**아! 카페**

데이터베이스 최종보고서

| Team 8 | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Team**  **member** | 이동건  김효균  공윤환  최유미 | **ID** | 201221005  201421006  201620614  201621036 |  |  |

**목차**

[**프로젝트 개요**](#_lz5rt5b7v8gz) **2**

[주제](#_gjdgxs) 2

[선정배경](#_30j0zll) 2

[**개발 환경**](#_1fob9te) **2**

[**데이터베이스 스키마 및 UML 다이어그램**](#_2et92p0) **2**

[데이터베이스 스키마](#_tyjcwt) 2

[UML다이어그램](#_1t3h5sf) 4

[**시스템 구성 및 기능**](#_2s8eyo1) **4**

[시스템](#_17dp8vu) 4

[Store](#_vv0og8we7n2y) 5

[User](#_ts00rorkjvx5) 5

[**시연화면**](#_y57ye0laqvp) **5**

[아! 카페 Web](#_35nkun2) 5

[아! 카페 App](#_1ylzdqtfhvzr) 6

[**소스코드 분석**](#_z337ya) **8**

[Trigger 분석](#_3j2qqm3) 8

[SP 분석](#_4i7ojhp) 8

[JDBC 연결 분석](#_1ci93xb) 9

[**추후 개선 및 보완할 점**](#_qsh70q) **10**

[**구현시 어려웠던 점**](#_3as4poj) **10**

**표 목차**

[Table 1 개발환경 3](#_3znysh7)

[Table 2 스키마 4](#_3dy6vkm)

[Table 3 시스템 기능 5](#_3rdcrjn)

[Table 4 Store 기능 5](#_26in1rg)

[Table 5 User 기능 6](#_lnxbz9)

[Table 6 아! 카페 Web 7](#_1ksv4uv)

[Table 7 아! 카페 App 8](#_2jxsxqh)

**그림 목차**

[Figure 1 UML 다이어그램 5](#_4d34og8)

[Figure 2 Trigger code 9](#_1y810tw)

[Figure 3 Stored procedure code 10](#_2xcytpi)

[Figure 4 JDBC 연결 code 10](#_3whwml4)

[Figure 5 SQL code 11](#_2bn6wsx)

# 프로젝트 개요

## 주제

아주대학교 재학생 및 교직원을 위한 교내 카페 주문예약 및 픽업시간 지정이 가능한 웹 및 모바일 앱을 구현할 예정입니다.

## 선정배경

교내에 있는 카페들은 현장주문만 가능하여 특정 시간에 손님이 몰려 대기열이 발생하는 경우가 빈번합니다. 학교 근처에 온라인 주문 서비스를 도입한 프랜차이즈 카페가 있으나 원하는 시간에 음료를 take out 할 수 있는 기능이 없습니다. 따라서 아주대학교 재학생 및 교직원을 위해 음료를 미리 주문하고 take out 시간까지 고객이 직접 지정할 수 있는 시스템을 구현할 예정입니다.

# 개발 환경

| Backend | Playframe2(HTTP Server) Java/Scala + JDBC(SQL)  선택 이유: 기존에 가장 사용해본 익숙한 framework이고, Strong/Static Type을 이용하여 프로그램 개발 시 에러를 줄일 수 있습니다.  간편하고 Async IO 기반으로 Thread/request 를 사용하는 타 프레임워크 대비 빠르게 작동합니다 |
| --- | --- |
| Frontend | Android Native App  선택 이유: 외부에서 주문을 하는 특성상 사용자가 PC가 아닌 모바일 환경에서 더 많이 사용할 것이라고 생각했습니다. 팀 내부에서 가장 원활하게 사용 가능한 안드로이드 개발환경을 프론트 엔드로 선택했습니다 |
| Database | MariaDB  선택 이유: 많은 user base와 Reference로 인해 Trouble Shooting이 편리합니다. |

