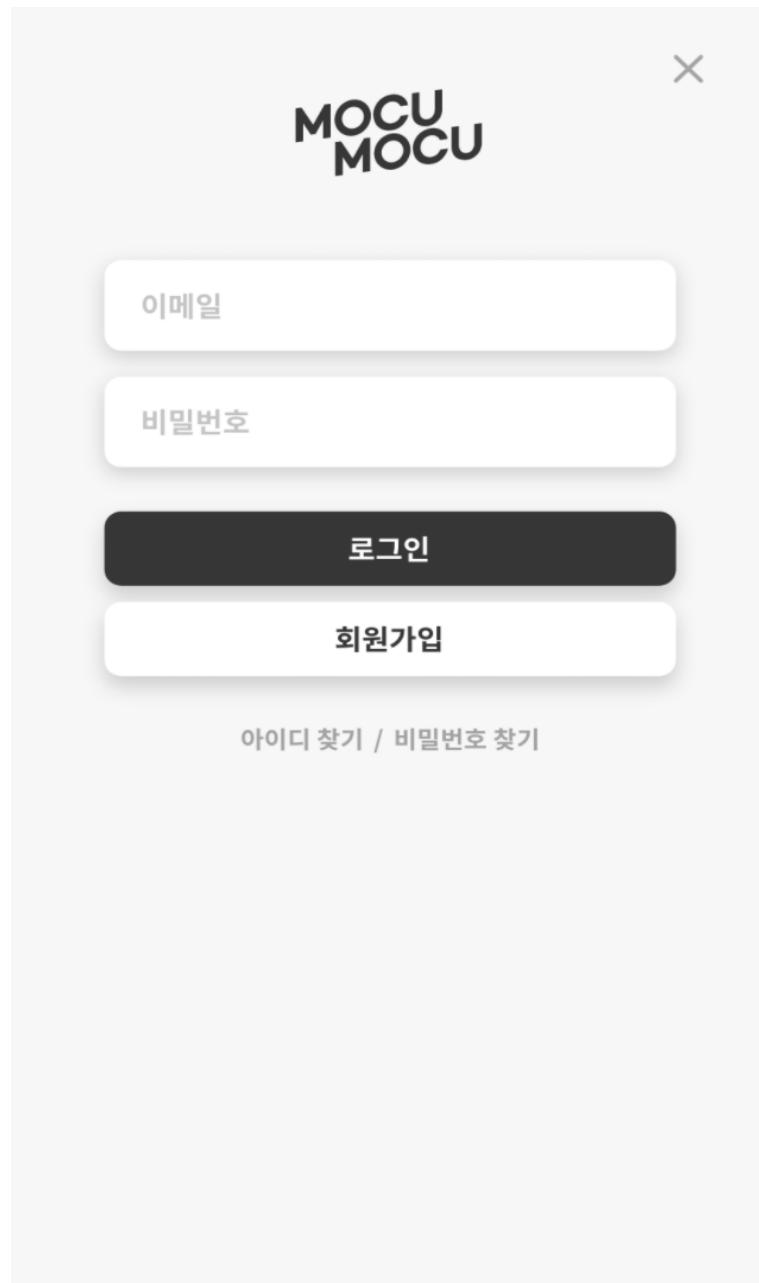


그림 86 로그인(접속) 화면

애플리케이션을 사용하기 위해 로그인을 요구하는 화면이다.

이메일과 비밀번호를 입력하는 텍스트 입력 창과  
로그인 요청을 보내기 위한 버튼, 회원가입을 하기 위한  
버튼이 존재한다.

### 7.3.2. 회원가입 화면



사용자가 애플리케이션을 사용하기 위해  
계정을 등록하기 위한 회원가입 화면이다.  
개인정보 수집 및 이용 동의 화면에서 동의 버튼을 눌러야만  
회원가입을 할 수 있는 화면이 출력된다.

### 7.3.3. 메인 화면



그림 88 메인(점주) 화면

로그인이 정상적으로 이루어지면 출력되는 점주 메인 화면이다.  
상단에는 어플리케이션 설정으로 들어가는 아이콘과 알림 아이콘이 존재하고,  
본 화면에는 점주가 등록한 매장의 리스트를 출력하는 ScrollView와  
매장별 일일 방문자 분석 카드 ScrollView가 존재한다.  
하단에는 적립/사용 화면, 홈 화면, 더보기 화면으로 이동할 수 있는  
Tab Bar가 존재한다.

