Smart Planner

- 3. Design Document -

소속 : 영남대학교 컴퓨터공학과
학번 : 22213489
이름 : 표주원
e-mail : pjwp0928w@yu.ac.kr

[Revision history]

Revision date	Version #	Description	Author
05/11/2025	#1.0.1	Fist Documentation	juwon pyo

= Contents =

1. Introduction

<그림 1> Software Devlopment Life Cycle

본 문서는 스마트 플래너 시스템의 소프트웨어 개발 생명주기(SDLC)에서 세 번째 단계인 Design 단계의 산출물로, 이전 단계인 Analysis Document를 기반으로 도출된 요구사항, 유스케이스(Use Case), 도메인 모델(Domain Model) 등을 바탕으로 시스템의 정적 구조와 동적 행위를 체계적으로 설계하고 정의한다. 설계 단계는 구현(Implementation) 단계로의 전환에 앞서 시스템의 핵심 구성 요소와 동작 방식을 시각적이며 명확하게 문서화하여, 개발자와 이해관계자 간의 기술적 커뮤니케이션을 가능하게 하고, 시스템의 일관성과 유지보수성을 확보하는 것을 주된 목적으로 한다.

본 프로젝트에서 개발된 스마트 플래너 시스템은 다양한 공공데이터(Open API)를 통합 활용하여 사용자 맞춤형 일정 관리 기능을 제공하는 통합형 개인 비서 플랫폼이다. 시스템은 대한민국 정부 및 공공기관에서 제공하는 날씨 데이터(Weather API), 식품영양소 데이터(Nutrition API), 지도 기반 위치 데이터(Map API), 그리고 일정 관리 기능을 위한 캘린더 API(Calendar API) 등과 연동된다. 이러한 공공데이터 기반 아키텍처는 최신 정보를 자동으로 수집, 갱신함으로써 사용자에게 실시간으로 유용한 생활 정보를 제공하고, 사용자 경험(UX)을 극대화하는 데에 중점을 두고 있다.

사용자는 시스템을 통해 회원가입 및 로그인을 진행하고, 일정을 등록하거나 즐겨찾기 장소를 추가하는 등 다양한 기능을 사용할 수 있다. 일정 등록 시에는 단순한 텍스트 일정뿐만 아니라, 식단 정보, 의상 계획, 휴가 및 장소 등록 기능 등 세부 항목들이 확

장 기능으로 제공되며, 이를 통해 사용자 일정이 보다 풍부하고 실생활에 밀접하게 연결되도록 설계되었다. 사용자는 등록된 일정을 일별, 주별, 월별 단위로 확인할 수 있으며, 저장된 식단 정보를 기반으로 영양소 정보를 검색하고, 외부 데이터를 참조하여 식단의 영양학적 균형을 스스로 점검할 수 있다. 날씨 정보를 기반으로 한 계획 조정이나 추천도 가능하도록 설계되어 있으며, 모든 기능은 직관적인 UI 흐름과 연결되어 사용자 중심의 상호작용을 가능케 한다.

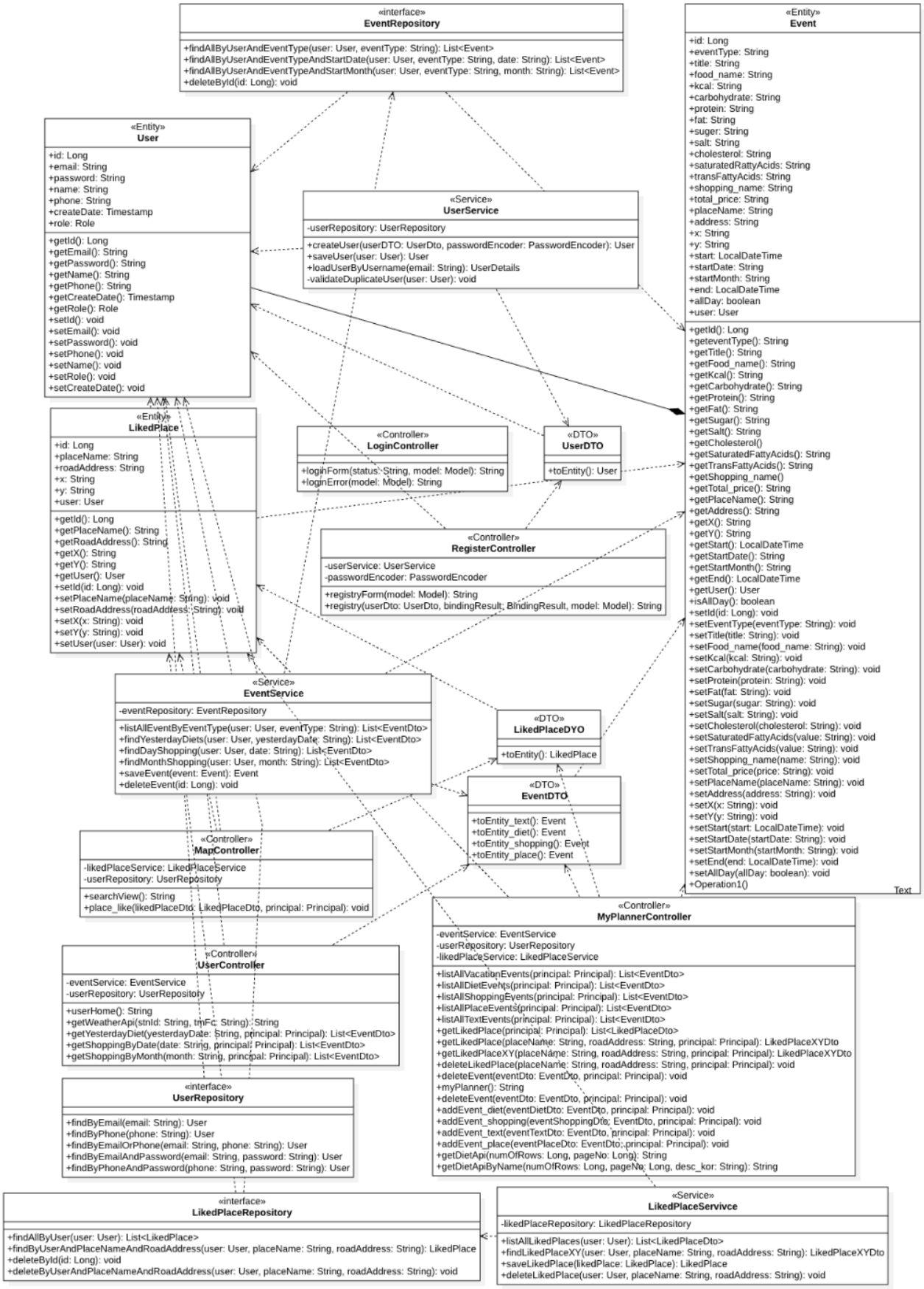
본 문서에서는 위와 같은 시스템 기능을 설계하기 위한 다음의 대표적인 UML 다이어그램들을 중심으로 구성된다.
먼저 Class Diagram은 시스템을 구성하는 클래스들의 속성(Attribute), 메서드(Operation), 클래스 간의 관계(연관, 일반화, 의존 등)를 명확히 정의하며, 정적인 시스템 구조를 기술하는 데에 중점을 둔다. 이를 통해 객체지향적 설계 원칙에 따라 시스템을 모듈화하고, 유지보수성과 재사용성을 극대화할 수 있다.

Sequence Diagram은 사용자 또는 시스템 내 객체 간의 메시지 흐름을 시간 순으로 표현하며, 기능 수행 시 객체 간의 상호작용을 시각적으로 분석하는 데 효과적이다. 본 프로젝트에서는 예를 들어 '영양소 검색'이나 '일정 등록'과 같은 핵심 기능의 수행 흐름을 구체적으로 설명한다.

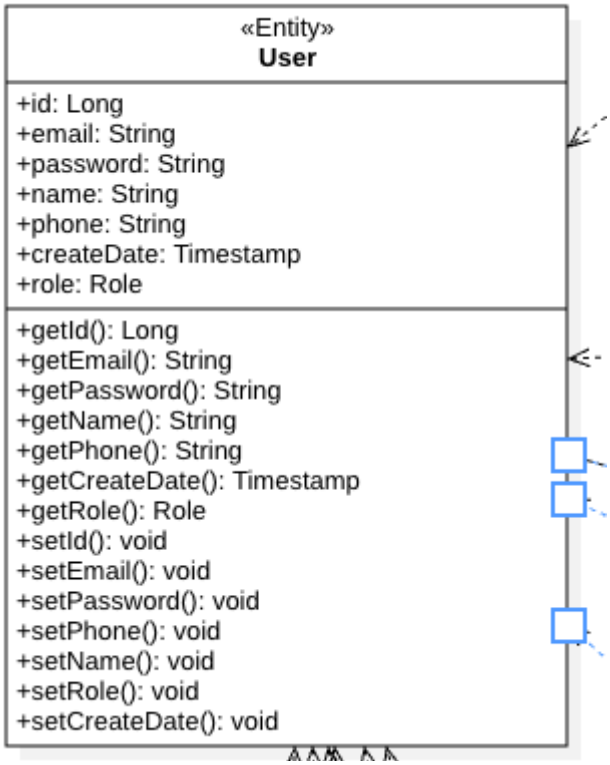
State Machine Diagram은 특정 객체나 서브시스템의 생애주기 동안 발생 가능한 모든 상태(State)와 그 전이 조건(Event, Guard 등)을 정의하여, 상태 기반의 동작 제어나 예외 처리를 명확히 설명할 수 있도록 한다.

결론적으로, 본 설계 문서는 스마트 플래너 시스템의 아키텍처 및 주요 기능을 구조적(Structural) 및 행위적(Behavioral) 관점에서 정교하게 정의하고, 구현 단계에서의 개발 방향성을 명확히 제시하기 위해 작성되었다. 이를 통해 개발팀은 설계 문서를 참조하여 통합적이고 안정적인 시스템을 구현할 수 있으며, 향후 확장성과 유지보수 측면에서도 일관성을 확보할 수 있다.

2. Class diagram



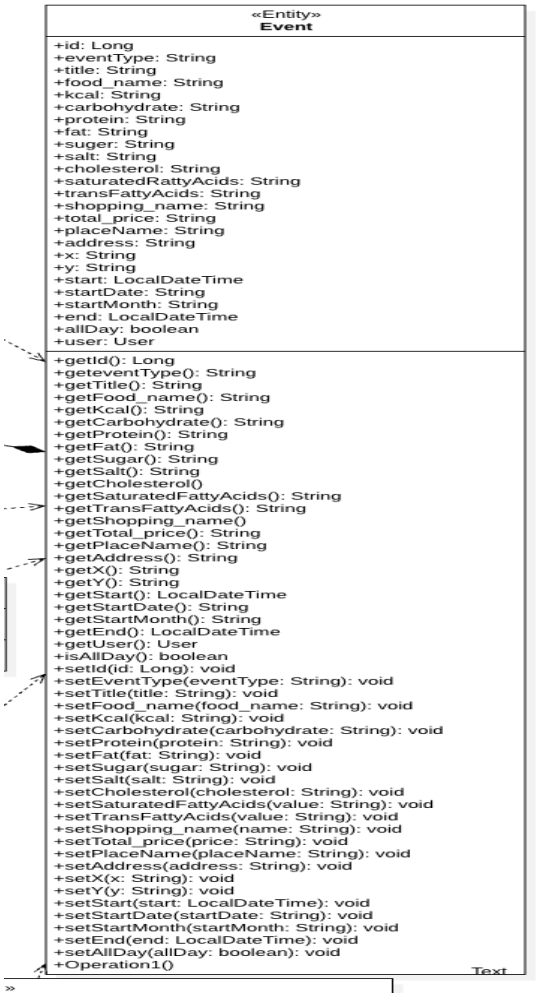
2.1 User



Attributes
id: Long : 사용자 식별자 (PK)
email: String : 사용자 이메일
password: String : 비밀번호
name: String : 사용자 이름
phone: String : 전화번호
createDate: Timestamp : 생성일시
role: Role : 사용자 권한
Methods
getId(): Long : ID 가져오기
getEmail(): String : 이메일 가져오기
getPassword(): String : 비밀번호 가져오기
getName(): String : 이름 가져오기
getPhone(): String : 전화번호 가져오기
getCreateDate(): Timestamp : 생성일시 가져오기
getRole(): Role : 권한 정보 가져오기
setId(id: Long): void : ID 설정
setEmail(email: String): void : 이메일 설정
setPassword(password: String): void : 비밀번호 설정

setName(name: String): void : 이름 설정 setPhone(phone: String): void : 전화번호 설정 setCreateDate(createDate: Timestamp): void : 생성일시 설정 setRole(role: Role): void : 권한 설정
Description
User 클래스는 시스템 내 사용자 정보를 저장하는 Model 클래스이며, 로그인 및 인증 주체로 사용되며, Event 및 LikedPlace와 연관 관계를 가집니다.

