Smart Energy Management System

Software Design Specification

소프트웨어공학개론

Team 9

2019311270 김서정

2019311188 김주원

2017312334 김현중

2016311561 송유호

2019310940 이예송

2016312429 임형진

2022.05.15

목차

표 목차

그림 목차

1. Preface

챕터 1에는 이 문서의 독자들을 위한 간략한 정보, 독자층, 범위, 목적 및 문서 구조가 포함되어 있다.

1.1. Readership

본 Software Design Specification(SDS) 문서는 각각 다양한 하위 챕터가 있는 10개의 챕터로 나뉜다. SDS 문서의 구조는 이 SDS의 [1.4. Document Structure] 하위 섹션에 기술된 바에 따라 찾을 수 있다. 이 문서의 주요독자는 Team9이다. 또한 소프트웨어공학개론 수업의 교수, 조교, 팀원 및 모든 관련 이해 관계자도 주요 독자가 될 수 있다.

1.2. Scope

본 SDS 문서는 에너지 절약을 도와주는 Smart Energy Management System을 구현하는데 사용되는 디자인의 정의로 소프트웨어공학(Software Engineering)과 소프트웨어품질공학(Software Quality Engineering)에서 사용된다.

1.3. Objective

본 SDS 문서의 주요 목적은 Smart Energy Management System의 기술적 디자인 측면에 대한 설명을 제공하는 것이다. 개발자들은 이 문서에 설명된 디자인에 따라 소프트웨어를 구현해야 하고 이해관계자들은 이 문서를 사용하여 프로젝트가 요구사항을 충족하는지 확인할 수 있다. 이 문서는 Smart Energy Management System 구현을 위한 소프트웨어 아키텍처(Software Architecture) 및 소프트웨어 디자인 결정(Software design decision)에 대해 기술한다. 또한, 시스템의 다양한 측면을 설명하기 위해 시스템의 아키텍처 개요를 제공하고 소프트웨어 요구사항 사양(SRS) 문서에서 논의된 여러 모듈들의 구조와 디자인을 더욱 구체화하며 use case들을 다이어그램들을 통해 보여준다. 이 문서의 대상 독자는 Smart Energy Management System 프로젝트의 이해관계자, 개발자, 디자이너 및 소프트웨어 테스터이지만 경우에 따라 더욱 늘어날 수 있다.

1.4. Document Structure

(1) Preface: 해당 챕터에서는 문서의 독자층, 적용 범주, 목적에 대해 기술하고 문서의 전체적 구조와 각 장에 대한 설명을 통해 전체적인 윤곽을 보여준다.

(2) Introduction: 해당 챕터에서는 문서에 사용된 여러 다이어그램과 툴에 대한 정의와 설명을 제공하고 본 문서를 이해하는데 필요할 수 있는 범위와 참고한 레퍼런스를 기술한다.

(3) System Architecture – Overall: context diagram, sequence diagram, use case diagram 을 사용하여 시스템의 전체 아키텍처를 표현한다.

(4) System Architecture – Frontend: class diagram, sequence diagram 을 사용하여 frontend 의 아키텍처를 표현한다.

(5) System Architecture – Backend: class diagram, sequence diagram 을 사용하여 backend 의 아키텍처를 표현한다.

(6) Protocol Design: subsystem 간의 상호작용에 사용되는 프로토콜의 구조와 interface 를 정의하는 프로토콜에 대해 기술한다.

(7) Database Design: Smart energy management system 의 데이터 구조에 대해 설명하고 해당 데이터 구조가 데이터베이스에서 표현되는 방식에 대해 기술한다.

(8) Testing Plan: 해당 챕터에서는 시스템이 누락된 사항이나 오류 없이 요구사항에 맞게 잘 작동하는지 확인하는 방법이 기술된다. 3 가지 테스트(development testing, release testing, user testing) 계획에 대해 설명한다.

(9) Developing Plan: 해당 챕터에서는 개발 환경 및 개발 툴에 대해 설명하고 제한사항, 가정사항 및 의존성에 대해 기술한다.

(10) Supporting Information: 해당 챕터에서는 본 문서의 기준과 문서 이력에 대해 기술한다.

2. Introduction

2.1. Objectives

이 챕터에서는 본 프로젝트의 디자인 단계에서 사용한 여러가지 툴과 다이어그램에 대해 설명한다. 또한, 이 프로젝트의 범위 및 참조에 대해 기술한다.

2.2. Applied Diagrams

2.2.1. UML

UML는 통합 모델링 언어(Unified Modeling Language)의 약자로 요구사항 분석, 시스템 설계, 시스템 구현과 같은 시스템 개발 과정에서 개발자 간의 의사소통이 순조롭게 진행되도록 하기 위한 표준화된 모델링 언어이다. UML은 모델링에 대한 표현력이 강하고 모순이 상대적으로 적은 표기법을 가졌기 때문에 개발자 간의 의사소통 불일치를 해소할 수 있고 개발하려는 시스템 크기에 상관없이 모두 적용 가능하다. UML은 use case diagram, class diagram을 포함한 여러가지 다이어그램을 기반으로 객체지향 소프트웨어를 개발하기 위한 풍부한 분석 및 설계 장치를 제공한다.

2.2.2. Use Case Diagram

Use Case Diagram은 시스템과 사용자의 가능한 상호작용을 시각적으로 표현한 것으로 시스템에서 제공한 기능 단위를 설명한다. Use Case Diagram의 구성요소는 시스템(system), 액터(actor), 유스케이스(usecase), 관계(relationship)이다. 본 diagram은 개발 팀이 시스템의 기능적 요구 사항들을 시각화 하는데 목적을 두고 있다. 시스템의 고급 기능과 개발 범위를 설명하는데 사용되며 이해 관계자에게 보다 단순화된 방식으로 시스템의 의도를 전달한다.

2.2.3. Class Diagram

Class Diagram은 사람, 제품, 데이터와 같은 엔터티(entity)들이 서로 어떻게 관계를 주고 받고 있는지를 나타내는 일종의 정적 구조 다이어그램이다. 클래스들은 서로 연관관계, 집합 연간관계, 상속관계와 같은 다양한 관계를 가질 수 있다. Class diagram은 클래스 간의 정적인 협력관계를 정의함으로써 시스템 이해를 돕고 처음부터 끝까지 일관된 형식으로 소프트웨어 시스템을 분석, 설계하는 방식을 제공한다.

2.2.4. Sequence Diagram

Sequence Diagram은 시스템의 동적인 면을 나타내는 대표적인 다이어그램이다. 기기의 동작순서와 상호관련을 이해하기 쉽게 전개하여 표시한 동작설명도로써 시간 순서로 배열된 프로세스 상호 작용을 보여준다. Sequence diagram의 구성요소는 액터(actor), 객체(object), 메시지(message), 생명선(lifeline), 활성박스(activation box)이다. 본 diagram은 현재 존재하는 시스템이 어떠하게 움직이고 있는지를 잘 표현해주고 런타임 중일 때의 시스템의 흐름을 한 눈에 보고자 할 때 용이하다.