### 7.3.4. 쿠폰 적립/사용 화면

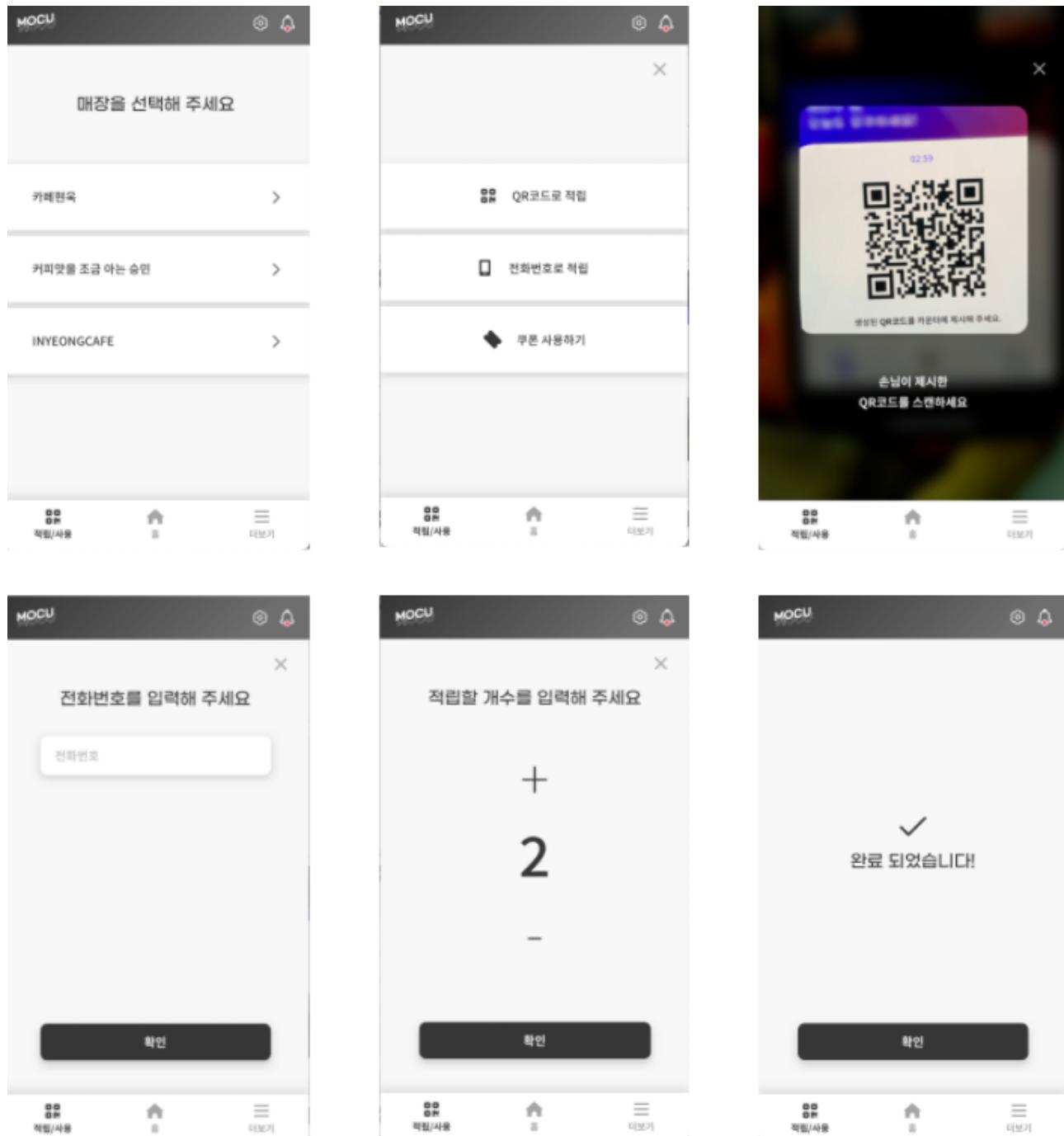


그림 89 쿠폰 적립/사용(점주) 화면

메인 화면에 존재하는 Tab Bar에서  
적립/사용 탭을 눌렀을 때 출력되는 화면이다.  
매장을 선택하고 어떤 작업을 할 지 선택하는 버튼들이 존재한다.  
쿠폰 적립은 QR 코드 스캔, 전화번호 입력으로 가능하고  
쿠폰 사용은 QR 코드 스캔으로만 가능하다.

### 7.3.5. 매장 관리 화면

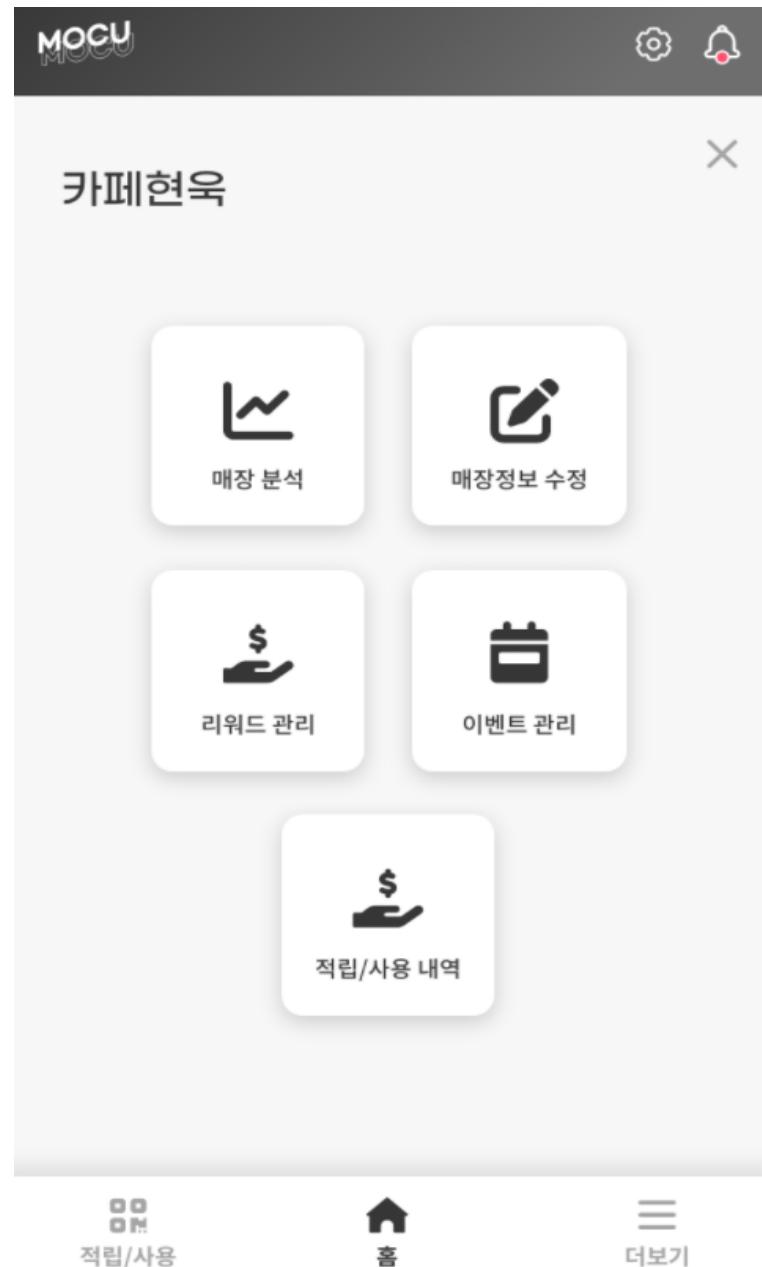


그림 90 매장 관리 화면

점주가 관리하는 매장 중 하나를 선택했을 때 보이는 매장 관리 화면이다. 매장 분석, 매장정보 수정, 리워드 관리, 이벤트 관리, 적립/사용 내역 화면으로 들어갈 수 있는 버튼이 존재한다.