**Table 1 개발환경**

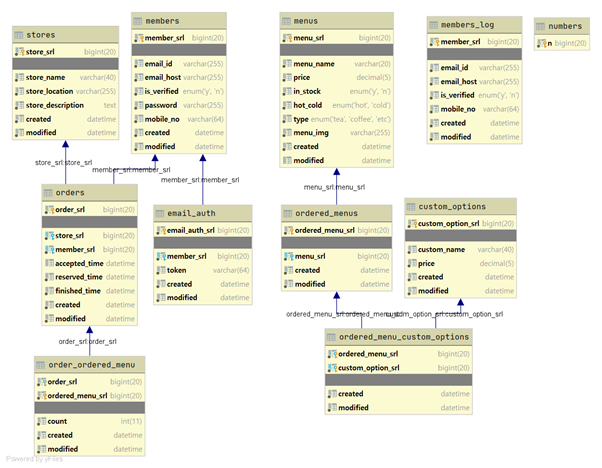
# 데이터베이스 스키마 및 UML 다이어그램

## 데이터베이스 스키마

| **Table1: custom\_options** | **Table2: members** |
| --- | --- |
| 각 음료별 옵션(사이즈 업그레이드, 샷추가, 테이크아웃) 등을 지정하는 테이블 | 회원가입 관련 기능을 하는 테이블 |
| **Table3: email\_auth** | **Table4: menus** |
| 이메일 인증 기능을 하는 테이블 | 메뉴 테이블 |
| **Table5: ordered\_menus** | **Table6: ordered\_menu\_custom\_options** |
| 주문된 메뉴 테이블 | 주문한 메뉴와 custom option을 붙인 테이블 |
| **Table7: stores** | **Table8: orders** |
| 상점 리스트 테이블 | 메인 주문 테이블 |
| **Table9: order\_ordered\_menu** |  |
| 주문과 주문된 메뉴 연동 테이블 |  |

**Table 2 스키마**

## UML다이어그램



**Figure 1 UML 다이어그램**

# 시스템 구성 및 기능

## 시스템

| 회원가입 | 회원가입을 할 수 있다. |
| --- | --- |

**Table 3 시스템 기능**

## Store

| 주문 내역 확인 | 전체 주문 내역을 확인할 수 있다. |
| --- | --- |

**Table 4 Store 기능**

## User

| 음료주문 | 교내카페, 음료를 선택하고 픽업시간을 남겨 주문을 할 수 있다. |
| --- | --- |
| 장바구니 | 여러 주문을 장바구니에 저장하고 나중에 한 번에 결제할 수 있다. |

**Table 5 User 기능**

# 시연화면

## 아! 카페 Web

| 1.회원가입  아주대 이메일, 비밀번호, 휴대폰 번호 입력 후 회원 가입 | 2. 로그인  아주대 이메일, 비밀번호 입력 후 로그인 |
| --- | --- |
|  |  |
| 3. 주문 - 1  카페선택, 픽업시간 선택 | 3. 주문 - 2  메뉴선택, 커스텀 선택, 장바구니 담기 |
|  |  |
| 4. 주문하기 주문내역 확인 후 주문하기 | 5. 주문내역 확인 전체 주문내역 확인 |
|  |  |

**Table 6 아! 카페 Web**

## 아! 카페 App

| 1. 앱 메인화면 | 2. 카페 선택 |
| --- | --- |
|  |  |
| 3. 음료 종류 선택 | 4. 음료 선택 후 장바구니에 추가 |
|  |  |
| 5. 장바구니에서 주문넣기 | 6. 주문목록에서 주문 확인 |
|  |  |

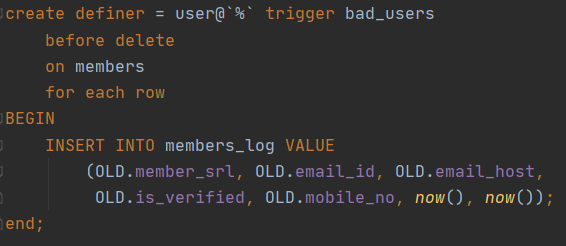
**Table 7 아! 카페 App**

# 소스코드 분석

## Trigger 분석

trigger bad\_users

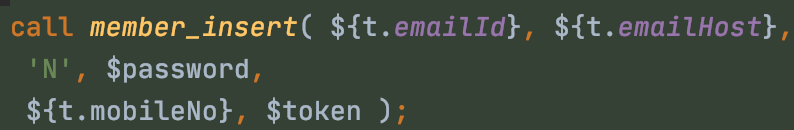
악성 유저 발생 시 회원 정보를 삭제하고 시스템을 이용할 수 없게 합니다. 회원 정보(member table)가 삭제될 때 Before DELETE trigger를 이용하여 회원 정보를 따로 백업합니다.