2.2.5. Context Diagram

Context Diagram은 시스템 또는 시스템의 일부와 환경 간의 경계를 정의 함으로써 시스템과 상호 작용하는 엔터티를 보여준다. Context Diagram은 시스템 전체와 외부 요인의 입력 및 출력을 보여주기 때문에 시스템에 대한 이해를 도울 수 있다. 본 diagram은 일반적으로 요구 사항 문서에 포함된다.

2.2.6. Entity Relationship Diagram

Entity Relationship Diagram은 엔터티(entity) 및 엔터티들의 관계를 네트워크 형태의 구조로 나타낸 다이어그램이다. Entity Relationship Diagram의 구성요소는 엔터티(entity), 속성(attribute), 관계(relationship)를 기본으로 한다. 본 diagram은 주로 데이터 베이스 모델링, 소프트웨어 엔지니어링 영역에서 사용되며 데이터 구조를 쉽게 파악할 수 있다는 장점을 가지고 있다.

2.2.7. Enhanced Entity Relationship Diagram

Enhanced Entity Relationship Diagram은 Entity Relationship Diagram을 기반으로 하여 일반화(여러 개체의 공통적인 특징을 상위 클래스 개체로 일반화), 세분화(상위 개체를 하위 개체로 나눌 수 있음), 집계(두 항목 간의 관계를 단일 항목으로 취급) 개념을 추가하였다.

2.3. Applied Tools

2.3.1. Microsoft Word

Microsoft Word는 본 프로젝트의 명세서들을 작성하는데 사용되었다. 세계 표준 워드프로세서이자 다양한 문서 편집 환경에서 가장 많이 사용되는 툴이기 때문에 문서작업 시에 Team9는 Microsoft Word를 기본적으로 사용한다.

2.3.2. Microsoft PowerPoint

Microsoft Powerpoint는 주로 발표자료를 제작하는데 사용되지만 여러 그리기 기능을 지원하고 있기때문에 본 문서의 일부 diagram과 그림을 작성하는데 사용하였다. 본 문서를 작성하는데 사용된 툴인 Microsoft word와 같은 포맷을 지원하여 높은 호환성을 갖는다.

2.3.3. draw.io

Draw.io는 웹 기반 diagram 소프트웨어이다. UML을 통한 diagram 작성을 지원하고 있기 때문에 본 문서의 diagram 중 일부를 작성하는데 사용하였다. 또한, 오프라인 모드로 사용할 수도 있으

며 다양한 저장공간(구글 드라이브, 깃헙, 드롭박스 등)를 지원하고 있으며 이를 통해 백업 및 동기화할 수 있다.

2.4. Project Scope

Smart Energy Management는 심각한 환경문제를 발생시키는 온실가스 배출에 큰 비중을 차지하고 있지만 에너지 절감 노력은 미미한 건물 부문에서의 에너지 절약을 도와 주기 위해 도입된 프로젝트이다. 이 프로젝트를 통해 사용자들의 에너지 절약을 도와주어 에너지 사용량 감소에 따른 비용 절감과 환경 보호 효과를 거둘 수 있다. iOS와 안드로이드 환경에서 실행되며 지원하는 대표적인 기능은 다음과 같다.

- 기기별/시간별 에너지 소비량 확인

- 스케줄러와 GPS를 이용한 사용자 기반 서비스를 통해 사용자 맞춤형 에너지 관리 시스템 제공

- 에너지 낭비 행동을 할 경우 이에 대한 경고 알림을 주고 사용자 에너지 소비형태에 따른 사용자별 에너지 절약을 위한 유용한 정보 제공 및 제안

- 에너지 절약 행동 실천을 위한 조건 및 조건 별 에너지 절약 행동 설정

- 조건 충족 시 설정된 작업 실행 또는 알림 보냄

2.5. References

- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library (http://ieeexplore.ieee.org/Xplore/guesthome.jsp)

- Skkuse (https://github.com/skkuse)

- Software Engineering 10th Edition, Ian Sommerville

- 네이버 지식백과 (https://terms.naver.com/)

- 위키피디아 (https://en.wikipedia.org/wiki/Main_Page)

3. System Architecture Overall

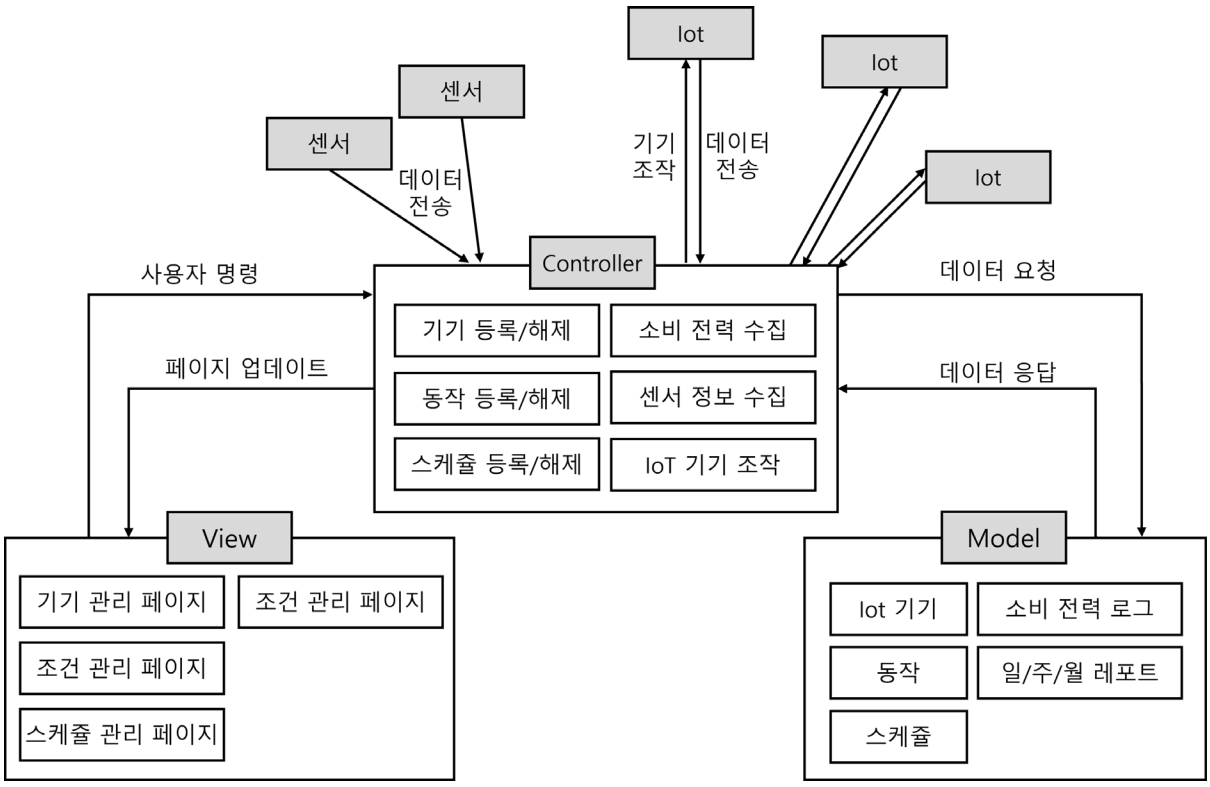
3.1. Objective

이 챕터에서는 frontend와 backend로 구성된 전체적인 시스템 구조를 기술한다.