### 7.3.6. 매장 분석 화면

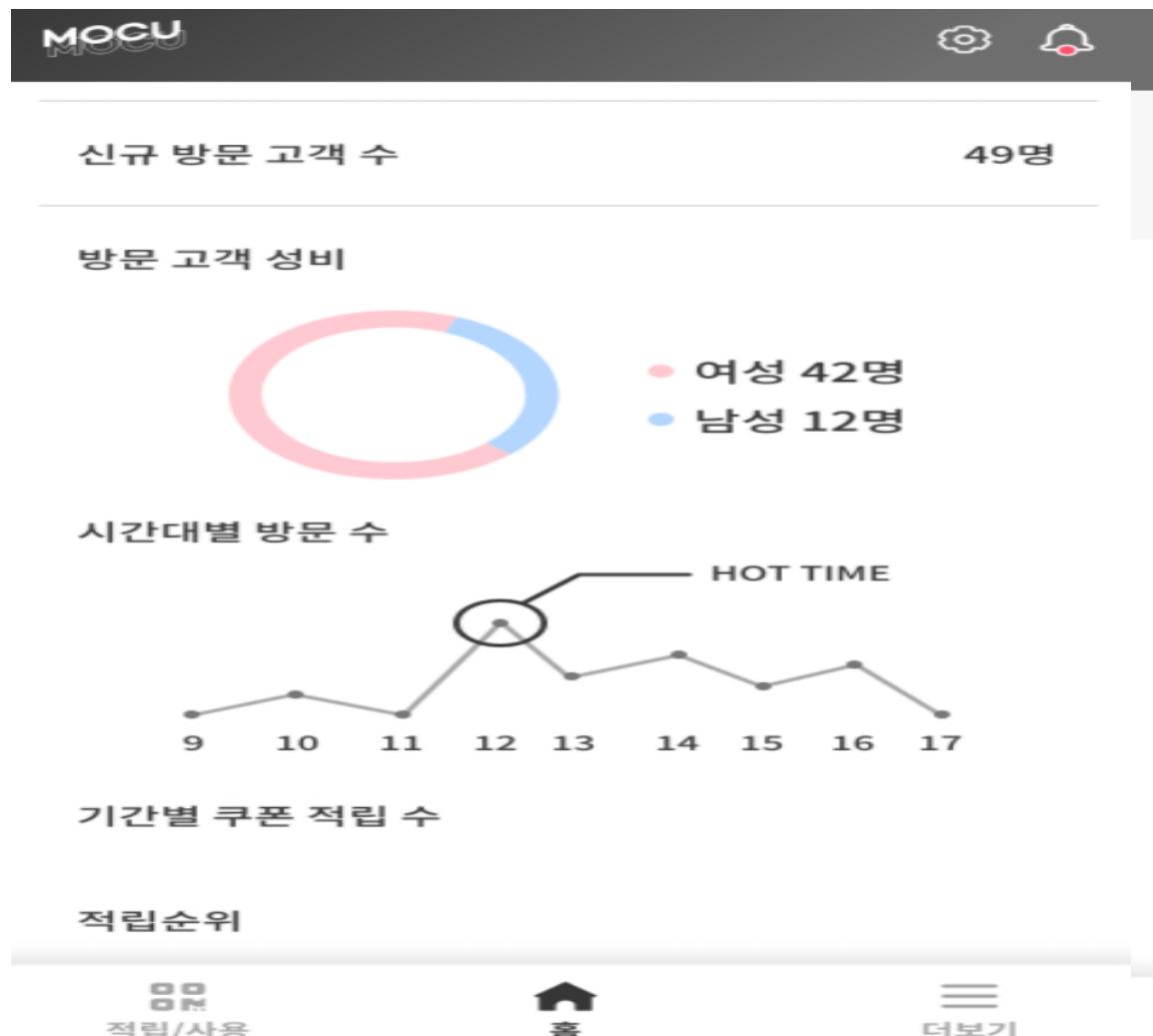


그림 91 매장 분석 화면

매장 관리 화면에서 매장 분석 버튼을 클릭했을 때 나오는 화면이다. 해당 매장의 일별 고객 수, 월별 고객수 월 평균 방문 고객 수, 신규 방문 고객 수, 방문 고객 성비, 시간대별 방문 수, 적립 순위를 보여준다.

