프로젝트 최종 보고서

[Domestic Grocery Manager]

201615007 컴퓨터공학과 문기태

201712845 컴퓨터공학과 류한길

201011359 컴퓨터공학과 임종화

Chart.

1. 프로젝트 소개
2. 사용한 오픈소스 소프트웨어 소개
3. 설계 및 개발 문서
4. 실행 및 사용방법
5. 개발 후 소감
6. 프로젝트 소개
7. 프로젝트 개요

DGM[:Domestic Grocery Manager]는 집 안 식자재를 관리해주는 서비스이다. 집 안의 식자재의 정보를 데이터베이스화 시켜서 웹으로 관리하고 해당 식자재가 부족하거나 유통기한이 지나는 등 식품의 상태를 사용하기에 적절하지 않게 되면 자동적으로 표시해줘서 한 눈에 볼 수 있도록 제공한다. 결국 사람들의 집 안내의 다양한 식자재를 한 눈에 관리할 수 있도록 편의성과 정확성을 제공해주는 서비스라고 할 수 있다.

1. 프로젝트 동기

보통 집 안에는 수많은 식자재들이 있다. 가족 구성원의 숫자가 크면 클수록 더욱 많은 식자재가 있을 확률이 크다. 그러다 보니 사 놓고도 다른 식품들에 의해 가려져 있거나 유통기한이 지나고 상한 제품이 방치되어 있어도 잘 인식하지 못하는 경우가 발생하기도 한다. 또한 집안 구성원 사이의 정보 불균형으로 이미 있는 식자재를 다른 구성원이 또 사오는 등 낭비가 발생하기도 한다. 그래서 집 안의 식품을 등록만 하면 웹을 통해 식품의 유통기한 및 수량 등을 관리할 수 있게 하여 한 눈에 어떤 식품이 집에 있고 어떤 식품이 현재 필요한지 필요와 집안 식자재를 빠르고 정확하게 비교할 수 있게 하자는 의도로 기획하였다.

1. 프로젝트 의도

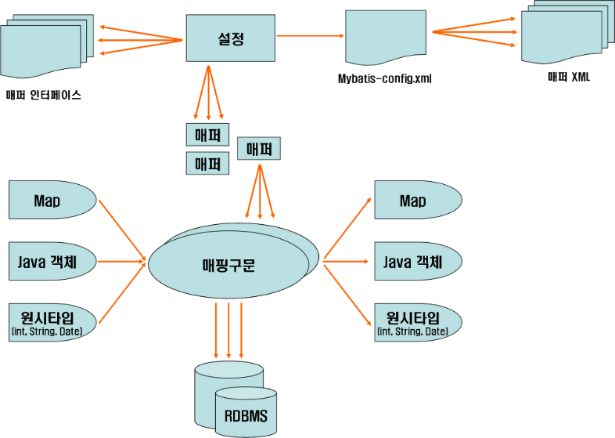
이번 학기에 수강하는 Open Source SW Project에서 사용하는 오픈소스인 Spring Boot를 이용해서 식품관리 웹 서비스를 개발하였다. 식품 정보를 데이터화하여 DataBase에 저장할 때 mySQL을 이용하고, 이를 MyBatis를 이용해서 Spring Boot에 연동하여 식품 데이터와 웹 서비스를 연계해 필요한 서비스를 제공하였다. Spring Boot를 이용하여 이번 학기에 배운 것을 다시 한 번 다지고, 미처 배우지 못한 부분을 탐색해보면서 더욱 학습해보고자 하였다.

1. 사용한 오픈소스 소프트웨어 소개
2. Spring Framework (Spring Boot)