3.2. System Organization

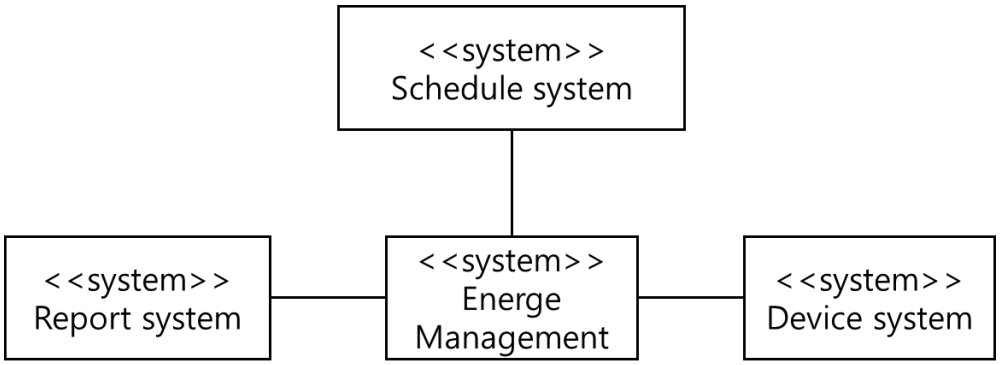
이 서비스는 Model-View-Controller에서 약간 수정된 구조로 설계되었다. View 에서는 사용자의 요청을 받아서 controller 에게 넘겨주고, 받은 정보를 사용자에게 보여준다. Controller 에서는 기본적인 동작과 view 로부터 오는 사용자의 명령을 보고 model 에게 데이터를 요청해 view 에게 넘겨준다. Model 은 controller 로부터 오는 요청에 맞는 데이터를 전달한다.

여기에 추가로 controller 에서는 등록한 IoT 나 센서 기기와 서로 통신하면서 데이터를 받거나 조작한다.



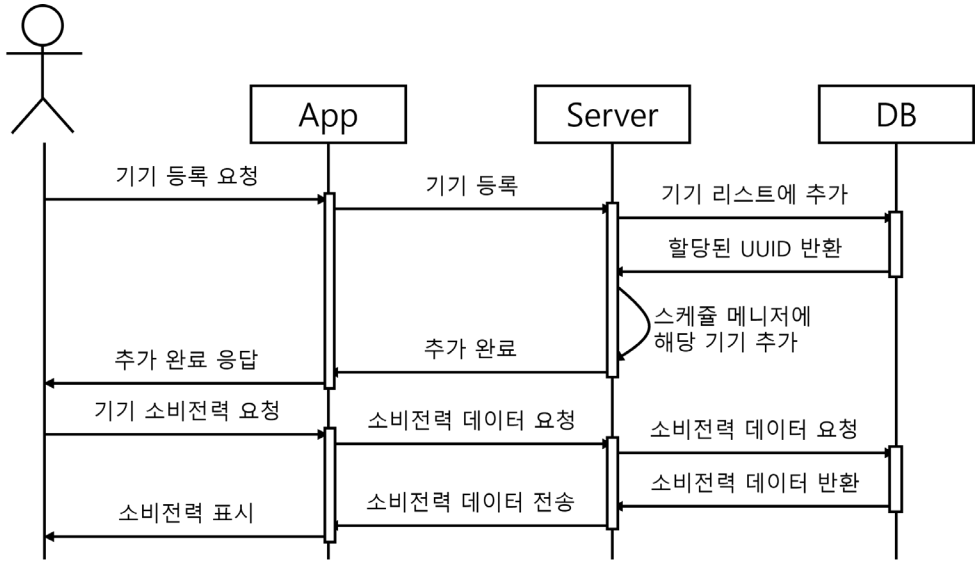
[그림 1] System architecture

3.2.1. Context Diagram



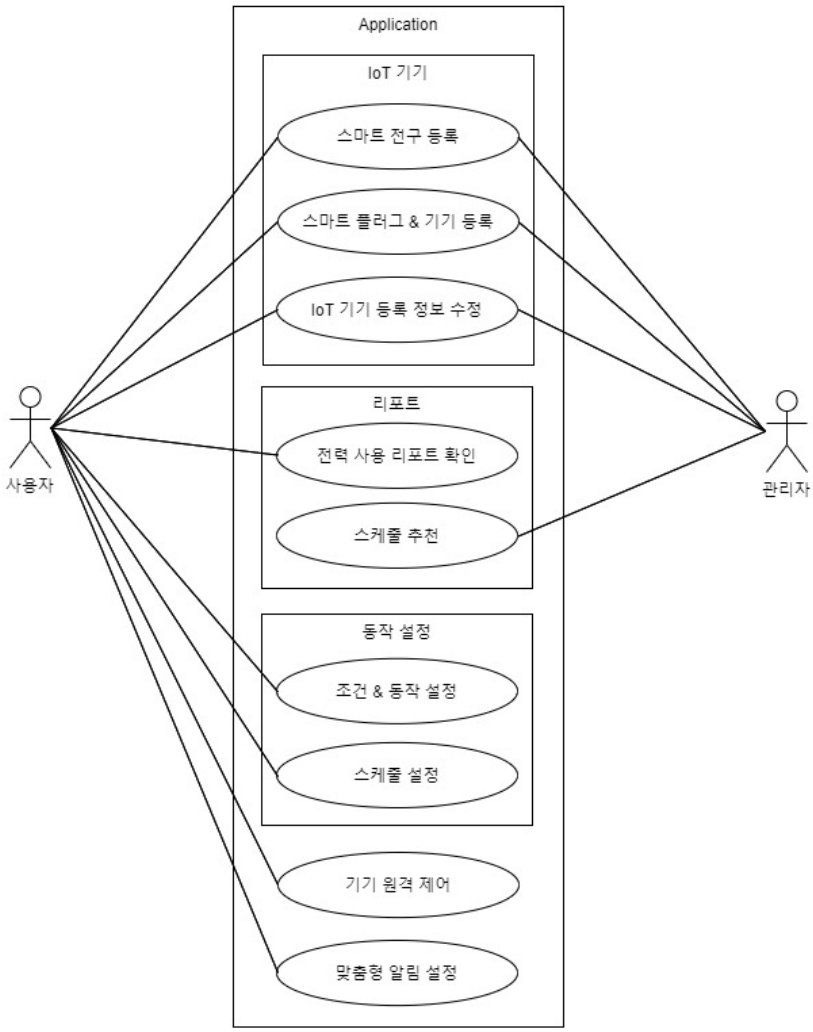
[그림 2] Context diagram

3.2.2. Sequence Diagram



[그림 3] Sequence diagram

3.2.3. Use Case Diagram



[그림 4] Use case diagram

4. System Architecture Frontend

4.1. Objectives

System Architecture에서 사용자 인터페이스에 해당하는 Frontend 시스템을 이루는 Component 들의 구성을 Class Diagram과 Sequence Diagram으로 도식화 및 설명한다.

