생활 환경에 따른 노인 기대 수명 및 삶의 만족도 프로젝트

프로젝트 수행 결과서

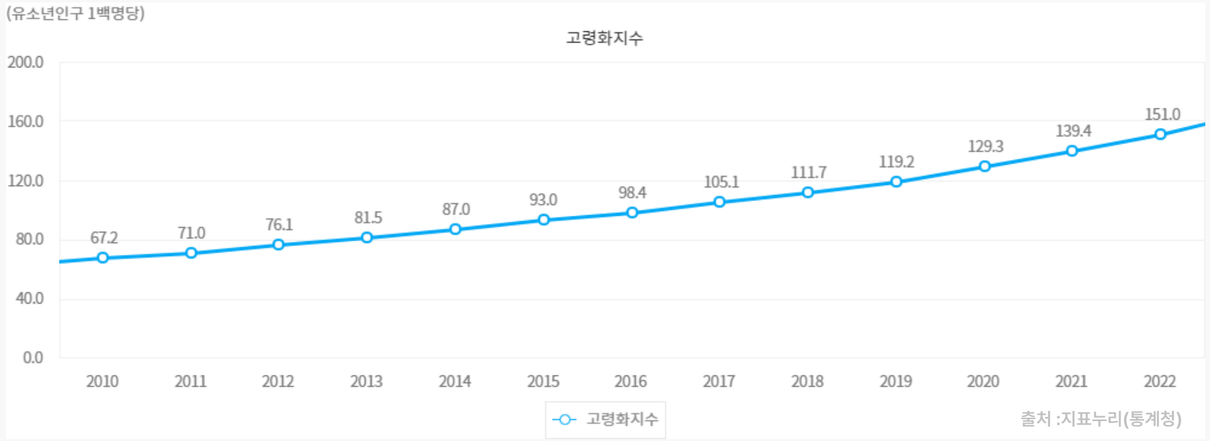
유정민

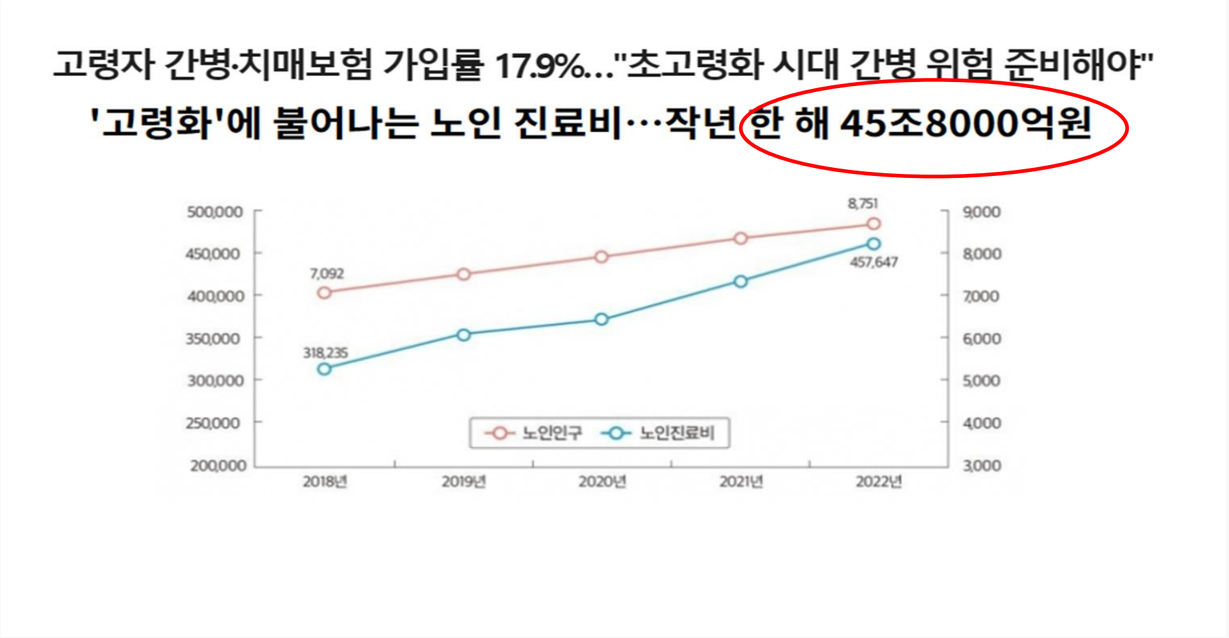
https://github.com/p1p2p1/Project_DLT

목차

기획 의도

프로젝트 기획서





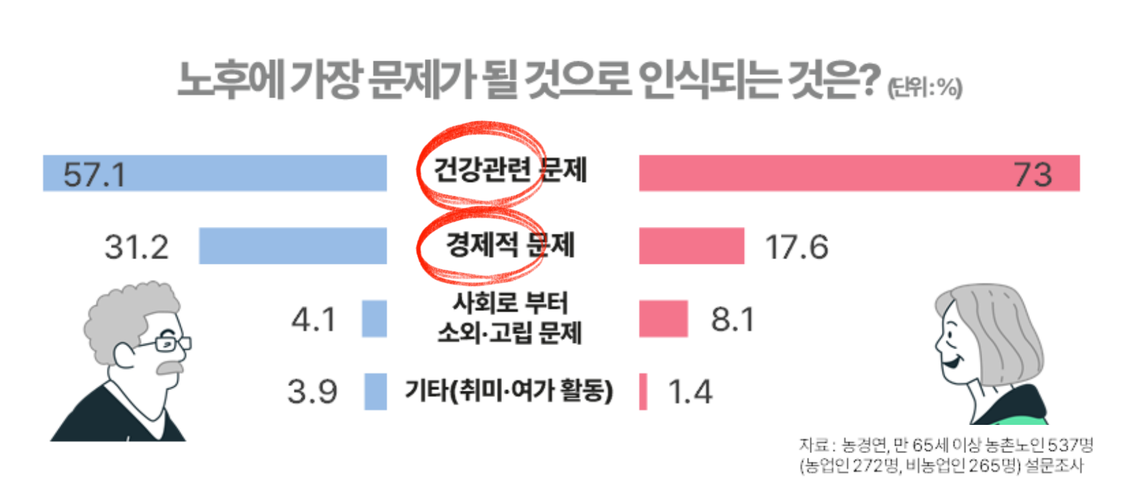
사회적 이슈로 계속해서 나타나는 **고령화 문제**

계속되는 고령화 문제로 인한 노년부양비가 계속해서 증가

실제로 노년 부양비 중 진료비는 상당한 비중을 차지

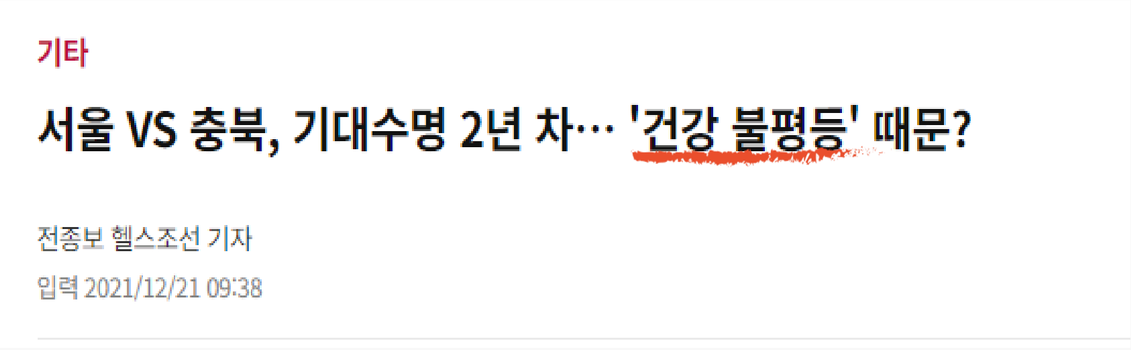
기획 의도

프로젝트 기획서



설문에서도 노후 걱정 중 건강, 경제적 문제를 가장 높게 인식

하지만 해당 정보의 접근성이 떨어진다.



기타

서울 VS 충북, 기대수명 2년 차… '건강 불평등' 때문?