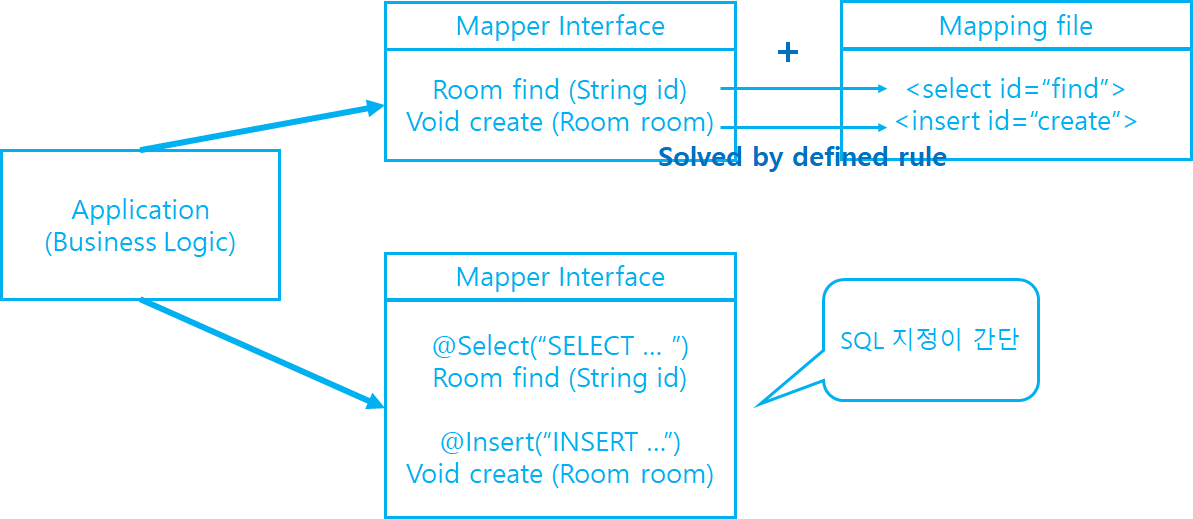
스프링 프레임워크는 Java Platform을 위한 오픈소스 어플리케이션 프레임워크이다. IoC를 통한 loosely Combination을 도모하고 AOP프로그래밍을 제공하여 풍부한 지원을 한다. 특별히 이번 프로젝트에서는 spring을 쉽게 실행할 수 있는 독립실행형 프로그램인 Spring boot를 사용하였다. 가장 큰 특징은 독창적인 종속성을 제공해 빌드 구성을 간소화할 수 있다는 것이다. 또한 자동적으로 Spring과 타 라이브러리를 구성하는 편리성이 있다. Spring에서의 다양한 복잡한 설정들이 한 번에 처리되어 Spring Boot에서는 애플리케이션 설정을 쉽게 할 수 있고, 간단한 기본 설정만으로 애플리케이션을 쉽게 구동할 수 있다. 본 프로젝트에서는 특히 스프링에서 지원하는 MVC패턴을 이용했는데, 간단한 Dependency와 Annotation 방식으로 프로젝트를 진행할 수 있었다.

1. MyBatis

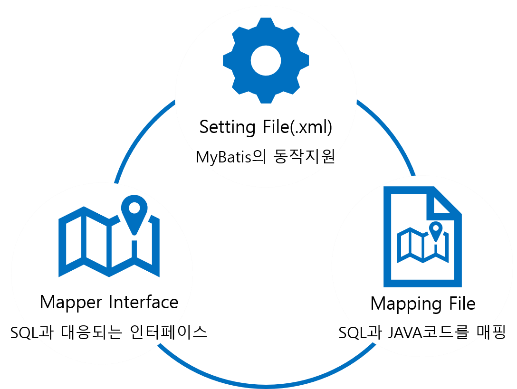
MyBatis는 관계형 데이터베이스를 쉽게 사용할 수 있도록 제공하는 오픈소스 개발 프레임워크이다. 개발자가 작성하는 SQL명령어, 즉 명령 쿼리를 자바 객체와 매핑하는 기능을 수행한다. JDBC의 반복되는 코드사용의 불편함을 쿼리문을 JAVA Code와 독립적으로 분리해서 XML파일로 관리함으로 커버하여 재사용성이 뛰어난다. 그리고 1~2줄의 명령어 코드로 DB를 관리할 수 있는 간결함도 있다.

첨부. MyBatis 구성

위 첨부된 사진은 MyBatis의 구성이다. DB를 Mapper파일로 관리를 하게 되는데 XML 설정파일을 통해 Mapper를 구성하고 해당 Mapper 구문에 필요한 SQL 명령어를 매핑하여 실제 해당 쿼리문을 수행하는 프로세스이다.