4.2. Subcomponents

4.2.1. 로그인 및 유저 정보 등록 / 수정

4.2.1.1. Attributes

해당 profile object 가 가진 attribute 는 다음과 같다.

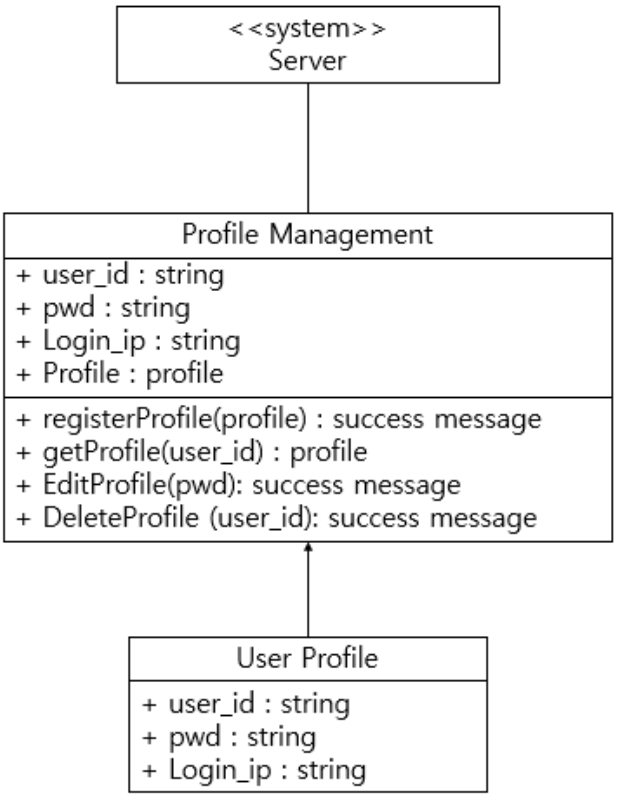
- user_id : 로그인에 사용되는 사용자의 ID 이다.
- pwd : 로그인에 사용되는 사용자의 비밀번호이다.
- Login_IP : 로그인시 사용자를 확인하기 위한 IP 주소이다.
- Profile : 로그인 정보를 수정할 때 가져오는 사용자 정보이다.

4.2.1.2. Methods

해당 Profile Object에서의 method는 다음과 같다.

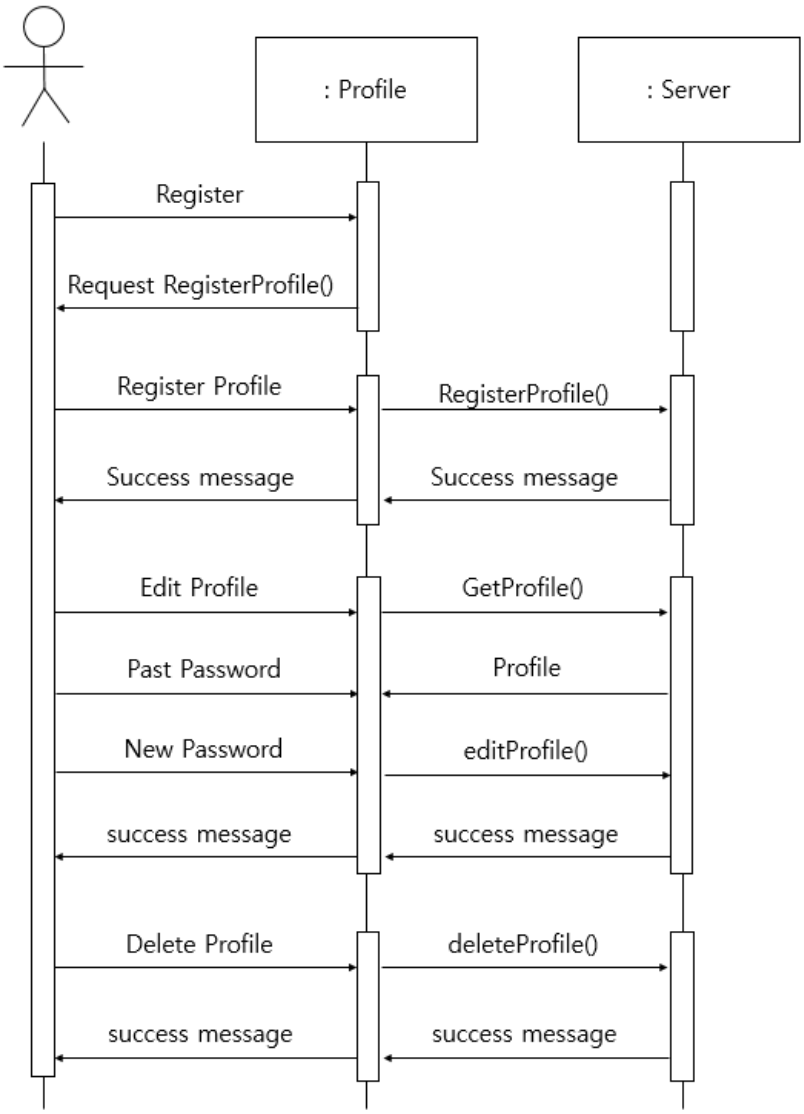
- registerProfile()
- getProfile()
- EditProfile()
- DeleteProfile()

4.2.1.3. Class Diagram



[그림 5] Class Diagram – 로그인 및 유저 정보 등록/수정

4.2.1.4. Sequence Diagram



[그림 6] Sequence Diagram – 로그인 및 유저 정보 등록/수정

4.2.2. 기기 등록 및 수정, 삭제

4.2.2.1. Attributes

해당 Device object에서의 attribute는 다음과 같다.

- plug: 등록, 수정, 삭제하려는 스마트 플러그이다.
- light: 등록, 수정, 삭제하려는 스마트 전구이다.

- type: 기기의 종류이다.
- plugList: 등록된 스마트 플러그의 목록이다.
- lightList: 등록된 스마트 전구의 목록이다.

해당 plug object가 가진 attribute는 다음과 같다.

- id: 기기의 id 이다.
- name: 사용자가 설정하는 기기의 이름이다.
- device: 스마트 플러그에 연결된 기기의 종류다.