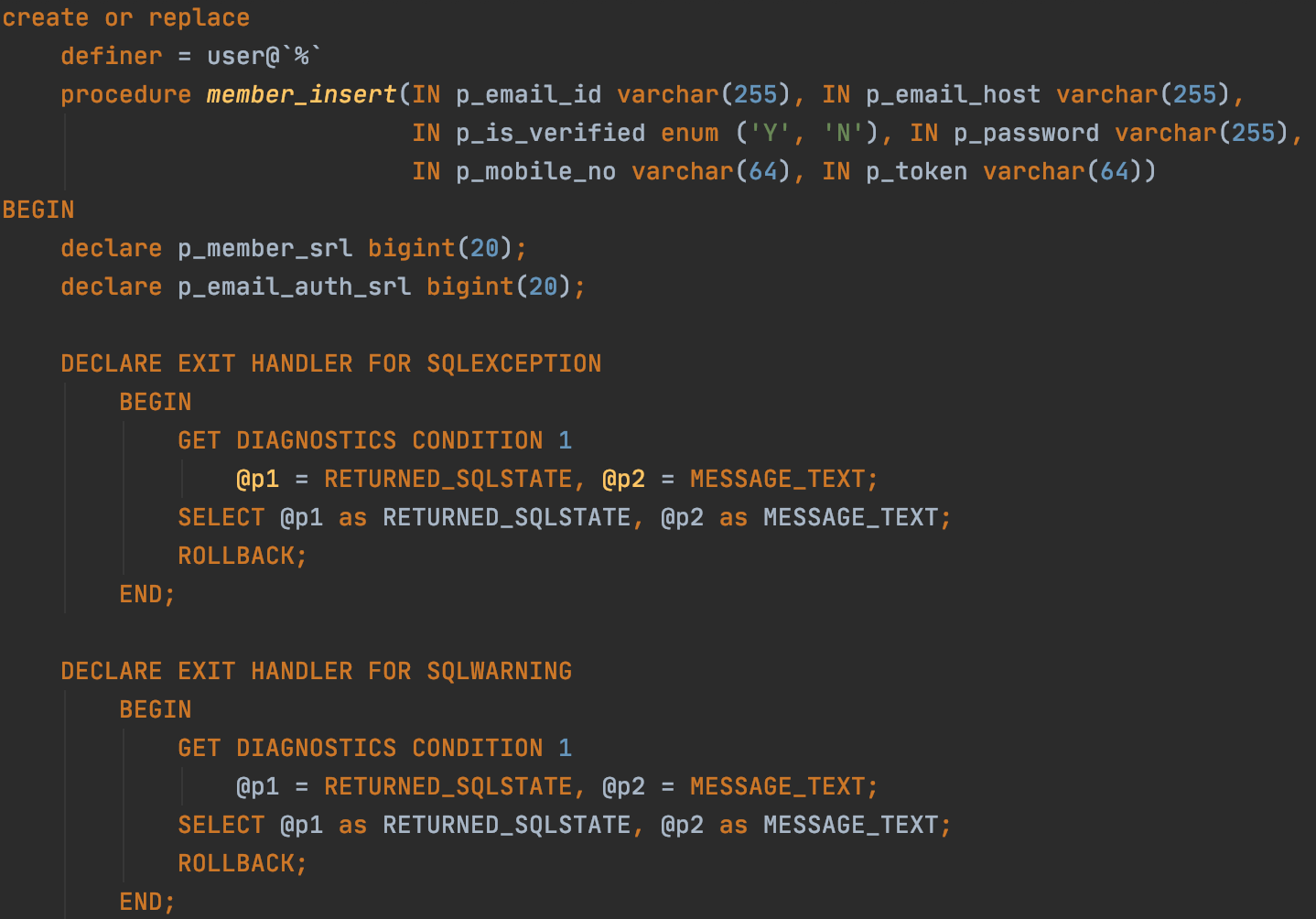


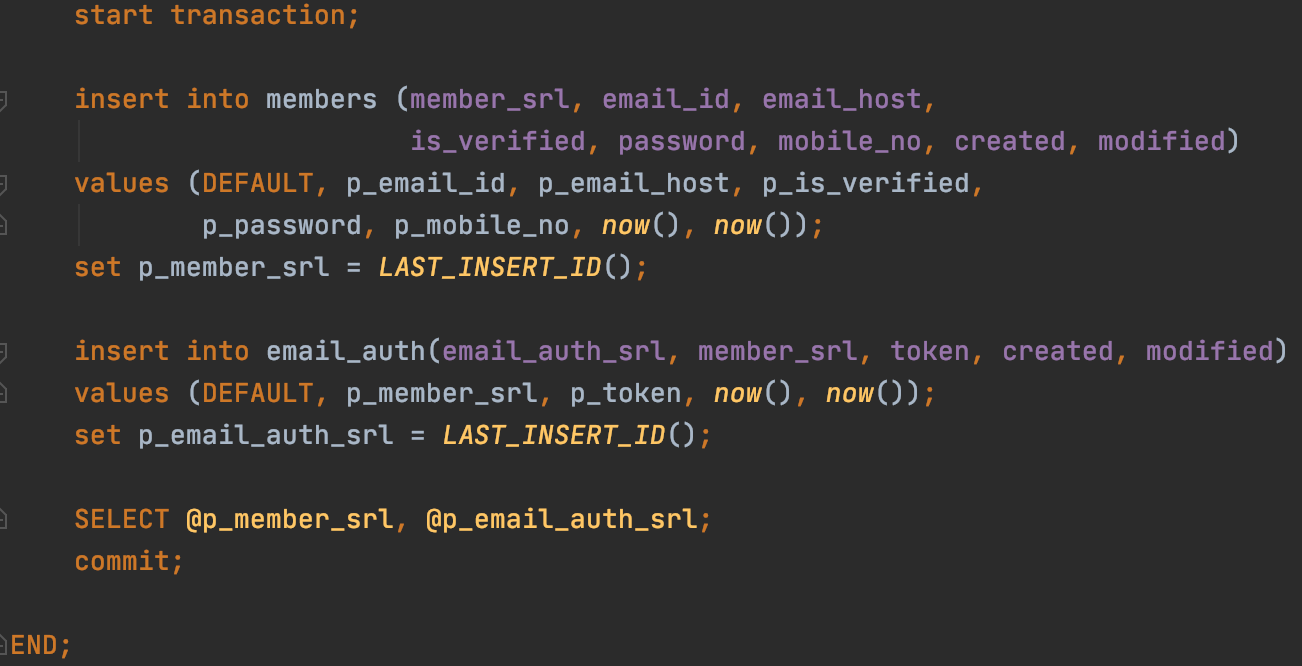
**Figure 2 Trigger code**

## SP 분석

회원 가입시 두 개의 테이블에 INSERT가 발생합니다. HTTP 서버 쪽에서 JDBC를 활용하여 INSERT를 할 경우 DB서버와의 통신이 여러번 일어나게 됩니다. 따라서 통신 시간을 줄이고, 빠른 Insert 를 위해 미리 Query를 Compile 하기 위해서 JDBC를 사용한 SQL문 통신 대신 Stored Procedure를 사용했습니다.