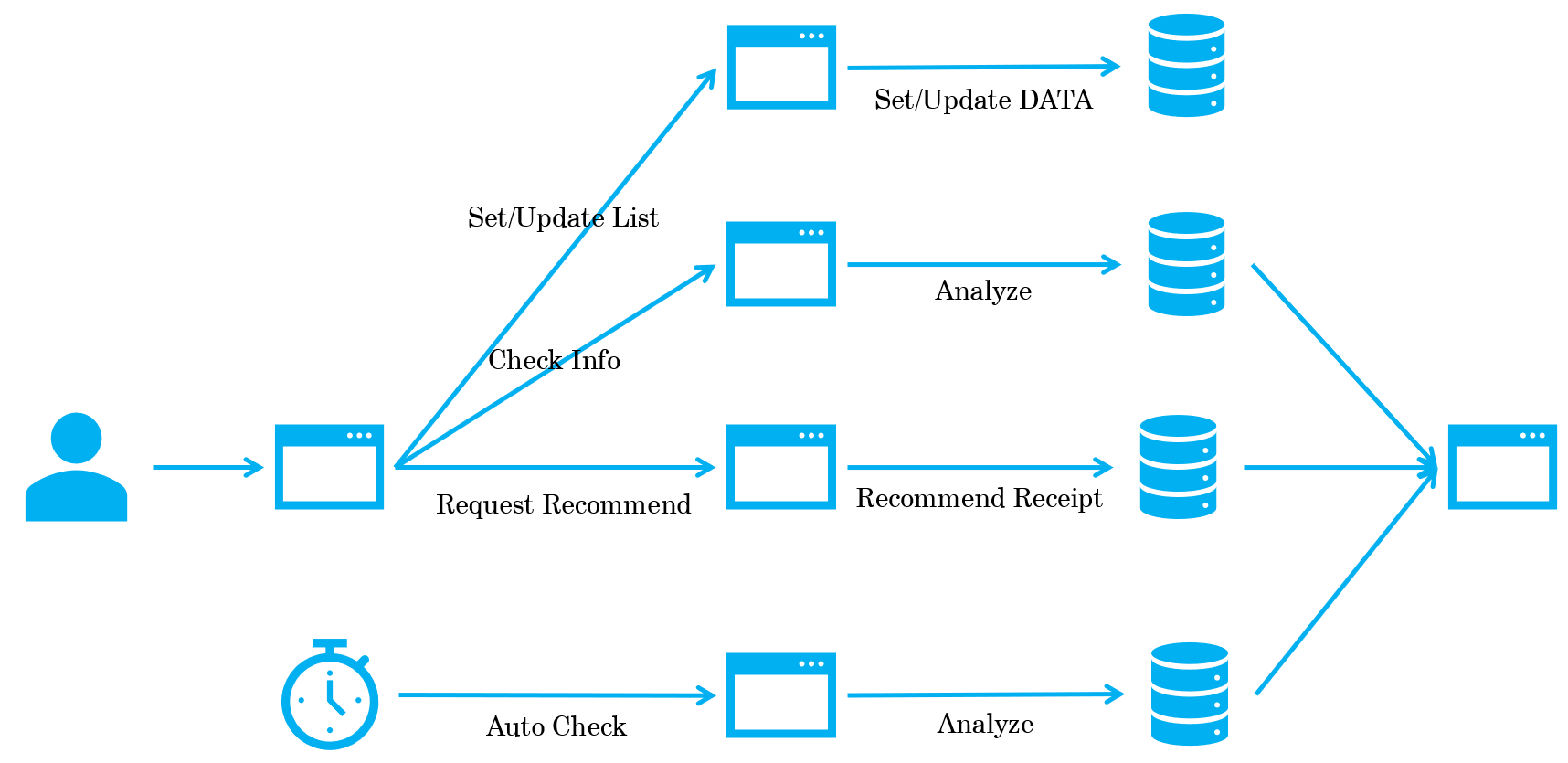
첨부. 매핑방법

Mapper 파일에 쿼리문을 매핑하는 방법은 크게 두 가지가 있다. 첫 번째는 Mapper Interface와 file을 구분해서 대응시켜주는 방법이 있다. 이 방법은 정의된 규칙에 의해 수행되고 MyBatis의 모든 기능을 사용할 수 있다. 두 번째 방법은 Mapper의 file을 따로 구성하지 않고 Interface자체적으로 Annotation을 추가하여 매핑하는 방법이다. 이 방법은 한 파일로 선언하기 때문에 직관적이고 편리한 장점이 있지만 MyBatis의 기능사용에 제한적이다. 그러나 이번 프로젝트에 사용할 DB활용에는 크게 고급기능이 필요하지 않기 때문에 직관적이고 편리한 Annotation 추가방식으로 매핑하기로 하였다.

첨부. MyBatis 구성파일

Mybatis를 사용하기 위해 필요한 파일형식은 크게 세 개가 있다. 프레임워크의 동작을 지원하는 xml파일, 실제 SQL명령을 JAVA코드와 매핑하는 매핑 파일, 이 매핑 파일에 대응되는 인터페이스인 Mapper 인터페이스가 있다.