해당 light object가 가진 attribute는 다음과 같다.

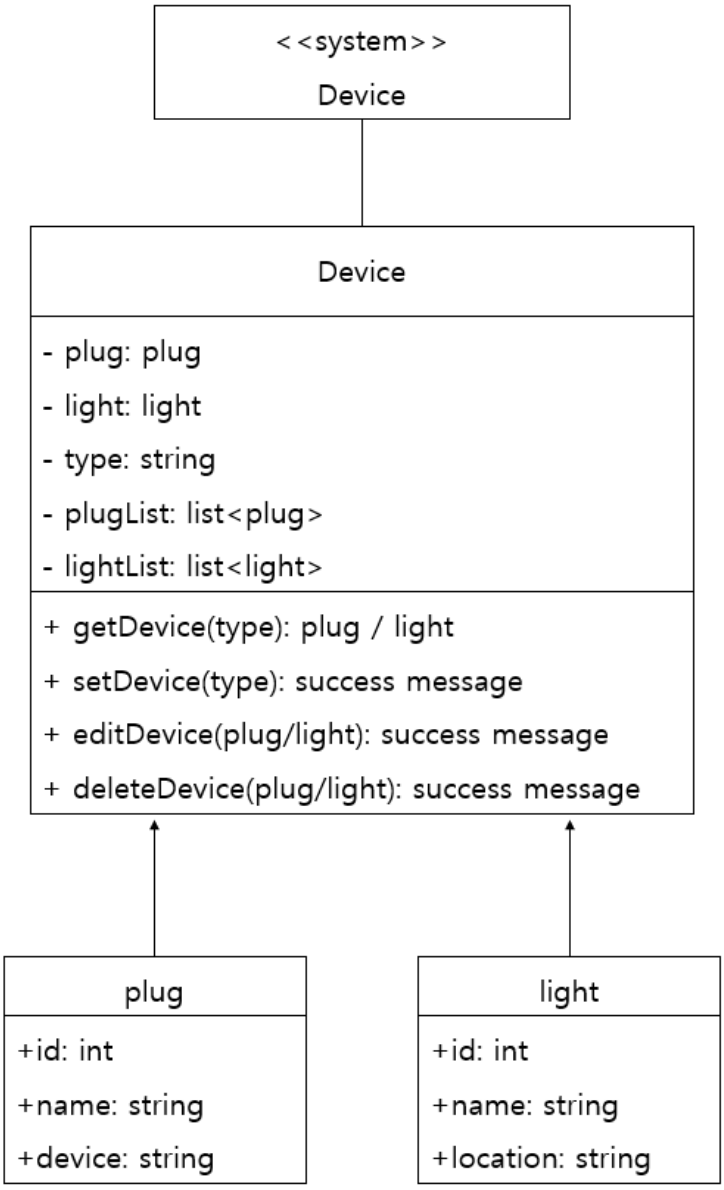
- id: 기기의 id 이다.
- name: 사용자가 설정하는 기기의 이름이다.
- location: 전구의 위치이다.

4.2.2.2. Methods

해당 Device object에서의 method는 다음과 같다.

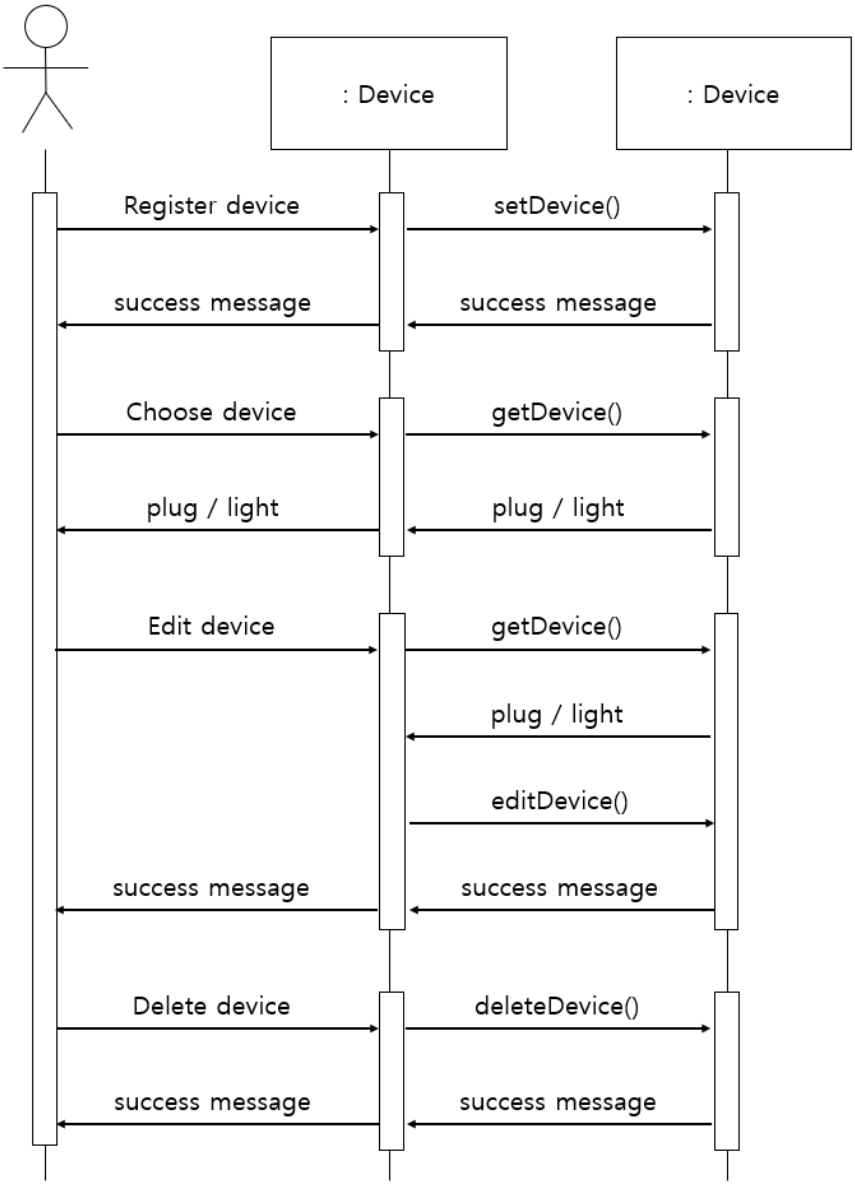
- getDevice()
- setDevice()
- editDevice()
- deleteDevice()

4.2.2.3. Class Diagram



[그림 7] Class Diagram – 기기 등록 및 수정, 삭제

4.2.2.4. Sequence Diagram



[그림 8] Sequence Diagram – 기기 등록 및 수정, 삭제

4.2.3. 사용량 데이터 확인

4.2.3.1. Attributes

- Plug : 확인하려는 스마트 플러그이다.
- PlugList : 등록된 스마트 플러그의 목록이다.

