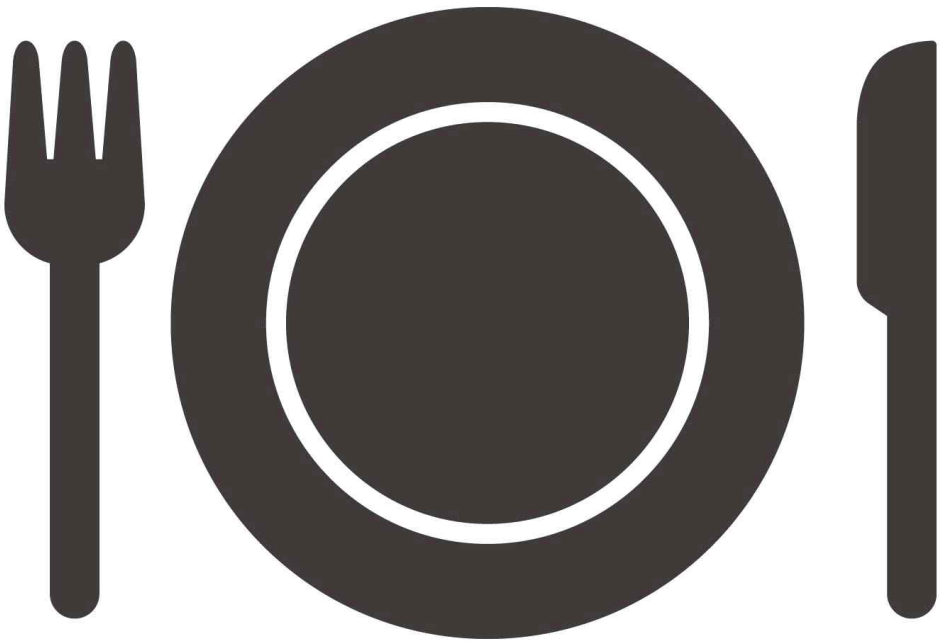
Yeungnam University

# 전 세계의 음식들

## -Design document-



# [ Revision history ]

| Revision date | Version # | Description | Author |
|---|---|---|---|
| | #1.1 | class diagram, sequence diagram에서 operation 이름 등 일부 변경 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

= Contents =

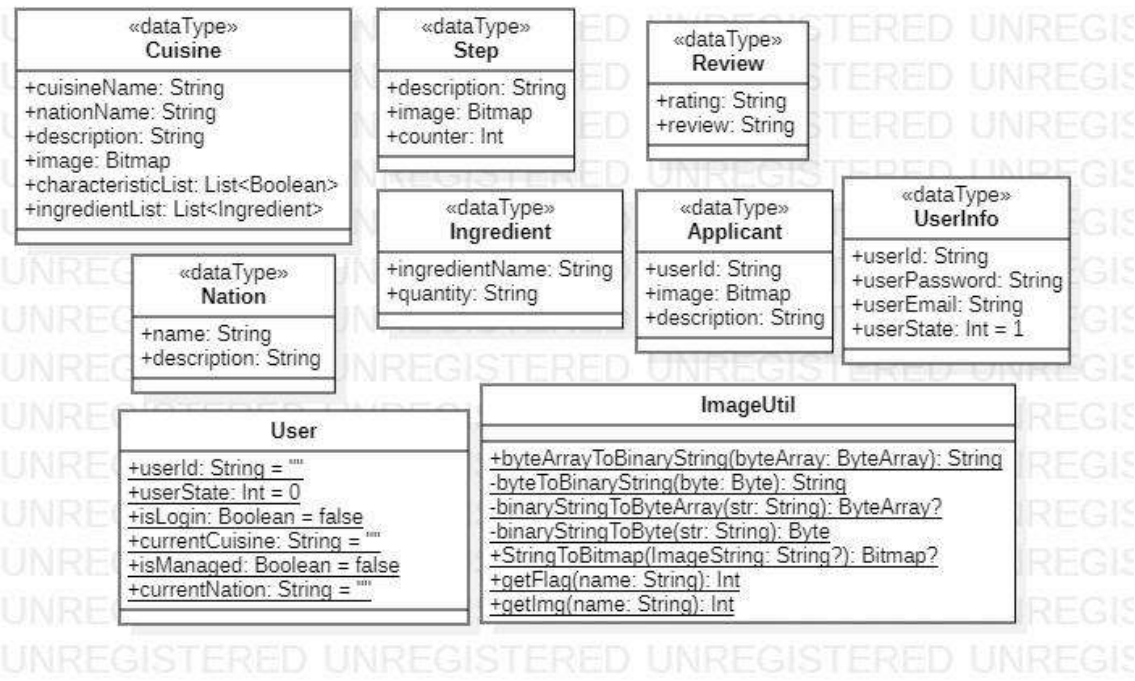
# 1. Introduction

지난 analysis문서에서 시스템이 '무엇을 하는가'에 맞춰 요구사항들을 분석했다. 본 보고서에서는 이전 보고서의 내용을 바탕으로 시스템의 기능들을 '어떻게 구현하는가'를 기술한 design보고서이다. 이 목적에 맞춰 보고서 전반에 걸쳐 class diagram, sequence diagram, state machine diagram을 그리고 각각에 맞는 설명을 적었다. 이를 통해 시스템을 구성하는 빌딩 블록들을 analysis문서에 이어 계속해서 쌓아 올렸다. 다만 본 시스템은 로컬과 서버를 이용한 데이터베이스도 구현할 생각이나 이를 문서에 기술하는 것은 수업의 범위를 벗어나는 것이기 때문에 비록 데이터베이스를 광범위하게 사용하긴 하지만 문서에 그 방법과 구조를 기술하진 않았다. 안드로이드 어플로 개발할 계획이고 MVVM 패턴을 사용한다. 문서를 다 읽고 나면 시스템을 어떤 식으로 구현할지에 대해 모두 이해할 수 있을 것이다.

## 2. Class diagram

안드로이드 MVVM 패턴에 기반을 둔 layered structure를 시스템의 전체적인 틀로 잡았다. UML model은 packages, classes, associations로 구성되어 있고 그것을 그래픽으로 표현한 것이 UML diagram이다.[1] 이에 맞춰 UML model의 class들을 다양한 관점으로 바라본 결과를 class diagram으로 최대한 표현하고자 했다.

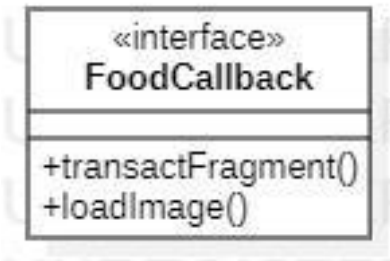
후에 다른 diagrams에서 상속 관계를 일일이 표시하는 것은 가독성을 해칠 것 같아 따로 빼서 첫 번째로 상속의 관점에서 본 class diagram을 소개한다. UML model에 있는 class들의 이름이 매우 직관적이기 때문에 상속에 관련된 class diagram 혹은 설명을 한 번 본다면 다른 diagrams에서도 상속 관계에 대해 충분히 파악할 수 있을 것이다. 두 번째는 기능의 관점에서 본 class diagram이다. 전 문서의 use case diagram을 이용해서 기능을 나눴다. 시스템의 전체적인 구조는 이 때 자세히 다뤘다. 이 문서에서 가장 핵심적인 부분이 될 것이다. class diagram에 앞서 시스템에 사용할 data type과 static method를 설명하겠다.




[그림 2-1] 시스템에 사용된 data type과 static methods

모두 클래스로 구현했고 data type이므로 association을 표시하진 않았다. Step, Ingredient는 Cuisine과 관련된 클래스이고 Applicant는 요리사가 되기 위한 지원자에 관한 클래스이다. image와 description으로 사용자는 자신의 자격을 증명할 자료를 관리자에게 보낼 수 있다. UserInfo는 회원가입에 쓰이는 dataType이다. static

methods를 갖고 있는 클래스는 User와 ImageUtil이다. 코틀린의 Object 키워드를 이용해서 구현할 예정이다. User는 시스템이 전반적으로 사용해야 하는 중요한 정보들을 담고 있다. 한편으로 서버로 이미지를 보낼 때와 서버에서 이미지를 받을 때 모두 이미지를 문자열로 변환하는데 이 때 필요한 method들과 앱에 drawable에 저장되어 있는 나라의 국기, 이미지의 id를 가져오는 methods를 ImageUtil에 담았다.
시스템에서 작성한 인터페이스는 하나이며 아래와 같다.

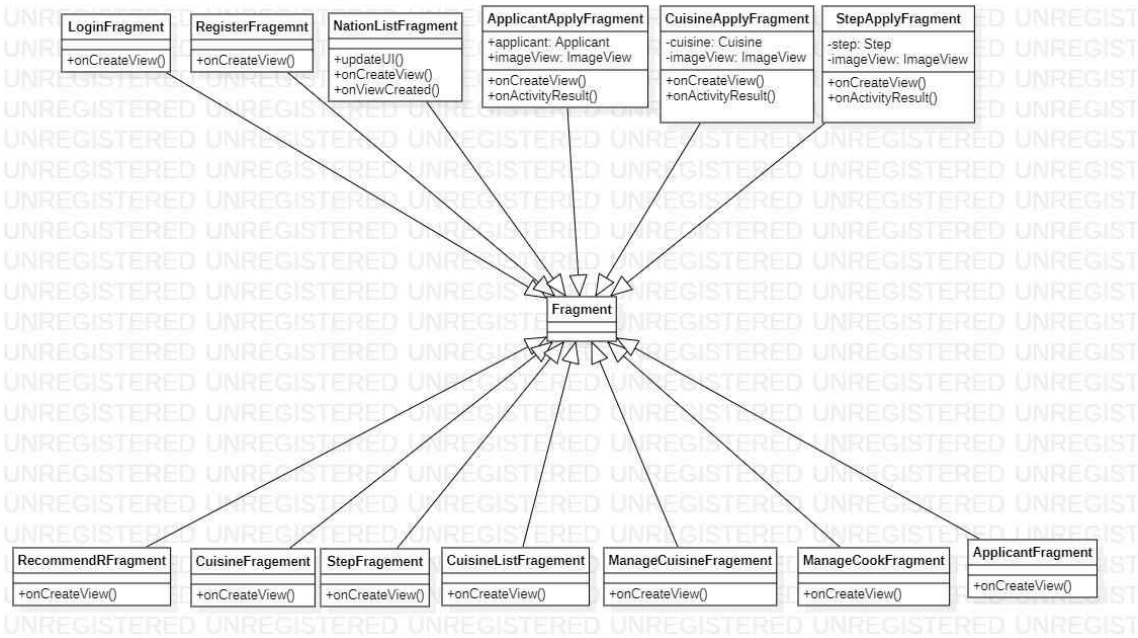



[그림 2-2] interface class

화면 전환과 이미지 불러오기는 logic에서 할 수 없기 때문에 business logic에 있는 클래스들이 fragment로부터 이를 실현한 객체를 받는다. 실현한 객체는 코틀린의 object 키워드를 이용해서 생성하기 때문에 class diagram에서 클래스로 표현하지 않았고 logic 클래스 안의 property로 작성했다.