### 7.3.7. 매장 정보 수정 화면

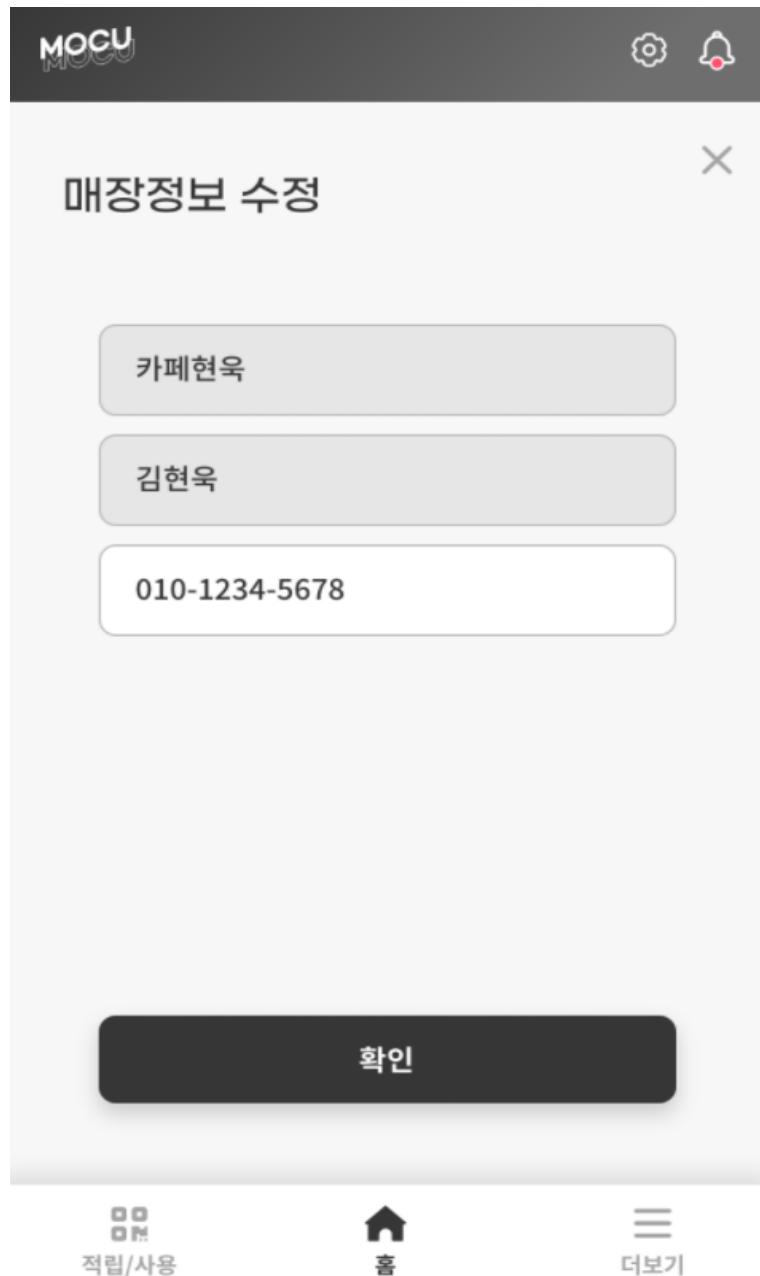


그림 92 매장 정보 수정 화면

매장 관리 화면에서 매장 정보 수정 버튼을 클릭했을 때 나오는 화면이다.  
매장 이름, 점주 이름을 출력하고 매장 전화번호를 수정할 수 있는 요소가 있다.

### 7.3.8. 리워드 관리 화면



그림 93 리워드 관리 화면

매장 관리화면에서 리워드 관리 버튼을 클릭했을 때 나오는 화면이다  
리워드를 등록하는 화면으로 넘어가는 버튼과  
리워드를 삭제하는 화면으로 넘어가는 버튼이 존재한다.

### 7.3.9. 리워드 등록 화면



그림 94 리워드 등록 화면

리워드 관리 화면에서 리워드 등록 버튼을 나왔을 때 출력되는 화면들이다.  
리워드 목록 화면에서 + 버튼을 누르면 리워드와 차감 개수를 설정 할 수 있는  
화면이 나오고 설정 후 확인 버튼을 누르면 리워드가 추가된다.