전종보 헬스조선 기자

입력 2021/12/21 09:38

이번 프로젝트를 통해,
지역별로 노인층의 질병 정보와 정책 정보를 제공하고,
건강과 직결되는 기대 수명에 대한 정보를 제공

개발 목표

프로젝트 기획서

개인별 기대 여명 예측

연령대별(성별, 기대 여명, 흡연 여부, 음주 여부, 고혈압 여부, 당뇨 여부) 데이터로 기대 여명을 예측할 수 있는 다중 선형 회귀 모델 훈련

사용자의 설문 조사를 진행

회귀 분석을 통해 만든 회귀식으로

실제 사용자의 기대 여명 예측

지역별 기대 수명 예측

지역별로 4대 만성 질환(고지혈증, 치매, 당뇨, 고혈압), 스트레스 인지율과 음주율, 당뇨 비율, 사망률 데이터를 분석

지역별 기대 수명과 연관성이 높은 특성 파악

비교하고 싶은 2개 지역과 미래 연도 항목을 선택

지역별 기대 수명을 그래프로 시각화

'이미지: Freepik'. 이 페이지의 커버는 Freepik의 자료를 사용해 디자인되었습니다

기술 스택

사용 기술 목록

사용 언어

HTML5

CSS3

javascript
[1.5]

java
[11.0.9]

JSP
[2.3]

사용 환경

Oracle DB
[11.2.0.2.0]

Apache tomcat
[9.0.84]

Spring Framework
[4.3.9]

Spring Boot
[3.0.1]

개발 도구

visual Studio Code
[1.85.2]

R Studio
[4.3.2]

Eclipse
[2019.09]

git
[2.43.0]

프로젝트 사용 기술 경험

사용 기술 목록

Java

- 다양한 미니 프로젝트 경험
- 주소록 관리 프로그램
  - 연락처 CRUD
  - 자바 컬렉션 프레임워크 사용 경험
    (ArrayList, Set, Map, Stack)
- 다양한 자료구조 구현 경험
  - Single Linked List 구현
    (노드 추가, 삭제, 역순으로 뒤집기 구현)
  - Queue 구현
    (enqueue, dequeue 구현)
  - Stack 구현
    (push, pop 구현)
  - Tree/Binary Tree 구현
- 다양한 알고리즘 구현 경험
  - Binary Search 구현
  - Linear Search 구현
  - Greedy Search 구현

JSP, Servlet

- 개인별, 지역별 기대 수명 예측 서비스
- JSTL, ES6을 사용하여 데이터 처리 및 출력
- 세션을 사용한 로그인 세션 기능 구현
- 회원 관리 기능, 커뮤니티 게시판 기능 구현
- request, response를 사용하여 클라이언트의 요청 처리

HTML5, CSS3

- 폼태그를 사용하여 서버와의 원활한 데이터 교환
- 그리드 시스템을 사용하여 반응형 구현
  - BootStrap을 사용하여 깔끔한 디자인 구성

R

- 공공데이터포털을 활용한 데이터 분석
  - ggplot2를 사용한 시각화 가능
- 데이터 전처리
  - 연평균 데이터를 0과 1 사이 값으로 스케일링
- 개인별 기대 수명 모델, 지역별 기대 수명 모델 훈련
  - 다중 선형 회귀 분석을 이용한 모델 훈련
- 단계적 선택법을 사용하여 특성 설정
  - stepWise() 함수를 사용하여 상관관계가 낮은 특성 제거

JavaScript

- 클라이언트에서 서버에 데이터를 비동기적으로 요청
  (AJAX, Fetch API 사용)
- Chart.js를 이용한 그래프 출력
  (Chart.js 사용)
- 정규표현식을 이용하여 유효성 검사 진행

Oracle DB

- 무결성 제약조건과 관계
  - 기본키, 외래키 설정
- DB 백업
- ERD 작성
- 인덱스 개념 이해
  - B-tree 개념 이해
- 부속질의 개념 이해
  - 중첩 쿼리
  - 스칼라 부속쿼리
- 데이터베이스 정규화 개념 학습

프로젝트 사용 기술 경험

사용 기술 목록

Spring

- STS3를 사용하여 스프링 프레임워크 사용
  - 서블릿 모델2 -> 스프링 프로젝트로 변경
- Spring IoC 개념 이해
- Spring DI
  - XML 사용 경험 및 이해
  - Annotation 사용 경험 및 이해
- Spring AOP
  - AOP 사용 경험 및 이해
- Spring JDBC
  - JDBC 사용 경험 및 이해
- Spring MVC
  - 실제 프로젝트 아키텍처에 MVC 패턴 적용 경험
- MyBatis
  - 프로젝트 코드에서 Query문 분리
- Encryption(암호화)
  - DB Connection에 암호화 기법 적용

AWS EC2

- 클라우드 서버 사용 경험
  - 인바운드, 아웃바인드 트래픽 제어
- VM(Virtual Machine)을 활용하여 서버 관리
  - 서버 환경 구성
- puTTY를 사용한 인스턴스 접속

Linux

- 리눅스 기본 명령어 사용 경험
  - Ubuntu 사용
  - Shell을 사용한 명령어 사용

Git

- 프로젝트 버전 관리
  - Git의 전체적인 구조 이해 및 사용
- Git을 활용하여 팀원 간의 협업
- Git의 다양한 커맨드 이해 및 사용
  - git add, git commit –m, git push과 같은 다양한 명령어 사용 경험
- 다양한 브랜치 생성, 병합 및 관리 경험

개발 스케쥴표

분류	담당자	작업	시작일	종료일	작업기간
기획	공통	아이디어 구상	2023-11-15	2023-11-21	6일
		기획안 작성	2023-11-21	2023-11-22	2일
		화면설계서 작성	2023-11-22	2023-11-30	8일
		요구사항 명세서/분석서 작성	2023-11-30	2023-12-04	4일
1차 UI	유정민	메인 페이지 구현	2023-12-05	2023-12-08	4일
		기대 수명 페이지 구현	2023-12-08	2023-12-12	4일
		게시판 서비스 구현	2023-12-12	2023-12-15	4일
		UI 페이지 최종 점검	2023-12-16	2023-12-18	2일
빅데이터 분석	유정민	[통계청]지역별 질병, 기대 수명, 건강 상태, 환경 데이터 수집	2023-12-18	2023-12-19	2일
		[기대 수명 예측]수집 데이터 전처리	2023-12-20	2023-12-20	1일
		[기대 수명 예측]지역 건강, 환경적 요소 분석	2023-12-21	2023-12-21	1일
		[기대 수명 예측]지역별 기대 수명 예측	2023-12-22	2023-12-25	3일
		[기대 수명 예측] 개인별 기대 수명 예측	2023-12-25	2023-12-27	3일
		[기대 수명 예측]빅데이터 분석서 작성 및 검토	2023-12-28	2023-12-28	1일

개발 스케쥴표

분류	담당자	작업	시작일	종료일	작업기간
2차 웹 개발	유정민	프로젝트 설계	2024-01-15	2024-01-16	2일
		UML 작성	2024-01-17	2024-01-17	1일
		데이터베이스 설계 및 데이터 임포트	2024-01-18	2024-01-19	2일
		FrontController 구현	2024-01-19	2024-01-19	1일
		커뮤니티 서비스 - 유저관리	2023-01-19	2023-01-22	3일
		커뮤니티 서비스 - , 게시판 글 작성, 수정, 조회 기능 구현	2023-01-22	2023-01-23	2일
		기대 수명 서비스 - View 구현	2023-01-23	2023-01-24	2일
		기대 수명 서비스 - DAO, DTO 구현	2023-01-24	2023-01-25	2일
		기대 수명 서비스 - 개인 기대 수명 제공 기능 구현	2023-01-25	2023-01-29	4일
		기대 수명 서비스 - 지역별 기대 수명 제공 기능 구현	2023-01-30	2023-02-02	4일
		코드 리팩토링	2023-02-03	2023-02-05	2일
		단위 테스트 및 통합 테스트 수행	2023-02-05	2023-02-06	2일
		버그 수정 및 코드 리팩토링	2023-02-06	2023-02-08	3일

요구사항 정의서

목차

0. 공통 기능

▶ 메뉴의 클릭을 통해 다른 기능으로 이동할 수 있어야 한다.

▷건강·정책(질병 정보, 정책 정보 확인), 삶의 질 분석(기대 수명 예측, 당신의 만족도는?), 커뮤니티, 로그인 순으로 메뉴가 구성되어 있어야 한다.

▷질병 정보는 서울과 전남의 4대 질병 데이터를 그래프로 확인할 수 있어야 한다.

▷기대수명 예측은 개인별 기대 수명 정보를 예측해주고 지역별로 예측한 기대수명 데이터를 비교하여 그래프로 확인할 수 있어야 한다.

▷커뮤니티는 사용자들끼리의 정보와 지식을 공유할 수 있어야 한다.

▷메뉴는 웹 페이지 상단에 고정될 수 있어야 한다.

▶ 로고 이미지 클릭 시, 메인화면으로 이동할 수 있어야 한다.

1. 메인 기능

▶움직이는 화면(캐러셀)이 있어야 하고 각 화면을 이동할 수 있는 버튼이 있어야 한다.

▷캐러셀 화면은 총 3개의 화면으로 구성될 수 있어야 한다.

▷각 캐러셀 화면에는 주요 기능인 지역별 통계, 기대수명 예측, 정책을 설명할 수 있어야 한다.

▷각 캐러셀 화면에는 주요 기능인 지역별 통계, 기대수명 예측, 정책 기능으로 이동할 수 있는 버튼이 있어야 한다.

▶ 커뮤니티로 이동할 수 있는 기능이 있어야 한다.

▷커뮤니티에 대한 간단한 설명과 기능 소개가 있어야 한다.

▷커뮤니티의 사용자가 쓴 글을 5개만 구성 되어야 한다.

▶ 각 서비스에 맞는 이미지와 서비스로 이동할 수 있는 기능이 있어야 한다.

2. 건강·정책

▶ 맵 기능을 통해 지역을 선택할 수 있어야 한다.

▷지역을 선택할 경우 작은 웹 페이지가 만들어지고 데이터를 보여줘야 한다.

2.1. 지역별 질병 정보 제공 기능

▶만들어진 웹 페이지 해당 지역의 그래프가 해당 조건에 맞추어 데이터를 제공해야 한다.

▶연도를 설정하면 해당 연도에 맞게 데이터 그래프가 정보를 제공해야 한다.