## 2.1 상속의 관점

시스템을 구성하는 classes들 중에 Fragment로 끝나는 이름을 가진 것들은 모두 fragment를 상속한다. 각각의 class들이 Frament의 어떤 methods를 overriding할 것인지 그리고 이유가 뭔지에 대해 설명함으로써 시스템 구조의 이해를 돕기 위해 class diagram을 작성했다. Fragment 자체를 설명하고 그 모든 methods를 적는 것은 본 문서의 범위를 벗어난 것으로 생각해서 생략했다. 가독성을 위해 오버라이딩 한 함수의 이름만 적고 인자, 반환값을 표시하지 않았다.



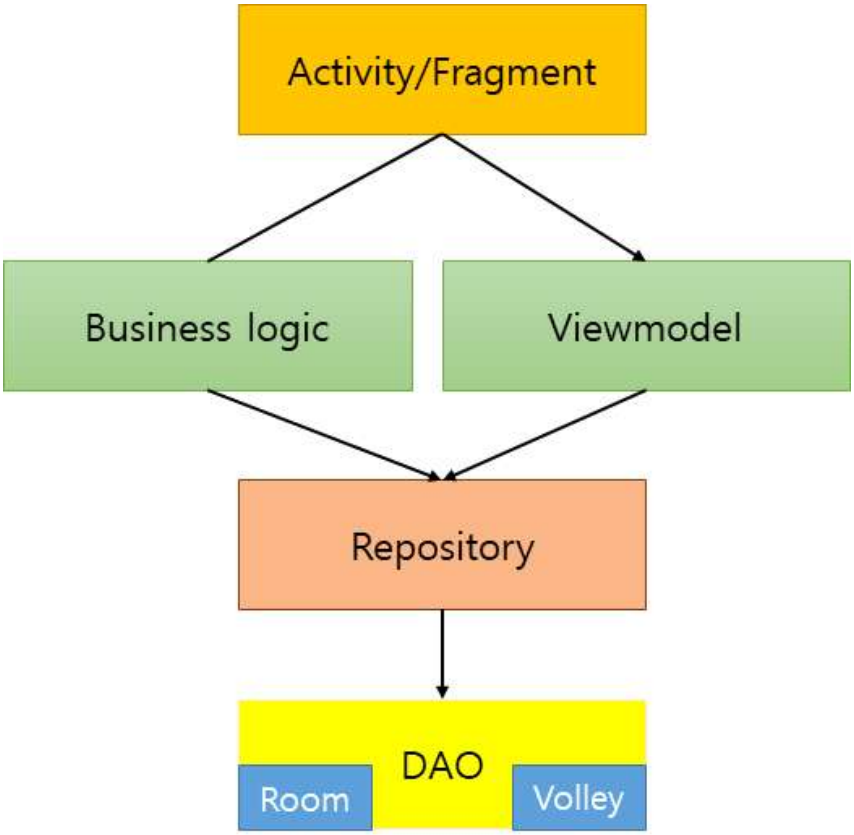

[그림 2-3] Fragment 상속관계

모두 onCreateView()를 오버라이딩한다. 여기서 databinding 혹은 observing과 관련된 작업을 수행한다. NationListFragment는 local database에서 나라의 정보를 가져오기 때문에 추가적인 operation을 오버라이딩하고 추가했다. Apply가 붙은 fragments는 이미지를 로딩해야하기 때문에 추가적인 attributes와 operations가 생긴다. 그 외에 이 diagram에는 없지만 Viewmodel이 붙은 classes는 모두 Viewmodel을, Request가 붙은 classes는 모두 StringRequest를 상속했다. 하지만 fragments와는 달리 상속 방법에 일관성이 있기 때문에 직관적으로 이해할 수 있을 것이라 생각해서 따로 diagram을 그리진 않았다. 상속은 아니지만 모든 fragments는 MainActivity클래스와 composition 관계에 있다. 왜냐하면 MainActivity가 되는 중심적인 한 화면에서 상단과 좌측은 메뉴를 가지고 있고 아래 부분만 fragment transaction으로 화면이 전환되기 때문이다. 이를 따로 diagram으로 표현하진 않았다.

## 2.2 기능의 관점

이 관점은 본 문서에 나오는 class diagram들 중에서도 가장 핵심적인 내용을 담고 있다. Use case diagram에서 class를 뽑고 그 것을 MVVM 패턴에 맞게 구조화했다. Kotlin은 항상 내부적으로 getter와 setter로 변수에 접근하기 때문에 특수한 목적에 의해 따로 작성하지 않는 한 get~~~()와 set~~~()를 구현할 필요가 없으므로

class diagram에 표시하지 않았다. 단, getter, setter의 visibility를 각각 설정해 줄 수 있기 때문에 구현 기본 원칙은 private으로 하되 변수를 외부에서 쓴다면 getter는 public, setter는 private으로 한다. Class diagram에서는 private으로 표현한다. Val 변수는 setter를 호출할 수 없으므로 외부에서 getter를 사용하는지에 따라 변수 전체를 public 아니면 private으로 설정했다. Viewmodel은 Viewmodel과 business logic으로 분리했는데, 이는 시스템과 상호작용하는 구체적인 logic을 수명 주기를 고려하여 UI 관련 데이터를 저장하고 관리하도록 설계[2]된 viewmodel에 담는 것은 viewmodel이 너무 많은 기능을 맡게 될 것이라 생각했기 때문이다. 따라서 시스템의 큰 구조는 [그림 2-4]와 같다.

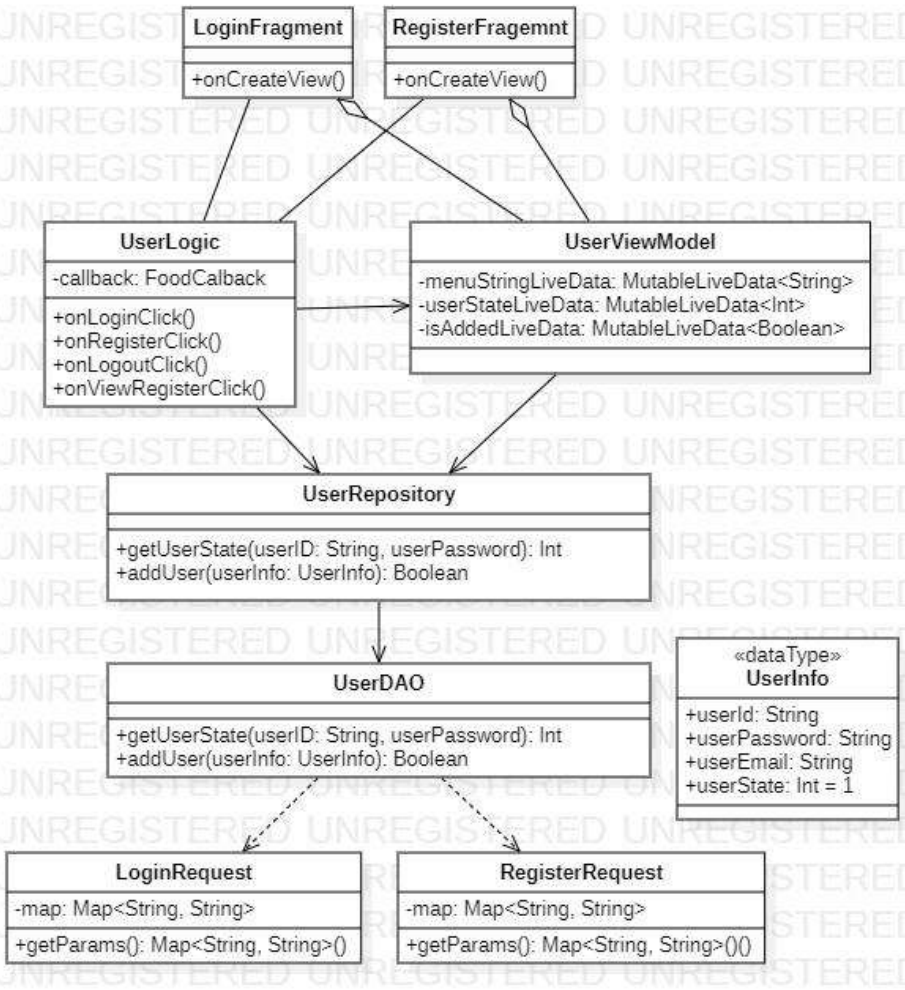



[그림 2-4] 시스템의 전체적인 구조

Android developers에서 권장하는 앱 아키텍처[3]를 조금 변형한 모습이다. Business logic은 view들과의 상호작용 알고리즘, Viewmodel은 값에 대한 읽기/쓰기를 맡는다. Fragment와 businessl logic은 callback을 통해 서로 소통하기 때문에 양방향이다. Repository는 기본적으로 DAO에서 제공하는 기능을 그대로 호출하지만 상황에 따라 반복적이고 불필요한 data accessing을 막기 위한 caching, 여러 dao methods를 조합하여 윗 계층에서 원하는 보다 적절한 데이터로 합치는 역할을 추가적으로 한다. Dao는 local인 경우 room, 서버인 경우 volley를 사용하여

데이터에 접근한다.

첫 번째 기능은 검증이다. class diagram은 [그림 2-5]와 같다.