- Date : 확인하려는 사용량 데이터의 날짜이다.
- Time : 확인하려는 사용량 데이터의 시간이다.
- Device : 세부 정보 페이지로 확인하려는 스마트 플러그에 연결된 디바이스이다.
- DeviceList : 등록 되어있는 기기 목록이다.

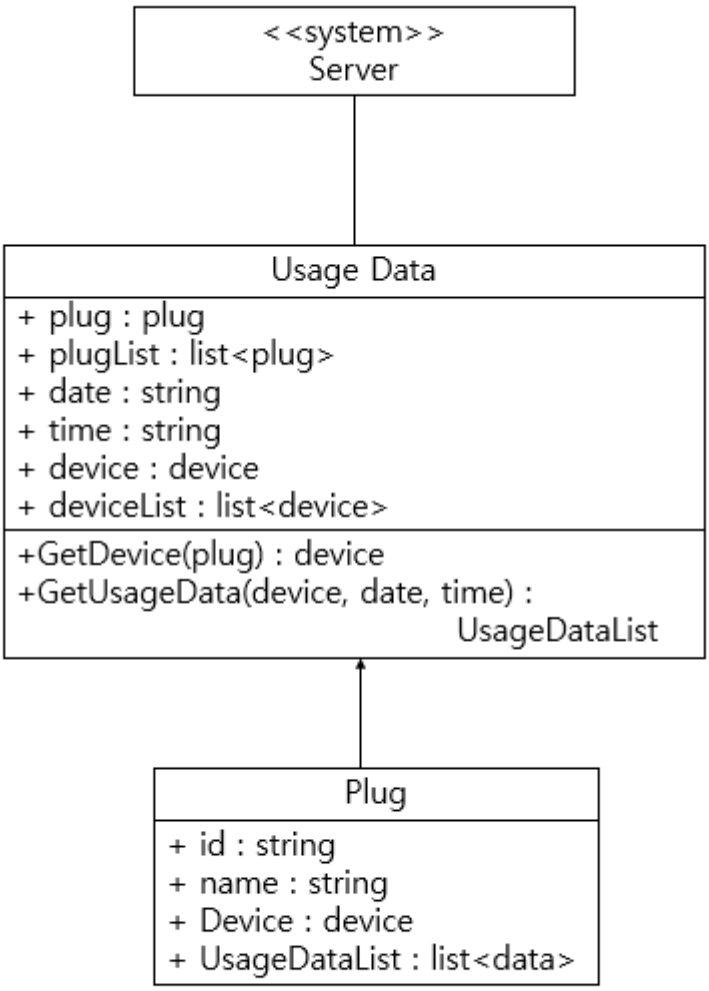
Plug Object 가 가진 attribute 는 다음과 같다.

- Device : 스마트 플러그에 연결된 기기이다.
- Id : 스마트 플러그의 ID 이다.
- Name : 사용자가 설정한 기기의 이름이다.
- UsageDataList : 해당 Plug 의 실시간 사용량 데이터가 저장된 리스트이다.

4.2.3.2. Methods

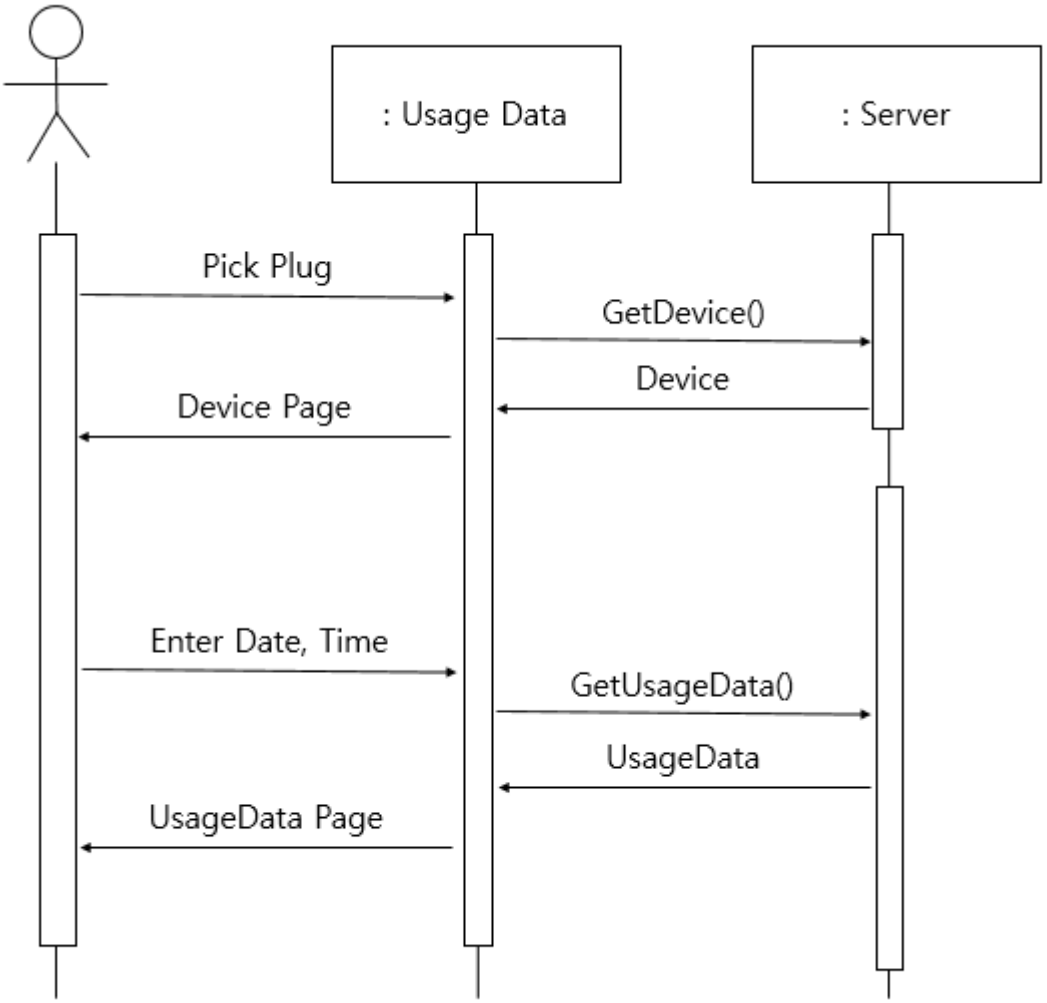
- GetDevice()
- GetUsageData()

4.2.2.3. Class Diagram



[그림 9] Class Diagram – 사용량 데이터 확인

4.2.2.4. Sequence Diagram



[그림 10] Sequence Diagram – 사용량 데이터 확인

4.2.4. 스케줄 / 동작 설정

4.2.4.1. Attributes

- Light : 확인하려는 전구이다.
- LightList : 등록된 전구의 목록이다.
- Plug : 확인하려는 스마트 플러그이다.
- Device : 스케줄과 동작을 설정하려는 기기이다.