2.2 Event

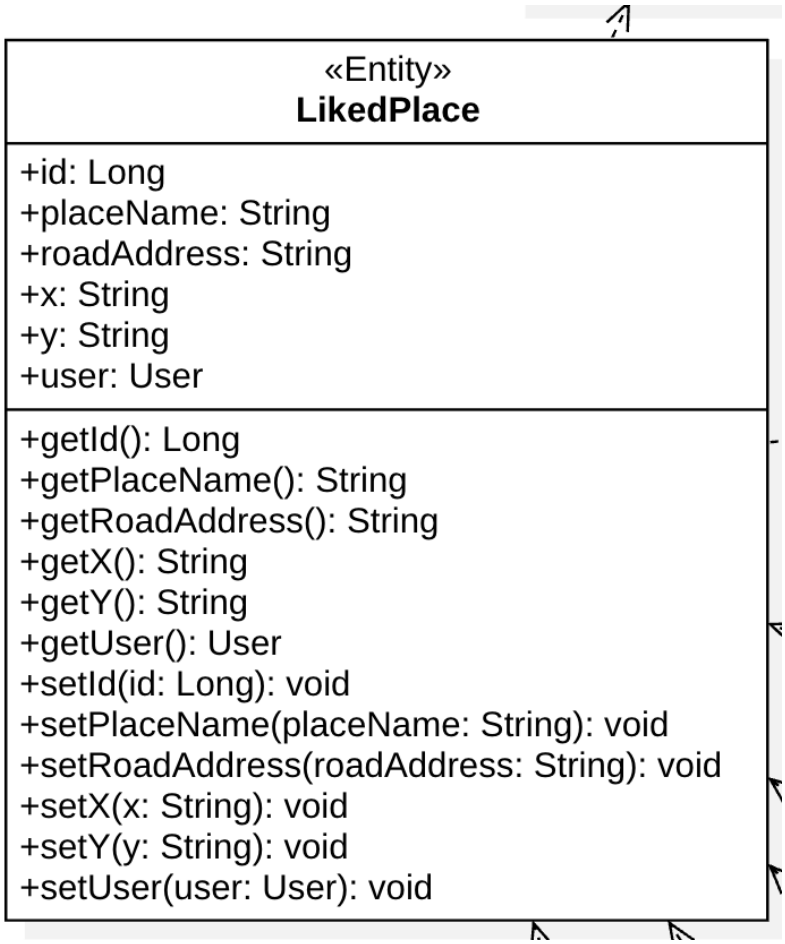


Attributes
id: Long : 이벤트 고유 식별자
eventType: String : 이벤트 유형 (예: 식단, 쇼핑, 장소 등)
title: String : 이벤트 제목
food_name: String : 음식 이름
kcal: String : 칼로리 정보
carbohydrate: String : 탄수화물 정보
protein: String : 단백질 정보
fat: String : 지방 정보
sugar: String : 당류 정보
salt: String : 나트륨 정보
cholesterol: String : 콜레스테롤 정보
saturatedFattyAcids: String : 포화지방산 정보
transFattyAcids: String : 트랜스지방 정보

shopping_name: String : 쇼핑 품목 이름
total_price: String : 총 가격
placeName: String : 장소 이름
address: String : 장소 주소
x: String : 경도 좌표
y: String : 위도 좌표
start: LocalDateTime : 시작 일시
startDate: String : 시작 날짜 (yyyy-MM-dd)
startMonth: String : 시작 월 (yyyy-MM)
end: LocalDateTime : 종료 일시
allDay: boolean : 하루 종일 여부
user: User : 이벤트 작성자 (User 객체)
Methods
getId(): Long : ID 가져오기
getEventType(): String : 이벤트 유형 가져오기
getTitle(): String : 제목 가져오기
getFood_name(): String : 음식 이름 가져오기
getKcal(): String : 칼로리 가져오기
getCarbohydrate(): String : 탄수화물 가져오기
getProtein(): String : 단백질 가져오기
getFat(): String : 지방 가져오기
getSugar(): String : 당류 가져오기
getSalt(): String : 나트륨 가져오기
getCholesterol(): String : 콜레스테롤 가져오기
getSaturatedFattyAcids(): String : 포화지방산 가져오기
getTransFattyAcids(): String : 트랜스지방 가져오기
getShopping_name(): String : 쇼핑 품목명 가져오기
getTotal_price(): String : 총 가격 가져오기
getPlaceName(): String : 장소 이름 가져오기
getAddress(): String : 주소 가져오기
getX(): String : 경도 가져오기
getY(): String : 위도 가져오기
getStart(): LocalDateTime : 시작 일시 가져오기
getStartDate(): String : 시작 날짜 가져오기
getStartMonth(): String : 시작 월 가져오기
getEnd(): LocalDateTime : 종료 일시 가져오기

getUser(): User : 사용자 객체 가져오기 isAllDay(): boolean : 하루 종일 여부 확인 setId(id: Long): void : ID 설정 setEventType(eventType: String): void : 이벤트 유형 설정 setTitle(title: String): void : 제목 설정 setFood_name(food_name: String): void : 음식 이름 설정 setKcal(kcal: String): void : 칼로리 설정 setCarbohydrate(carbohydrate: String): void : 탄수화물 설정 setProtein(protein: String): void : 단백질 설정 setFat(fat: String): void : 지방 설정 setSugar(sugar: String): void : 당 설정 setSalt(salt: String): void : 나트륨 설정 setCholesterol(cholesterol: String): void : 콜레스테롤 설정 setSaturatedFattyAcids(value: String): void : 포화지방산 설정 setTransFattyAcids(value: String): void : 트랜스지방 설정 setShopping_name(name: String): void : 쇼핑 품목명 설정 setTotal_price(price: String): void : 총 가격 설정 setPlaceName(placeName: String): void : 장소 이름 설정 setAddress(address: String): void : 주소 설정 setX(x: String): void : 경도 설정 setY(y: String): void : 위도 설정 setStart(start: LocalDateTime): void : 시작 일시 설정 setStartDate(startDate: String): void : 시작 날짜 설정 setStartMonth(startMonth: String): void : 시작 월 설정 setEnd(end: LocalDateTime): void : 종료 일시 설정 setAllDay(allDay: boolean): void : 하루 종일 여부 설정
Description
Event 클래스는 일정, 식단, 소비, 장소 등 다양한 형태의 사용자 이벤트를 저장하는 도메인 클래스이다. 각 이벤트는 User와 연관되어 있으며, 다양한 유형별 데이터를 저장할 수 있도록 설계되어 있다.

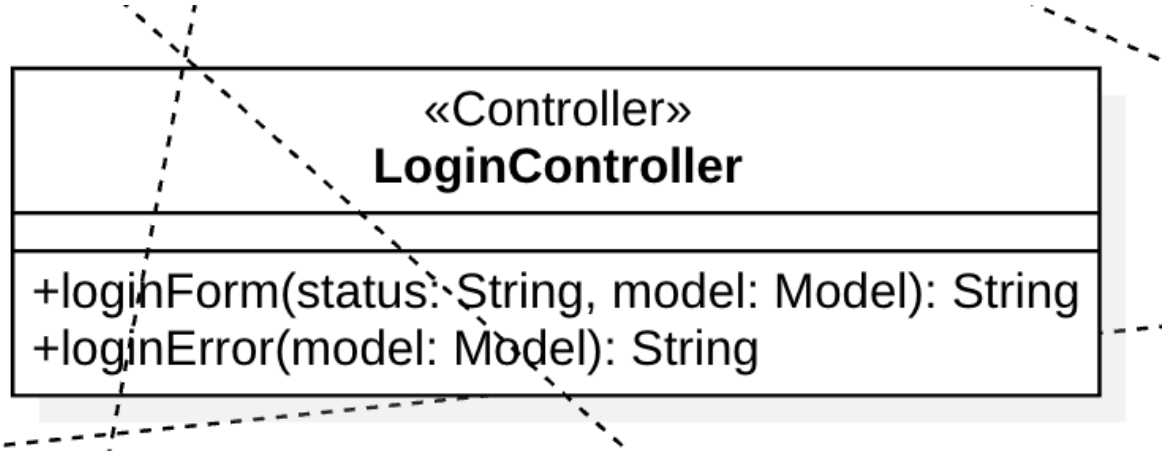
2.3 LikedPlace



Attributes
id: Long : 장소 고유 ID
placeName: String : 장소 이름
roadAddress: String : 도로명 주소
x: String : 경도 (X좌표)
y: String : 위도 (Y좌표)
user: User : 해당 장소를 등록한 사용자
Methods
getId(): Long : ID 가져오기
getPlaceName(): String : 장소 이름 가져오기
getRoadAddress(): String : 도로명 주소 가져오기
getX(): String : 경도 가져오기
getY(): String : 위도 가져오기
getUser(): User : 사용자 객체 가져오기
setId(id: Long): void : ID 설정
setPlaceName(placeName: String): void : 장소 이름 설정

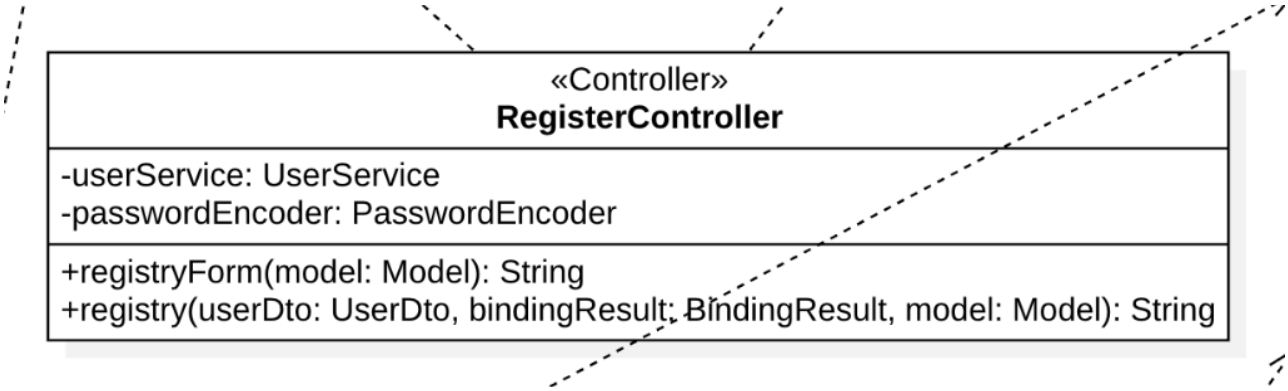
setRoadAddress(roadAddress: String): void : 도로명 주소 설정 setX(x: String): void : 경도 설정 setY(y: String): void : 위도 설정 setUser(user: User): void : 사용자 객체 설정
Description
LikedPlace 클래스는 사용자가 즐겨찾기로 저장한 장소 정보를 저장하는 도메인 클래스입니다. 각 장소는 사용자(User)와 연결되며, 지도 기반 기능 및 마이플래너 서비스에서 사용됩니다.

2.4 LoginController



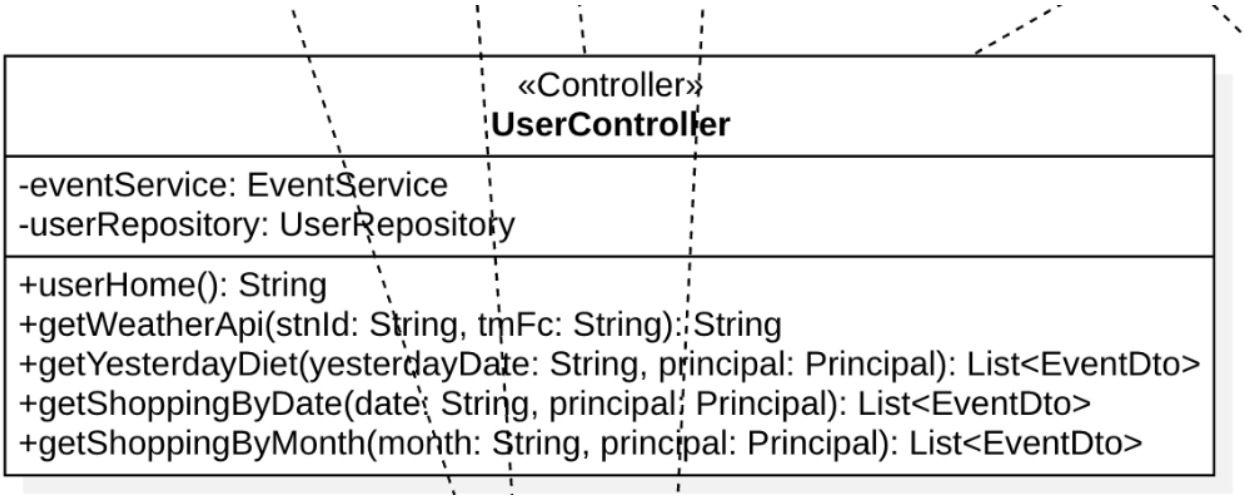
Methods
loginForm(status: String, model: Model): String : 로그인 페이지 반환 (성공 메시지 포함) loginError(model: Model): String : 로그인 실패 시 오류 메시지 반환
Description
LoginController는 로그인 화면 처리와 로그인 실패 메시지를 처리하는 역할을 수행합니다. 스프링 시큐리티의 인증 실패 시 사용자에게 적절한 피드백을 전달하는 데 사용됩니다.

2.5 ResgisterController



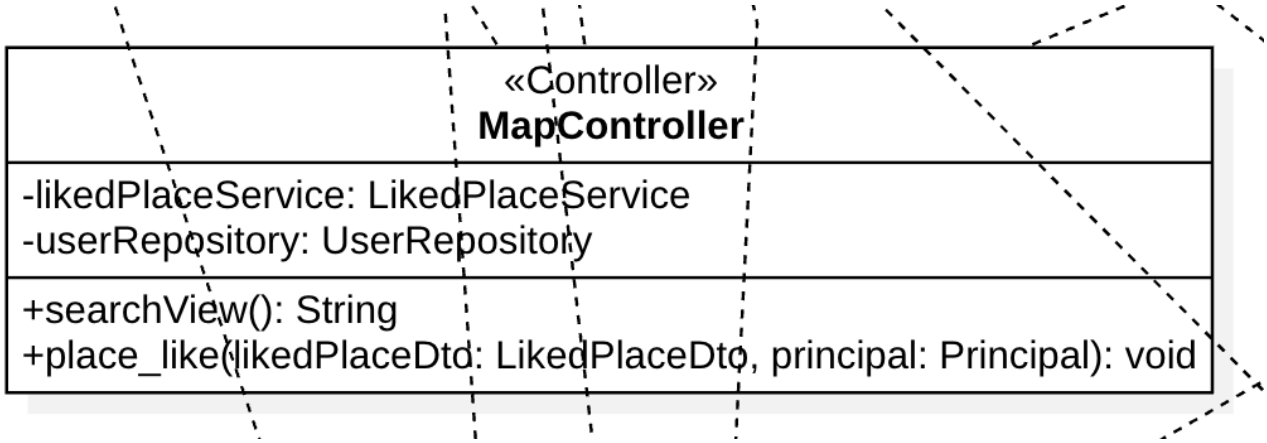
Attributes
userService: UserService : 회원 생성 서비스 passwordEncoder: PasswordEncoder : 비밀번호 암호화 처리기
Methods
registryForm(model: Model): String : 회원가입 화면 출력 registry(userDto: UserDto, bindingResult: BindingResult, model: Model): String : 회원가입 처리
Description
RegisterController는 사용자 회원가입 처리를 담당하며, 유효성 검사, 비밀번호 확인, 중복 이메일 체크 등 로직을 포함합니다.