### 7.3.10. 리워드 삭제 화면

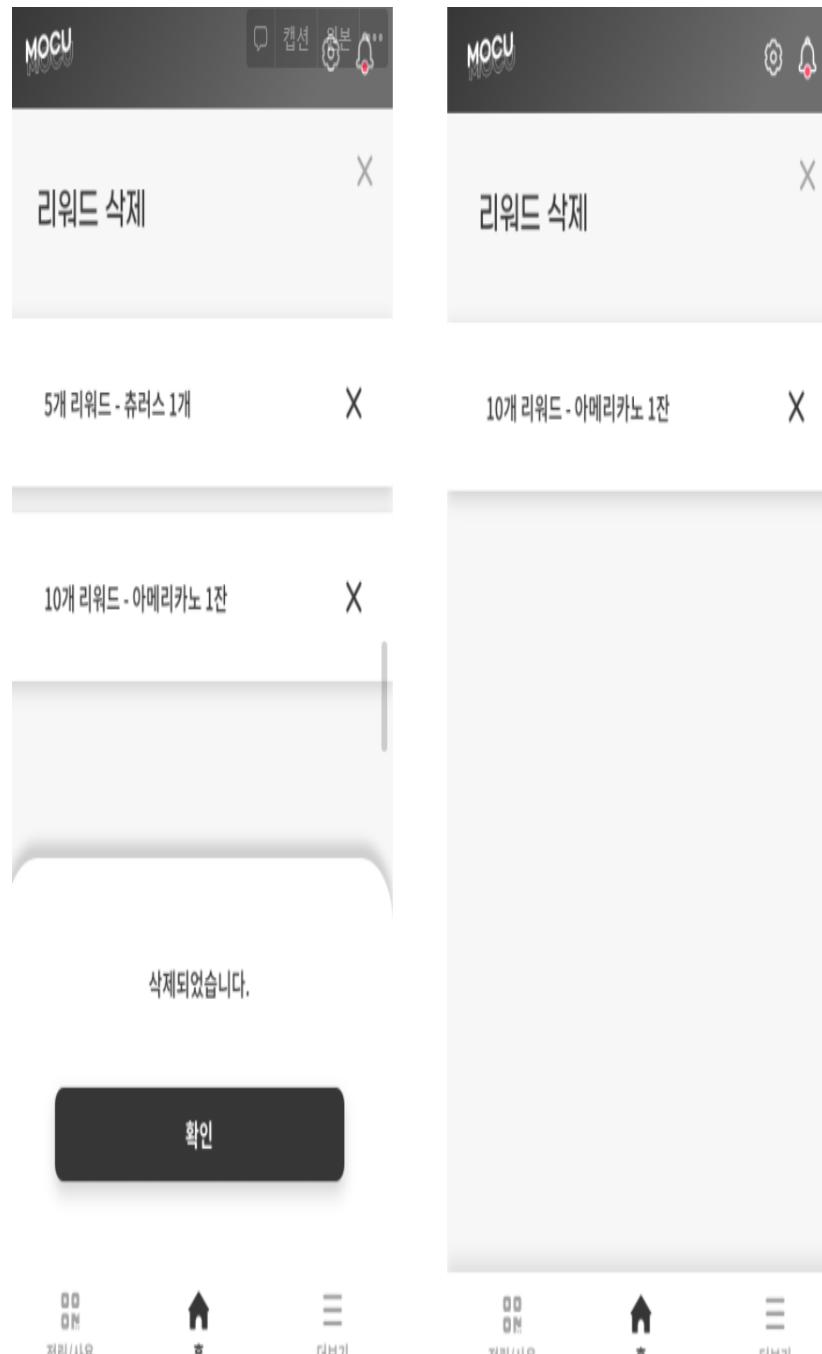


그림 95 리워드 삭제 화면

리워드 관리 화면에서 리워드 삭제 버튼을 나왔을 때 출력되는 화면  
리워드 목록화면에서 삭제하고자 하는 리워드 템에 있는 x 버튼을 누르면  
삭제 확인 메세지가 나오고 확인 버튼을 누르면 정상적으로 삭제된다.

### 7.3.11. 이벤트 관리 화면

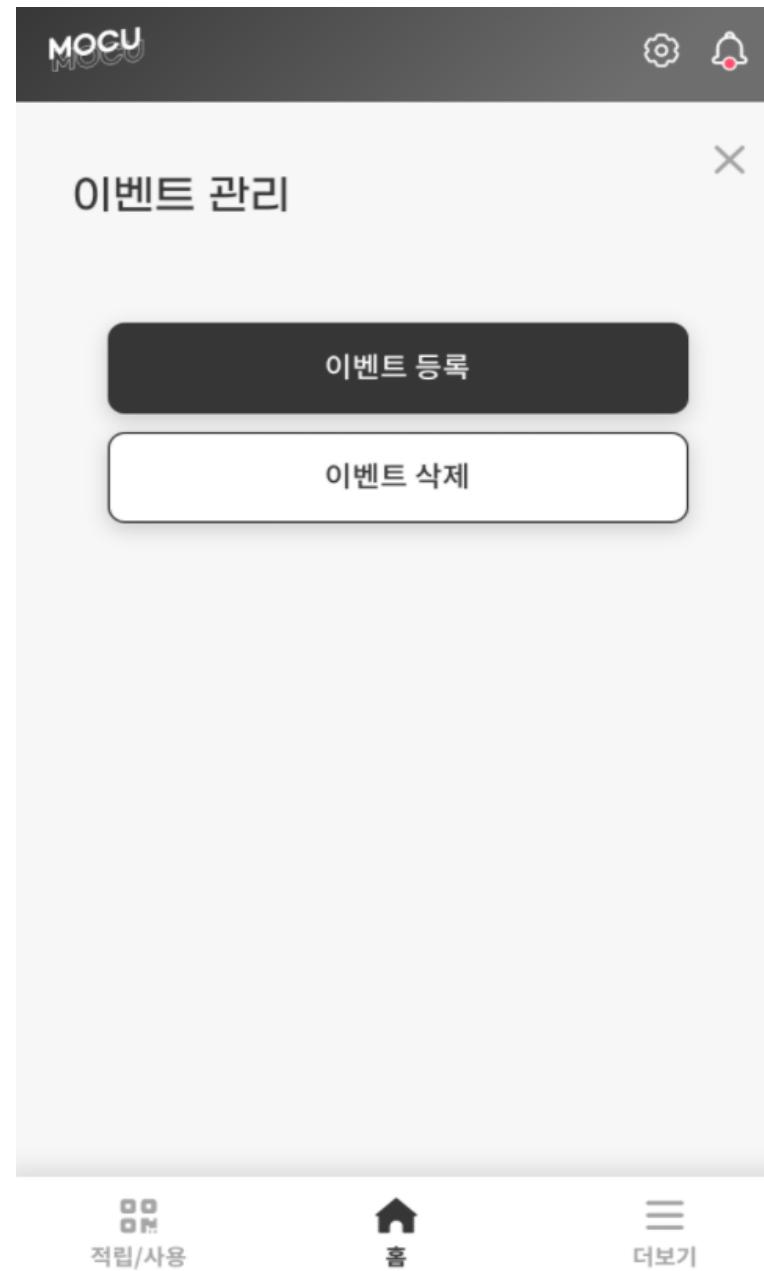


그림 96 이벤트 관리 화면

매장 관리화면에서 이벤트 관리 버튼을 클릭했을 때 나오는 화면이다.

이벤트를 등록하는 화면으로 넘어가는 버튼과  
이벤트를 삭제하는 화면으로 넘어가는 버튼이 존재한다.