- DeviceList : 등록 되어있는 기기 목록이다.
- PlugList : 등록된 스마트 플러그의 목록이다.
- Date : 확인하려는 사용량 데이터의 날짜이다.
- BeginTime : 스케줄의 시작 시간이다.
- EndTime : 스케줄의 종료 시간이다.
- TemperatureLowerBound : 스케줄이 실행되는 온도 하한선이다.
- TemperatureUpperBound : 스케줄이 실행되는 온도 상한선이다.
- HumidityLowerBound : 스케줄이 실행되는 습도 하한선이다.
- HumidityUpperBound : 스케줄이 실행되는 습도 상한선이다.
- AnotherDevice : 기기의 스케줄 설정에 사용되는 또다른 기기이다.
- Action : 스케줄 만족 시 실행되는 동작이다.
- ScheduleList : 설정된 스케줄의 목록이다.
- Schedule : 선택한 스케줄이다.

Schedule Object 가 가진 attribute 는 다음과 같다.

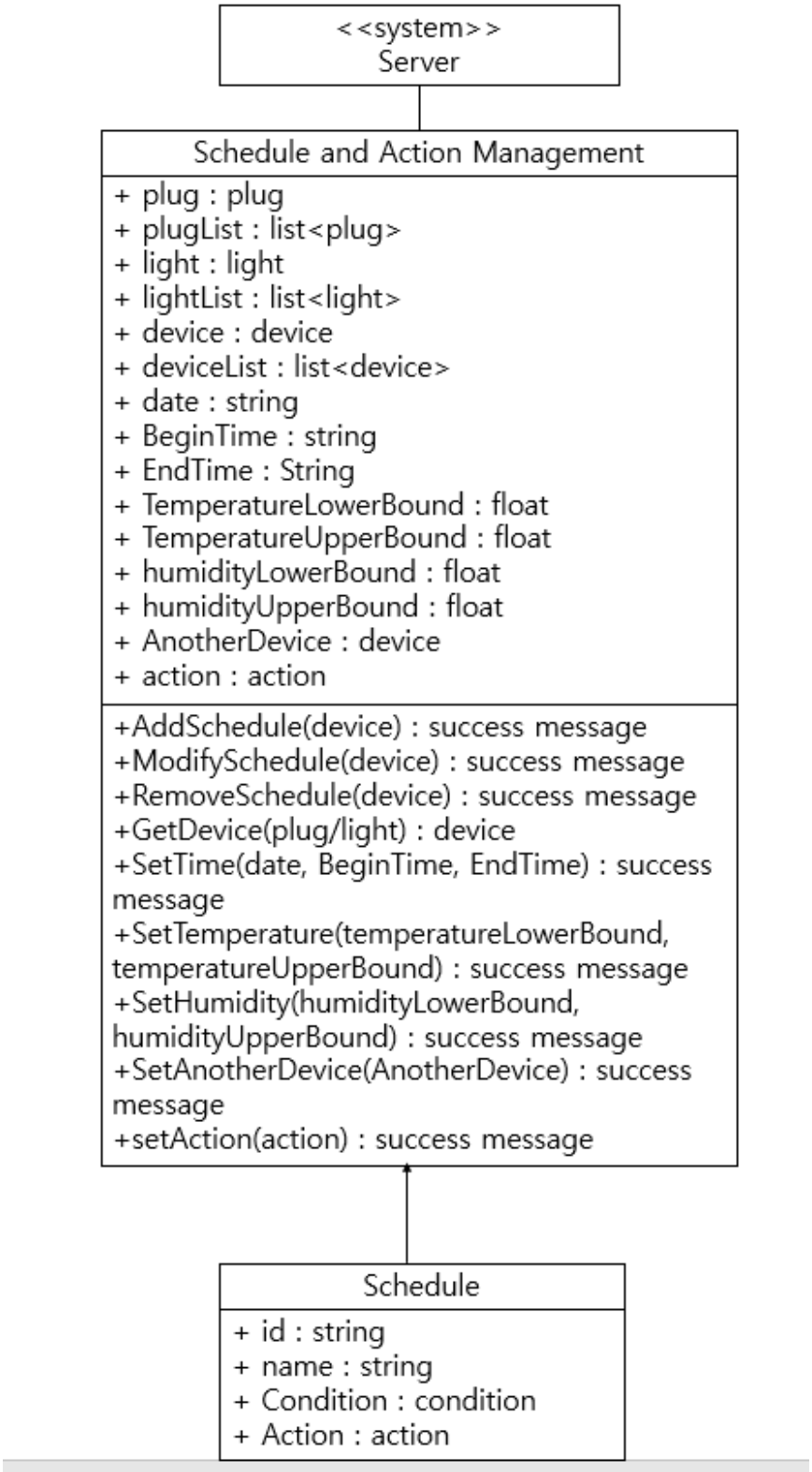
- Id : 해당 스케줄의 id 이다.
- Name : 유저가 설정해둔 스케줄의 이름이다.
- Condition : 스케줄이 가진 조건이다.
- Action : 스케줄 만족 시 실행되는 동작이다.

4.2.4.2. Method

- RemoveSchedule()
- ModifySchedule()
- AddSchedule()
- GetDevice()
- SetTime()
- SetTemperature()