2.6 UserController



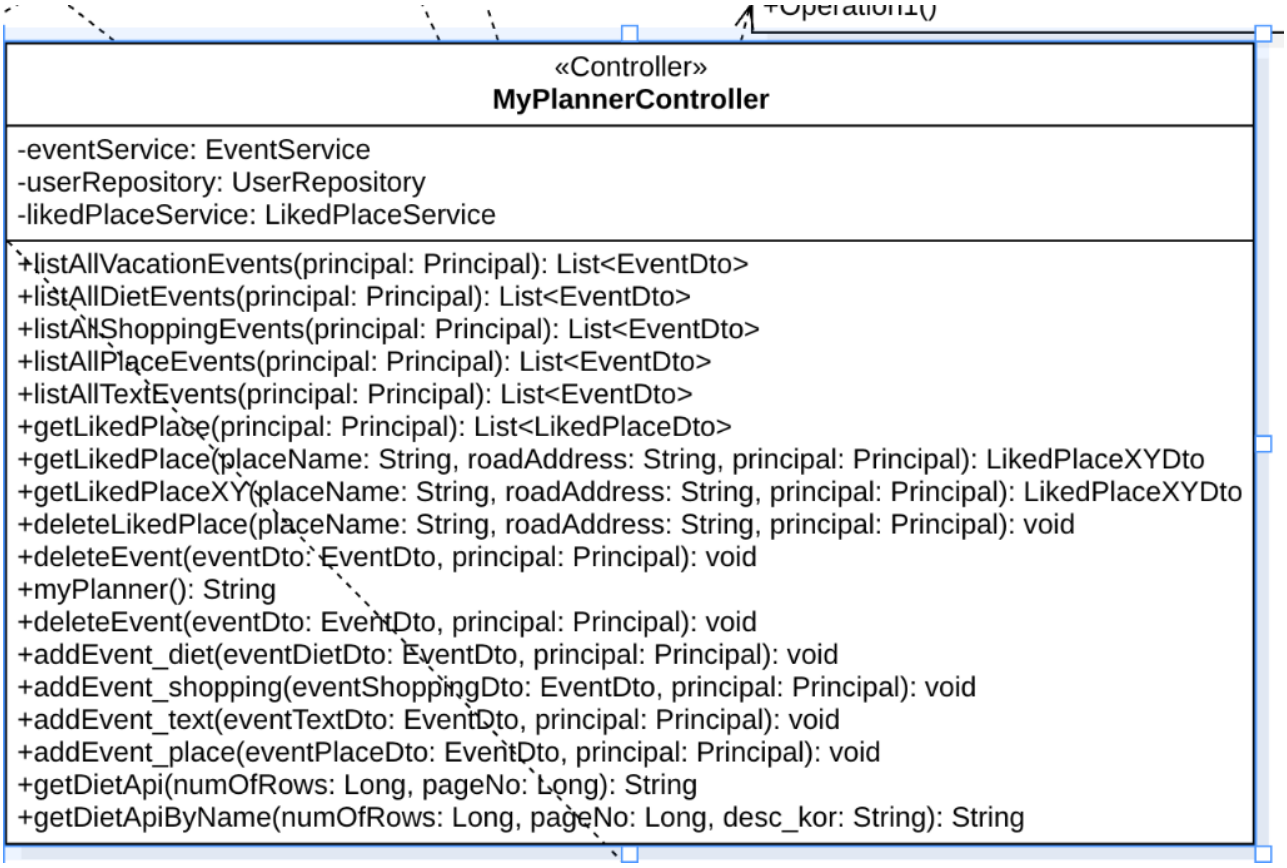
Attributes
eventService: EventService : 이벤트 관련 서비스
userRepository: UserRepository : 사용자 조회용 리포지토리
Methods
userHome(): String : 사용자 홈 화면 반환
getWeatherApi(stnId: String, tmFc: String): String : 날씨 API 호출 결과 반환
getYesterdayDiet(yesterdayDate: String, principal: Principal): List<EventDto> : 어제 식단 이벤트 조회
getShoppingByDate(date: String, principal: Principal): List<EventDto> : 날짜별 쇼핑 데이터 조회
getShoppingByMonth(month: String, principal: Principal): List<EventDto> : 월별 쇼핑 데이터 조회
Description
UserController는 사용자 홈 뷰, 날씨 API 요청, 이벤트별(식단, 쇼핑 등) 데이터 조회를 제공하는 기능을 담당합니다.

2.7 MapController



Attributes
likedPlaceService: LikedPlaceService : 즐겨찾기 등록 서비스 userRepository: UserRepository : 사용자 조회용 리포지토리
Methods
searchView(): String : 지도 검색 페이지 반환 place_like(likedPlaceDto: LikedPlaceDto, principal: Principal): void : 장소 즐겨찾기 등록 처리
Description
MapController는 사용자가 장소를 검색하거나 즐겨찾기에 등록하는 기능을 제공합니다. 카카오맵 또는 공공 위치 API와의 연동을 통해 사용자 지정 장소를 저장합니다.

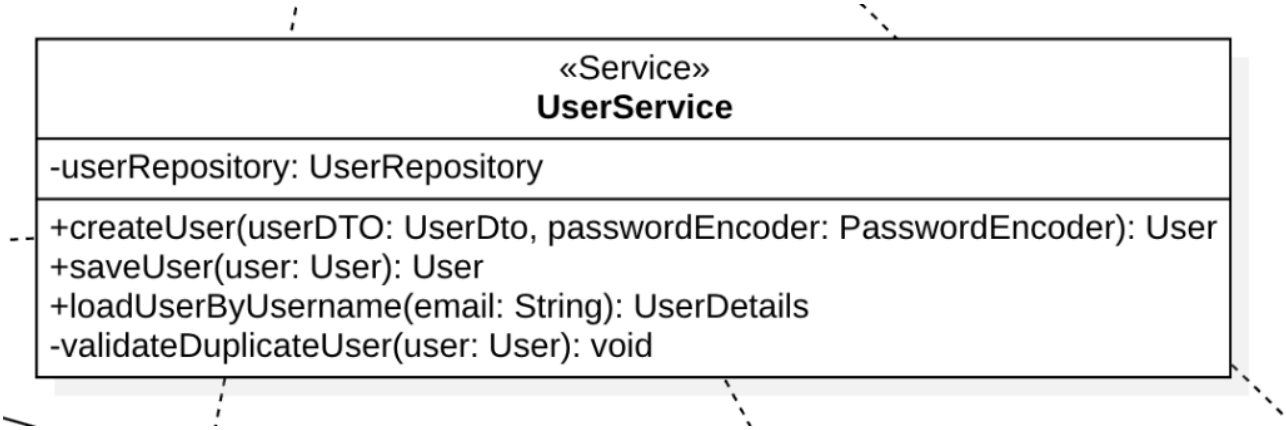
2.8 MyPlannerController



Attributes
eventService: EventService : 이벤트 CRUD 서비스
userRepository: UserRepository : 사용자 조회용 리포지토리
likedPlaceService: LikedPlaceService : 즐겨찾기 서비스
Methods
listAllVacationEvents(principal): List<EventDto> : 휴가 이벤트 조회
listAllDietEvents(principal): List<EventDto> : 식단 이벤트 조회
listAllShoppingEvents(principal): List<EventDto> : 쇼핑 이벤트 조회
listAllPlaceEvents(principal): List<EventDto> : 장소 이벤트 조회
listAllTextEvents(principal): List<EventDto> : 텍스트 이벤트 조회
getLikedPlace(principal): List<LikedPlaceDto> : 즐겨찾기 전체 조회
getLikedPlace(placeName, roadAddress, principal): LikedPlaceXYDto : 특정 즐겨찾기 좌표 조회
deleteLikedPlace(placeName, roadAddress, principal): void : 즐겨찾기 삭제
addEvent_text(dto, principal): void : 텍스트 이벤트 저장
addEvent_diet(dto, principal): void : 식단 이벤트 저장
addEvent_shopping(dto, principal): void : 쇼핑 이벤트 저장

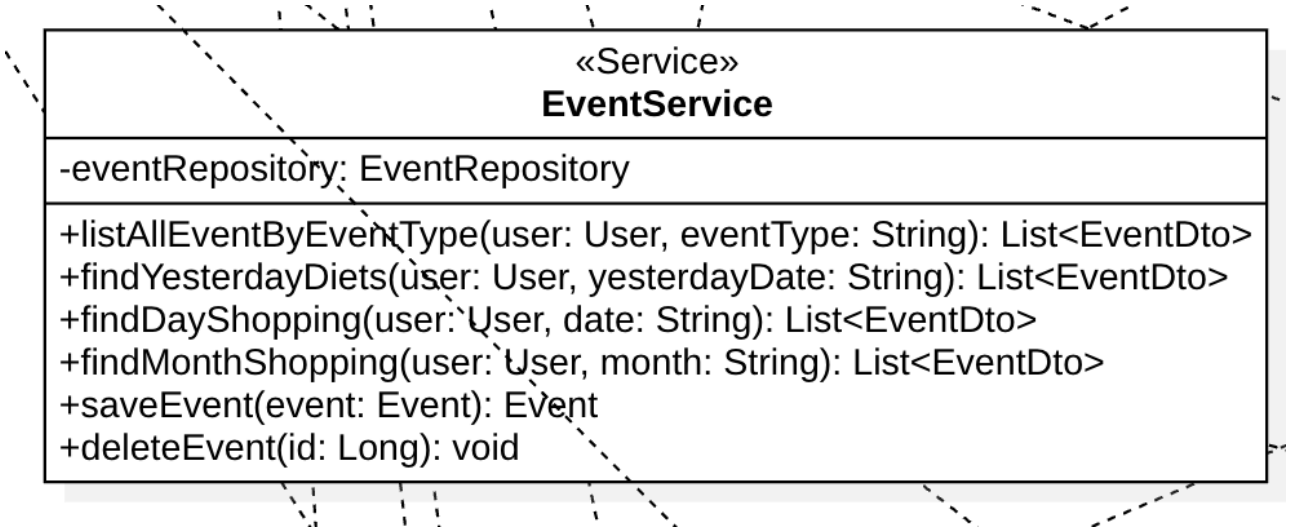
addEvent_place(dto, principal): void : 장소 이벤트 저장 deleteEvent(dto, principal): void : 이벤트 삭제 getDietApi(numOfRows, pageNo): String : 공공 API - 식품 영양 정보 페이지 단위 조회 getDietApiByName(numOfRows, pageNo, desc_kor): String : 공공 API - 식품명 기반 영양 정보 검색
Description
MyPlannerController는 사용자 일정(텍스트/식단/쇼핑/장소)과 즐겨찾기 장소를 관리하는 핵심 기능을 제공하며, 공공데이터 API도 연동합니다.

2.9 UserServive



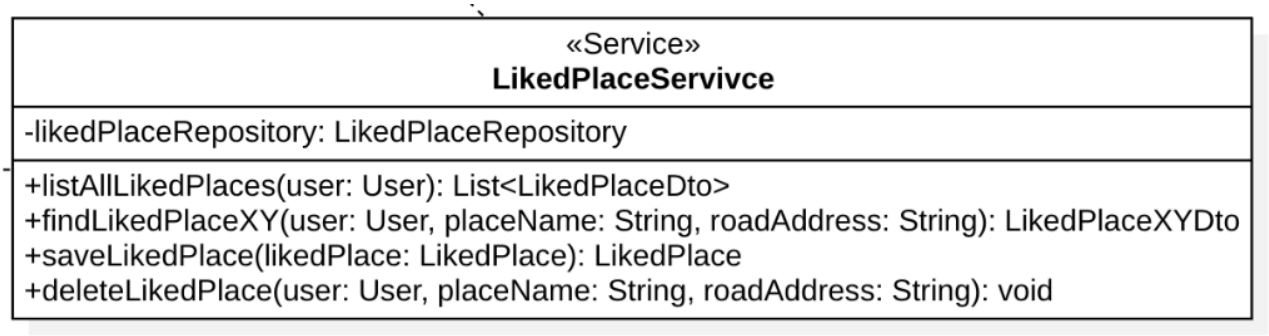
Attributes
userRepository: UserRepository : 사용자 저장소
Methods
createUser(userDto: UserDto, passwordEncoder: PasswordEncoder): User : 비밀번호 암호화 및 User 엔티티 생성 saveUser(user: User): User : 사용자 저장 loadUserByUsername(email: String): UserDetails : 사용자 정보 로딩 (Spring Security) validateDuplicateUser(user: User): void : 중복 사용자 검사 (private)
Description
UserService는 회원가입 및 사용자 인증을 위한 로직을 수행하며, Spring Security의 UserDetailsService를 구현하여 인증 기능도 제공합니다.