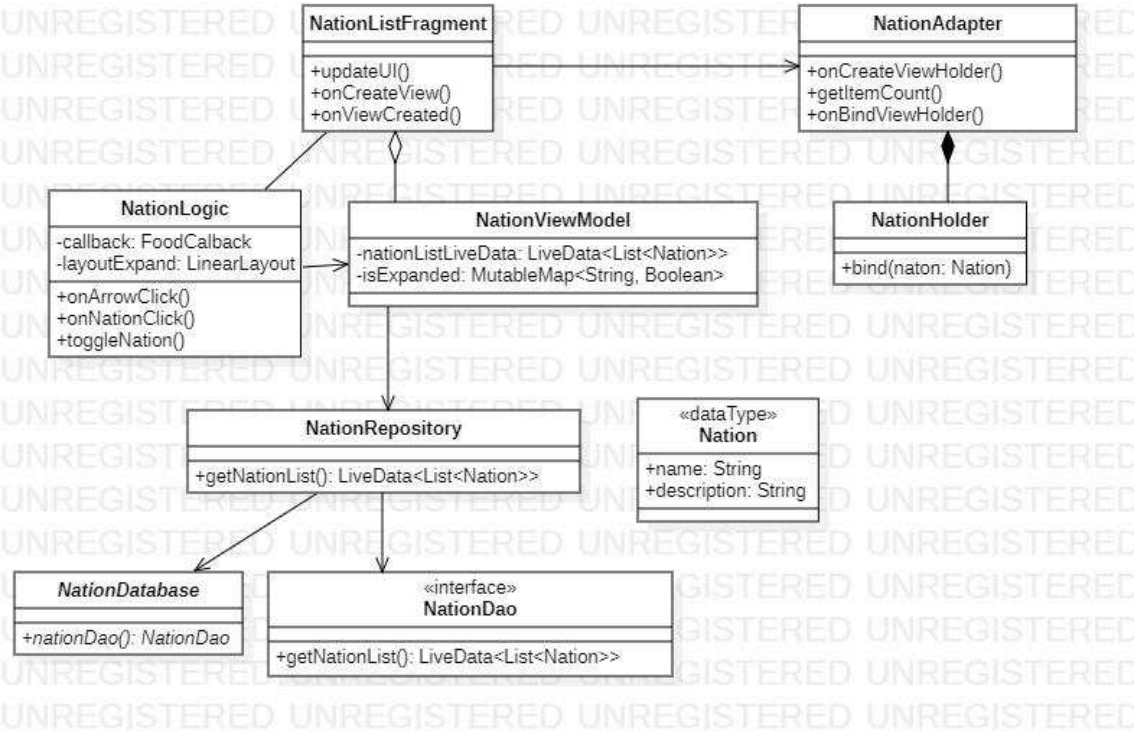



[그림 2-5] 검증 기능을 표현한 class diagram

UserViewModel에서 menuString은 로그인 상태에 달라지는 menu의 인사말과 관련된 attribute이다. Viewmodel의 attributes중에 observe를 사용하는 경우에는 자료형을 Livedata로 감싼다.

UserRepository에서 getUserState()는 로그인을 한 후 그 사용자의 상태를 가져오는 operation이다. 0은 일반 사용자, 1은 요리사, 2는 관리자로 할 계획이며 -1은 로그인 실패 코드이다. addUser()는 회원 가입시에 사용되고 성공여부를 boolean으로 반환한다.

Use case diagram에서 Log in, Register, Log out 기능과 연결된다. 다음은 나라를 표현한 class diagram이다.

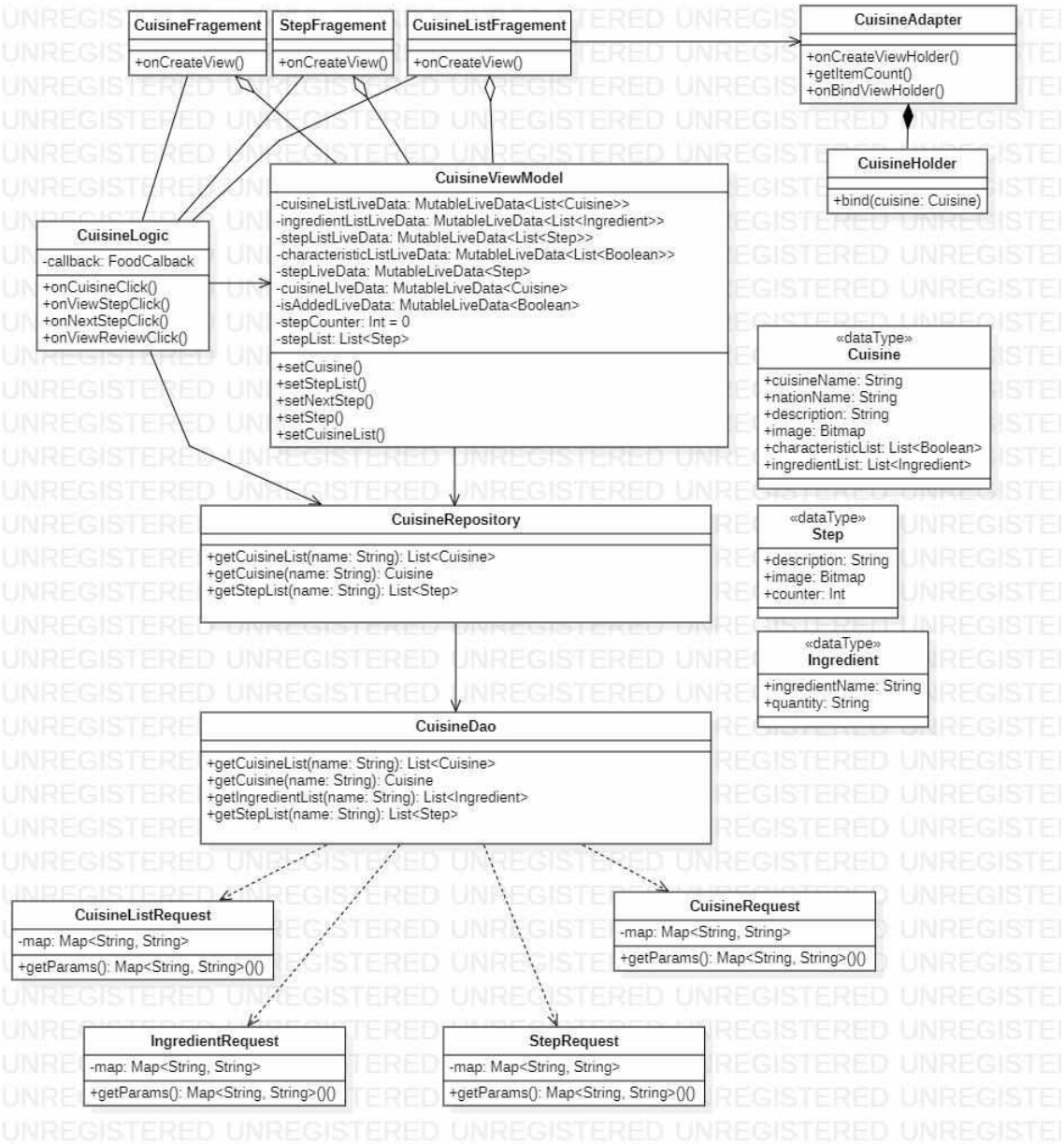



[그림 2-6] Nation 기능을 표현한 class diagram

Recycler view를 이용해 나라 목록을 띄울 계획이므로 Adapter, Holder를 사용한다. Holder에 있는 bind는 직접 제작하는 operation이고 Adapter에 있는 operations는 오버라이딩한 것들이다. 인자와 반환값은 가독성을 위해 적지 않았다.

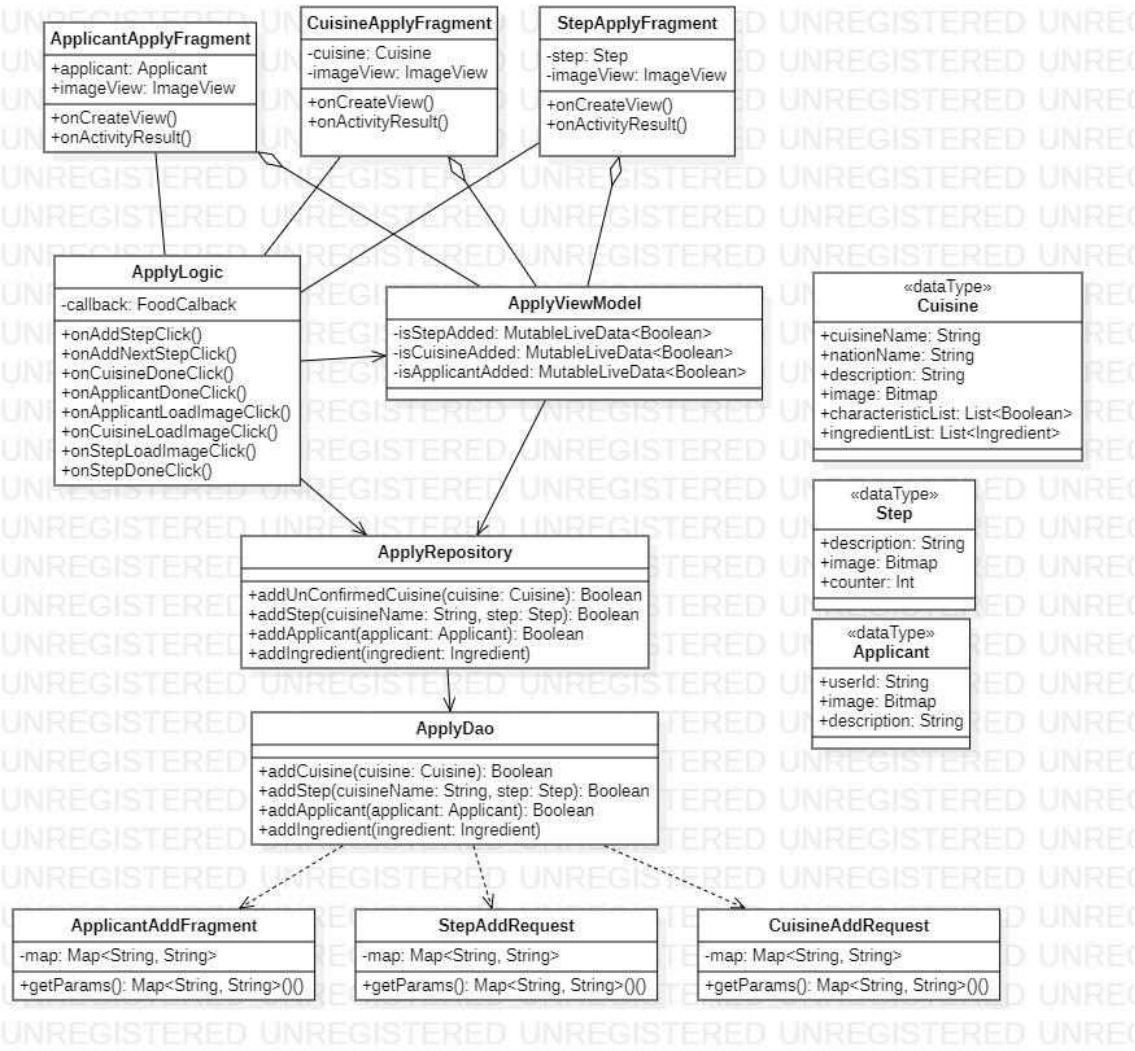
나라를 눌렀을 때 나라의 대한 사진과 설명이 확장되는 식으로 구현하기 위해 NationViewModel에서 isExpanded 변수를 추가했다. 만약 isExpanded[“Korea”]가 true라면 사용자는 한국을 눌렀고 시스템이 한국에 대한 사진과 설명을 확장한 상태일 것이다. 그리고 NationLogic은 NationHolder에서 각 나라별로 할당한다. 나라별로 추가 설명을 하는 LinearLayout을 준다. 나라에 대한 모든 정보는 local database에 저장한다.