▷시작 연도를 설정할 수 있어야 한다.

▷마지막 연도를 설정할 수 있어야 한다.

▶해당 지역의 시/군/구를 설정할 수 있어야 한다.

▶질병을 선택할 수 있는 버튼이 있어야 한다.

▷고지혈증 버튼을 이용하여 선택할 수 있도록 하고 데이터를 그래프에서 제공해야 한다.

▷치매 버튼을 이용하여 선택할 수 있도록 하고 데이터를 그래프에서 제공해야 한다.

▷당뇨 버튼을 이용하여 선택할 수 있도록 하고 데이터를 그래프에서 제공해야 한다.

▷고혈압 버튼을 이용하여 선택할 수 있도록 하고 데이터를 그래프에서 제공해야 한다.

▶차트 단위를 확인할 수 있는 버튼이 있어야 하고 단위에 대한 설명을 제공해야 한다.

▶각 질병들에 대한 설명이 있어야 한다.

▶왼쪽 아래 화살표 버튼을 통해 화면의 상단으로 이동할 수 있어야 한다.

2.2. 정책 정보 제공 기능

▶ 가장 먼저 보여지는 화면에는 해당 페이지에 대한 소개 글이 있어야 한다.

▶ 제공하는 정책을 보여주어야 한다.

요구사항 정의서

▷제공하는 정책은 어르신 일자리, 생활 안정 지원, 여가 복지 지원, 어르신 건강 증진으로 정책에 대한 간단한 설명이 있어야 한다.

▶ 맵 기능을 통해 지역을 선택할 수 있어야 한다.

▷ 지도에서 지역을 선택할 수 있도록 안내하는 문구가 있어야 한다.

▷ 지역을 선택할 경우 정책을 확인 할 수 있는 화면으로 이동 되어야 한다.

▶ 어르신 일자리, 생활 안정 지원, 여가 복지 지원, 어르신 건강 증진 정책은 지역별로 보여줄 수 있어야 한다.

▷ 어르신 일자리 정책은 서울시 정책과 전라남도 정책을 보여줄 수 있어야 한다.

▷ 생활 안정 지원 정책은 서울시 정책과 전라남도 정책을 보여줄 수 있어야 한다.

▷ 여가 복지 지원 정책은 서울시 정책과 전라남도 정책을 보여줄 수 있어야 한다.

▷ 어르신 건강 증진 정책은 서울시 정책과 전라남도 정책을 같이 보여줄 수 있어야 한다.

3. 기대 수명 예측 기능

▶사용자의 정보를 받아서 개인별 기대 수명 데이터를 예측하여 제공할 수 있도록 해야 한다.

▷개인별 기대 수명을 예측하고 해당 기대 수명 데이터에 대한 회귀식과 출처에 대한 정보를 제공해야 한다.

▶두 지역의 미래의 예측한 기대수명 데이터를 동시에 제공하여 비교할 수 있도록 그래프 제공해야 한다.

▶지역을 설정할 수 있어야 한다.

▷첫 번째 지역을 설정할 수 있어야 한다.

▷두 번째 지역을 설정할 수 있어야 한다.

▶연도를 설정하면 해당 연도에 맞게 데이터 그래프가 정보를 제공한다.

▷기준 연도를 설정할 수 있어야 한다.

▶과거 기대수명 데이터로 그래프를 제공해야 한다.

▶지역을 설정할 수 있어야 한다.

▶차트에 대한 부가 설명이 있어야 한다.

▷각 데이터에 대한 출처와 예측에 사용된 데이터에 대한 설명이 있어야 한다.

4. 만족도 제공 기능

▶ 구글 설문지를 통해 삶의 만족도 설문을 진행할 수 있어야 한다.

▷ 설문지 문항은 17개로 구성 되어야 한다.

▷ 설문지 문항에는 성별, 연령대, 경제적 상황 만족도, 사회적 관계 만족도, 건강 상태 만족도를 알 수 있는 문항으로 구성 되어야 한다.

▷ 경제적 상황 만족도, 사회적 관계 만족도, 건강 상태 만족도에 대한 문항은 각 5개로 구성 되 어야 한다.

▷ 설문이 완료되면 결과 보기 버튼을 통해 만족도를 조회 할 수 있어야 한다.

▶ 만족도 설문 결과는 점수로 환산하여 나타낼 수 있어야 한다.

▷ 삶의 만족도, 경제적 상황 만족도, 사회적 관계 만족도, 건강 상태 만족도는 점수로 나타낼 수 있어야 한다.

▷ 경제적 상황 만족도, 사회적 관계 만족도, 건강 상태 만족도 중 점수가 가장 높은 만족도와 가장 낮은 만족도를 나타낼 수 있어야 한다.

▷ 만족도 조회 후 평균 그래프 보기 버튼을 통해 평균 만족도 그래프를 조회 할 수 있어야 한 다.

▶ 평균 만족도 그래프는 설문자와 성별, 연령대가 동일한 사람들의 평균 점수와 설문자의 점수를 비교할 수 있게 같이 보여줄 수 있어야 한다.

▷ 평균 데이터를 그래프로 제공해야 한다.

▷ 설문자의 데이터를 그래프로 제공해야 한다.

▷ 평균보다 높은 설문자의 만족도와 낮은 설문자의 만족도를 글로 알려주어야 한다.

▷ 설문을 통해 낮게 나온 만족도에 대한 정책을 추천해 줄 수 있어야 한다.

▷ 정책 보기 버튼을 통해 추천 해준 정책을 확인할 수 있어야 한다.

5. 게시판 기능

▶게시판에 등록된 게시글 목록을 불러올 수 있어야 한다.

▶게시판에 글을 등록하려면 로그인을 할 수 있어야 한다

▷아이디, 비밀번호에 대한 유효성 검사를 진행할 수 있어야 한다.

▶로그인을 하기 위한 회원 가입 기능이 있어야 한다.

▷아이디, 비밀번호, 이메일, 휴대폰 번호에 대한 유효성 검사를 진행할 수 있어야 한다.

▶게시판에 글을 수정할 수 있어야 한다.

▷아이디, 비밀번호, 이메일, 휴대폰 번호에 대한 유효성 검사를 진행할 수 있어야 한다.

▶게시판에 글을 삭제할 수 있어야 한다.

▶게시판에 글에 댓글을 작성할 수 있어야 한다.

▶게시판에 글에 답글을 작성할 수 있어야 한다.

화면 설계서

프로젝트 명	생활 환경에 따른 노인 건강 분석 및 정책 제공 서비스	페이지	14-1
설명	기대수명 예측 페이지 (상단)		

기대수명 예측 페이지 Description	
1	현재 본인의 나이를 직접 입력 혹은 ▲▼ 아이콘을 클릭하여 최소 20세에서 최대 84세까지 입력 가능하도록 설정
2	체크박스를 통해 본인에게 맞는 선택지를 선택할 수 있음
3	작성 완료 버튼 클릭 시 기대 수명, 기대 여명 확인 가능
4	값 도출 식 보기 버튼을 사용하여 기대 여명 예측 계산식에 대한 정보 확인 가능
5	

[기대 수명 제공 페이지(개인)]

프로젝트 명	생활 환경에 따른 노인 건강 분석 및 정책 제공 서비스	페이지	14-2
설명	기대수명 예측 페이지 (상단, 결과 화면)		

기대수명 예측 페이지 Description	
1	작성 완료 버튼을 누르고 남은 기대 수명 수치와 기대 여명 수치 값을 표시
2	값 도출 식에 대한 설명으로 이동할 수 있는 버튼 표시
3	
4	
5	

[기대 수명 제공 페이지(개인, 결과)]

화면 설계서

프로젝트 명	생활 환경에 따른 노인 건강 분석 및 정책 제공 서비스	페이지	14-3
설명	기대수명 예측 페이지 (하단)		

기대수명 예측 페이지 Description	
1	기대수명 예측 정보를 차트로 나타냄
2	셀렉트 박스를 이용해 확인할 지역과 비교할 지역 선택 가능
3	셀렉트 박스를 이용해 2024년 부터 2026년까지 선택 가능
4	
5	

[미래 기대 수명 제공 페이지(지역)]