2.10 EventService



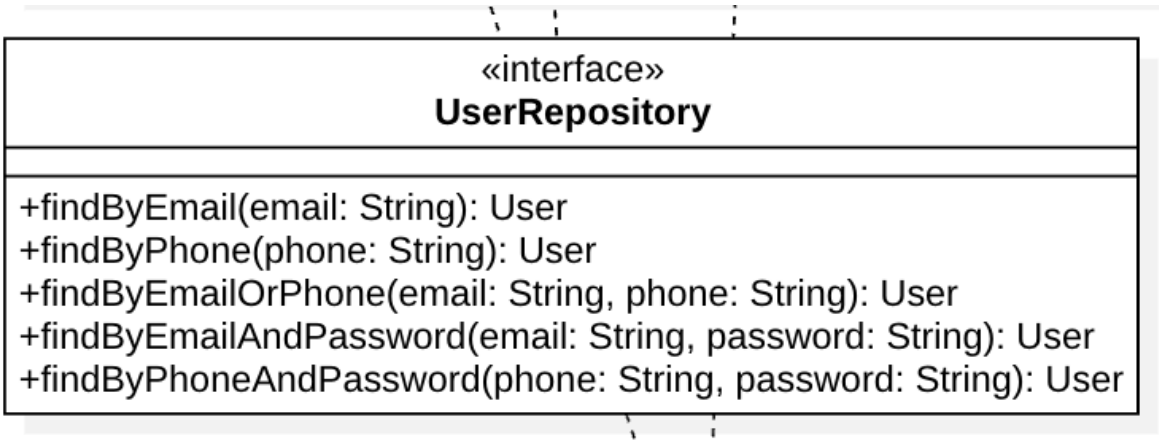
Attributes
eventRepository: EventRepository : 이벤트 저장소
Methods
listAllEventByEventType(user: User, eventType: String): List<EventDto> : 특정 타입의 이벤트 전체 조회
findYesterdayDiets(user: User, date: String): List<EventDto> : 어제 식단 이벤트 조회
findDayShopping(user: User, date: String): List<EventDto> : 하루 쇼핑 이벤트 조회
findMonthShopping(user: User, month: String): List<EventDto> : 월별 쇼핑 이벤트 조회
saveEvent(event: Event): Event : 이벤트 저장
deleteEvent(id: Long): void : 이벤트 삭제
Description
EventService는 사용자의 일정/식단/쇼핑/장소 등 다양한 유형의 이벤트 데이터를 CRUD 처리하는 핵심 서비스입니다.

2.11 LikedPlaceService



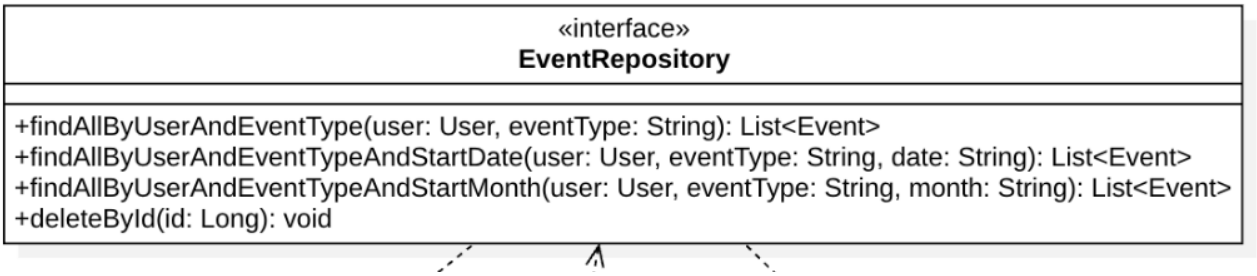
Attributes
likedPlaceRepository: LikedPlaceRepository : 즐겨찾기 장소 저장소
Methods
listAllLikedPlaces(user: User): List<LikedPlaceDto> : 즐겨찾기 전체 조회 findLikedPlaceXY(user: User, placeName: String, roadAddress: String): LikedPlaceXYDto : 특정 장소 좌표 조회 saveLikedPlace(likedPlace: LikedPlace): LikedPlace : 장소 저장 deleteLikedPlace(user: User, placeName: String, roadAddress: String): void : 즐겨찾기 삭제
Description
LikedPlaceService는 사용자가 저장한 즐겨찾기 장소의 조회, 추가, 삭제 등의 기능을 담당하는 서비스 클래스입니다.

2.12 UserRepository



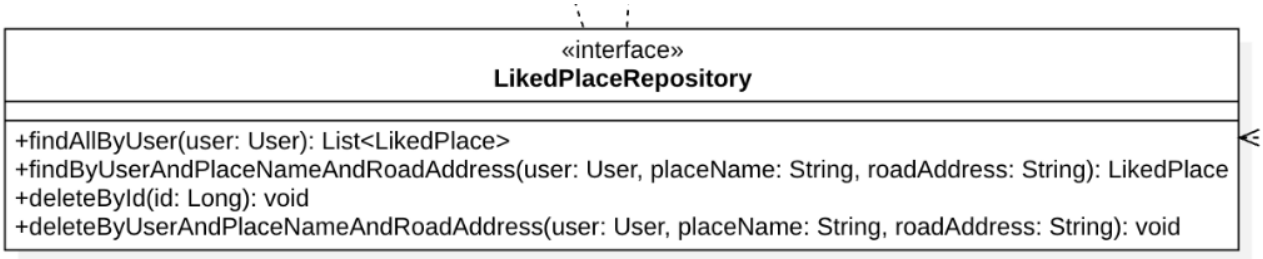
Methods
findByEmail(email: String): User : 이메일로 사용자 조회
findByPhone(phone: String): User : 전화번호로 사용자 조회
findByEmailOrPhone(email: String, phone: String): User : 이메일 또는 전화번호로 조회
findByEmailAndPassword(email: String, password: String): User : 이메일과 비밀번호로 조회
findByPhoneAndPassword(phone: String, password: String): User : 전화번호와 비밀번호로 조회
Description
UserRepository는 JPA를 통해 사용자(User) 엔티티를 이메일, 전화번호 등 다양한 조건으로 조회하는 인터페이스입니다.

2.13 EventRepository



Methods
findAllByUserAndEventType(user: User, eventType: String): List<Event> : 사용자와 이벤트 유형으로 조회 findAllByUserAndEventTypeAndStartDate(user: User, eventType: String, date: String): List<Event> : 날짜 기반 이벤트 조회 findAllByUserAndEventTypeAndStartMonth(user: User, eventType: String, month: String): List<Event> : 월별 이벤트 조회 deleteById(id: Long): void : ID 기반 이벤트 삭제
Description
EventRepository는 사용자의 이벤트 데이터를 조회/삭제할 수 있도록 복합 키 조건 (User, eventType, 날짜 등)에 맞춰 메서드를 정의한 JPA 인터페이스 입니다.

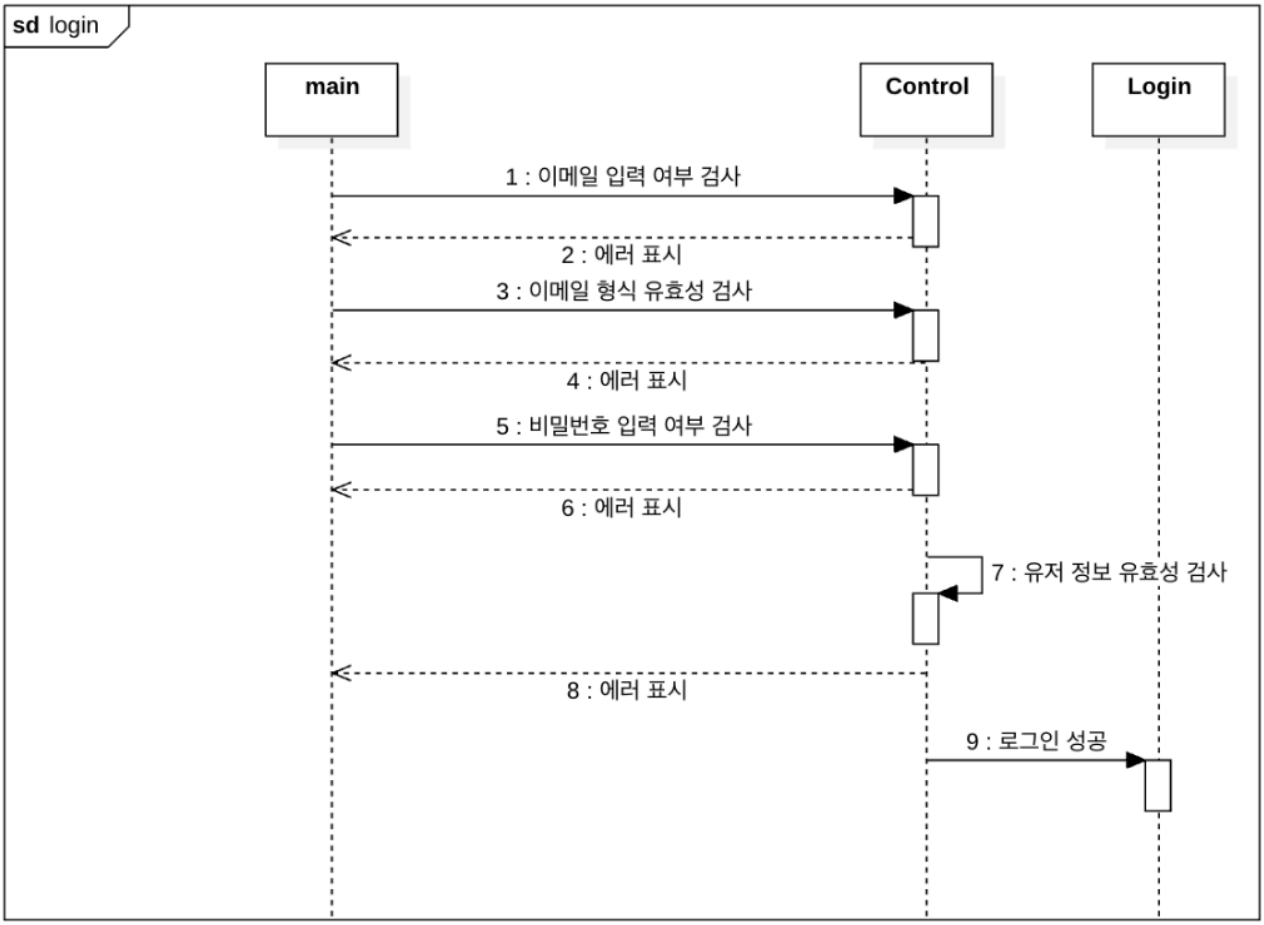
2.14 LikedPlaceRepository



Methods
findAllByUser(user: User): List<LikedPlace> : 특정 사용자의 즐겨찾기 전체 조회 findByUserAndPlaceNameAndRoadAddress(user: User, place: String, addr: String): LikedPlace : 특정 장소 단건 조회 deleteById(id: Long): void : ID 기반 삭제 deleteByUserAndPlaceNameAndRoadAddress(user: User, place: String, addr: String): void : 사용자+장소로 삭제
Description
LikedPlaceRepository는 즐겨찾기 장소(LikedPlace) 데이터를 사용자 및 장소 조건으로 조회하거나 삭제하는 기능을 제공합니다.

3. Sequence diagram

1) Login : 로그인



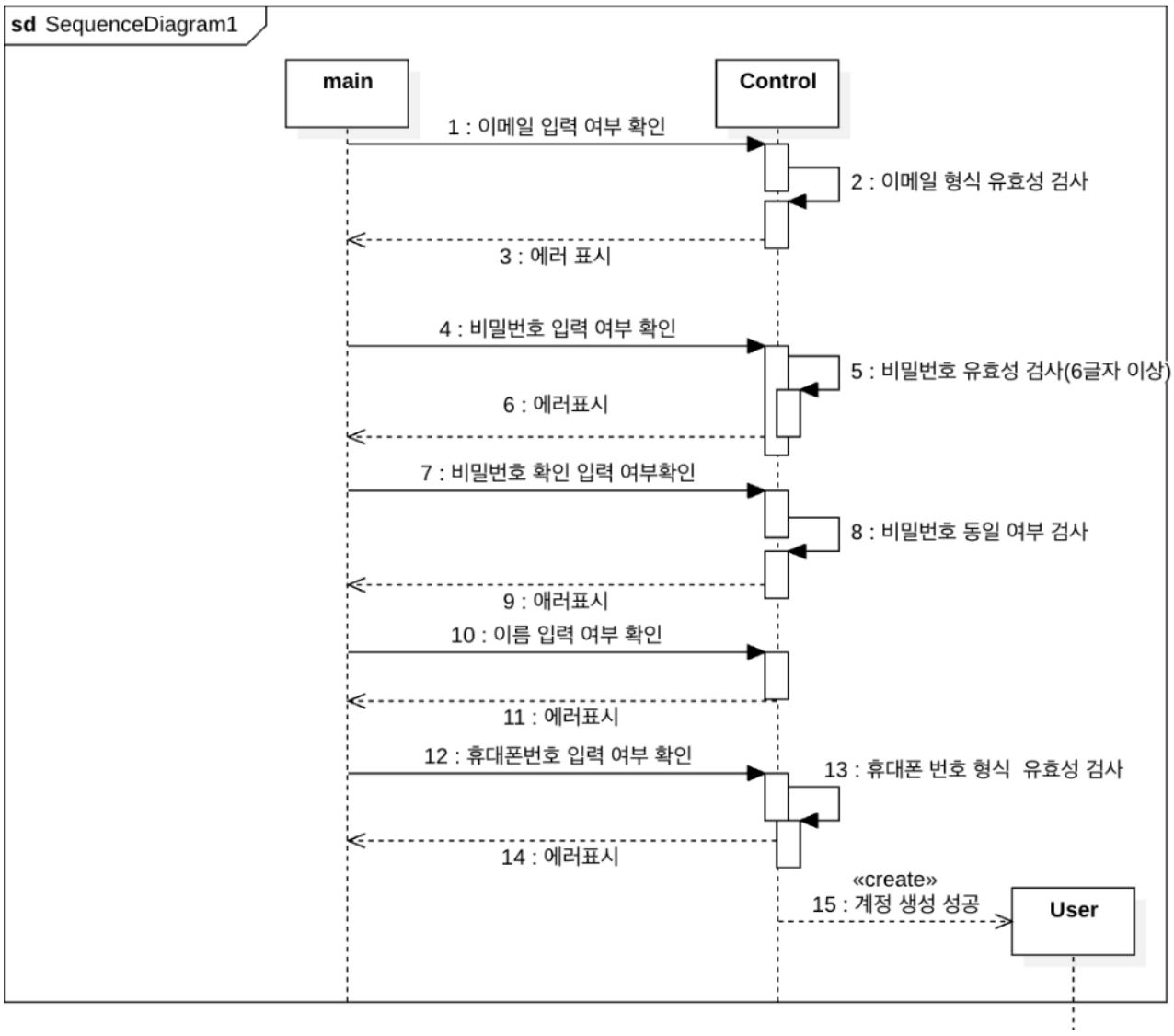
Login 시퀀스 다이어그램은 본 프로젝트인 공공데이터 기반 스마트 플래너에서 사용자가 로그인할 때 수행되는 전체 과정을 시각적으로 표현한 것이다. 사용자는 웹 브라우저에서 이메일과 비밀번호를 입력하며, 시스템은 이를 검증한 후 로그인 성공 여부를 판단하고 그 결과를 사용자에게 전달한다.