1. 설계 및 개발 문서
2. 설계 디자인



웹 서비스에 필요한 기능은 총 네 가지이다. 사용자의 식품을 등록하고 수정하는 식품 관리기능, 각 식품마다 구체적인 정보를 조회할 수 있는 정보확인기능, 매일 날짜와 식품수량을 체크하여 유통기한이 지나거나 수량이 소진된 식품에 대해 알려주는 Auto알림기능, 데이터에 있는 식품을 가지고 만들 수 있는 레시피 추천 기능으로 구성되었다.

식품 관리 기능은 데이터베이스에 들어있는 해당 사용자의 식품 테이블에 추가할 수 있고 또 특정 식품 데이터를 수정할 수 있도록 설계했다.   
그리고 식품 테이블에서 원하는 특정 데이터에 대한 정보를 추출하는 기능을 통해 정보확인 기능을 설계하였다.   
또한 매일 날짜와 식품의 유통기한 날짜를 비교하고 또 수량을 체크하여 더 이상 사용할 수 없는 식품에 대해 알려주는 기능을 설계하였다.  
마지막으로 레시피와 레시피 재료가 담겨있는 DB와 사용자가 가진 식품의 DB를 대응하여 만들 수 있는 레시피를 추천하는 기능도 설계했다.

* 1. 유스케이스

|  |  |
| --- | --- |
| Use case | 1. login |
| Actor | User |
| Purpose | User가 시스템 사용을 위해 로그인한다. |
| Overview | 프로그램에 접근하기 전 DB에 저장된 아이디와 패스워드로 인증해 접근한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 사용자가 해당 시스템에 접근하고자 한다.  2.(S) 자동으로 로그인 화면을 띄운다.  3.(A) 사용자가 아이디와 패스워드를 입력하고 로그인 버튼을 누른다.  4.(S) 스프링 시큐리티를 통해 DB에 저장된 내용을 확인하고 인증을 처리한다.  5.(S) 성공 시 프로그램의 메인 화면을 띄우고, 실패 시 같은 로그인 화면을 다시 띄우되 에러 메시지를 출력한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 2. logout |
| Actor | User |
| Purpose | User가 시스템 사용을 종료하기 위해 로그아웃한다. |
| Overview | User가 시스템 사용을 종료하기 위해 로그아웃한다. |
| Pre-requisites | main화면이 띄워진 상태여야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 메인 화면의 로그아웃 버튼을 누른다.  2. (S) 스프링 시큐리티가 로그아웃을 처리하고, 로그인 화면을 띄운다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 3. checkfood |
| Actor | User |
| Purpose | User가 자신이 가진 식품 목록 현황을 확인한다. |
| Overview | User가 자신이 가진 식품 목록 현황을 확인한다. |
| Pre-requisites | main화면이 띄워져 있어야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 메인 화면의 ‘식품 목록 현황’ 버튼을 누른다.  2. (S) 새로운 화면으로 이동해 DB에서 유저가 가진 음식 정보를 받아와 출력한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 4. updatefood |
| Actor | User |
| Purpose | User가 자신이 가진 식품 목록을 갱신한다. |
| Overview | User가 자신이 가진 식품 목록에서 정보를 갱신한다. |
| Pre-requisites | main화면이 띄워져 있어야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 메인 화면의 ‘목록 수정’ 버튼을 누른다.  2. (S) 새로운 화면으로 이동해 DB에서 유저가 가진 음식 정보를 받아와 출력하고, 식품 추가를 위한 정보 입력 란을 출력한다.  3. (A) 출력된 자신이 가진 음식 정보를 수정하고 EDIT버튼을 누른다.  3. (S) 해당 정보로 DB를 갱신한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