Use case diagram에서 View nations와 연결된다. 다음은 음식을 표현한 class diagram이다.



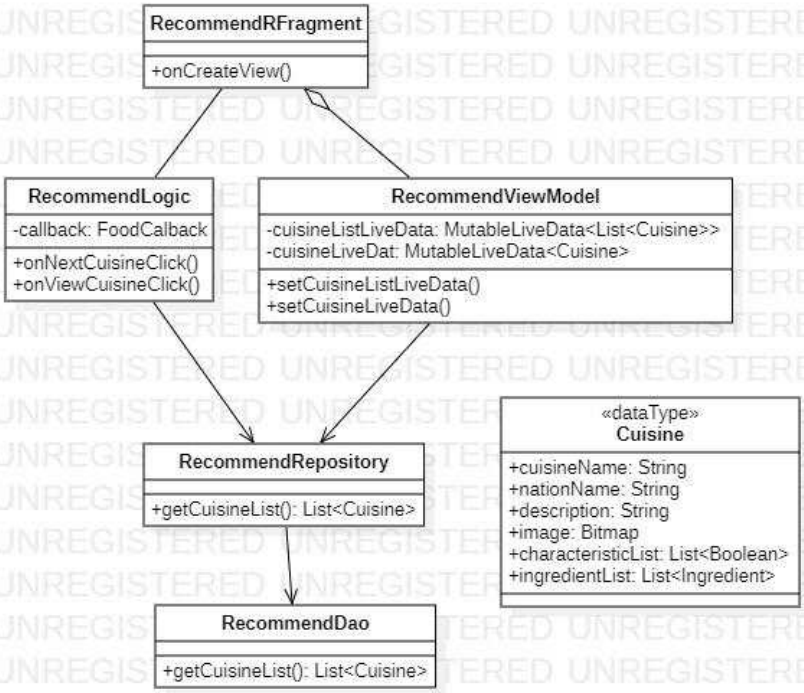

[그림 2-7] Cuisine 기능을 표현한 class diagram

음식 목록, 음식, 조리 단계, 재료 쇼핑 정보에 필요한 것들을 모았다. Use case diagram에서 View cuisines, View cuisine, View recipes와 연결된다. 다음은 신청을 표현한 class diagram이다.



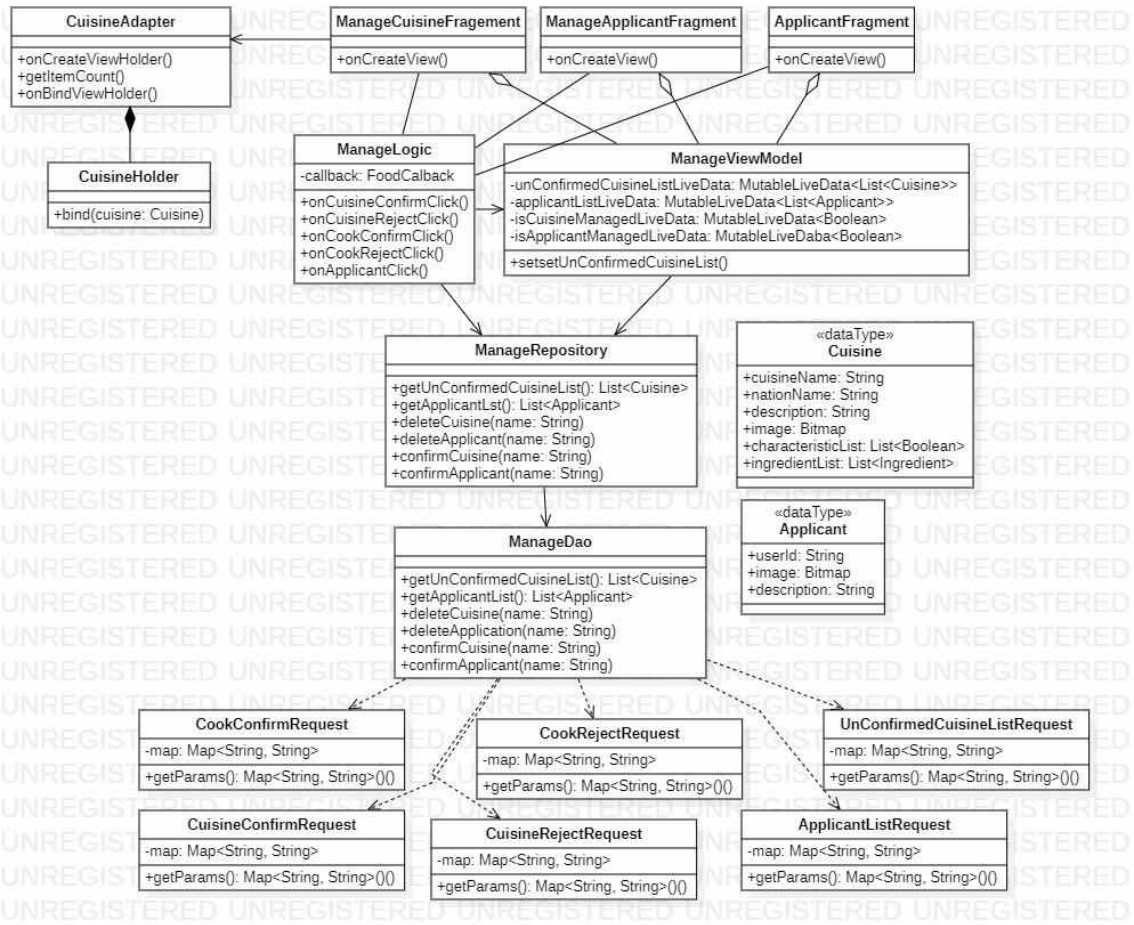

[그림 2-8] Apply 기능을 표현한 class diagram

Use case diagram에서 apply to be a cook, apply for cuisine, apply for recipes와 연결된다. 다음은 추천에 관련된 class diagram이다. 무작위로 음식을 추천해준다. Use case diagram에서 recommend cuisine과 연결된다.



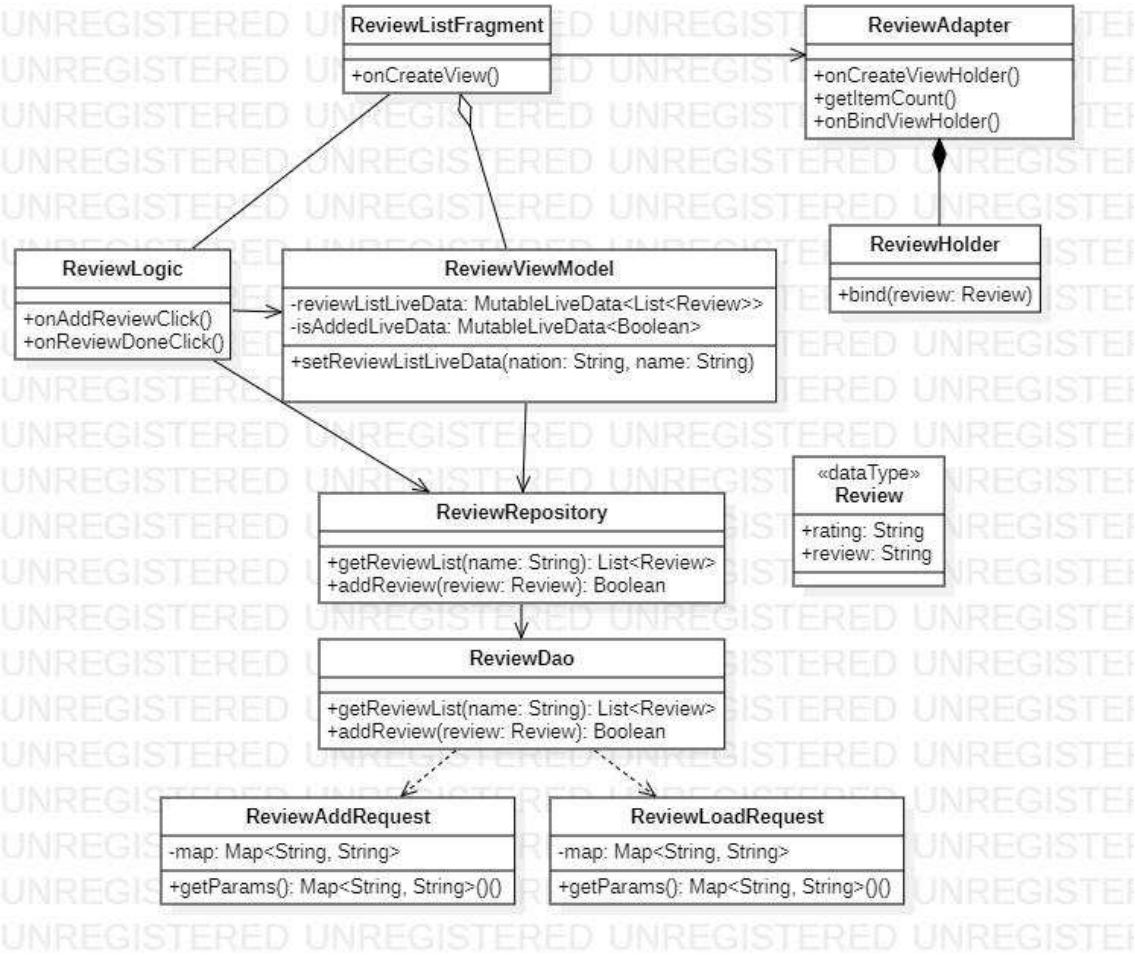

[그림 2-9] Recommend 기능을 표현한 class diagram

다음은 관리와 관련된 class diagram이다.




[그림 2-10] Manage 기능을 표현한 class diagram

구조는 다른 기능들과 유사하다. Use case diagram에서 Mange cooks, Manage cuisines에 해당한다. 마지막은 리뷰를 표현한 class diagram이다.