우선 사용자가 이메일을 입력하지 않았을 경우, 클라이언트 화면을 나타내는 main 객체는 Control 객체에 이메일 입력 여부를 검사하도록 요청한다. Control은 입력값이 비어 있는지를 확인하고, 비어 있을 경우 오류 메시지를 다시 main으로 반환하여 사용자에게 알린다. 이후 이메일 형식의 유효성을 검사하며, 정규식 등을 통해 올바른 이메일 구조가 아니라고 판단되면 동일하게 오류 메시지를 전달한다.

다음 단계에서는 비밀번호 입력 여부를 검사한다. 사용자가 비밀번호를 입력하지 않은 경우에도 Control에서 판단하여 오류 메시지를 사용자에게 반환한다. 이와 같이 입력 유효성 검사가 모두 통과된 후에는 Control이 Login 객체에 사용자 정보의 유효성을 검사하도록 요청한다. 이 과정에서는 내부적으로 UserRepository의 findByEmail 메서드나

UserService의 loadUserByUsername 메서드가 호출되어 입력된 사용자 정보와 실제 저장된 정보가 일치하는지를 확인한다.
사용자 정보가 일치하지 않으면 다시 main으로 오류 메시지가 전달되고, 로그인 실패로 처리된다. 반대로 정보가 일치하여 인증에 성공한 경우에는 로그인 성공 메시지가 반환되며, 이후 시스템은 사용자에게 홈 화면 등으로 이동시키는 응답을 제공한다. 실제로는 Spring Security와 연동되어 로그인 실패 시에는 "/login/error" 경로로 리디렉션되며, 로그인 성공 시에는 "/login?status=success"와 같은 응답 경로가 표시된다.

2) Register : 회원가입



해당 시퀀스 다이어그램은 공공데이터 기반 스마트 플래너 시스템에서 사용자가 회원가입을 진행할 때 수행되는 절차를 시각적으로 나타낸 것이다. 이 다이어그램은 사용자가 입력한 정보를 기반으로 서버 측에서 순차적으로 유효성 검사를 수행하고, 최종적으로 계정을 생성하는 전 과정을 보여준다.

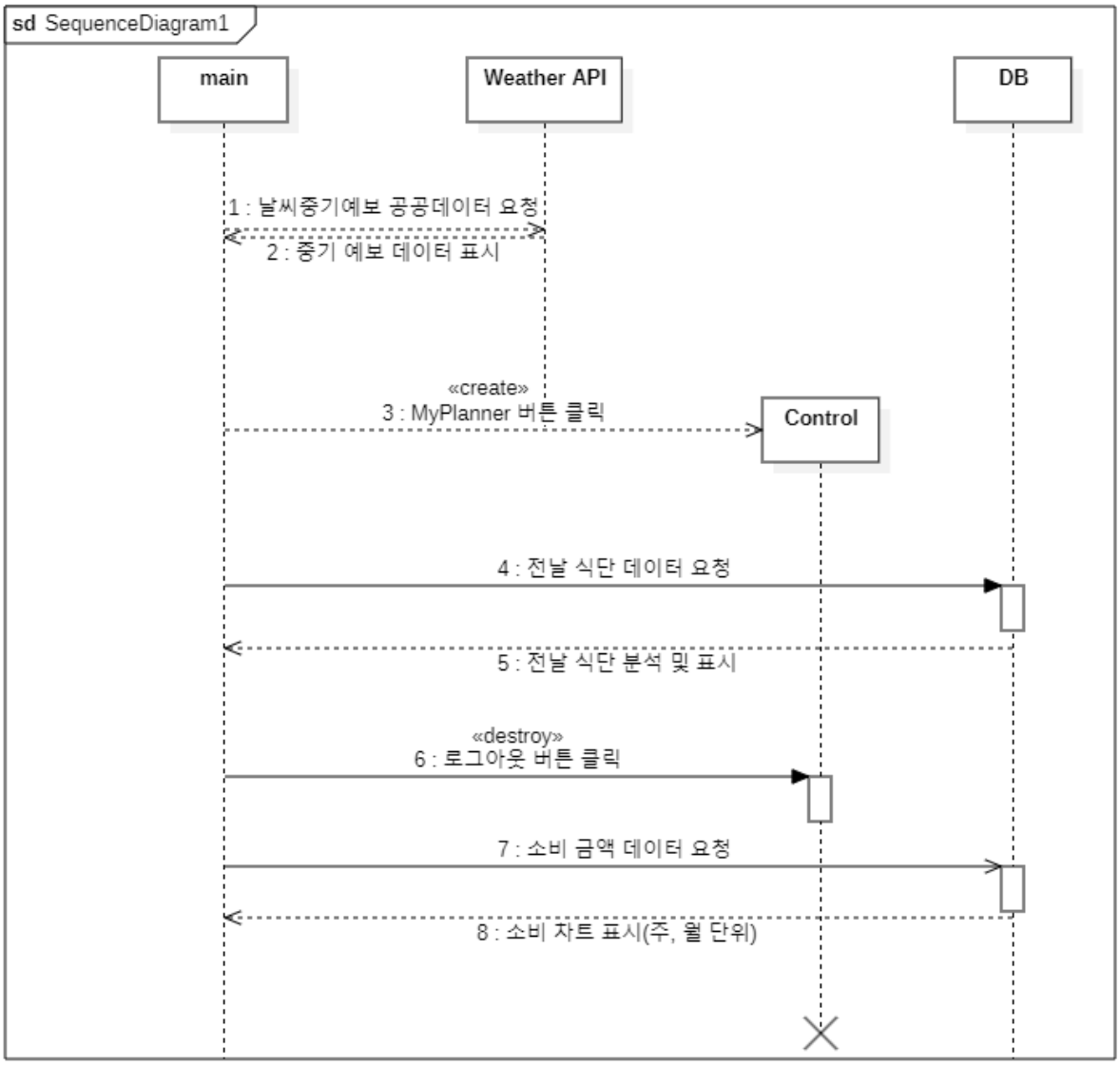
첫 번째 단계로, 사용자가 회원가입 폼에 이메일을 입력하면 main 객체는 Control 객체에 해당 이메일이 비어 있는지를 확인하도록 요청한다. 이어서 Control 객체는 이메일 형식이 올바른지를 검사하고, 조건에 부합하지 않을 경우 오류 메시지를 사용자 화면(main)으로 반환하여 입력 수정 유도를 수행한다.

이메일 검사가 완료되면 비밀번호 입력 여부가 검사된다. 비밀번호가 입력되지 않은 경우에도 에러 메시지가 반환되며, 입력된 경우에는 최소 6자 이상이라는 조건에 따라 유효성 검사가 수행된다. 조건을 충족하지 않으면 다시 에러가 표시된다. 그 다음 단계로는 비밀번호 확인 입력란이 비어 있는지 확인되며, 이후 원래 입력한 비밀번호와 동일한지 비교하여 일치 여부를 검사한다. 이 또한 불일치 시 오류 메시지로 응답한다.

비밀번호 항목을 통과한 후에는 이름 입력 여부 검사가 이어진다. 이름이 비어 있을 경우 사용자에게 입력을 유도하는 에러 메시지가 반환된다. 마지막으로 휴대전화번호 입력 여부를 확인하고, 형식이 유효한지를 검사하게 된다. 전화번호 형식은 일반적으로 하이픈(-)을 포함한 국내 휴대전화 번호 정규식을 기반으로 검증되며, 유효하지 않은 경우 에러가 표시된다.

이상의 모든 입력값이 정상적으로 확인되고 유효성 검사를 통과하면, Control 객체는 User 객체에 계정 생성을 요청하며 실제 사용자 정보가 저장된다. 다이어그램에서는 이 과정을 <<create>> 메시지를 통해 시각적으로 나타내었으며, 최종적으로 계정 생성 성공 응답이 반환된다.

3. userHome : 유저홈



이 시퀀스 다이어그램은 사용자가 애플리케이션을 이용하면서 발생하는 주요 기능 흐름을 시간 순서에 따라 설명하고 있다. 전체 흐름은 중기 예보 조회, MyPlanner 기능 실행, 식단 분석, 소비 분석, 로그아웃 처리의 다섯 가지 주요 과정으로 구성되어 있다. 처음에 main 객체는 외부의 Weather API에 날씨 중기예보 데이터를 요청한다. 이 요청은 시스템이 초기 구동되었을 때 자동으로 발생하는 동작이다. Weather API는 요청을 수신한 뒤 중기예보 데이터를 응답으로 반환하며, main은 받은 데이터를 화면에 표시한다. 사용자는 이 과정을 통해 별도의 조작 없이도 최신 예보 정보를 확인할 수 있다.

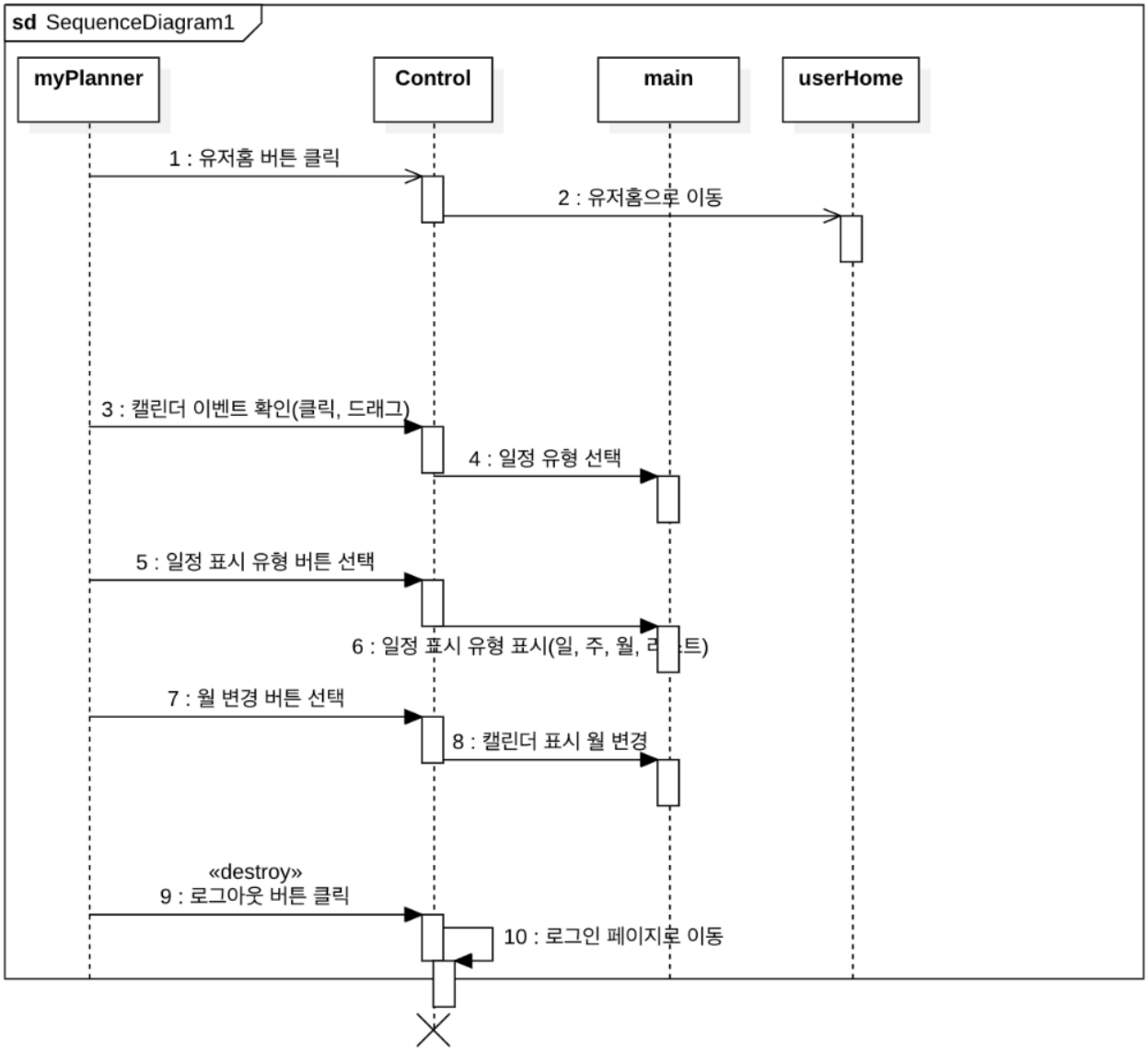
그 다음 단계에서는 사용자가 MyPlanner 버튼을 클릭하는 이벤트가 발생한다. 이 클릭 동작은 Control 객체를 생성하는 트리거가 되며, Control 객체는 이후 사용자 맞춤 기능을 처리하는 역할을 맡는다. 생성된 Control 객체는 DB에 전날 식단 데이터를 요

청한다. DB는 해당 요청에 응답하여 데이터를 반환하고, Control 객체는 반환받은 데이터를 기반으로 분석을 수행한 뒤, 그 결과를 사용자에게 시각적으로 표시한다. 표시되는 내용은 식단 구성이나 영양 정보 등으로 구성된다.