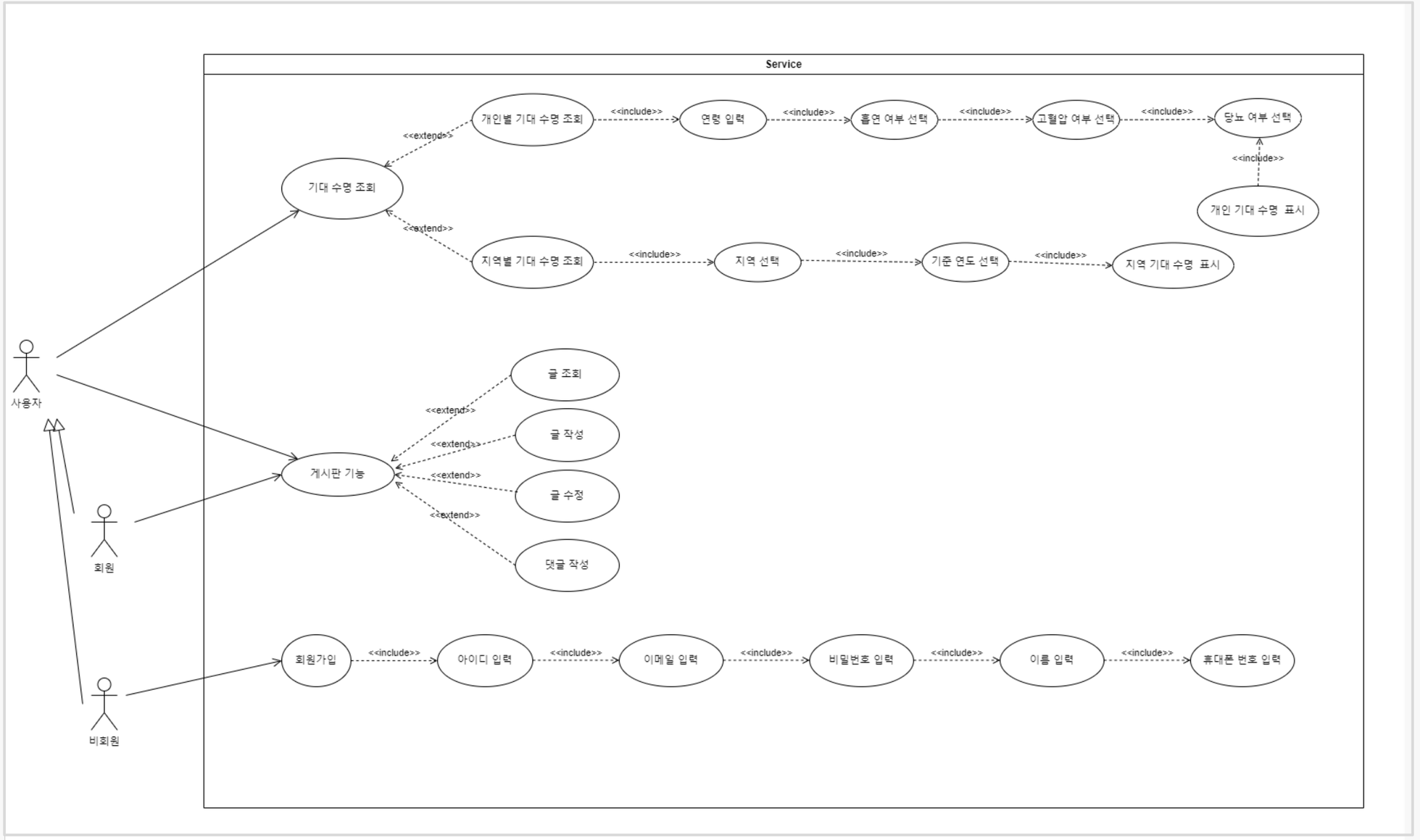
프로젝트 명	생활 환경에 따른 노인 건강 분석 및 정책 제공 서비스	페이지	14-4
설명	기대수명 예측 페이지 (중단)		

기대수명 예측 페이지 Description	
1	과거 수명 정보를 차트로 나타냄
2	셀렉트 박스를 이용해 지역 설정 가능
3	
4	
5	

[과거 실제 수명 제공 페이지(지역)]

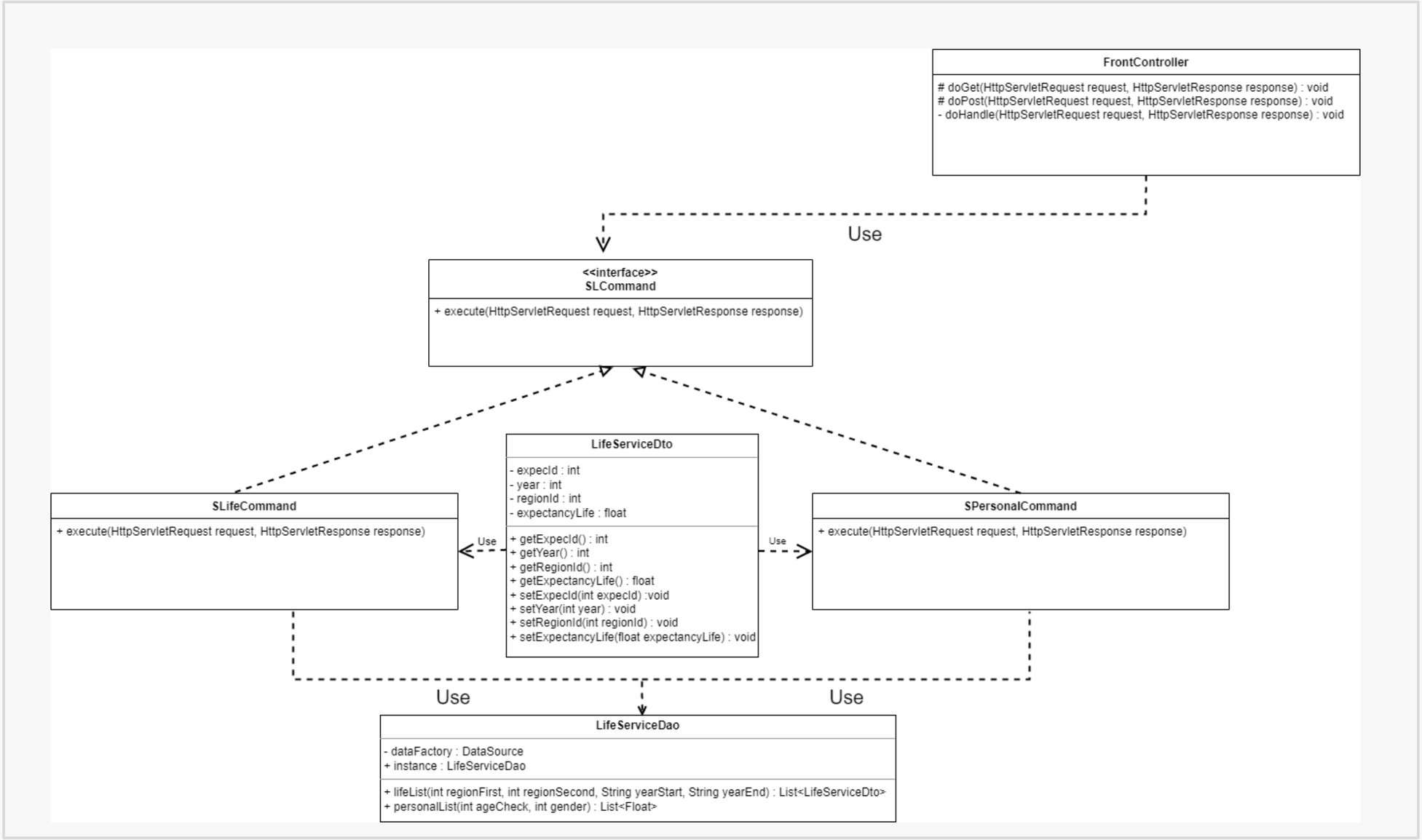
UML

Usecase Diagram



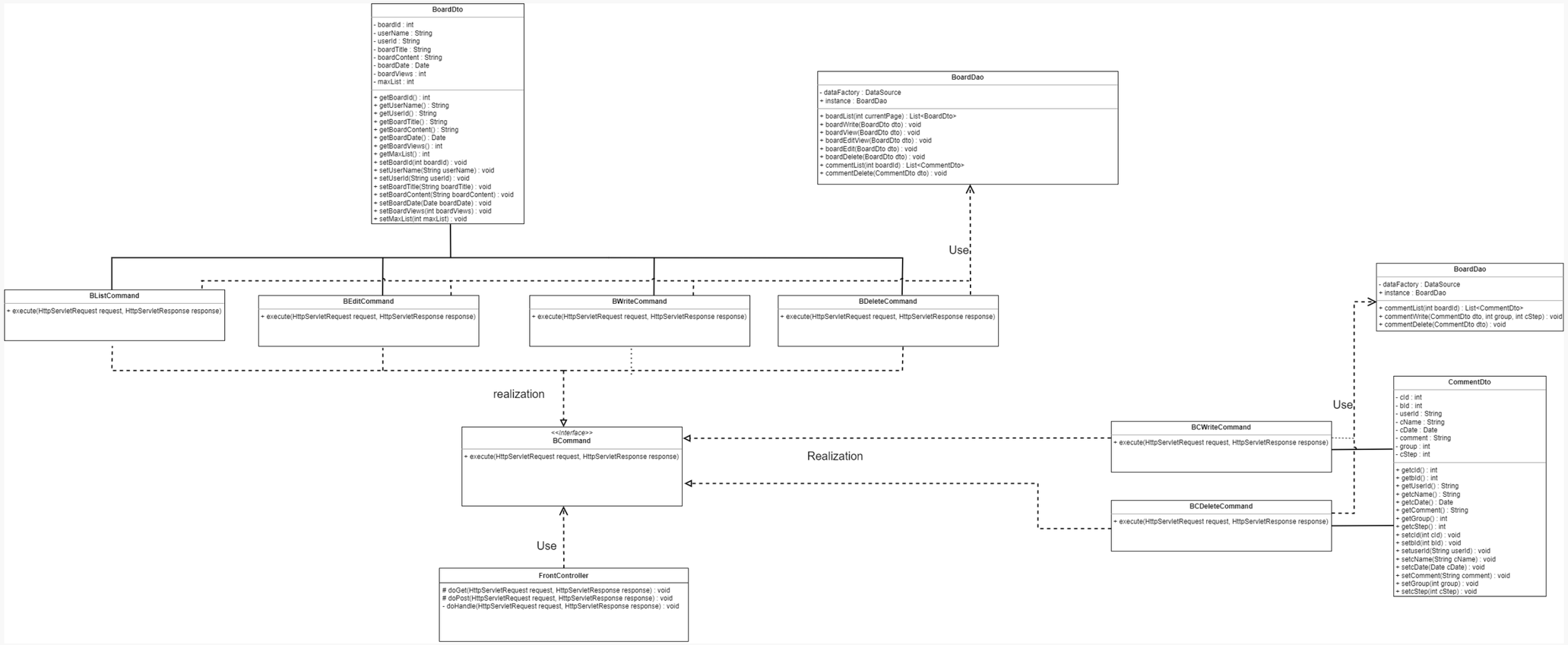
UML

Class Diagram(기대 수명 서비스)