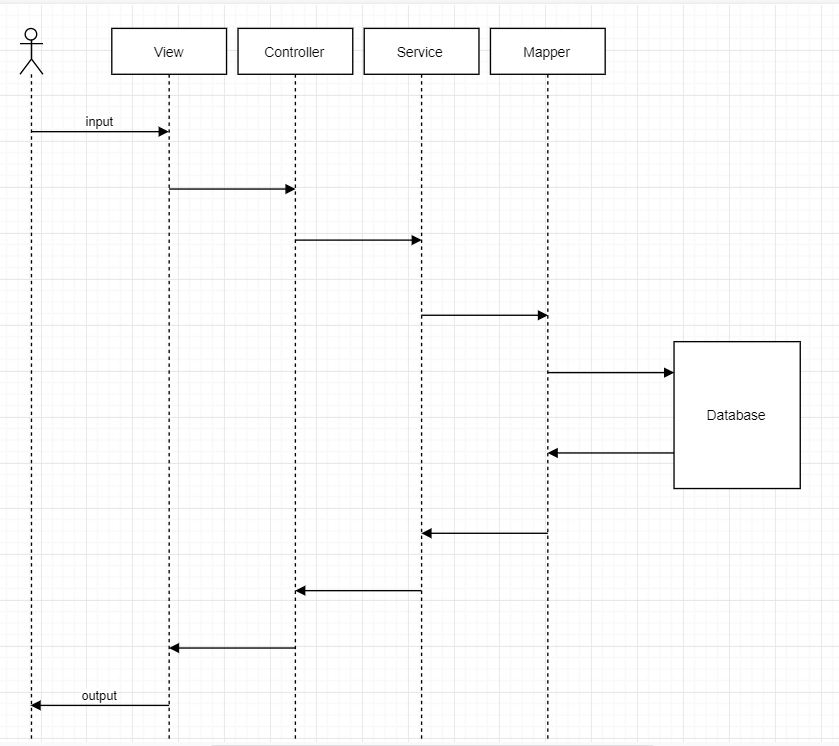
|  |  |
| --- | --- |
| Use case | 5. addfood |
| Actor | User |
| Purpose | User가 자신이 가진 식품 목록을 갱신한다. |
| Overview | User가 식품 목록에 새로운 식품을 추가한다. |
| Pre-requisites | main화면이 띄워져 있어야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 메인 화면의 ‘목록 수정’ 버튼을 누른다.  2. (S) 새로운 화면으로 이동해 DB에서 유저가 가진 음식 정보를 받아와 출력하고, 식품 추가를 위한 정보 입력 란을 출력한다.  3. (A) 식품추가를 위해 정보를 입력하고 목록추가 버튼을 누른다.  4. (S) 정보를 데이터베이스 Food 테이블에 추가한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 6. deletefood |
| Actor | User |
| Purpose | User가 자신이 가진 식품 목록을 갱신한다. |
| Overview | User가 자신이 가진 식품 목록에서 음식 정보를 삭제한다. |
| Pre-requisites | main화면이 띄워져 있어야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 메인 화면의 ‘목록 수정’ 버튼을 누른다.  2. (S) 새로운 화면으로 이동해 DB에서 유저가 가진 음식 정보를 받아와 출력하고, 식품 추가를 위한 정보 입력 란을 출력한다.  3. (A) 출력된 음식 리스트에서 삭제를 원하는 음식에 대해 DEL버튼을 누른다.  4.(S) 데이터베이스에서 해당 정보를 삭제한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 7. AutoCheck |
| Actor | User |
| Purpose | User에게 유통기한이 다 된 식품을 알려준다. |
| Overview | User가 가지고 있는 음식 중 유통기한이 다 된 식품은 main화면에 띄워지는 쇼핑리스트에 자동으로 리스트업해 알려준다. |
| Pre-requisites | N/A |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 사용자가 main화면으로 이동한다.  2.(S) DB로부터 사용자가 가진 음식 정보를 모두 받아오고, 그 중 유통기한이 지난 음식들에 대해 shoppingList로 출력한다. |
| Alternative Courses of Events | 1-1. 사용자가 main화면으로 이동해오는 것이 아니라 main화면을 계속 띄워둔다면 24시간마다 페이지를 갱신해준다. |
| Exceptional Courses of Events | N/A |

|  |  |
| --- | --- |
| Use case | 8. recommendrecipe |
| Actor | User |
| Purpose | User가 가진 음식 현황과 비교해 제작 가능한 레시피를 추천해준다. |
| Overview | User가 가지고 있는 음식과 비교해 시스템이 가지고 있는 레시피의 재료 중 70퍼센트 이상을 유저가 가지고 있다면 해당 레시피를 추천해준다. |
| Pre-requisites | main화면이 띄워져 있어야 한다. |
| Typical Courses of Events | (A): Actor / (S) : System  1. (A) 사용자가 main화면에서 ‘레시피 추천’ 버튼을 누른다.  2.(S) DB에서 레시피 정보와 유저가 가진 음식 정보를 가져와 비교해 레시피 재료 중 유저가 가진 것이 70%가 넘으면 레시피를 출력한다. |
| Alternative Courses of Events | N/A |
| Exceptional Courses of Events | N/A |

* 1. Flow Diagram



각각에 책임을 나눔으로써 재사용성과 변경가능성을 확보하였다.

View/Controller는 app계층으로, Model/Service/Mapper는 domain계층으로

분리해 생각하여 구현하였다.