식단 분석 이후 사용자는 로그아웃 버튼을 클릭한다. 이 동작은 Control 객체를 소멸시키는 동작으로 연결되며, 사용자 세션이 종료되고 관련 기능이 더 이상 유지되지 않도록 처리된다. Control 객체가 사라진 뒤에도 main 객체는 DB에 소비 금액 데이터를 요청한다. 이 요청은 소비 내역을 주 단위 또는 월 단위로 집계하고 시각화하기 위한 용도로 이루어진다. DB는 요청에 대한 응답으로 소비 데이터를 제공하고, main은 이 데이터를 차트 형태로 구성하여 사용자에게 보여준다. 차트는 시간 흐름에 따른 소비 경향을 한눈에 확인할 수 있도록 구성된다.

이 다이어그램은 사용자의 서비스 이용 과정에서 발생하는 주요 기능들을 순차적으로 분리하고, 각 기능이 어떤 객체에 의해 처리되며 어떤 방식으로 데이터를 주고받는지를 명확하게 표현한다. 각 메시지 전달은 실제 시스템에서 발생하는 요청 및 응답 흐름과 일치하며, 사용자의 조작에 따라 시스템 내부의 컨트롤러와 외부 API, 데이터베이스가 어떻게 연동되는지를 단계별로 보여준다.

4. myPlanner : 나의플래너

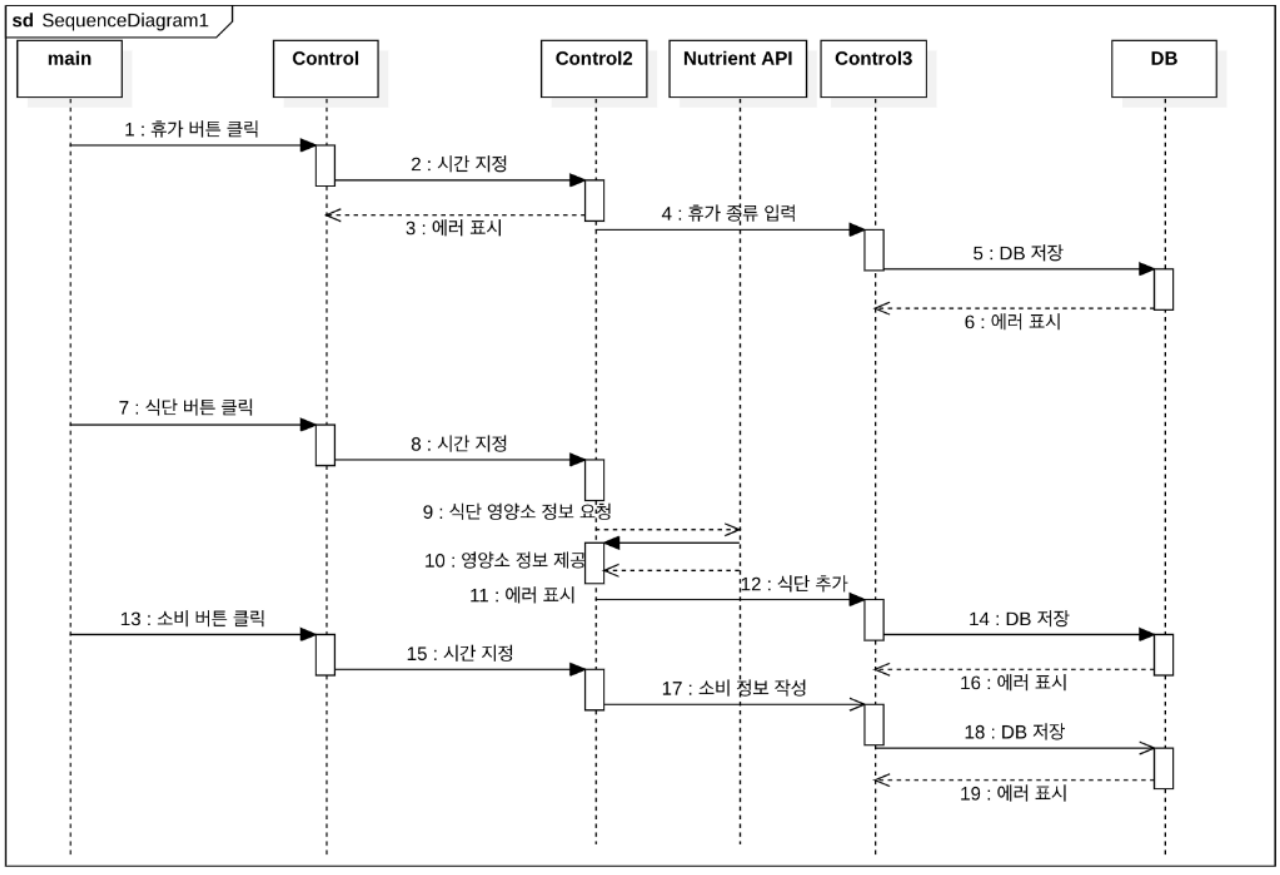


이 시퀀스 다이어그램은 사용자가 일정 관리 애플리케이션을 이용하면서 발생하는 기능 흐름을 시간 순서에 따라 설명하고 있다. 전체 흐름은 유저 홈으로 이동, 캘린더 이벤트 확인, 일정 유형 선택, 월 변경, 로그아웃 처리의 다섯 가지 주요 단계로 구성되어 있다. 사용자는 먼저 myPlanner 화면에서 유저홈 버튼을 클릭하며 상호작용을 시작한다. 이 클릭 이벤트는 Control 객체로 전달되고, Control은 main을 통해 userHome으로 화면을 전환한다. 유저는 이후 캘린더 화면에서 이벤트를 클릭하거나 드래그하며 일정을 확인하게 된다. 이 동작은 다시 Control을 통해 main으로 전달되어 일정 유형을 선택하는 과정으로 이어진다.

Control은 일정 표시 유형 버튼의 선택을 main 객체에 전달하고, main은 이에 따라 일간, 주간, 월간, 리스트 등의 형태로 일정을 시각적으로 표시한다. 이어서 사용자가 월 변경 버튼을 선택하면, 이 동작 역시 Control을 통해 main에 전달되며 캘린더의 월 표

시가 변경된다. 월 변경은 사용자 인터페이스 상에서 캘린더의 내용을 갱신하는 기능으로 동작한다.
일정 확인 및 월 변경이 끝난 후, 사용자는 로그아웃 버튼을 클릭한다. 이 동작은 Control 객체의 소멸로 이어지며, 이후 main은 로그인 페이지로의 전환을 처리한다. 로그아웃 이후 사용자 세션은 종료되며 애플리케이션은 초기 상태로 되돌아간다.
이 시퀀스 다이어그램은 사용자 일정 관리 기능에서 발생하는 주요 행위들을 시간 흐름에 따라 나열하고, 각 행위가 어떤 객체 간의 메시지 교환으로 구현되는지를 명확히 보여준다. 사용자 인터페이스 요소의 선택이 내부 컨트롤러와 시스템 로직으로 어떻게 연결되는지를 파악할 수 있으며, 전체적으로 단순한 일정 관리 기능의 흐름을 구조화된 방식으로 표현하고 있다.

5. 일정 유형 선택 1 : 휴가, 식단, 소비



이 시퀀스 다이어그램은 사용자가 애플리케이션에서 휴가 기록, 식단 등록, 소비 정보 입력과 같은 주요 기능을 사용하는 과정에서 시스템 내부 객체들이 상호작용하는 흐름을 시간 순서에 따라 설명하고 있다. 전체 흐름은 세 가지 주요 기능 흐름인 휴가 처리, 식단 등록, 소비 정보 입력으로 구성되어 있으며, 각 기능은 사용자의 버튼 클릭을 기점으로 시작되어 컨트롤 객체와 API, DB 사이에서의 메시지 교환을 통해 처리된다.

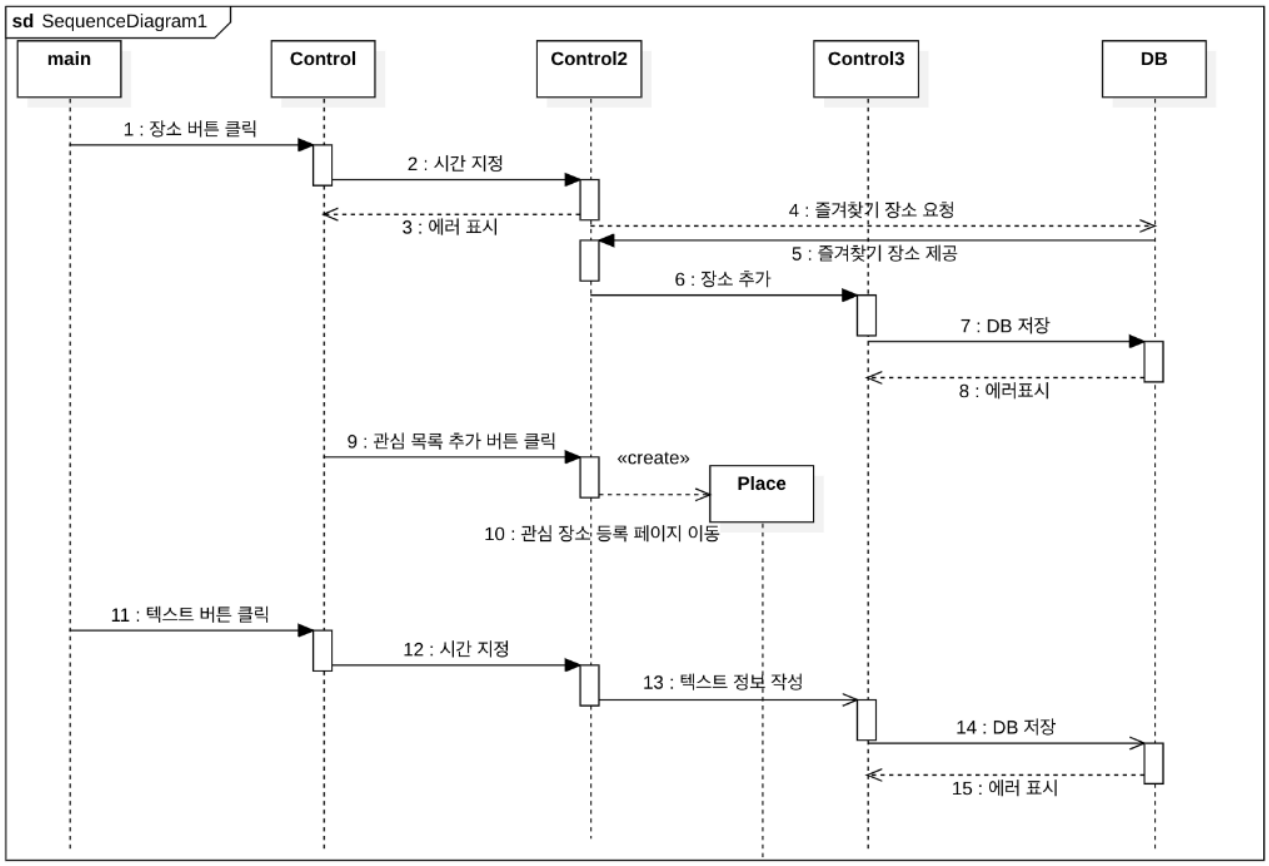
첫 번째 흐름은 휴가 버튼 클릭에서 시작된다. 사용자가 main에서 휴가 버튼을 클릭하면, 이 요청은 Control 객체로 전달되고, Control은 사용자의 입력에 따라 Control2 객체로 시간을 지정하는 요청을 보낸다. 이후 Control2는 휴가 종류를 입력하기 위해 Control3 객체로 메시지를 전달하고, Control3은 이를 DB에 저장한다. 그러나 저장 과정에서 문제가 발생할 경우 DB는 오류를 응답으로 반환하며, 이에 따라 Control3는 에러 메시지를 표시하고, 최종적으로 사용자는 시스템이 해당 오류를 인지했음을 확인할 수 있다. 또는 시간 지정 단계에서 문제가 발생할 경우 Control은 바로 에러를 표시하도록 설계되어 있다.

두 번째 흐름은 사용자가 식단 버튼을 클릭하는 이벤트로 시작된다. main에서 클릭된 요청은 Control로 전달되고, 다시 시간 지정 작업이 수행된다. 이후 Control은 식단에 포함된 음식의 영양소 정보를 확인하기 위해 외부의 Nutrient API로 정보를 요청하고, API는 해당 정보를 제공한다. 이 때 API 응답이 정상적으로 이루어지지 않거나 정보 해석 과정에서 문제가 생길 경우 에러가 표시될 수 있으며, 정상적으로 정보가 수신되었을 경우 Control3을 통해 식단 항목이 추가된다. Control3은 해당 정보를 DB에 저장하며, 저장 성공 여부에 따라 다시 에러 메시지를 표시하거나 성공적으로 완료되었음을 시스템적으로 처리하게 된다.

세 번째 흐름은 소비 버튼 클릭으로부터 시작된다. 사용자의 클릭 이벤트는 Control로 전달되고, 시간 지정 과정을 거친 후, Control3을 통해 소비 정보를 작성한다. 작성된 정보는 DB에 저장되며, 이 과정에서도 오류가 발생할 경우 사용자에게 에러가 표시된다. 이 흐름 역시 사용자의 간단한 입력을 기반으로 다양한 내부 모듈이 순차적으로 동작하는 구조를 갖는다.

이 시퀀스 다이어그램은 각 기능 처리에 있어 시간 지정, 외부 API 요청, 내부 데이터 저장이라는 공통된 구조를 따르며, 각 단계에서의 실패 시 에러 처리가 명시적으로 설계되어 있는 점이 특징이다. 특히 Nutrient API와 같은 외부 요소와의 연동, 그리고 여러 개의 Control 객체를 활용한 기능 분산 설계는 실제 시스템에서의 복잡성을 시각적으로 명확히 보여주며, 사용자의 상호작용이 어떻게 시스템 전반에 걸쳐 반응하게 되는지를 단계별로 파악할 수 있게 한다.