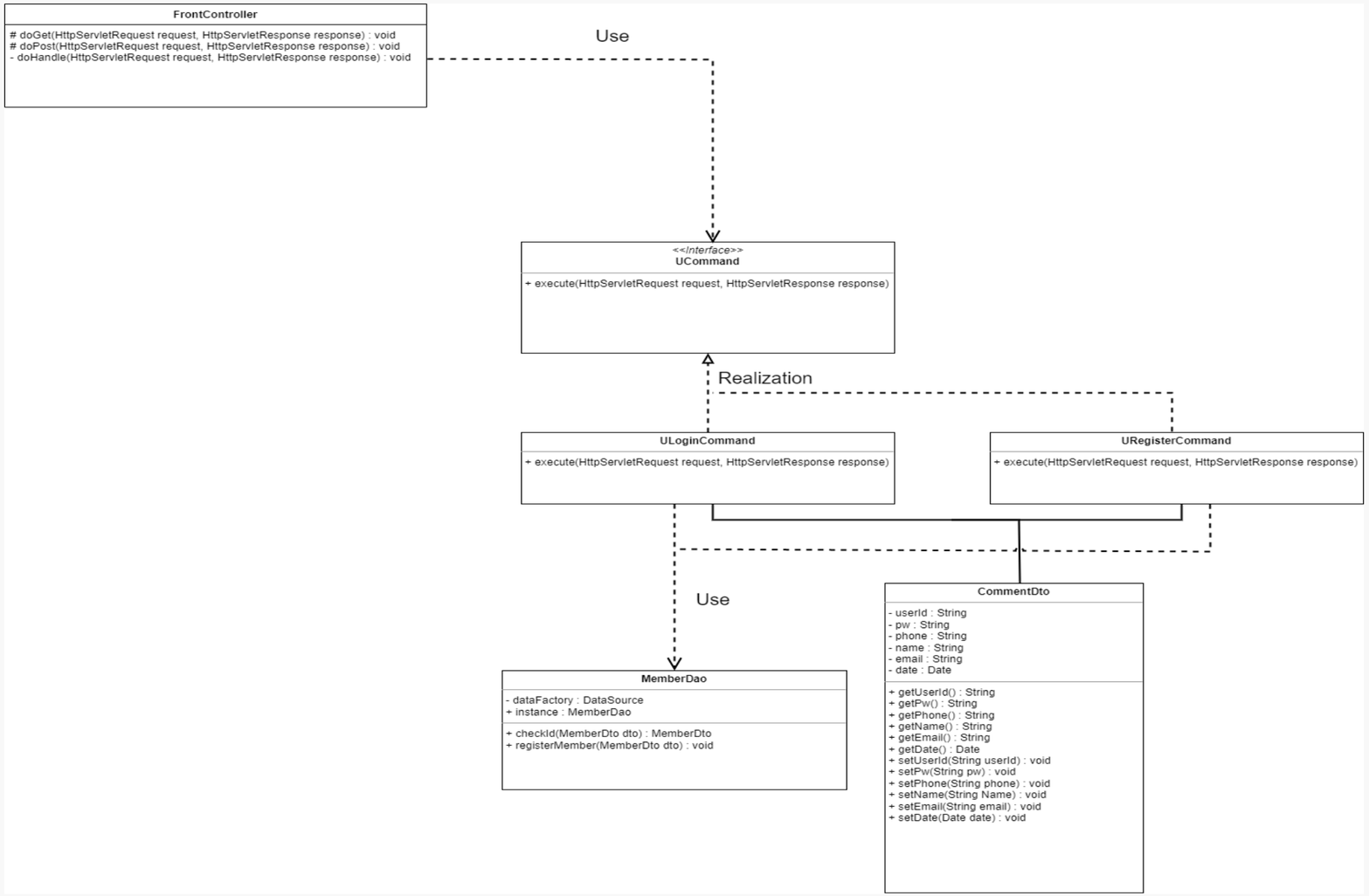
UML

Class Diagram(게시판 서비스)



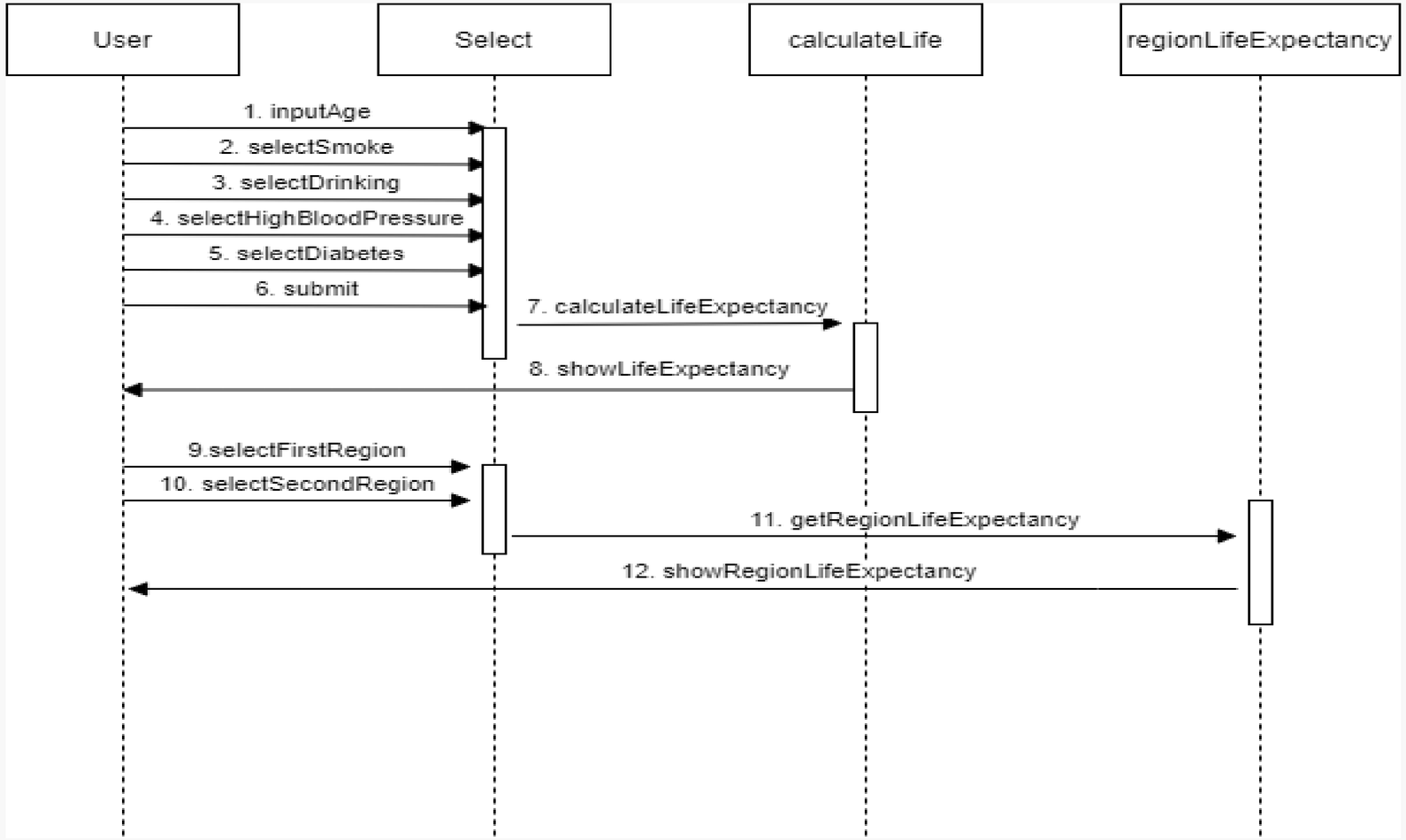
UML

Class Diagram(기대 수명 서비스)