* 1. DB

DB는 세 개의 테이블, food, user, recipe로 구성되어 있다.

food는 식별아이디(foodId) / 소유자아이디(userName) / 음식이름(foodName) / 카테고리(category) / 수량(num) / 유통기한(expirationDate)으로 구성되어 있다. 각 음식이 가져야 하는 정보 외에 소유자 아이디는 그 음식을 소유한 사용자를 식별하는 아이디를 나타내며, 카테고리는 레시피에서 해당 음식을 재료로 인식할 수 있도록 추가한 정보이다.

음식이름을 사용자가 자유롭게 정할 수 있게 해 융통성을 주되, 레시피의 재료와 비교하는 과정에서 문제가 생길 수 있어 레시피의 재료를 옵션으로 선택해 정하는 카테고리 항목을 두어 레시피와의 연결고리를 두었다.

User는 식별아이디(userName)/비밀번호(password)/roleName 로 이루어져 있다. 본 프로젝트에서 로그인은 간단하게 데이터베이스에 해당 아이디와 비밀번호에 해당하는 정보가 있느냐에 대해서만 인증을 처리하기 때문에 간단히 구성하였다. roleName은 후에 ADMIN을 추가할 경우를 고려해 넣었던 것이다.(현재 프로젝트는 ADMIN에 대한 기능은 없다.)

recipe는 식별아이디(recipeName)/재료(neededFood)로 이루어져 있다. 괜찮다면 조리과정 등의 내용도 넣고 싶었지만, 시간이 부족해 간단하게 구성해서 구현하였다.

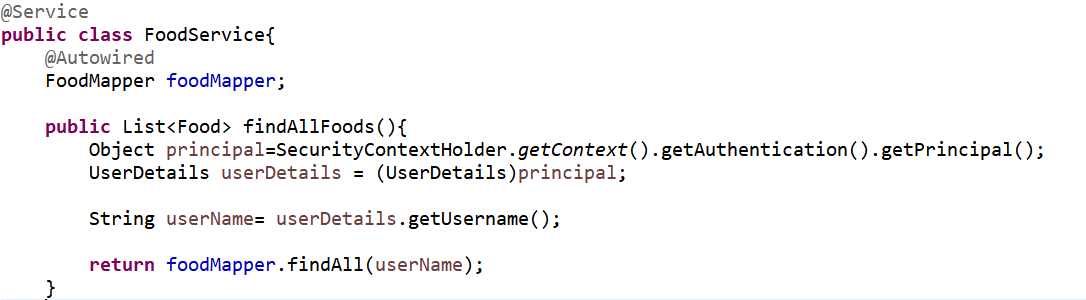
* 1. 기능구현 내용

각 기능 별로 자세한 구현을 서술하였다.

ㄱ) 로그인/로그아웃

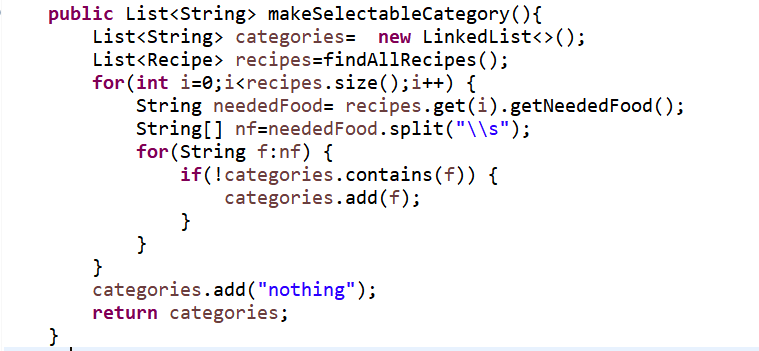
ㄴ) 식품 목록 조회

Service에 모든 식품 목록을 가져오는 findAllFoods함수를 구현하였다. 이 때 현재 로그인한 유저의 정보에 따라 해당 유저의 음식만을 가지고 와야 하기 때문에 전역으로 정의된 SecurityContextHolder를 이용해 User정보를 가지고 Mapper에 넘겨 DB에 접근해 데이터를 가져오도록 했다.