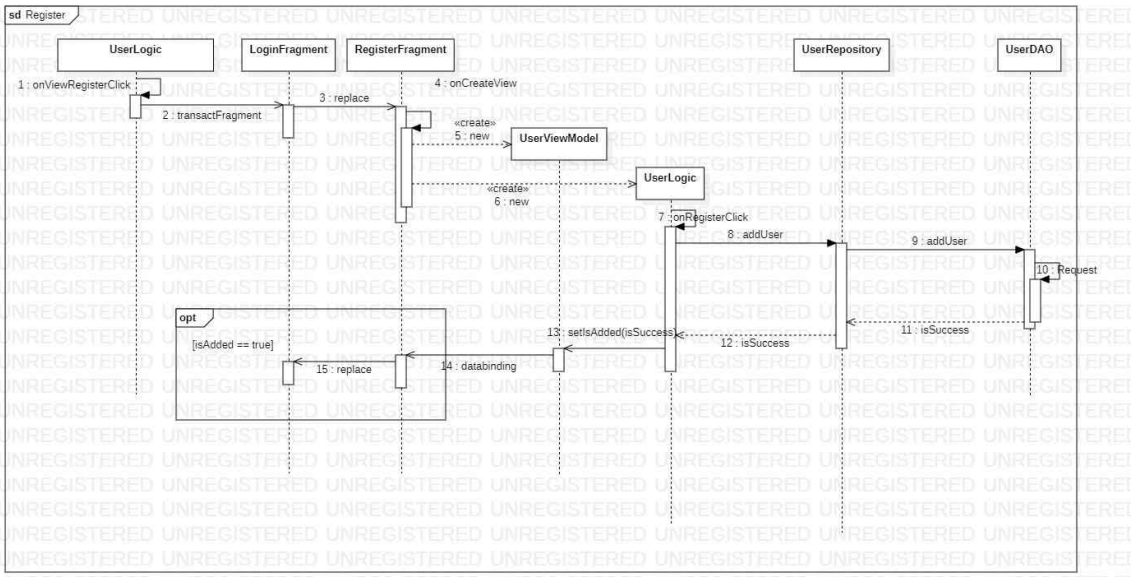



[그림 2-10] Review 기능을 표현한 class diagram

구조는 다른 기능들과 유사하다. Use case diagram에서 view reviews, add review와 연결된다.

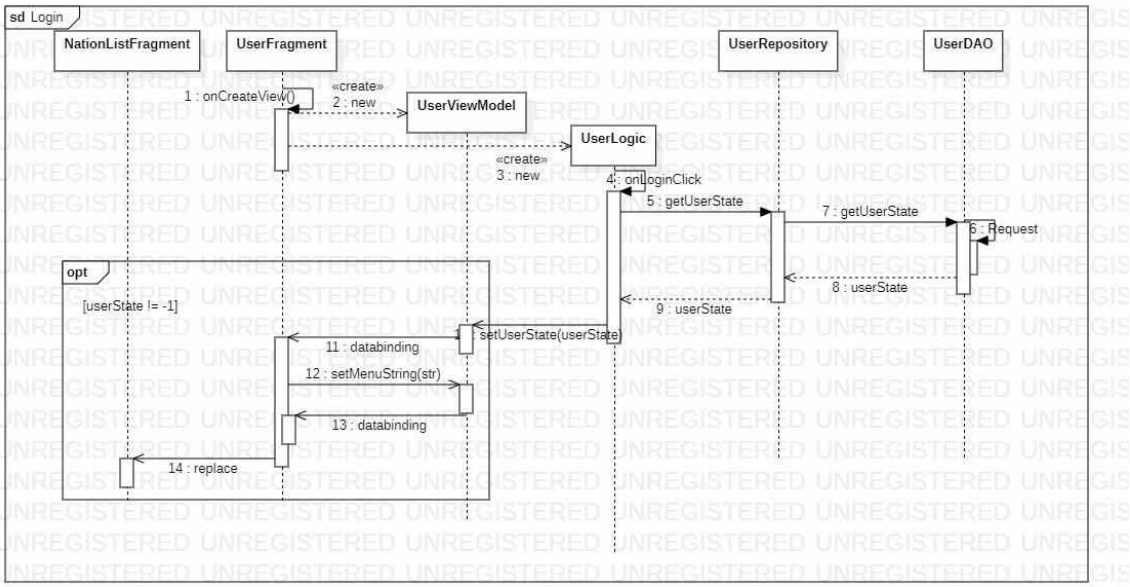
# 3. Sequence diagram

이 장에선 use case별로 추출한 sequence diagram을 각각 소개하고 설명한다. 2장에서 설명했듯이 화면은 크게 MainActivity 하나가 fragment를 감싸는 식으로 되어있다. 화면 전환은 fragment의 전환으로 구현한다. 그렇기 때문에 sequence diagram에서도 fragment의 전환이 빈번히 나오는데 전환마다 fragment 클래스가 새로 생성된다는 점에서 한 fragment가 다른 하나를 생성하는 것으로 표시할 수 있다. 하지만 예를 들어 로그인과 회원가입의 경우엔 서로가 서로를 생성하기 때문에 이를 sequence diagram에서 서로 <<create>>로 묘사하는 것엔 제한이 있다. 따라서 통일감을 위해 fragment 전환은 replace 비동기 메시지를 사용했다. Repository 클래스는 싱글톤 패턴으로 시스템이 처음 시작할 때 한 번만 생성되고 거기서 dao클래스도 생성되기 때문에 <<create>>에서 묘사하지 않았다. Recycler view를 위해 사용하는 adapter와 holder클래스는 객체들의 소통을 표현하는 것이 목적인 이 diagram에서 상세히 설명할 필요가 없다고 생각해서 생략했다. 본 시스템은 객체 이름에 큰 의미를 두지 않기 때문에 모두 클래스를 기준으로 diagram을 작성했다. 최대한 class diagram에서 정의한 operation이 모두 쓰이도록 설계했지만 흐름상 sequence diagram에 표시하지 않은 것들이 있다. 예를 들어 view cuisine의 트리거는 나라를 눌렀을 때 나온 음식 리스트를 누를 때, 관리자가 등록 요청 음식을 누를 때, 추천된 음식 보기를 누를 때 총 3가지인데 모두 표현할 수 없어 diagram에 표시하지 않았다. 그렇지만 operation들은 모두 직관적인 이름을 가지고 있고 모든 sequence diagram이 일관적인 흐름을 가지고 있기 때문에 표시되지 않았더라도 충분히 operation의 시스템상의 용도나 흐름을 알 수 있을 것이다. Transactfragment와 loadImage는 모두 logic에서 callback attribute를 통해 호출하는 Foodcallback의 operation이다. 첫 번째 sequence diagram은 회원 가입이다.



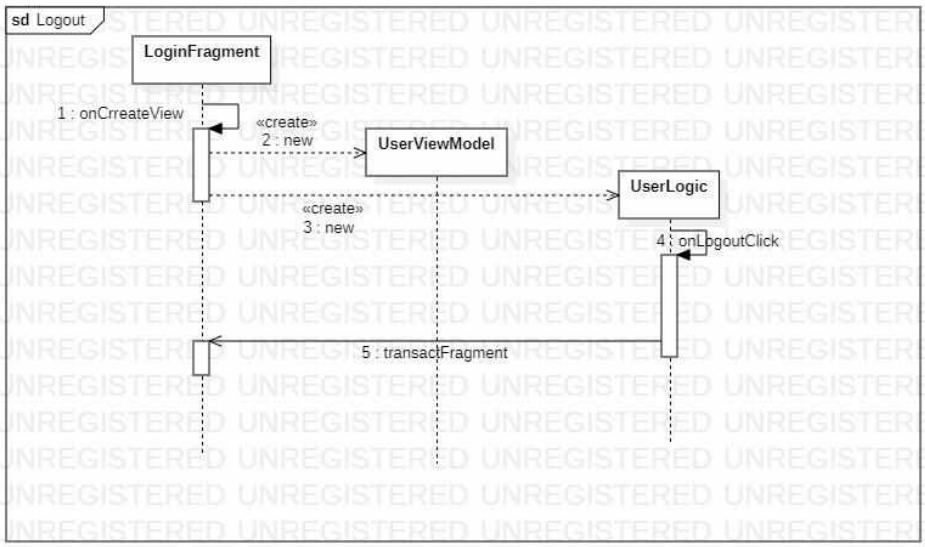

[그림 3-1] Register sequence diagram

로그인 화면에서 회원가입 버튼을 누르면 UserLogic에서 callback을 통해 회원가입 화면으로 전환한다. 회원가입 화면에서 회원가입을 누르면 데이터베이스에 접근해서 유저 정보를 등록한다. 성공하면 로그인 화면, 실패하면 회원가입 화면에 계속 있는다. Databinding은 xml과 클래스의 데이터를 직접적으로 연결해주는 기능으로 MVVM패턴에서 자주 사용한다. 거의 모든 sequence diagram에서 볼 수 있는데 이 diagram의 12번처럼 viewmodel의 값이 변경되면 fragment클래스에서 변화된 값을 감지해 어떤 행동을 하거나 혹은 뷰와의 연결을 통해 화면의 값을 즉각적으로 바꾸는 역할을 한다. 다음은 로그인이다.



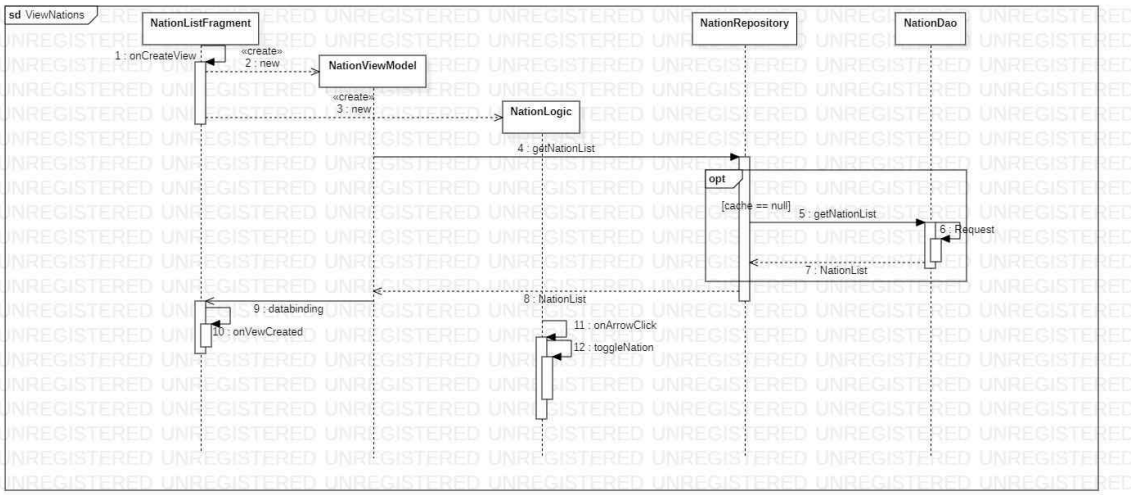

[그림 3-2] Login sequence diagram

가져온 userState를 통해 로그인 성공 시에 메뉴에 환영메시지를 띄우고 나라 리스트 화면으로 넘어간다. 다음은 로그아웃이다.