6. 일정 유정 선택 2 : 장소, 텍스트



이 시퀀스 다이어그램은 사용자가 애플리케이션 내에서 장소를 등록하거나 관심 목록에 추가하고, 텍스트 정보를 작성하는 일련의 상호작용을 시간 순서대로 설명하고 있다. 주요 흐름은 장소 등록, 관심 장소 추가, 텍스트 작성이라는 세 가지 기능을 중심으로 구성되어 있으며, 각 기능은 사용자 인터페이스 상의 버튼 클릭을 기점으로 시작되어 여러 컨트롤 객체와 데이터베이스 간의 메시지 교환을 통해 처리된다.

첫 번째 흐름은 사용자가 장소 버튼을 클릭하면서 시작된다. main 객체는 이 이벤트를 Control 객체로 전달하며, Control은 시간 지정 과정을 처리하기 위해 Control2로 요청을 보낸다. 이 과정에서 오류가 발생할 경우 Control은 사용자에게 에러 메시지를 표시한다. 이후 정상적으로 시간 지정이 완료되면 Control2는 Control3 객체를 통해 즐겨찾기 장소 요청 메시지를 전달하며, Control3는 이를 받아 DB에 요청하고 해당 정보를 조회한다. DB는 즐겨찾기 장소 정보를 응답으로 반환하고, Control3는 이 정보를 Control2를 통해 사용자에게 제공한다. 제공된 장소 정보는 시스템에 장소를 추가하는 동작으로 이어지고, 그 결과는 다시 DB에 저장된다. 이 과정에서 저장 중 문제가 발생하면 DB는 에러를 반환하게 되며, Control 객체는 사용자에게 에러 상태를 전달한다.

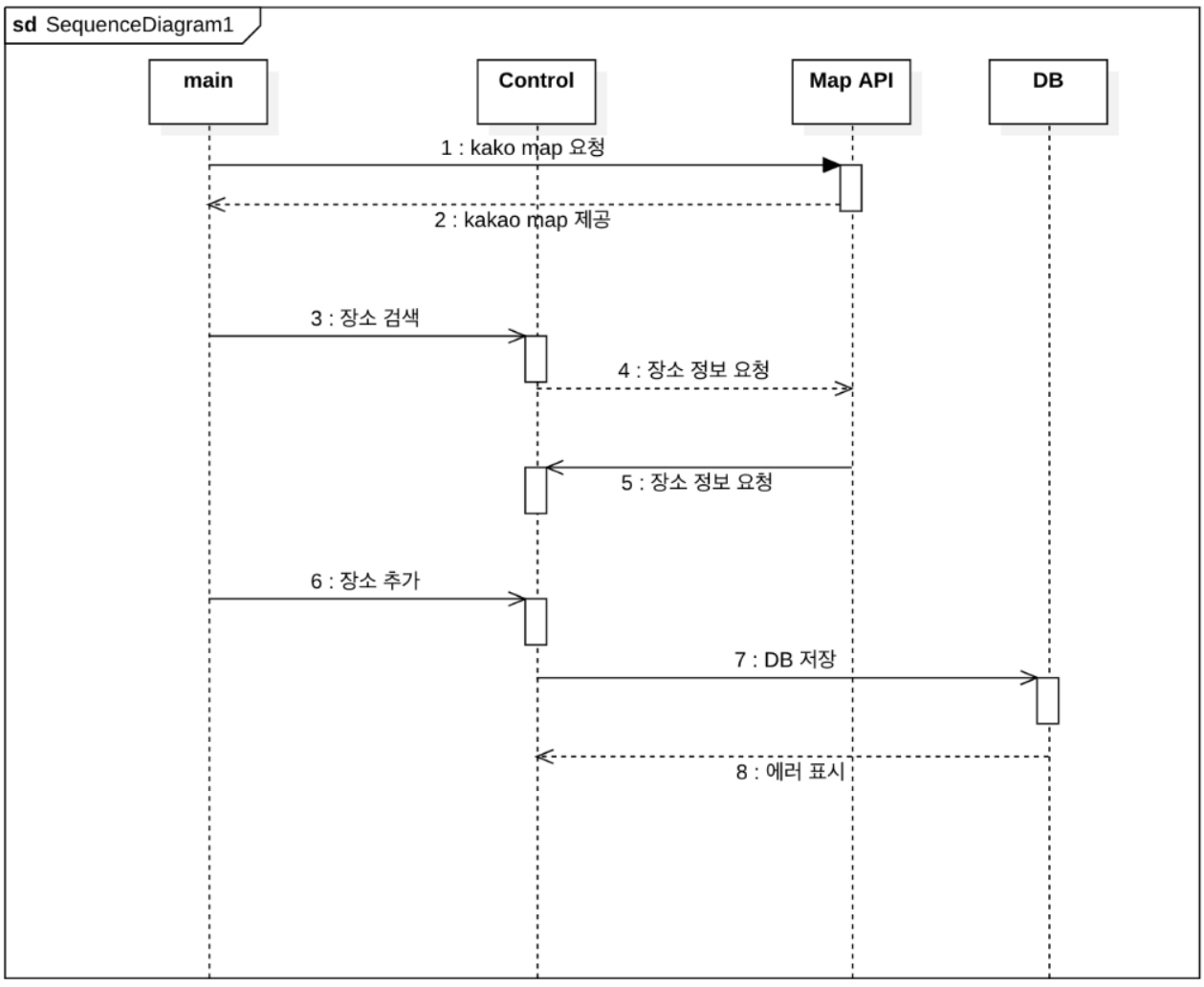
두 번째 흐름은 관심 목록에 장소를 추가하는 기능으로 구성된다. 사용자가 관심 목록 추가 버튼을 클릭하면, 해당 요청은 Control로 전달되고, Control은 새로운 Place 객체를 생성하며 관심 장소 등록 페이지로 이동시킨다. 이 과정은 사용자 맞춤 정보를 등

록하거나 수정하는 인터페이스로의 전환을 의미하며, 장소 정보를 직접 구성하는 절차로 이어질 수 있다.

세 번째 흐름은 텍스트 버튼 클릭 이벤트로부터 시작된다. 사용자가 main에서 텍스트 버튼을 클릭하면, Control 객체는 다시 시간 지정 절차를 실행하고, 이후 텍스트 정보를 작성한다. 작성된 텍스트 정보는 Control 객체를 통해 DB에 저장되며, 저장 처리 과정에서 오류가 발생할 경우 사용자에게 에러 메시지가 표시된다.

이 시퀀스 다이어그램은 사용자 입력을 기반으로 시스템이 장소 기반 데이터와 텍스트 데이터를 처리하는 과정을 구조화된 방식으로 보여주며, 각 단계마다 오류 발생 가능성과 그에 따른 예외 처리를 포함하여 설계되어 있다. 다양한 Control 객체를 통한 역할 분리와 객체 생성(예: Place), 외부 데이터 요청 및 저장 흐름이 명확히 드러나 있으며, 사용자의 기능 수행이 시스템 내부적으로 어떻게 구현되는지를 단계적으로 파악할 수 있게 한다.

7. 관심 장소 등록



이 시퀀스 다이어그램은 사용자가 애플리케이션을 통해 지도를 활용하여 장소를 검색

하고, 해당 장소 정보를 등록하는 일련의 과정을 시간 순서대로 설명하고 있다. 주요 객체는 main, Control, Map API, DB로 구성되어 있으며, 외부 지도 API와의 연동과 내부 데이터베이스 저장 흐름이 함께 나타난다.

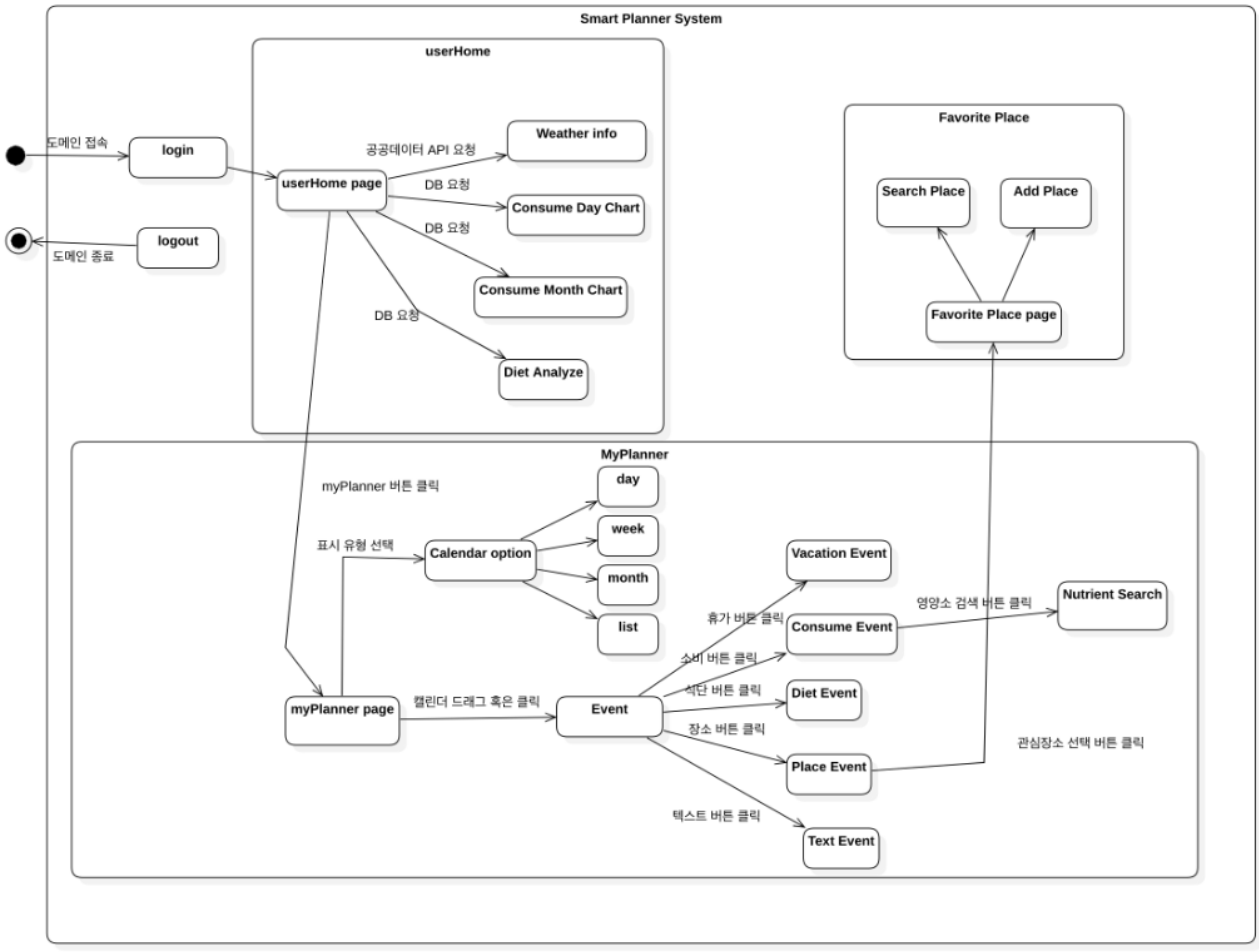
초기 단계에서 사용자는 main 인터페이스를 통해 Kakao 지도를 요청하게 된다. 이 요청은 Control 객체로 전달되며, Control은 외부 Map API에 kakao map 요청을 수행한다(1). Map API는 요청을 수신한 뒤 지도 데이터를 제공하고(2), Control은 해당 지도를 main 인터페이스에 전달하여 사용자에게 지도를 시각적으로 표시한다.

이후 사용자는 지도 상에서 특정 장소를 검색하고(3), 이 동작은 다시 Control 객체로 전달된다. Control은 검색된 장소 정보를 얻기 위해 Map API에 장소 정보 요청을 보낸다(4). Map API는 요청을 수신하고 해당 장소의 상세 정보를 반환한다(5). 이 정보는 사용자 화면에 표시되거나 장소 추가에 활용된다.

사용자가 선택한 장소를 저장하기 위해 '장소 추가' 동작을 수행하면(6), Control은 해당 장소 데이터를 DB에 저장한다(7). 저장 과정에서 문제가 발생할 경우, DB는 에러를 반환하고, Control은 이를 사용자에게 에러 메시지로 전달하여 처리 흐름을 종료한다(8).

이 시퀀스 다이어그램은 외부 지도 API와의 연동을 기반으로 사용자가 장소 정보를 검색하고 저장하는 흐름을 명확히 보여준다. 지도 데이터를 활용한 사용자 중심 기능을 중심으로, 시스템 내부의 컨트롤 객체와 외부 API, 데이터베이스 간의 역할과 상호작용이 구체적으로 시각화되어 있으며, 사용자의 명령이 시스템 내부 동작으로 어떻게 연결되는지를 이해하는 데 도움을 준다. 또한 각 단계에서 오류 처리 절차가 포함되어 있어 실제 서비스에서 발생할 수 있는 예외 상황까지 고려한 설계가 반영되어 있다.

4. State machine diagram



이 다이어그램은 "Smart Planner System"의 상태 흐름(state flow)을 시각적으로 설명하고 있으며, 사용자가 시스템에 접근한 이후 로그아웃하기까지의 전체 사용자 활동 흐름을 다룬다. 시스템은 크게 세 개의 주요 구성 요소로 나뉜다: userHome, MyPlanner, 그리고 Favorite Place.

사용자는 도메인 접속을 통해 시스템에 로그인하며, 이때 userHome page로 진입하게 된다. 해당 페이지는 공공데이터 API를 통해 날씨 정보를 요청하고, 데이터베이스를 통해 소비 일간 차트, 소비 월간 차트, 그리고 식단 분석 데이터를 각각 요청할 수 있다. 이는 사용자의 생활 패턴을 시각화하여 사용자 맞춤형 정보를 제공하려는 목적을 가진다.