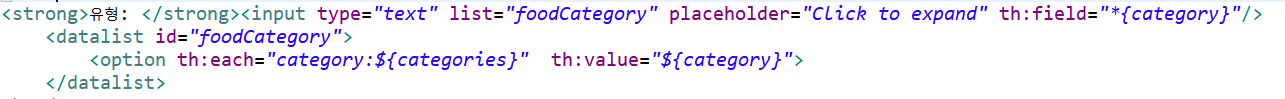


ㄷ) 식품 추가

(recipeService)



(updateFood.html)



레시피의 재료를 이용해 카테고리의 옵션을 만드는 함수이다. 이를 통해 레시피와 각 사용자가 설정하는 음식 간의 연결고리를 만들어 추후 레시피 추천 기능도 원활하게 구현할 수 있도록 하였다. 또한, 해당 사항이 없는 것에 대해서는 nothing을 선택할 수 있게 하기 위해 마지막에 nothing을 추가하여 옵션을 구성하도록 하였다.

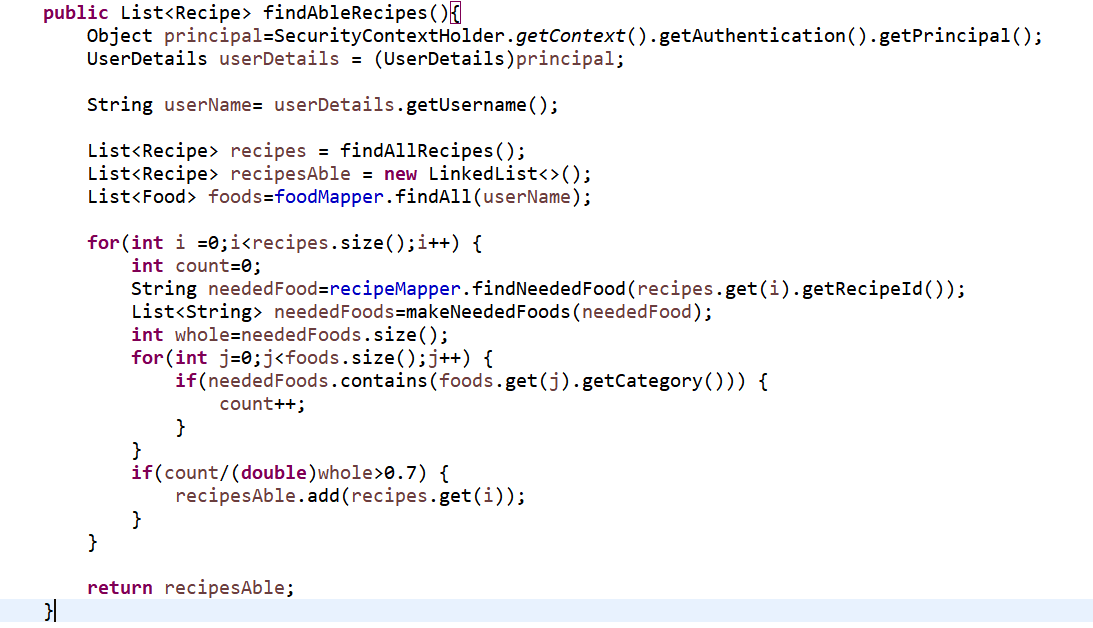
ㄹ) 식품 갱신

사용자의 식품 리스트에 대해 각 항목을 수정하고 EDIT 버튼을 누르면 위의 그림에 나타난 흐름을 따라 데이터베이스를 UPDATE해준다.

ㅁ) 식품 삭제

위와 마찬가지로, 사용자의 식품 리스트에 대해 DEL 버튼을 누르면 위의 그림에 나타난 흐름을 따라 데이터베이스의 항목을 DELETE해준다.

ㅂ) 레시피 추천



역시 현재 로그인 중인 사용자의 정보를 SecurityContextHolder를 이용해 가져와 음식 정보를 가져오고, 모든 레시피 각각의 재료와 비교해 레시피 재료의 70퍼센트 이상을 가지고 있는지 확인하고 이에 대한 리스트를 반환한다. makeNeededFoods는 recipeService에 있는 함수로, recipe테이블의 neededFood항목을 우리가 사용할 수 있는 foodName(String)의 리스트로 가공해주는 함수이다.

1. 개발 디자인

개발에 필요한 프로그램은 세 가지이다.

우선 웹에 대한 전체적인 구현과 MVC모델의 구현을 Spring Boot를 통해 개발하였다. 웹 서비스를 구현하고 View와 DB를 연결하는 연결자의 역할도 수행하도록 도와준다.

두 번째로 사용자와 음식, 레시피의 실제의 데이터를 담아두고 사용할 수 있도록 구현하기 위해 MyBatis를 사용하였다.

마지막으로 View를 Html을 통해 직관적인 UI로 서비스를 표현하였다.