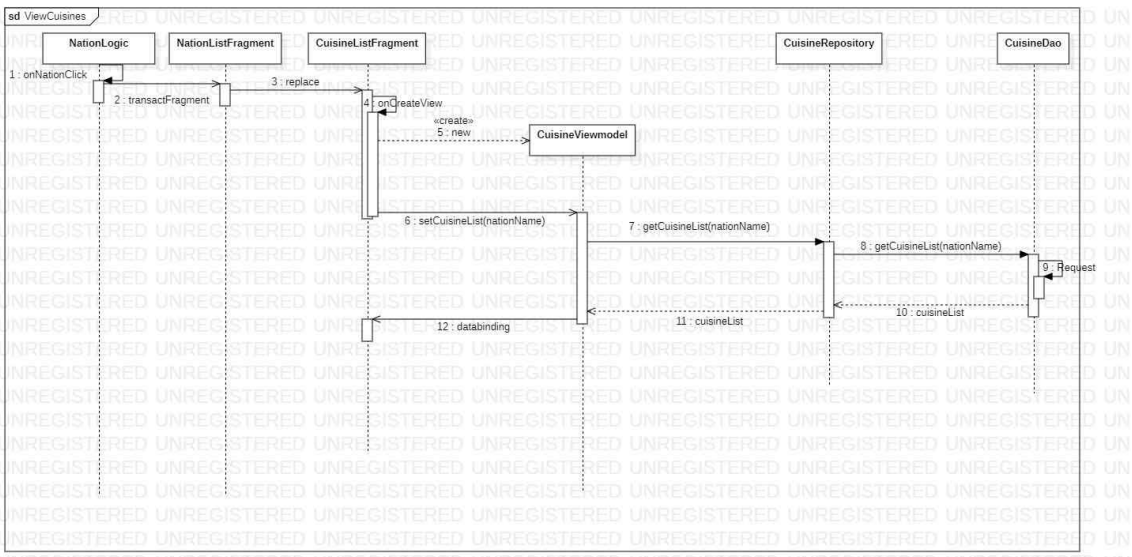

[그림 3-3] Logout sequence diagram

Loginfragment는 로그인 상태에 따라 로그인/로그아웃 페이지를 각각 띄운다. 5번에서 fragment 전환이 일어나면 다시 LoginFragment가 띄워지지만 화면은 로그인 페이지로 바뀐다. 다음은 나라 리스트 보기이다.



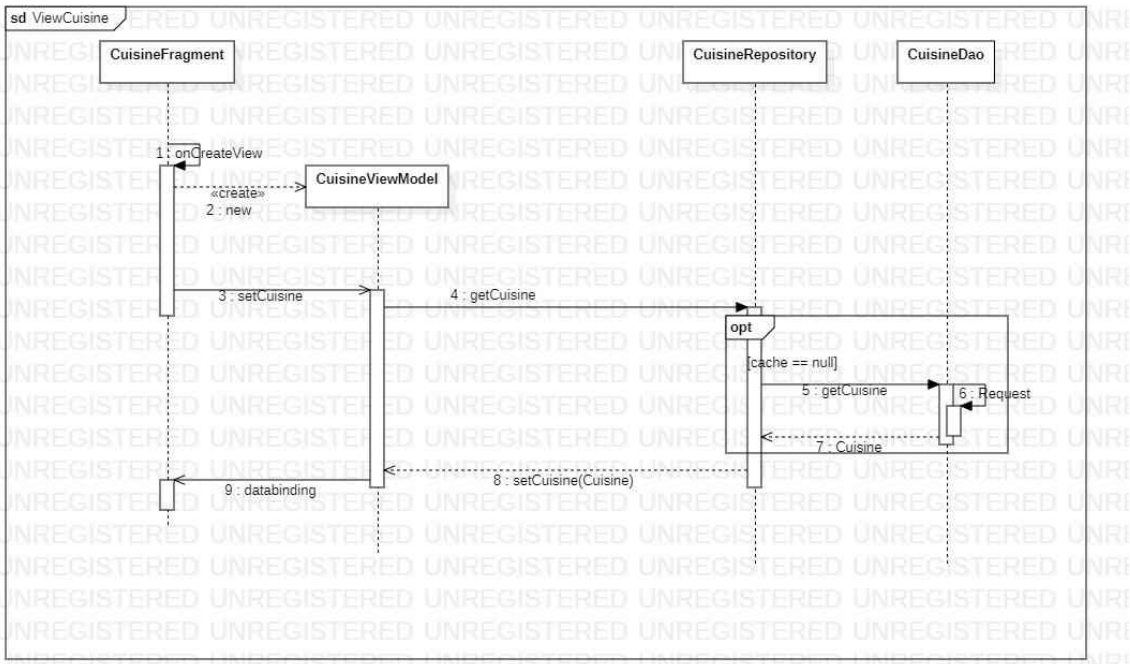

[그림 3-4] View nations sequence diagram

화면 전환마다 로컬DB에서 나라 리스트를 가져오는 것은 비효율적이므로 처음 불러왔을 때 캐시에 저장하고 그 후엔 캐시에 있는 값을 반환한다. NationViewModel의 생성자에서 getNationList가 호출된다. 다음은 음식 리스트 보기다.



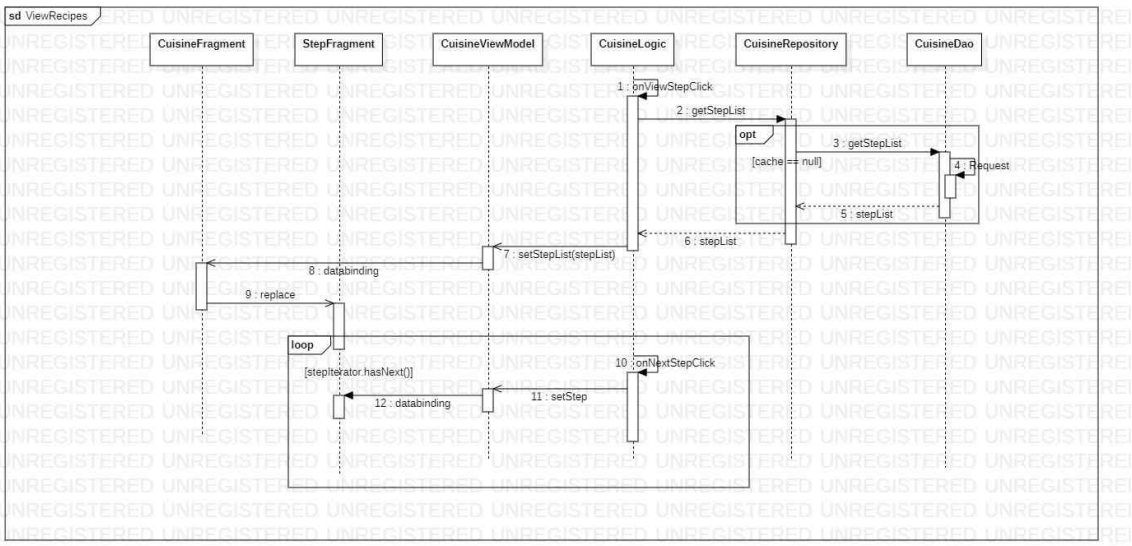

[그림 3-5] View cuisines sequence diagram

나라를 클릭하면 나라 리스트 화면에서 음식 리스트 화면으로 전환된다. 서버에서 정보를 가져온다. 다음은 음식 보기이다.




[그림 3-6] View cuisine sequence diagram

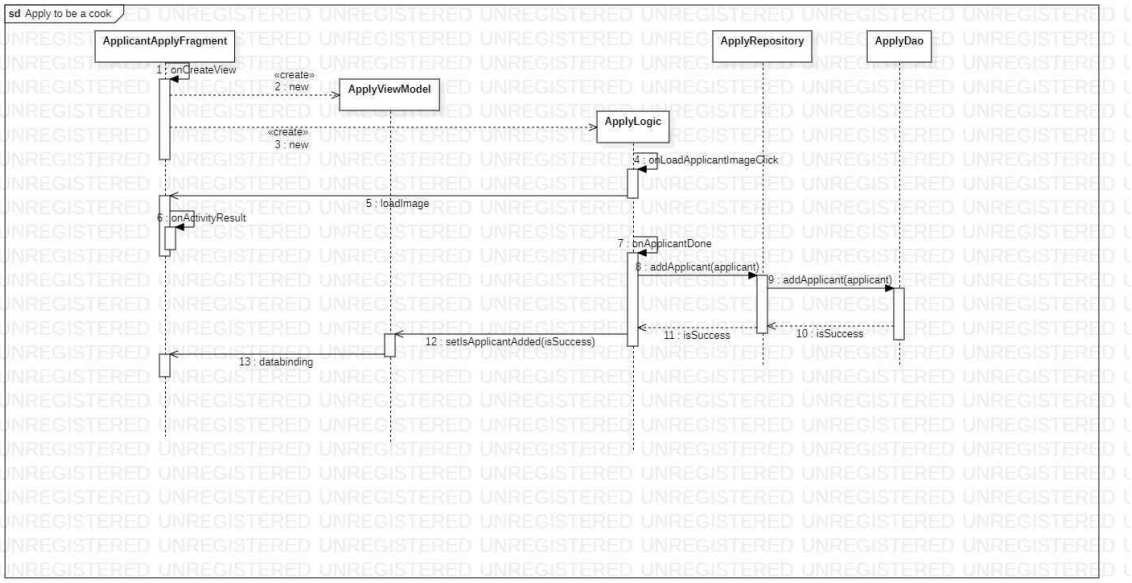
음식 정보는 바뀔 일이 거의 없으므로 캐시를 사용한다. 음식 리스트에서 음식을 선택했을 때, 음식 관리하기에서 음식을 선택했을 때, 음식 추천 페이지에서 음식 보기를 눌렀을 때 시작된다. 다음은 조리법 보기이다.




[그림 3-7] View recipe sequence diagram

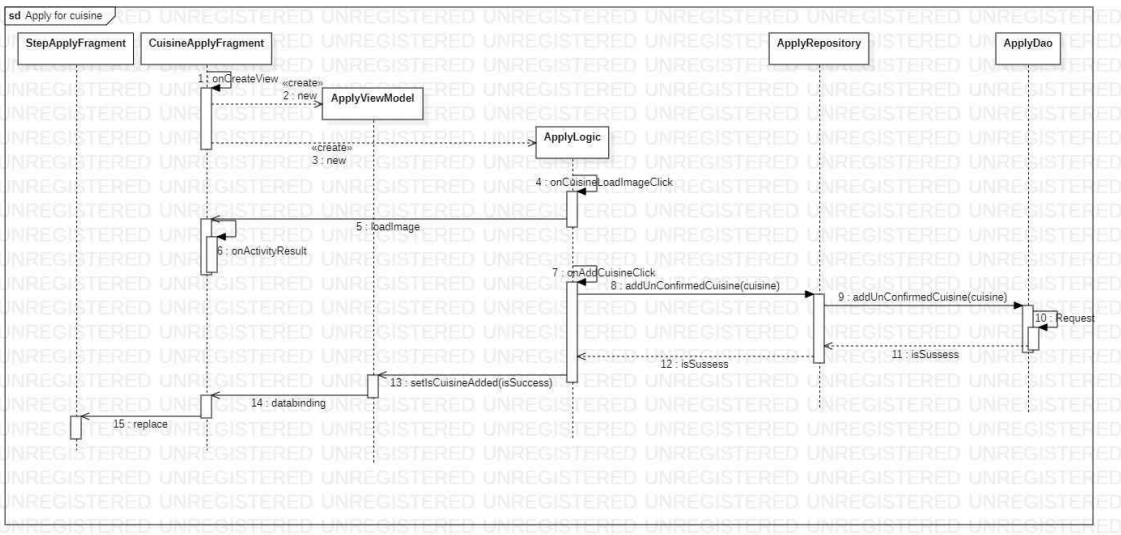
조리법 보기 버튼을 누르면 일단 조리법을 단계별로 저장한 리스트를 가져온 다음 조리법 화면으로 넘어간다. 그 후 다음 조리법 보기 버튼을 눌러서 조리법이 끝날

때까지 단계별로 볼 수 있게 한다. 다음은 요리사 신청하기이다.