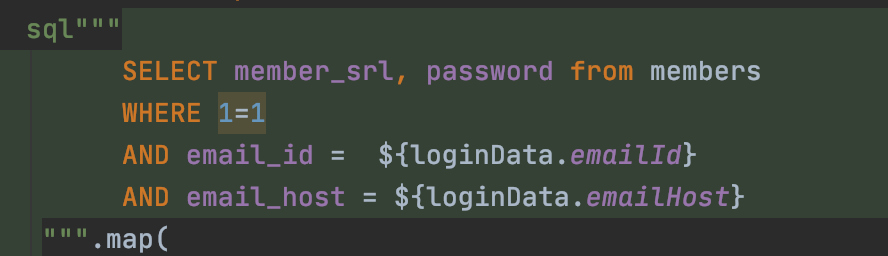


**Figure 3 Stored procedure code**

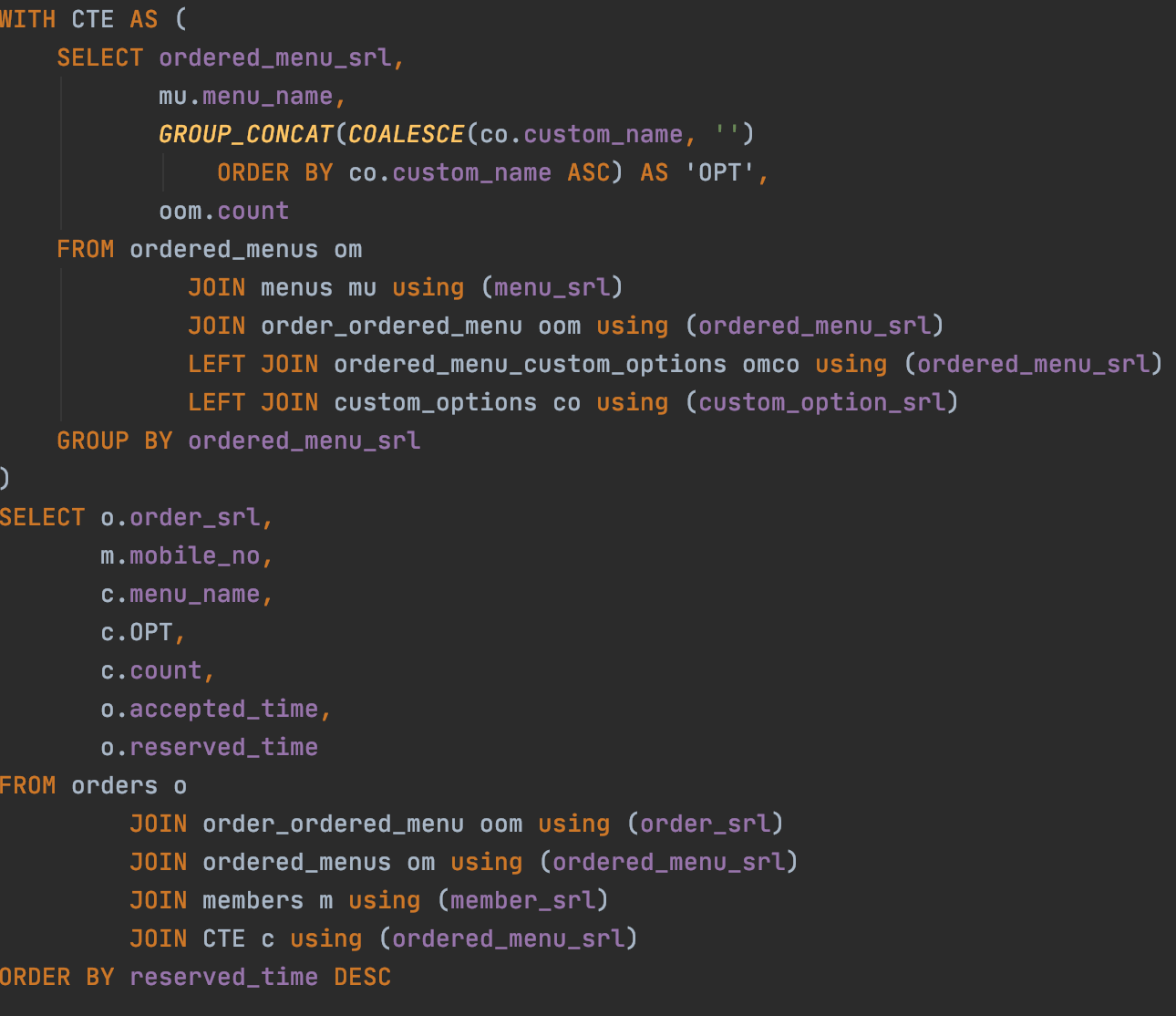
## JDBC 연결 분석

사용자의 SQL Injection 공격을 막기 위하여 모든 SQL 문장 실행시에 Parameter Binding을 자동으로 실행하도록 하였습니다. 일반 string 과는 다르게 sql”””을 사용합니다.

sql””” 로 시작하는 String에 일반 값을 평소 처럼 집어넣으면 scalikejdbc라는 데이터베이스 라이브러리에서 자동으로 parameter binding을 적용해 줍니다.



**Figure 4 JDBC 연결 code**

작성한 SQL 쿼리문 중에서 복잡했던 SQL 입니다.

**Figure 5 SQL code**

주문한 메뉴에 커스텀(사이즈업, 진하게, 테이크 아웃)이 있을수도, 없을수도 있어서, 주문한 메뉴들과 커스텸을 join 할때는 LEFT JOIN을 사용하였습니다. 또한 여러줄의 커스텀을 한줄로 보여 주기 위하여 GROUP\_CONCAT 함수를 사용하였으며, NULL 값을 빈 STRING으로 바꿔 주기 위하여 COALESCE 함수를 사용했습니다.

# 추후 개선 및 보완할 점

해당 서비스의 구현은 서비스의 시연을 위한 데모버전으로 구현이 되었기 때문에 정식으로 서비스를 하기 위해서는 사용자의 편의를 위한 UI 업데이트가 필요 합니다. 또한 현재는 여러가지로 UX 및 보안 면에서 문제점이 있어서, 실제 서비스로 업데이트 하기 위해서는, 보안 및 UX 부분을 개선해야 합니다.

# 구현시 어려웠던 점

DB(MariaDB) 와 Java/Scala/JVM과 연동하는 부분과 그것을 JSON으로 바꿔서 통신하는 부분에는 큰 어려움은 없었으나, DB에서 가져온 데이터를 웹 및 안드로이드에 연동하는 과정에서 어려움이 있었습니다. 또한 Stored Procedure과 Trigger는 많이 사용해 보지 않아서, 작성하는데 익숙하지 않은 면이 있었습니다.