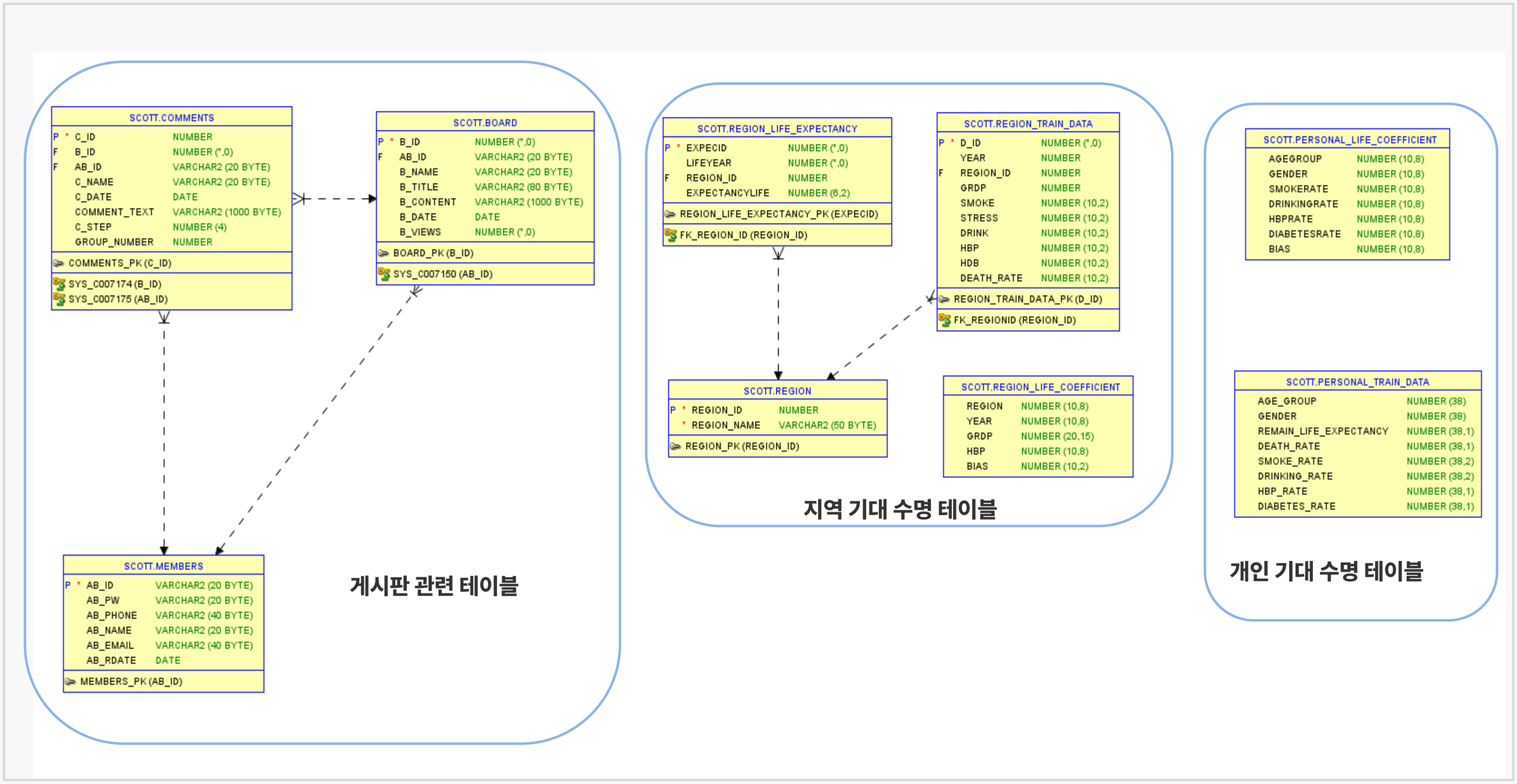


UML

Sequence Diagram(기대 수명 서비스)



ERD(Entity Relationship Diagram)



주요 기능 설명

주요 서비스 기능(개인별 기대 여명 정보 제공)

사용자가 입력한 값(나이, 성별, 흡연 여부, 음주 여부, 고혈압 여부)을 회귀 분석을 사용하여 사용자의 기대 여명을 예측

사용자의 기대 여명에 대한 정보는
비동기 방식을 사용하여 서버에 요청

기대 여명 계산 방법에 대한 정보를 버튼을 통해서
누구든지 접근 가능

주요 기능 설명

주요 서비스 기능(지역별 기대 여명 정보 제공)

통계청의 연평균 데이터를 사용하여 훈련 시킨 회귀 모델을 사용

(지역, 연도별 GRDP, 고혈압 비율, 스트레스 인지율, 음주율, 당뇨 비율 데이터 사용)

지역별 기대 수명 정보를 시각화하여 정보 제공

사용자의 기대 수명과 지역별 기대 수명과 비교 가능

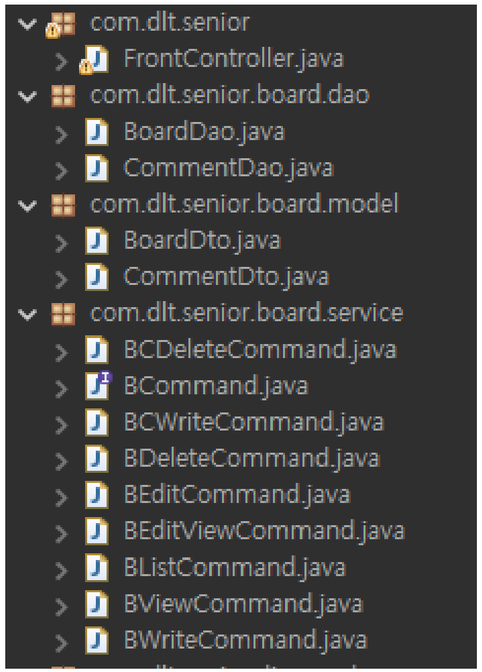
사용자가 비교하고 싶은 지역을 선택하여
지역별로 기대 수명을 비교 가능

차트 정보는 비동기 방식을 사용하여 제공

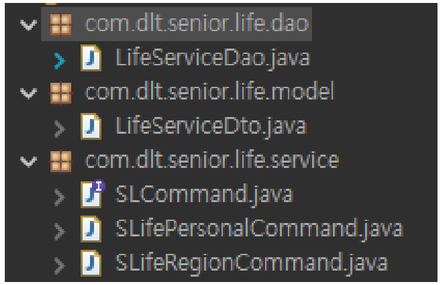
주요 기능 설명

프로젝트 패키지 구조

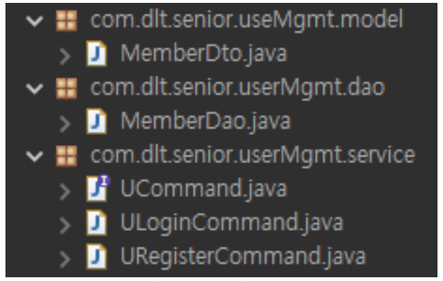
Controller, Model



[FrontController, 게시판 패키지]

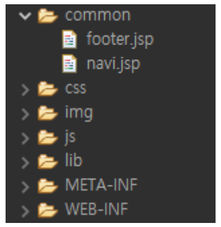


[기대 수명 관련 패키지]

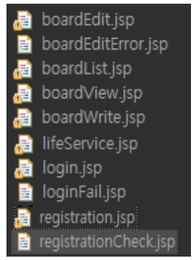


[유저 관리 패키지]

View



[Static 폴더, 공통 View]



[게시판, 기대 수명 View]

[프로젝트 패키지]

Controller

com.dlt.senior
- FrontController.java : 클라이언트의 요청 처리

(model2 -> Spring 전환 시에 컨트롤러를 서비스별로 분산 예정)

View

common 디렉토리

- footer.jsp : 푸터 작성 JSP

- navi.jsp : 네비게이션 작성 JSP

- boardEdit.jsp : 게시글 수정 페이지

- boardEditError.jsp : 게시글 작성 중 예외 처리 페이지

- boardList.jsp : 게시글 목록 페이지

- boardWrite : 게시글 작성 페이지

- lifeService : 기대 수명 서비스 페이지

- login : 로그인 페이지

- loginFail : 로그인 실패 알림

- registration : 회원가입 페이지

- registrationCheck : 회원가입 유효성 검사 페이지

Model

[게시판]

com.dlt.senior.board.model
- BoardDto.java
- CommentDto.java

com.dlt.senior.board.dao
- BoardDao.java
- CommentDao.java

com.dlt.senior.board.service
-BCommand.java : 인터페이스

-BCDeleteCommand.java
-BCWriteCommand.java

-BDeleteCommand.java
-BEditCommand.java
-BEditViewCommand.java
-BListCommand.java
-BViewCommand.java
-BWriteCommand.java

[기대 수명]

com.dlt.senior.life.model
- LifeServiceDto.java

com.dlt.senior.life.dao
- LifeServiceDao.java

com.dlt.senior.life.service
- SLCommand.java : 인터페이스
- SLifePersonalCommand.java
- SLifeRegionCommand.java

[유저 관리]

com.dlt.senior.userMgmt.model
- MemberDto.java

com.dlt.senior.userMgmt.dao
- MemberDao.java

com.dlt.senior.userMgmt.service
- Ucommand.java : 인터페이스
- ULoginCommand.java
- URegisterCommand.java