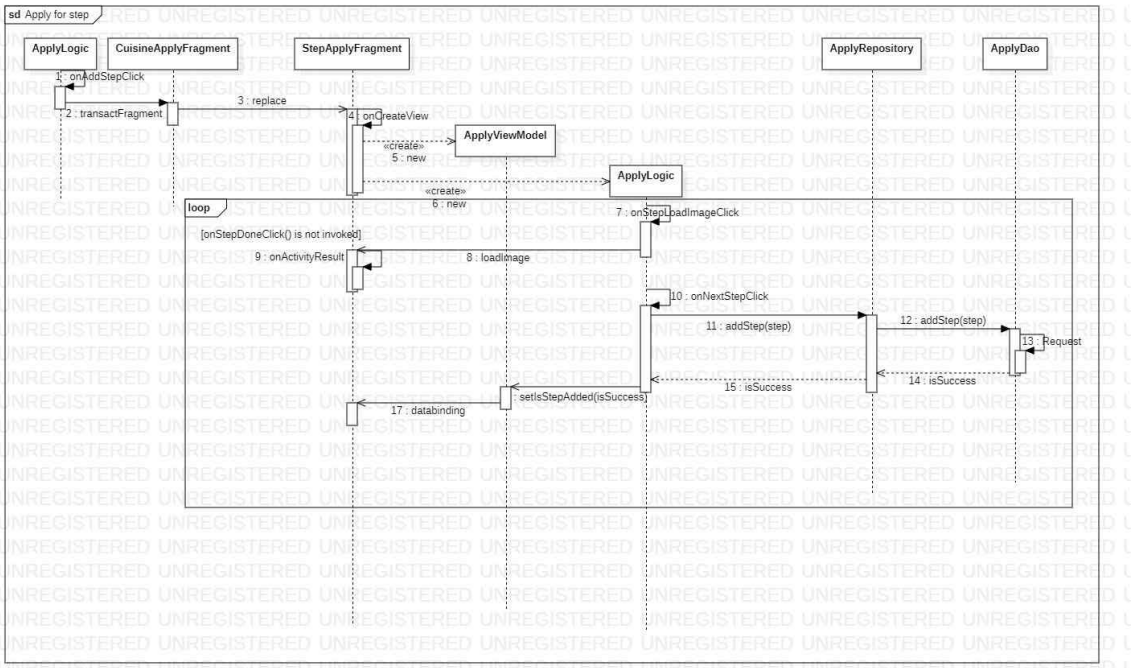

[그림 3-8] Apply to be a cook sequence diagram

내부 저장소에서 사진을 가져와야 하는 경우엔 callback을 이용해 fragment에서 작업을 처리한다. 그 외의 흐름은 다른 diagram과 큰 차이가 없다. 다음은 음식 등록 신청이다.



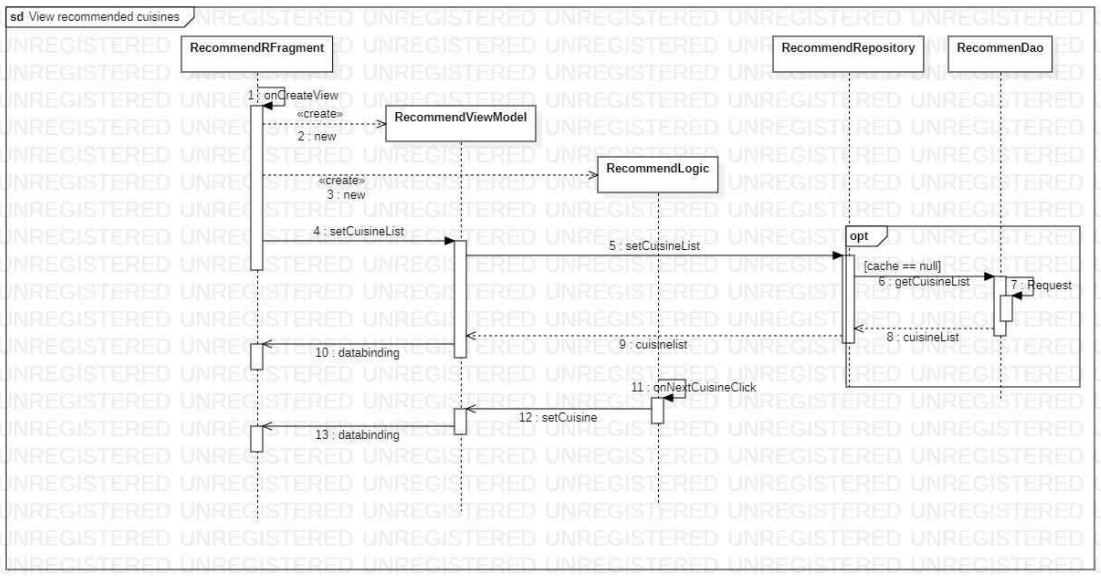

[그림 3-9] Apply for cuisine sequence diagram

요리사 신청과 거의 흡사하다. 다음은 조리법 단계 신청이다.



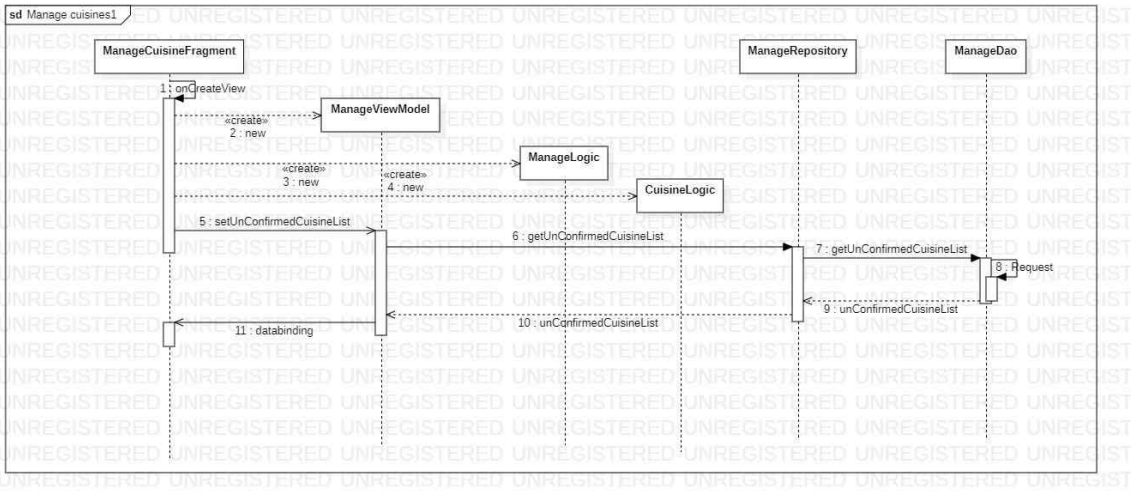

[그림 3-10] Apply for step sequence diagram

요리 신청과 흡사하지만 단계별로 등록하기 때문에 등록 완료 버튼을 누르기 전까지 계속 반복한다. 다음은 음식 추천이다.

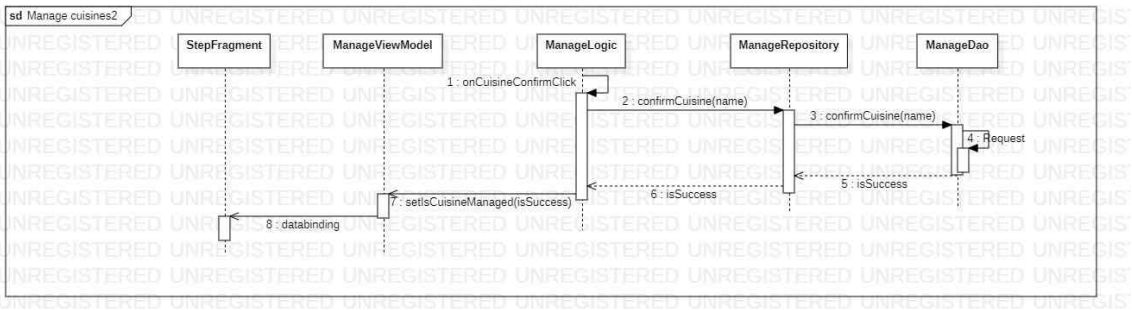



[그림 3-11] View recommended cuisines sequence diagram

Diagram에 묘사하지 않았지만 onViewCuisineClick이 호출되면 [그림 3-6]으로 넘어간다. 다음은 요리 신청 관리이다.