### 7.3.12. 이벤트 등록 화면

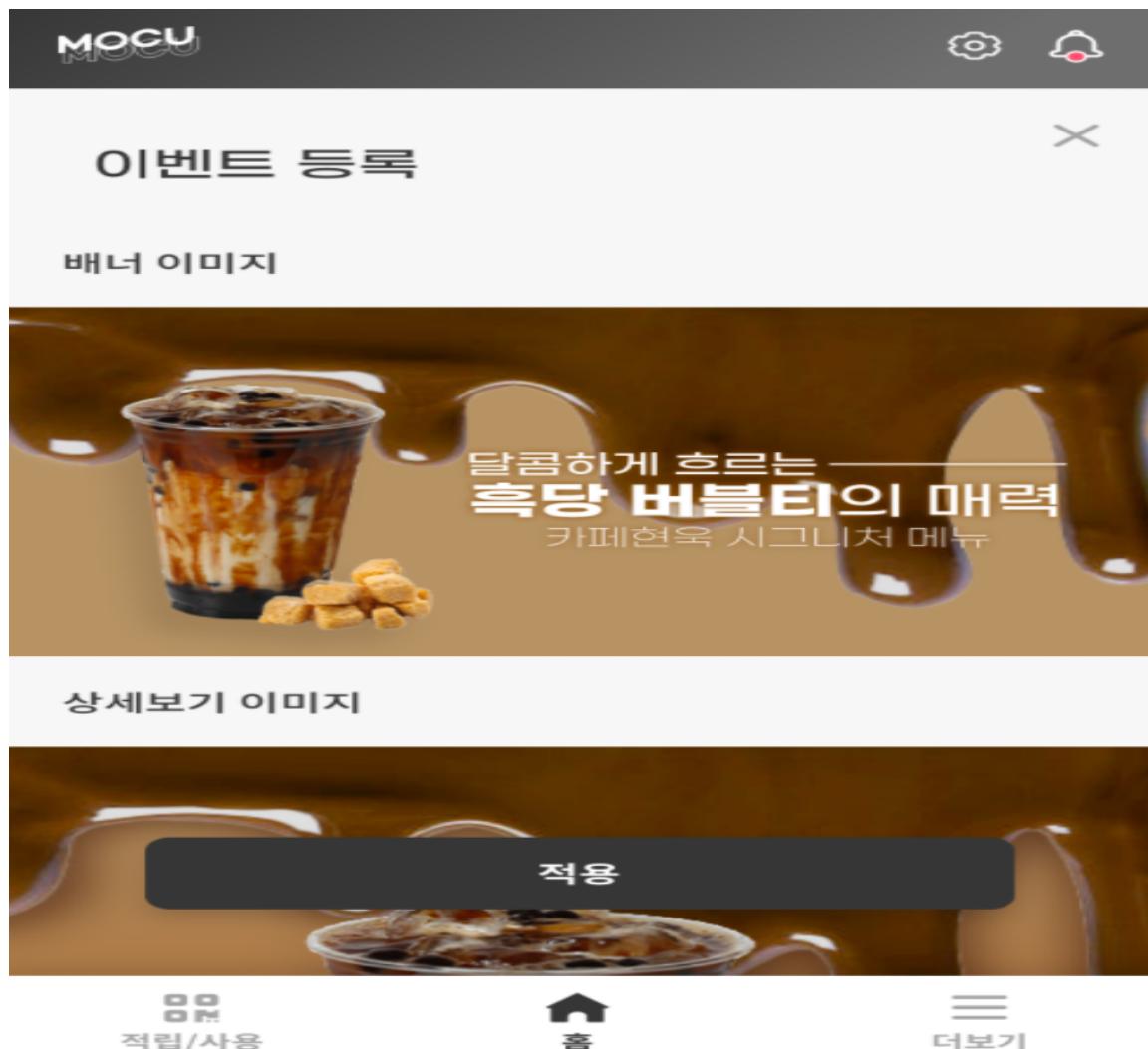


그림 97 이벤트 등록 화면

이벤트 관리 화면에서 이벤트 등록 버튼을 누르면 나오는 화면이다.  
회원 애플리케이션 메인화면에 출력되는 배너이미지와  
배너 이미지를 클릭했을 때 출력되는 이벤트 상세 이미지를  
업로드 할 수 있는 요소가 존재한다.