코드 설명

[Controller] Java/ FrontController.java

```java
@WebServlet("*.do")
public class FrontController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    //API로 가져오는 데이터
    @Override
    public void init() throws ServletException {
        // 데이터가 이미 존재하는지 여부를 확인합니다.
        SCCheckData checkData = new SCCheckData();

        // 데이터 초기화를 위해 서비스 클래스를 호출합니다.
        checkData.initializeIfNeeded();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doHandle(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doHandle(request, response);
    }

    private void doHandle(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset = utf-8");

        //session
        HttpSession sess = request.getSession();
        BCommand board;

        //mapping path
        String path = request.getContextPath();
        String uri = request.getRequestURI();
        String command = uri.substring(path.length());
        System.out.println("사용한 커맨드  : " + command);

        /*main page*/
        if(command.equals("/") || command.equals("/index.do")) {
            board = new BListCommand();
            request.setAttribute("page", 1);

            board.execute(request, response);

            RequestDispatcher dispatch = request.getRequestDispatcher("index.jsp");
            dispatch.forward(request, response);
        }
```

@WebServlet(“*.do”)

클라이언트의 모든 요청에 대한 처리

substring을 사용하여 URI 정보에서 *.do만 분리하여

if문을 사용해, 모든 요청을 처리

클라이언트로 response를 보낼 때는

redirect와 forward 방식을 주로 사용

[View] JavaScript/ lifeService.js

```javascript
else if (diaCheck == undefined) {
    alert("당뇨병 여부를 선택해주세요");
}
else {
    let sendData = {
        personal_age: ageCheck,
        genderCheck: genderCheck,
        smokeCheck: smokeCheck,
        drinkingCheck: drinkingCheck,
        hbpCheck: hbpCheck,
        diabetesCheck: diaCheck
    }
    fetch('personalLife.do', {
        method: 'POST',
        body: JSON.stringify(sendData)
    })
        .then(response => response.json())
        .then(data => {
            const lifeContainer = document.getElementById('inputLife');
            lifeContainer.innerHTML = '<div class = "col mb-1" style="fon
        })
        .catch(error => {
            console.log(error);
            console.log("fetch가 제대로 작동하지 않습니다.");
        });
}
```

Fetch API를 사용하여 비동기 방식을 사용하여

서버와 데이터 교환

[View] navi.jsp

```jsp
<div>
    <c:forEach var="dto" items="${boardList}" varStatus="status">
        <div class="num">${dto.getBoardId() }</div>
        <div class="title">
            <a href="./boardView.do?boardId=${dto.getBoardId()}" style="text-decoration:none">${dto.getBoardTitle()}</a>
        </div>
        <div class="writer">${dto.getUserName() }</div>
        <div class="date">${dto.getBoardDate()}</div>
        <div class="count">${dto.getBoardViews()}</div>
    </c:forEach>
</div>
```

JSTL와 EL을 사용하여 데이터를

출력하거나 로그인 세션 관리

[Model] LifeServiceDao.java

```java
public List<LifeServiceDto> lifeList(int regionFirst, int regionSecond, String yearStart, String yearEnd){
    Connection con = null;
    PreparedStatement stmt = null;
    try {
        con = dataFactory.getConnection();
        String query = "SELECT * FROM region_life_expectancy WHERE lifeyear >= ? "
                + "AND lifeyear <= ? AND (region_id = ? OR region_id = ?)";
        stmt = con.prepareStatement(query);
        int start = Integer.parseInt(yearStart);
        int end = Integer.parseInt(yearEnd);
        stmt.setInt(1, start);
        stmt.setInt(2, end);
        stmt.setInt(3, regionFirst);
        stmt.setInt(4, regionSecond);
        ResultSet rs = stmt.executeQuery();

        List<LifeServiceDto> list = new ArrayList<LifeServiceDto>();

        LifeServiceDto lifeDto;

        while(rs.next()) {
            lifeDto = new LifeServiceDto();
            String lifeyear = rs.getString("lifeyear");
            int regionId = rs.getInt("region_id");
            float lifeExpectancy = rs.getFloat("expectancylife");
            lifeDto.setYear(Integer.parseInt(lifeyear));
            lifeDto.setRegionId(regionId);
            lifeDto.setExpectancyLife(lifeExpectancy);
            list.add(lifeDto);
            //현재 여기에서 두번 도는 문제 -> 지역이 2개니까 연도도 2배임
        }
        return list;
    }catch(Exception e) {
        e.printStackTrace();
    }finally {
        try {
            if(stmt != null) {
                stmt.close();
            }if(con != null) {
                con.close();
            }

        }catch(Exception e2) {
            e2.printStackTrace();
        }
    }
```

쿼리문을 사용하여 DB의 데이터를 로드

substring을 사용하여 URI 정보에서 *.do만 분리하여 if문을 사용해, 모든 요청을 처리

[Model] SLifeRegionCommand.java

```java
public class SLifeRegionCommand implements SLCommand{

    @Override
    public void execute(HttpServletRequest request, HttpServletResponse response) {
        LifeServiceDao lifeDao = LifeServiceDao.getInstance();

        Map<String, Integer> regionMap = new HashMap<String, Integer>();

        //맵 지역 매핑
        regionMap.put("서울특별시", 1);
        regionMap.put("전라남도", 2);
        regionMap.put("강원도", 3);
        regionMap.put("경기도", 4);
        regionMap.put("경상남도", 5);
        regionMap.put("경상북도", 6);
        regionMap.put("광주광역시", 7);
        regionMap.put("대구광역시", 8);
        regionMap.put("대전광역시", 9);
        regionMap.put("부산광역시", 10);
        regionMap.put("세종특별자치시", 11);
        regionMap.put("울산광역시", 12);
        regionMap.put("인천광역시", 13);
        regionMap.put("전라북도", 14);
        regionMap.put("제주특별자치도", 15);
        regionMap.put("충청남도", 16);
        regionMap.put("충청북도", 17);

        String regionFirst = request.getParameter("regionFirst");
        String regionSecond = request.getParameter("regionSecond");
        String yearStart = request.getParameter("yearStart");
        String yearEnd = request.getParameter("yearEnd");

        int firstRegionId = regionMap.getOrDefault(regionFirst, 0);
        int secondRegionId = regionMap.getOrDefault(regionSecond, 0);

        List<LifeServiceDto> list = lifeDao.lifeList(firstRegionId, secondRegionId, yearStart, yearEnd);
        request.setAttribute("lifeList", list);

    }
}
```

비즈니스 로직을 처리하는 Service 계층으로써, DAO에서 데이터를 가져와서 계산

컬렉션 Map을 사용하여 데이터를 저장

[Model] LifeServiceDto.java

```java
public class LifeServiceDto {
    private int expecId;
    private int year;
    private int regionId;
    private float expectancyLife;

    public int getExpecId() {
        return expecId;
    }
    public int getYear() {
        return year;
    }
    public int getRegionId() {
        return regionId;
    }
    public float getExpectancyLife() {
        return expectancyLife;
    }
```

DTO를 사용하여 DB와 DAO, Service 간의 데이터 교환

코드 설명

프로젝트 코드 설명(R, 개인별 기대 수명 회귀 모델)

활용 데이터

- 연령대, 성별 기대 여명 데이터(통계청)
- 연령대, 성별 연평균 흡연율, 음주율, 고혈압 유병률, 당뇨 비율(통계청)

```r
#package loading
library(tidyverse) #NA값 처리
library(readxl) #excel 파일 읽어오기
library(knitr) #데이터 표 형태로 출력하기
library(ggplot2) #데이터 시각화
library(MASS) #특성 선택을 위한 라이브러리
library(scales) #x축 눈금 표시 방식 정의

#데이터 불러오기
dataTotal <- data.frame(read_excel("데이터 총합.xlsx"))
dataTotal_origin <- dataTotal
dataTotal

#전체 지역 기대 수명 예측 모델 훈련
life_expectancy_model <- lm(life.expectancy ~ ., data = dataTotal)
life_expectancy_model

summary(life_expectancy_model)

#전체 지역 기대 수명 예측 모델 단계적 선택법 적용
life_expectancy_model2 <- stepAIC(life_expectancy_model)
life_expectancy_model2

summary(life_expectancy_model2)
```

lm 함수를 사용하여 모델 훈련

stepAIC(단계적 선택법)을 활용하여 모델 성능 향상

[R Script]

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.66644 -0.34194 -0.04513  0.25277  0.99177

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   81.08273    1.09380  74.130  < 2e-16 ***
age_group     -0.88041    0.01550 -56.800  < 2e-16 ***
gender         1.87036    0.39384   4.749 0.000123 ***
drinking_rate -0.09449    0.01672  -5.651 1.57e-05 ***
hbp_rate      -0.07766    0.01061  -7.318 4.49e-07 ***
diabetes_rate  0.06718    0.02628   2.556 0.018834 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 0.4556 on 20 degrees of freedom
Multiple R-squared:  0.9995,    Adjusted R-squared:  0.9993
F-statistic:  7466 on 5 and 20 DF,  p-value: < 2.2e-16
```

[Model Summary]

코드 설명

프로젝트 코드 설명(R, 지역별 기대 수명 회귀 모델)

활용 데이터

- 지역별 0세 기준 기대 여명 데이터(통계청)
- 연도, 지역별 GRDP, 흡연율, 스트레스 인지율, 음주율, 고혈압 유병률, 당뇨율, 사망률(통계청)