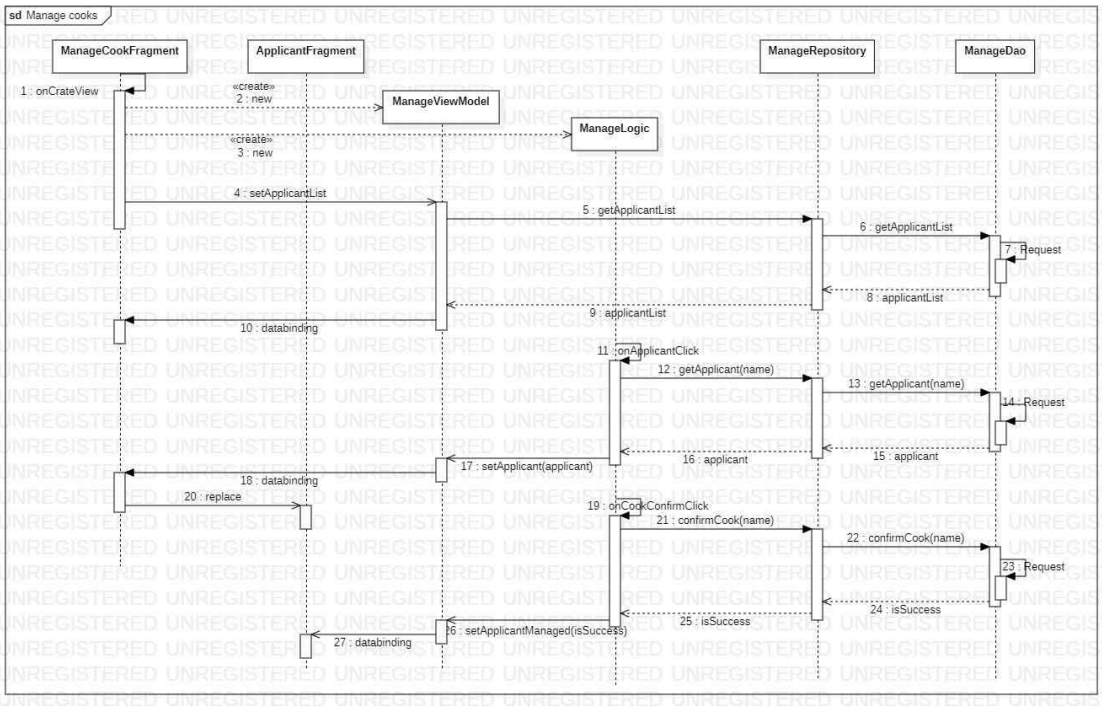



[그림 3-12] Manage cuisine sequence diagram 1



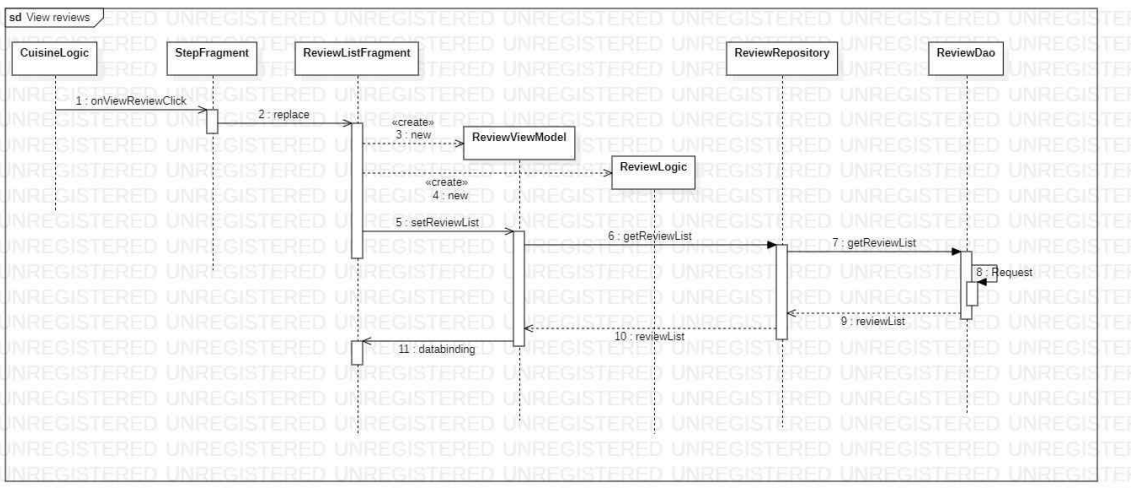

[그림 3-13] Manage cuisine sequence diagram 2

[그림 3-12]에서 허가되지 않은 음식 리스트를 가져와 화면에 띄운다. [그림 3-12]와 [그림 3-13]사이에 [그림 3-5], [그림 3-6]을 수행한다. Reference로 표현하고 싶었지만 툴 사용이 미숙해 이런 방식을 택했다. [그림 3-13]은 음식 추가를 허락했을 때의 흐름이다. 거절도 똑같은 흐름으로 진행된다. 다음은 요리사 신청 관리이다.



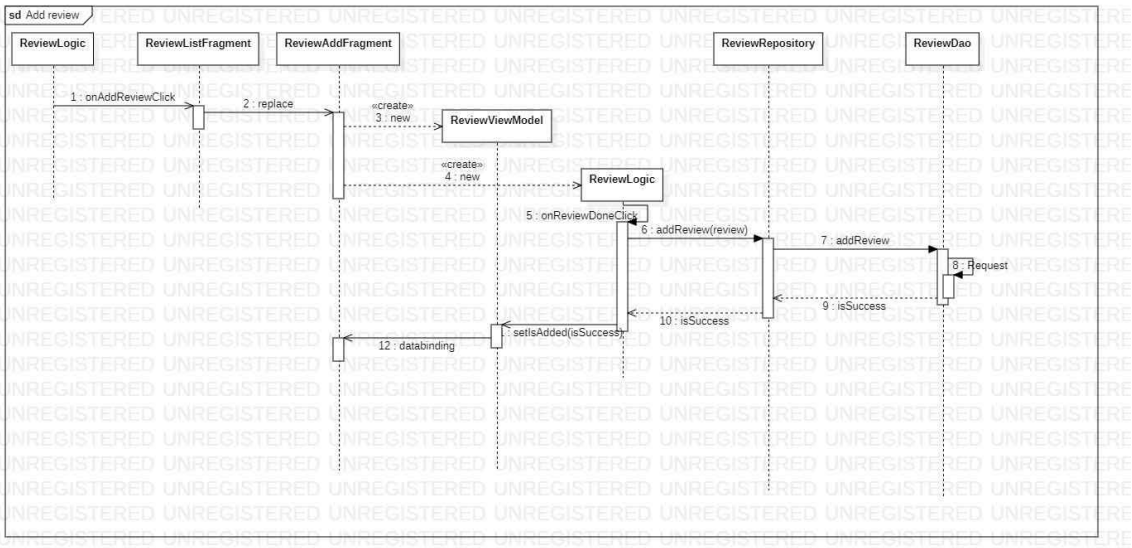

[그림 3-14] Manage cooks sequence diagram

먼저 지원자 리스트를 띄운다. 지원자를 클릭하면 지원자 정보가 뜨는 Applicant Fragment로 화면이 전환된다. 허가와 거절은 전과 같다. 다음은 리뷰 보기이다.




[그림 3-15] View reviews sequence diagram

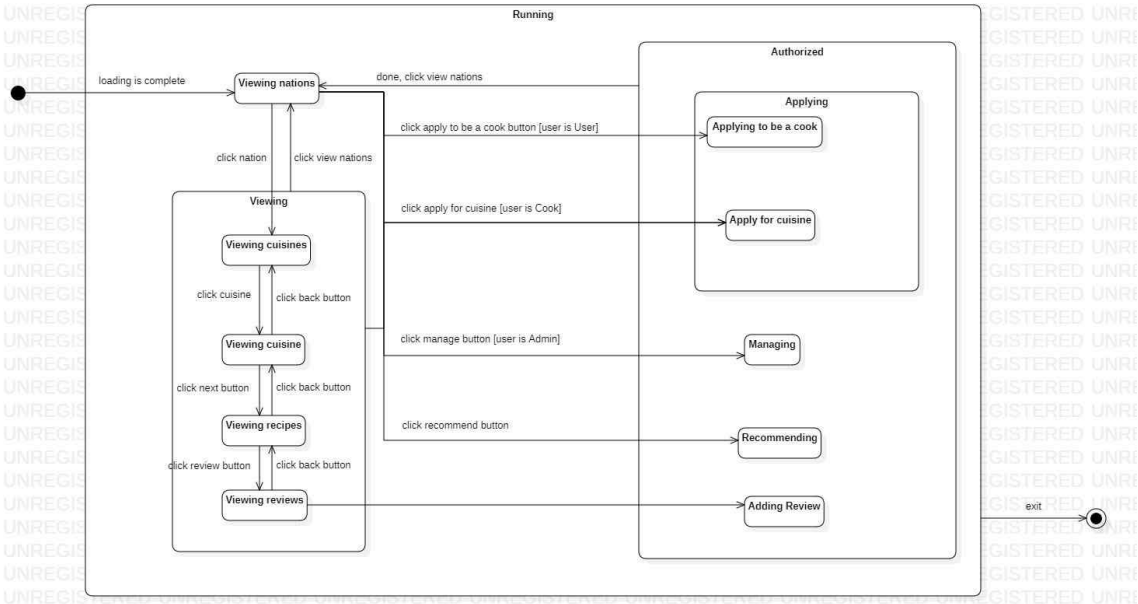
리뷰를 가져와서 보여준다. 다음은 리뷰 추가이다.




[그림 3-16] Add review sequence diagram

# 4. State machine diagram

이 장은 시스템 전체를 하나의 객체로 보고 그린 state machine diagram에 대한 장이다. State는 객체의 행동에 대한 축적된 결과라고 한다.[4] Diagram을 그리기 전에 이 시스템의 상태는 뭐로 정의할 수 있을까 생각해봤다. 그 결과 이러한 사용자와의 상호작용이 중요한 소프트웨어에서 행동에 대한 결과란 보이는 화면이라고 생각했다. 따라서 시스템에서 어떤 화면을 띄우고 전환 관계가 어떻게 되는지에 중점을 맞춰서 diagram을 그렸다. [그림 4-1]과 같다.




[그림 4-1] State machine diagram

프로그램을 시작한 후 제일 먼저 뜨는 화면은 나라 리스트 화면이다. 그 후, 메뉴를 통해 apply, manage, recommend로 갈 수 있다. 나라를 클릭하면 음식 리스트가 나온다.

# 5. Implementation requirements

두 가지 환경에서 테스트를 진행했다.

1. 갤럭시 S10+ 안드로이드 11
2. Android studio virtual device (Pixel 2 API 28)

안드로이드에서 구동가능하며 최소 SDK버전은 23이다.

# 6. Glossary

| 용어 | 설명 |
|---|---|
| 전 세계의 음식들 | 이 프로젝트로 만들어지는 앱의 이름. |
| 조리법 | 조리를 하는 방법 또는 기술이다. 페이지 별로 한 단계씩 보여준다. |
| 관리자 / An administrator | 앱을 관리하는 사람. |
| 요리사 / A Cook | 일반 이용자 중에 신청과 검증을 통해 요리사가 된 이용자. |
| 이용자 / A user | 일반 사용자. |
| DB 서버 | 회원 정보, 별점과 리뷰 등을 저장하기 위한 서버. |

# 7. References


1) Object Management Group. OMG® Unified Modeling Language® (OMG UML®). An OMG® Unified Modeling Language® Publication, n.d..
2) "ViewModel 개요," android developers, n.d. 수정, https://developer.android.com/topic/libraries/architecture/viewmodel?hl=ko.
3) “앱 아키텍처 가이드”, n.d. 수정, 2021년 05월 9일 접속, https://developer.android.com/jetpack/guide
4) Grady Booch 외, Object-oriented analysis and design with applications Third edition (n.p.: addison-wesley, n.d.), 600.