### 7.3.13. 이벤트 삭제 화면

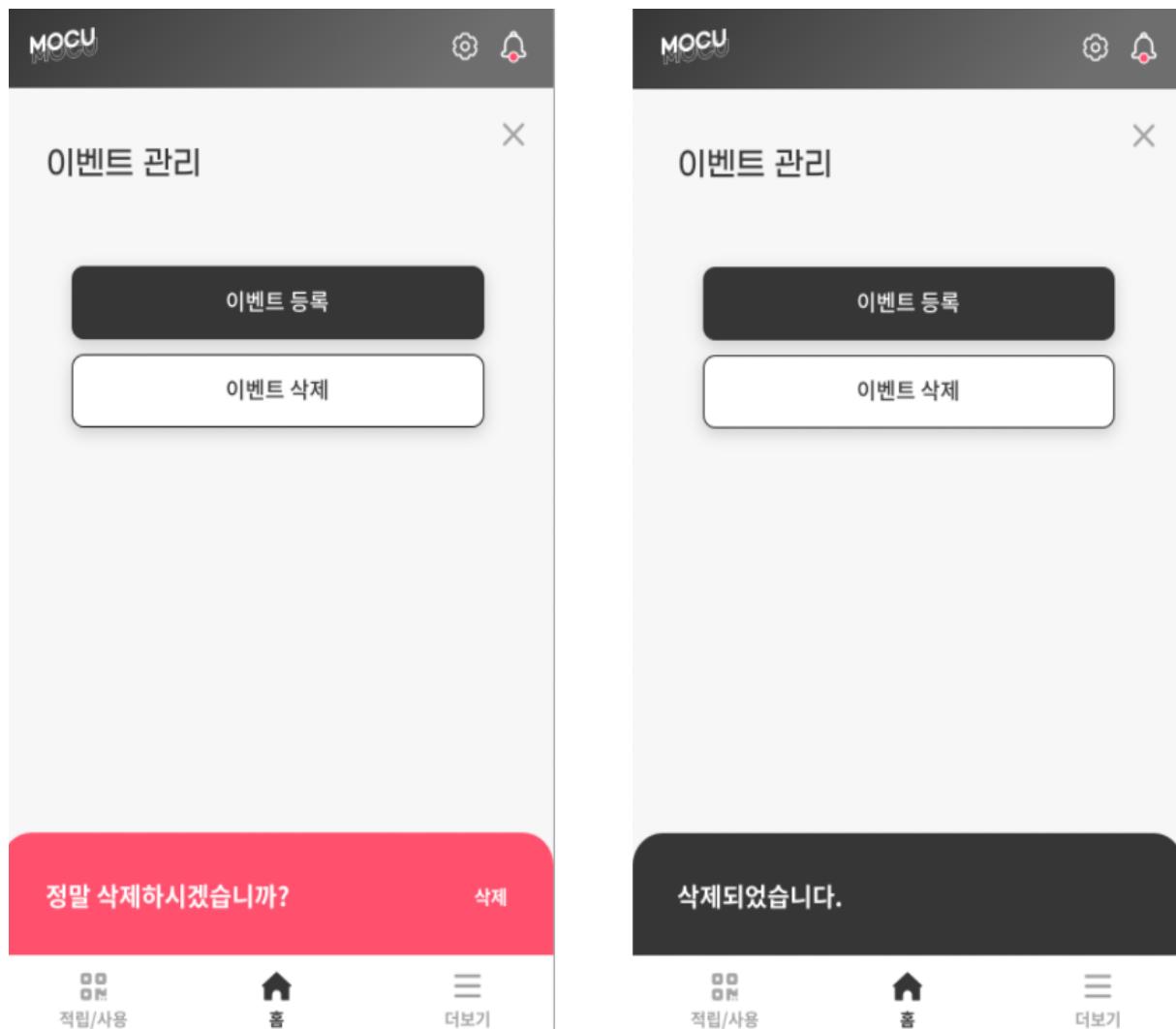


그림 98 이벤트 삭제 화면

이벤트 관리 화면에서 이벤트 삭제 버튼을 누르면  
삭제 여부를 확인하는 알림 창을 출력하고  
삭제 버튼을 누르면 정상적으로 삭제된다.

### 7.3.14. 쿠폰 적립/사용 내역 화면

The screenshot shows the 'MOCU' app interface. At the top, there is a dark header bar with the 'MOCU' logo on the left and two icons on the right: a gear and a bell with a red dot. Below the header is a light gray navigation bar with a close button ('X') on the right side. The main content area displays a list of transaction history items:

- 카페현욱  
적립/사용 내역**
- 전체 ▼** **방금 전 C**
- 5월 15일**
  - 이승민 - 리워드 A 사용**  
16:02
  - 여민수 - 1개 적립**  
16:02
- 5월 10일**
  - 김준서 - 1개 적립**  
12:45
- 5월 4일**
  - 장인영 - 2개 적립**  
09:05
- 5월 3일**
  - 박기홍 - 리워드 B 사용**

At the bottom of the screen, there are three navigation icons: a square icon labeled '적립/사용', a house icon labeled '홈', and a three-line icon labeled '더보기'.

그림 99 쿠폰 적립/사용 내역 화면

매장 관리화면에서 적립/사용 내역 버튼을 클릭했을 때 나오는 화면이다.  
매장별 적립/사용 내역을 출력한다.

### 7.3.15. 더보기 화면

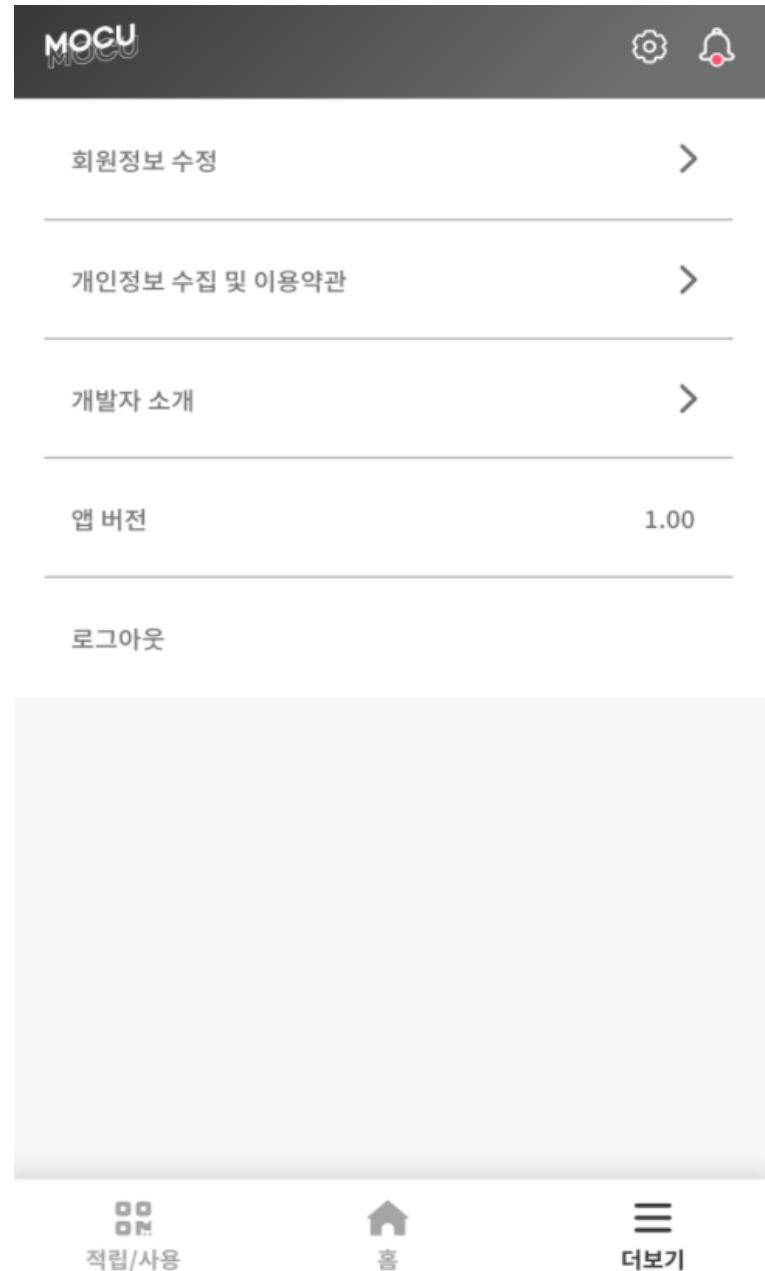


그림 100 더보기(점주) 화면

메인 화면에 존재하는 Tab Bar에서 더보기 탭을 눌렀을 때 출력되는 화면이다.  
회원 정보 수정을 할 수 있는 화면으로 들어가는 버튼,  
개인정보 수집 및 이용약관을 볼 수 있는 화면으로 들어가는 버튼,  
개발자 소개를 볼 수 있는 화면으로 들어가는 버튼,  
앱 버전 출력 텍스트,  
로그아웃을 할 수 있는 버튼이 존재한다.