```r
#전체 지역 기대 수명 예측 모델 훈련
life_expectancy_model <- lm(life.expectancy ~ ., data = dataTotal)
life_expectancy_model

summary(life_expectancy_model)
```

```r
#전체 지역 기대 수명 예측 모델 단계적 선택법 적용
life_expectancy_model2 <- stepAIC(life_expectancy_model)
life_expectancy_model2

summary(life_expectancy_model2)
```

[R Script]

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.633e+02  6.749e+02  -1.427   0.1840
year         5.260e-01  3.393e-01   1.550   0.1521
region      -4.059e+00  1.998e+00  -2.031   0.0697 .
GRDP        -8.226e-06  7.897e-06  -1.042   0.3221
smoking     -2.025e-02  1.579e-01  -0.128   0.9005
stress      -4.860e-02  7.148e-02  -0.680   0.5119
alcohol     -7.208e-03  5.463e-02  -0.132   0.8977
hbp         -9.726e-02  2.804e-01  -0.347   0.7359
diabetes    -2.257e-01  8.339e-01  -0.271   0.7921
death.rate  -6.208e-04  3.038e-03  -0.204   0.8422
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2716 on 10 degrees of freedom
Multiple R-squared:  0.9835,    Adjusted R-squared:  0.9687
F-statistic: 66.42 on 9 and 10 DF,  p-value: 9.818e-08
```

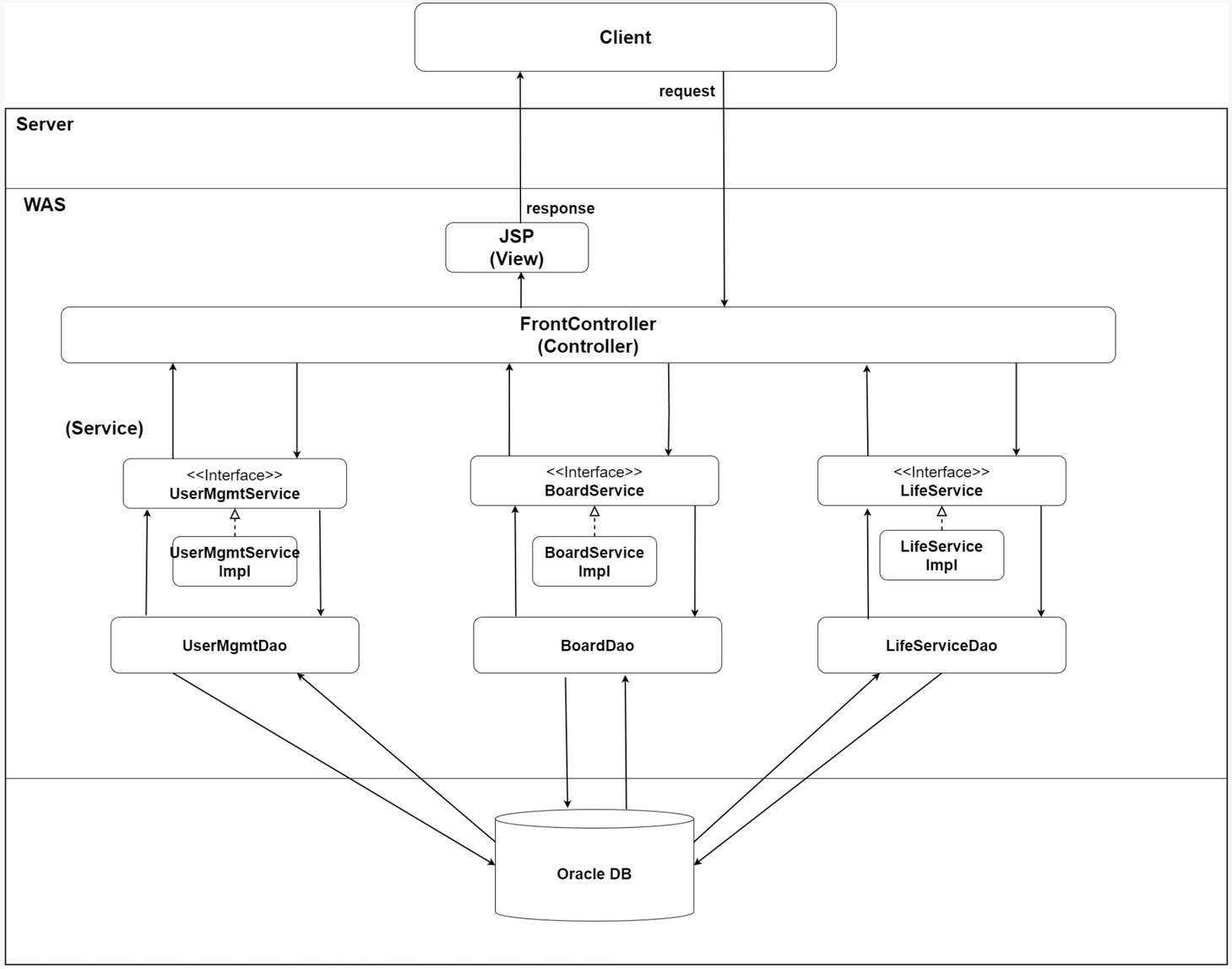
stepwise(단계적 선택법)

```
> summary(life_expectancy_model2)

Call:
lm(formula = life.expectancy ~ year + region + GRDP + hbp, data = dataTotal)

Residuals:
     Min       1Q   Median       3Q      Max
-0.79972 -0.03190  0.02279  0.10866  0.17409

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -8.406e+02  1.861e+02  -4.516 0.000410 ***
year         4.627e-01  9.358e-02   4.944 0.000176 ***
region      -3.601e+00  8.793e-01  -4.096 0.000955 ***
GRDP        -5.633e-06  2.973e-06  -1.895 0.077547 .
hbp         -1.269e-01  1.003e-01  -1.266 0.224996
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2282 on 15 degrees of freedom
Multiple R-squared:  0.9826,    Adjusted R-squared:  0.9779
F-statistic: 211.4 on 4 and 15 DF,  p-value: 5.394e-13
```

stepAIC(단계적 선택법)을 활용하여 모델 성능 향상

[Model Summary]

Software Architecture



향후 개발 계획

“Spring Security 프레임워크 적용”

- Spring Security 프레임워크를 사용하여 프로젝트의 잠재적 보안 위협을 식별하고 보안 체계를 강화
- Session -> JWT 로그인 전환, 보안 헤더 설정을 통해 애플리케이션의 보안 강화

“React를 사용한 클라이언트 최신화”

- React 라이브러리의 Virtual DOM을 도입하여 불필요한 화면의 갱신을 최소화하여 클라이언트 측의 성능을 향상

“ 훈련 모델에 다양한 기법을 적용”

- 확률적 경사 하강법과 같은 최적화 알고리즘을 학습하여 서비스에 사용되는 모델의 정확도를 향상

“ Spring -> Spring boot로 전환”

- Spring Boot로 전환함으로써 XML 기반의 구성을 줄이며 프로젝트의 종속성 관리를 단순화

프로젝트 수행 소감

선형 회귀 모델 개발 경험

선형 회귀 모델을 개발하는 도중 어려웠던 점은 데이터 탐색, 데이터 전처리 과정이었습니다. 모델을 훈련시키기 위한 맞춤형 데이터는 존재하지 않았고, 데이터를 여러 곳에서 가져오며 **사용했기에** 모델의 훈련이 원활하게 진행되지 않았습니다.

하지만, 계속해서 모델의 정확도를 높이기 위해서 부족한 부분에 대해서 공부를 진행하며 특성들의 스케일을 조정하고, 부족한 데이터를 탐색하여 모델에 추가해보는 여러 과정을 거쳐 원하는 수치에 가까운 수치를 얻을 수 있었습니다.

이 경험을 통해 프로젝트 진행 중이라 할지라도 지속적인 학습이 필요하고 학습을 통해 쌓은 지식을 실제로 사용하였을 때의 즐거움을 깨달았습니다.

팀장이라는 역할에 대한 고민

이번 팀 프로젝트에서 팀장을 처음으로 맡게 되었습니다. 처음 진행하는 웹 개발 프로젝트에 팀장을 맡으며 여러가지 걱정이 되었습니다.

프로젝트 진행 도중에도 제 서비스도 구현하며 다른 팀원들의 작업 일정 조율이나 팀원들의 서비스 코드들을 합치면서 프로젝트의 문제가 발생하는 등 예상치 못한 시간적 문제가 발생했습니다.

하지만 팀원들과 상의하고 협업하며 여러 문제들을 하나씩 해결하면서 무사히 프로젝트를 마치게 되었습니다.

팀장으로서 성공한 경험을 통해 예상치 못한 상황에 대한 대처 능력 전체적인 상황을 넓게 볼 수 있는 안목과 자신감을 얻을 수 있었습니다.

설계에 대한 중요성

프로젝트를 시작하고 기획에 대한 문서를 작성했습니다. 프로젝트 기획서부터 화면 설계서, 개발 스케줄 등 프로젝트의 전체적인 설계를 진행하고 프로젝트 코드 구현을 진행했습니다.

대학 재학 중에는 프로젝트를 완료하고 문서를 작성했다면, 프로젝트를 진행하기 위해 문서를 작성하는 과정은 조금 버거웠습니다.

하지만, 이런 문서화 과정으로 인해서 프로젝트를 실제로 구현하는 과정에서 작성한 문서들을 참고하여 진행했기에 프로젝트가 원활하게 진행되었습니다.

이 경험을 통해 프로젝트 기획과 설계에 대한 문서화 작업의 중요성을 깨달았습니다.

감사합니다

유정민

https://github.com/p1p2p1/Project_DLT