사용자가 myPlanner 버튼을 클릭하면 myPlanner page로 진입하며, 이어서 Calendar option을 통해 캘린더의 표시 유형을 선택할 수 있다. 표시 유형은 day, week, month, list의 네 가지 방식으로 제공되며, 사용자의 필요에 따라 일정을 시각적으로 구성할 수 있도록 설계되었다. 캘린더에서 특정 일정을 드래그하거나 클릭하면 Event 상태로 전환되며, 이는 사용자가 일정을 추가하거나 편집할 수 있음을 의미한다.

이벤트 추가는 다양한 유형의 버튼 클릭을 통해 이루어진다. 예를 들어, 휴가 버튼을 클

릭하면 Vacation Event, 소비 버튼은 Consume Event, 식단 버튼은 Diet Event, 장소 버튼은 Place Event, 텍스트 버튼은 Text Event로 각각 전환된다. 특히 장소 버튼을 클릭했을 경우에는 관심장소 선택 버튼을 통해 Favorite Place page로 진입할 수 있으며, 여기서 관심 장소를 검색(Search Place)하거나 추가(Add Place)할 수 있다. 또한 식단 관련 이벤트 생성 시 Nutrient Search 기능을 통해 영양소 검색 기능도 활용할 수 있다.

전체 시스템은 사용자가 로그아웃할 때까지 다양한 기능을 통해 상호작용을 이어가며, 각 기능은 데이터 요청, 시각적 구성, 관심 정보 저장 및 관리라는 핵심 흐름을 중심으로 구성되어 있다. 이러한 상태 다이어그램은 사용자의 주요 활동을 흐름에 따라 체계적으로 정리하여, 시스템의 논리적 구조와 사용자 인터페이스 간의 관계를 명확히 보여준다.

5. Implementation requirements

본 시스템은 공공데이터를 기반으로 동작하는 스마트 플래너 웹 애플리케이션으로서, 사용자의 일정을 통합 관리하는 데에 목적이 있다. 사용자는 식단, 소비, 장소, 텍스트, 휴가 등의 다양한 유형의 이벤트를 생성하고 저장할 수 있으며, 이를 통해 개인 일정 및 활동 내역을 효율적으로 통합하여 관리할 수 있다. 또한 즐겨찾기 장소 등록 기능 및 공공 API와의 연동을 통해 확장된 사용자 경험을 제공한다. 본 시스템의 구현에는 다음과 같은 운영 환경 및 기술 요구사항이 필요하다.

우선, 백엔드 개발 환경은 Java 기반의 Spring Boot 프레임워크를 사용하였으며, 웹 서버는 내장형 Tomcat을 기본으로 구동된다. 데이터베이스는 JPA(Hibernate)를 통해 H2 또는 MySQL과 연동 가능하도록 설계되어 있으며, 시스템 운영 시 실제 서비스 환경에서는 MySQL을 권장한다. API 호출을 위한 HTTP 통신은 Java의 HttpURLConnection 클래스를 기반으로 구현되었으며, 공공데이터포털의 기상청 예보 API와 식품영양정보 API가 포함된다.

프론트엔드는 Thymeleaf 템플릿 엔진을 활용하여 서버 렌더링 기반의 화면을 구성하였으며, 사용자 인터페이스에서는 Bootstrap을 활용하여 반응형 레이아웃을 구성할 수 있도록 지원한다. 전체 프로젝트는 Gradle을 기반으로 빌드되며, 개발 툴은 IntelliJ IDEA 또는 VSCode 환경을 기준으로 한다. 자바 버전은 JDK 11 이상을 권장하며, Spring Boot는 2.7.x 버전대가 사용되었다.

보안 측면에서는 Spring Security를 통해 로그인 인증 기능을 구현하였고, 비밀번호는 BCrypt 알고리즘을 사용하여 안전하게 암호화되어 저장된다. 사용자 가입 시 이메일, 비밀번호, 비밀번호 확인, 이름, 전화번호 입력값은 모두 서버단에서 정규식을 통한 유효성 검사를 수행하며, 에러 메시지는 컨트롤러에서 모델로 반환되어 화면에 표시된다. 데이터 모델은 사용자(User), 이벤트(Event), 즐겨찾기 장소(LikedPlace)를 중심으로 구성되어 있으며, 이들은 클래스 다이어그램을 통해 구조화되었다. 이벤트는 다대일 연관관계를 통해 사용자와 연결되며, DTO 클래스를 통해 클라이언트와의 데이터 교환이 이루어진다. 이벤트 생성은 유형별로 구분되며, 각각에 대해 별도의 builder 메서드와 toEntity 변환 메서드가 구성되어 있다.

또한, 주요 기능별 흐름은 시퀀스 다이어그램을 통해 명확히 정리되었다. 로그인 절차의 경우 입력 유효성 검사를 포함한 인증 흐름이 정의되어 있으며, 회원가입 시에는 이메일, 비밀번호, 이름, 전화번호의 각 항목에 대해 입력 여부 및 형식 검사를 수행한 후 최종적으로 계정이 생성된다. 이와 같은 구조는 MVC 패턴을 기반으로 구현되어 있

어 유지보수 및 확장에 유리하며, 컨트롤러-서비스-리포지토리 구조가 명확히 분리되어 있다.

운영 환경 측면에서 본 시스템은 Java 11 이상, Spring Boot 2.7 이상, Gradle, MySQL 8.0 이상, HTML/CSS/JavaScript, 그리고 공공데이터 API 통신을 위한 외부 인터넷 연결이 필요하다. 개발 및 테스트는 macOS 및 Windows 10 기반의 환경에서 수행되었으며, 실제 서비스 배포 시에는 Linux 기반의 서버 환경을 권장한다.

결론적으로, 본 스마트 플래너 시스템은 공공데이터의 실시간 활용과 사용자 맞춤 일정을 결합한 유연하고 확장 가능한 플랫폼이다. 설계 단계에서부터 유스케이스 모델, 클래스 다이어그램, 시퀀스 다이어그램을 바탕으로 구조화되었으며, 실제 구현은 모듈화된 구조를 통해 높은 재사용성과 안정성을 확보하였다. 본 구현 환경 요건은 해당 시스템의 원활한 개발, 테스트 및 배포를 위한 기준으로서 정의된다.

항목	내용
시스템 개요	공공데이터 기반 스마트 플래너 (일정·식단·쇼핑·장소·텍스트 관리 및 즐겨찾기 기능 포함)
백엔드 환경	백엔드 환경: Java 11 이상, Spring Boot 2.7.x, Gradle 빌드, Spring Security 인증
프론트엔드	프론트엔드: Thymeleaf 템플릿 엔진, HTML5, CSS3, JavaScript, Bootstrap (반응형 지원)
데이터베이스	개발 환경에서는 H2 DB, 운영 환경에서는 MySQL 8.0 이상 (JPA + Hibernate ORM 연동)
빌드 및 도구	Gradle, IntelliJ IDEA 또는 VSCode, Git, Postman 등 개발 및 테스트 툴

보안 방식	Spring Security 기반 로그인 인증, BCrypt 암호화, 서버 유효성 검사 및 정규식 검사 적용
외부 API 연동	기상청 중기예보 API, 식품영양정보 API (공공데이터포털), 지도 기반 위치 API 사용
데이터 구조 모델	User, Event, LikedPlace 중심의 Entity 설계, 클래스 다이어그램 기반 구조화
DTO 처리 방식	EventDto, LikedPlaceDto 등을 통해 클라이언트와 데이터 송수신 및 변환 처리 수행
운영체제 환경	개발 환경은 macOS 및 Windows 10, 배포 환경은 Linux 기반 서버 권장

6. Glossary

본 문서에서 사용된 주요 용어들은 시스템의 구조와 기능을 명확하게 이해하는 데 필수적인 개념들이다. 각 용어는 프로그램 설계, 데이터 처리, API 연동 등 다양한 관점에서 정의되며, 아래에 그 의미를 구체적으로 설명한다.

User
시스템에 회원가입 및 로그인을 통해 접근하는 실제 사용자로, 이름, 이메일, 전화번호, 비밀번호 등의 개인 정보를 기반으로 식별된다. 사용자는 여러 개의 이벤트를 생성하거나 즐겨찾기 장소를 등록할 수 있으며, 해당 정보는 User 엔티티와 연관되어 저장된다.

Event
사용자의 활동이나 일정을 나타내는 핵심 정보 단위이다. Event는 유형(eventType)에 따라 텍스트(text), 식단(diet), 쇼핑(shopping), 장소(place), 휴가(vacation) 등으로 분류되며, 각각의 세부 항목(예: 음식 이름, 칼로리, 장소 좌표 등)을 포함한다. Event는 User와 다대일 관계로 연결되어 있어, 한 명의 사용자가 여러 개의 이벤트를 가질 수 있다.

LikedPlace
사용자가 자주 방문하거나 즐겨찾기로 등록한 장소 정보를 저장하는 객체이다. 장소 이름(placeName), 도로명 주소(roadAddress), 좌표 정보(x, y) 등으로 구성되며, User와 연관되어 있다. 지도 기반 기능에서 해당 데이터를 호출하여 시각화하거나 이벤트 작성 시 자동 완성 기능에 활용된다.

DTO (Data Transfer Object)
클라이언트와 서버 간에 데이터를 전송할 때 사용되는 전용 데이터 구조이다. 시스템 내에서는 EventDto, LikedPlaceDto, UserDto 등이 존재하며, Entity 객체의 필드를 클라이언트와의 통신 목적에 맞게 가공하거나 필터링하여 전달한다. DTO는 사용자 입력 값을 수집하거나 백엔드에서 반환할 데이터를 담는 데에 사용된다.

Controller
사용자의 요청을 받아 처리하고, 서비스 계층과 연결하여 적절한 로직을 수행하는 역할을 한다. 예를 들어, LoginController는 로그인 요청을 처리하며, MyPlannerController는 이벤트와 즐겨찾기 관련 기능을 담당한다. 각 컨트롤러는 URL 매핑과 응답 반환을 포함하는 Spring MVC의 핵심 구성 요소이다.

Service
비즈니스 로직을 수행하는 계층으로, 컨트롤러로부터 요청을 받아 실제 처리를 담당한다.

예를 들어, EventService는 이벤트 저장, 조회, 삭제 등의 기능을 제공하며, LikedPlaceService는 즐겨찾기 장소 관련 로직을 처리한다. 서비스 계층은 DB 접근과 뷰 렌더링을 분리하여 유지보수성을 높인다.

Repository

데이터베이스와 직접 통신하는 계층으로, JPA를 통해 테이블과 연동된 CRUD 작업을 수행한다. 예를 들어 EventRepository, UserRepository, LikedPlaceRepository는 각각 이벤트, 사용자, 즐겨찾기 데이터를 관리하는 인터페이스이다.

API (Application Programming Interface)

외부 시스템과의 데이터 통신을 위한 인터페이스이다. 본 시스템에서는 기상청 예보 API, 식품영양정보 API, 지도 API 등이 사용되며, HTTP 프로토콜을 통해 요청과 응답을 주고받는다. 공공데이터포털에서 제공하는 API는 JSON 형식으로 데이터를 반환하며, 이를 파싱하여 사용자의 일정과 연계한다.

Sequence Diagram

시스템 내 객체들 간의 메시지 흐름과 순서를 시각적으로 표현한 다이어그램이다. 본 문서에서는 로그인 절차, 회원가입 절차, 이벤트 생성 과정을 시퀀스 다이어그램으로 제시하였으며, 각 메시지는 시간의 흐름에 따라 상호작용을 설명한다.

Class Diagram

시스템 내 클래스 간의 관계, 속성, 메서드를 정리한 정적 구조 다이어그램이다. 본 프로젝트에서는 User, Event, LikedPlace 등의 클래스가 Entity로 정의되며, 각각의 DTO, Service, Controller와 함께 연계되어 전체 구조가 설계된다. 이는 코드 구조를 명확하게 시각화하고 유지보수 및 확장에 유리한 설계를 가능하게 한다.

7. References

본 시스템의 설계, 구현 및 보고서 작성 과정에서 다음의 자료들을 참고하였다. 각 자료는 Java 및 Spring Boot 기반 웹 애플리케이션 개발, 공공데이터 활용, 데이터베이스 설계, UML 모델링, 사용자 인증 구현 등에 필요한 기술적 기반을 제공하였다.

.1. 공공데이터포털 (www.data.go.kr): 기상청 중기예보 API와 식품영양정보 API에 대한 기술 문서와 샘플 요청 URL을 참고하여 외부 API 연동 기능을 구현하였다.

2. Spring 공식 문서 (https://docs.spring.io): Spring Boot, Spring MVC, Spring Security, Spring Data JPA와 관련된 설정 및 구성 방식, 인터페이스 설계에 대한 레퍼런스를 기반으로 개발을 진행하였다.

3. Baeldung 온라인 튜토리얼 (www.baeldung.com): 로그인 인증 처리(Spring Security), 컨트롤러와 REST 컨트롤러의 차이, DTO 사용 방식 등에 대한 실용 예제와 설명을 참고하였다.

4. Christian Bauer, Gavin King, "Java Persistence with Hibernate, 2nd Edition": ORM 기반의 데이터 매핑과 복합 조건 조회, 엔티티 설계 방식에 대한 이론적 근거로 활용하였다.

5. GitHub 및 Stack Overflow: 실제 구현 중 발생한 오류 해결, 코드 예제, API 파싱 방법, 시큐리티 설정 문제 등을 해결하기 위해 다양한 개발자 커뮤니티의 질의응답을 참고하였다.'

6. Laurentiu Spilca, "Spring Security In Action": 사용자 인증 구조, UserDetailsService 구현, 비밀번호 암호화 방식 등의 보안 처리 방식 설계 시 참고하였다.

7. Oracle Java Documentation (https://docs.oracle.com): Java 11 문법, HttpURLConnection 클래스 사용법, 표준 API 구조에 대한 이해를 위해 참조하였다