### 7.3.16. 회원정보 수정 화면

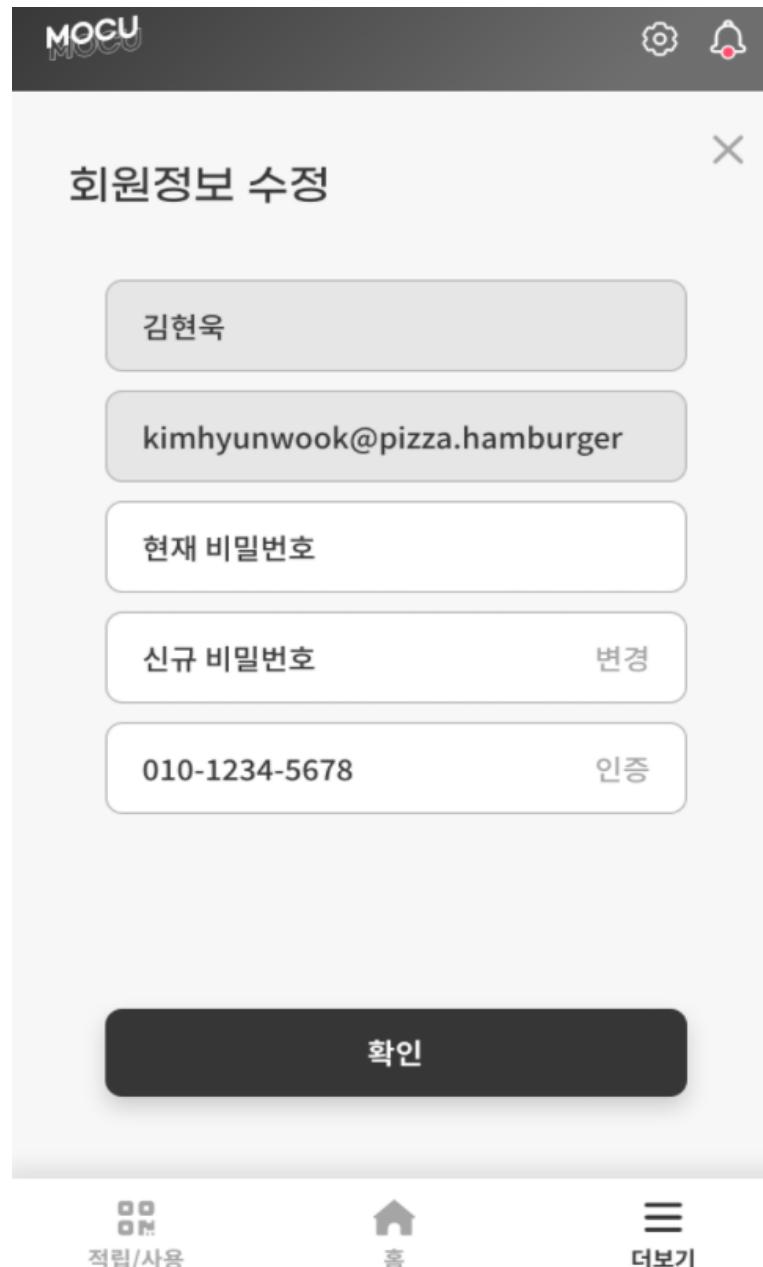


그림 101 회원 정보 수정(점주) 화면

더보기 화면에서 회원정보 수정 버튼을 눌렀을 때 나오는 화면이다.  
이름과 이메일을 출력하는 텍스트와  
비밀번호 및 전화번호를 변경할 수 있는 요소가 있다.

## 8. Implementation Requirements

### 8.1. Frontend

- ◎ S/W: HTML, CSS, TypeScript, React Native, Visual Studio Code, Adobe Illustrator
- ◎ H/W: Window10/11, Mac OS BigSur, 스마트폰 혹은 안드로이드 에뮬레이터 클라이언트 PC

### 8.2. Backend & DataBase

- ◎ S/W: Java 8, Spring Boot, Spring Security, Spring Data JPA, IntelliJ IDEA, MySQL
- ◎ H/W: Mac OS BigSur, AWS 리눅스 서버

### 8.3. Configuration Management

- ◎ Github, Notion

## 9. Glossary

Term	Description
쿠폰 판	쿠폰 도장을 적립하는 판
도장	오프라인 매장에서 리워드를 받기 위해 모으는 것
QR 코드	컴퓨터가 만든 흑백 격자무늬 패턴 코드
리워드	도장을 모아서 오프라인 매장으로부터 받는 보상
스킨	앱 내 포인트를 모아 얻을 수 있는 쿠폰 판과 도장 이미지
점주	가게를 관리하고 가게의 대부분의 일을 담당하는 사람
회원	애플리케이션을 이용하는 소비자
Mock Up	실제품을 만들어 보기 전, 디자인의 검토를 위해 실물과 비슷하게 시제품을 제작하는 작업의 프로세스, 결과물
커스터마이징	고객이 기호에 따라 제품을 요구하면 생산자가 요구에 따라 제품을 만들어 줌
스플래시 스크린	애플리케이션 실행 시 페이지의 컨텐츠가 로딩되기까지 일시적으로 보여주는 화면

## 10. Reference

[QR 코드 정의]

[https://ko.wikipedia.org/wiki/QR\\_코드](https://ko.wikipedia.org/wiki/QR_코드)

[Adobe XD]

<https://helpx.adobe.com/xd/help/cloud-documents.html>

[Spring]

<https://docs.spring.io/spring-framework/docs/current/reference/html/>

[StarUML]

<http://staruml.io/>