1. 실행 및 사용방법

MySql에 create database testdb;

create USER 'foodmanager'@'localhost' IDENTIFIED BY '1234';

GRANT ALL PRIVILEGES on testdb.\* TO 'foodmanager'@'localhost';

FLUSH PRIVILEGES;

를 입력해 데이터 베이스를 만들고 실행파일을 실행시킨다.

1. 개발 후 소감

문기태: 프로젝트를 진행하면서 먼저 계획했던 부분과 필요했던 기능들을 잘 구현했던 것과 필요한 오픈소스를 적절히 잘 사용했던 것이 좋았다. 오픈소스를 사용하는 것을 처음엔 어렵고 부담스럽게 생각했는데 오히려 오픈소스를 사용하면서 개발 속도가 빨라진 것을 느낄 수 있었고 흥미로웠다. 다만 오픈소스들을 연동하고 잇는 부분에서 시간이 오래 걸렸고 어려움을 많이 느꼈다. 그리고 처음 프로젝트를 설계하는 부분을 더 중점을 두고 하면 좋았을 것이라는 아쉬움이 들었다. 물론 필요한 Use-case와 Sequence-Diagram과 같이 유닛 단위의 설계는 했지만 각 객체들이 어떻게 작용하고 필요한 함수들이 어떤 객체에서 매개변수를 가져오는지에 대해 구체적으로 설계해 놓지 않아 실제 개발 때 팀원 간의 의견차이가 발생했고 이를 하나로 모으는데 시간이 꽤 걸렸다. 앞으론 개발 전의 설계하는 과정에도 시간에 투자를 해야겠다. 웹 서비스 자체는 생각한대로 직관적으로 구현이 돼서 좋았지만 개발시간이 촉박하다 보니 몇 가지 유닛 테스트 예외 상황에 대해서는 해결하지 못한 부분이 아쉬웠다. 그래서 좀 더 시간이 있었다면 남은 테스트 오류에 대한 부분도 해결할 수 있었을 것이라는 생각이 들었다. 그래도 프로젝트 아이디어 도출부터 개발까지 팀원 간의 의견교류와 개발방향 설정하는 전반적인 부분을 통해 배울 점이 많았다.

임종화: 오픈소스 프로젝트 수업 시간 및 실습 시간에 배운 Spring Boot를 실제로 적용해보면서 더 잘 이해할 수 있었다. 팀과 함께 데이터베이스와 연동하는 과정까지 진행하면서 처음에는 어려웠던 Spring Boot를 이제는 어느 정도 잘 사용할 수 있을 것 같다는 생각이 든다. html로 view부분을 구현하면서 다양한 기능을 지원하는 것에 놀랐고 스스로 하나하나 구현해가면서 어떻게 그런 것들을 구현하고 사용할 수 있는지 알게 되었다. 다만 그렇다 보니 디자인적인 면에서는 직관적인 요소가 다소 부족한 부분이 있는 등 아쉬운 부분이 생겼다. 이에 대해서는 추후 사용자의 이해도를 높일 수 있도록 식품 정보와 함께 사진 이미지를 업로드할 수 있게 한다거나 하는 기능을 추가해보고 싶다.

류한길: 스프링 부트의 강력함을 체험할 수 있었다. 맡은 부분이 스프링 부트이기 때문에 html과의 연결이나 마이바티스를 통한 DB와의 연결 등을 생각하면서 그 부분들 또한 공부했어야 했는데, 각 팀원들의 도움을 받아 잘 이해할 수 있었다. 하나하나 기능을 개발해가면서 객체 지향이 얼마나 편리한 것인지도 느낄 수 있었는데, 각 책임을 다 나눠놓고 시작하니 기능을 추가할 때도 머리 아프지 않게 하나하나 해갈 수 있었고, 나중에 팀 내 회의로 변경사항이 생겼을 때에도 각각 빠르게 고칠 수 있었다. 이 장점을 배우고 앞으로 사용할 수 있게 되었다는 점이 이 프로젝트에서 얻은 큰 혜택인 것 같다. 아쉬운 점은 팀끼리의 소통이 조금 부족했었다는 점과 이 프로젝트에 쓸 시간이 별로 없었다는 점이다. 다른 프로젝트나 일들이 많아 시간을 별로 쓰지 못했고, 팀끼리의 소통에 있어서 서로 다른 것을 생각해 갈아엎고 다시 진행하는 부분도 있었어서 기능을 더 추가하고 싶었는데 결국 원래 기획했던 기본적인 기능밖에 구현을 못 했다는 점이 아쉽다. 가능하다면 좀 더 기능을 추가하고 싶다.