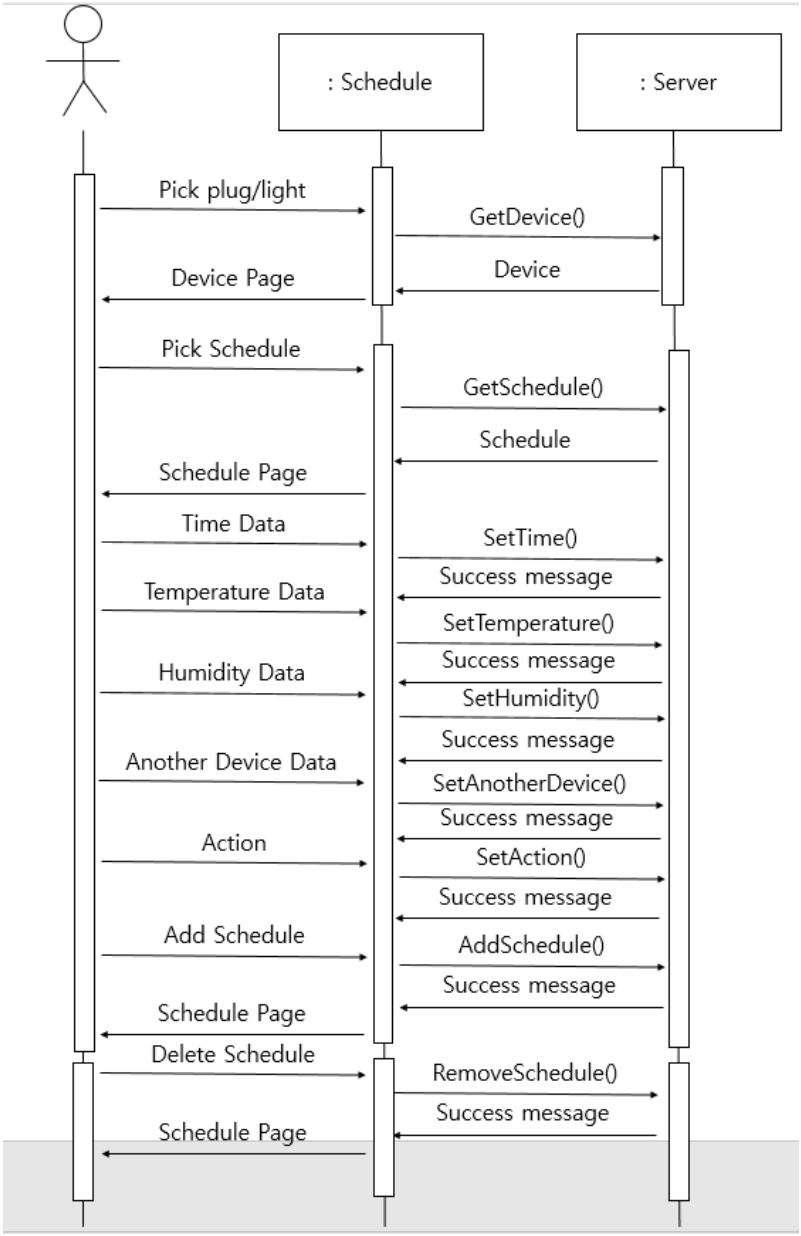
- SetHumidity()
- SetAnotherDevice()
- SetAction()
- GetSchedule()

4.2.4.3. Class Diagram



[그림 11] Class Diagram – 스케줄 / 동작 설정

4.2.4.4. Sequence Diagram



[그림 12] Sequence Diagram – 스케줄 / 동작 설정

4.2.5. IoT 기기 원격 제어

4.2.5.1. Attributes

해당 Remote control object에서의 attribute는 다음과 같다.

- control: 제어하려는 기기이다.

- deviceList: 제어 가능한 등록 되어있는 기기 목록이다.

해당 control object에서의 attribute는 다음과 같다.

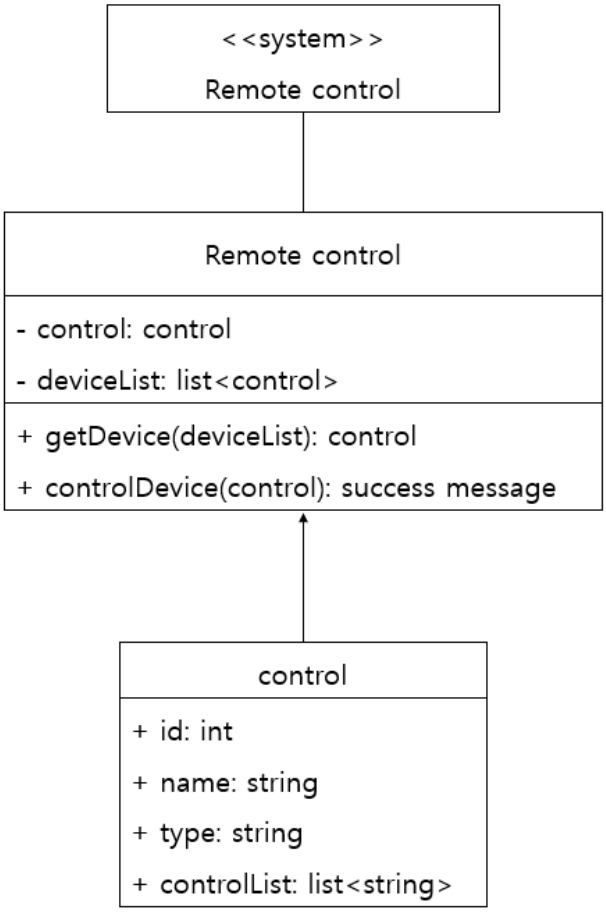
- id: 기기의 id 이다.
- name: 기기의 이름이다.
- type: 기기의 종류이다.
- controlList: 기기의 원격 제어 목록이다.

4.2.5.2. Method

해당 Remote control object에서의 method는 다음과 같다.

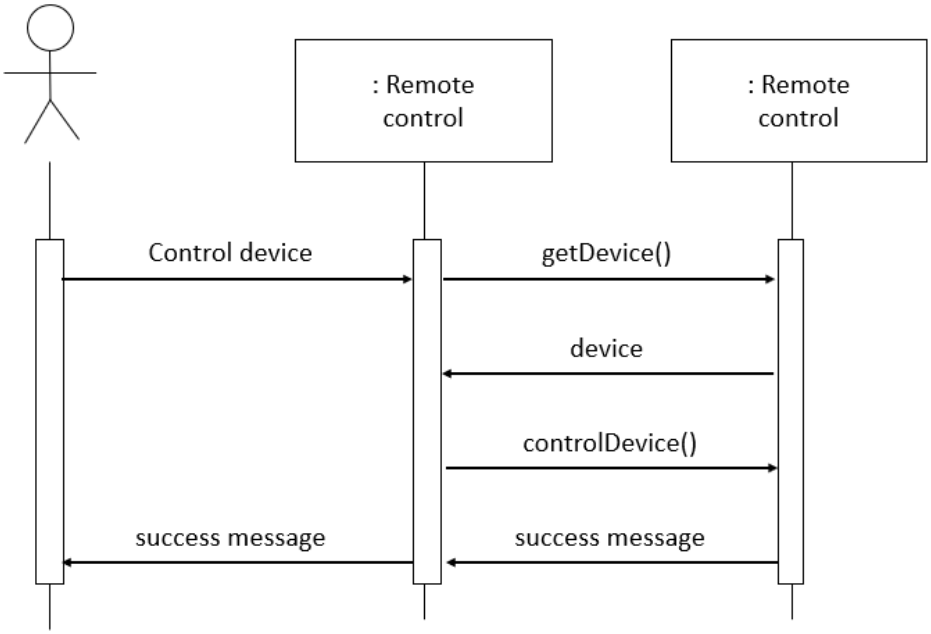
- getDevice()
- controlDevice()

4.2.5.3. Class Diagram



[그림 13] Class Diagram – IoT 기기 원격 제어

4.2.5.4. Sequence Diagram



[그림 14] Class Diagram – IoT 기기 원격 제어

4.2.6. 맞춤형 알림 설정

4.2.6.1. Attribute

해당 Customized notifications object에서의 attribute는 다음과 같다.

- notification: 선택한 알림이다.
- notificationList: 알림 목록이다.

해당 Notification object에서의 attribute는 다음과 같다.

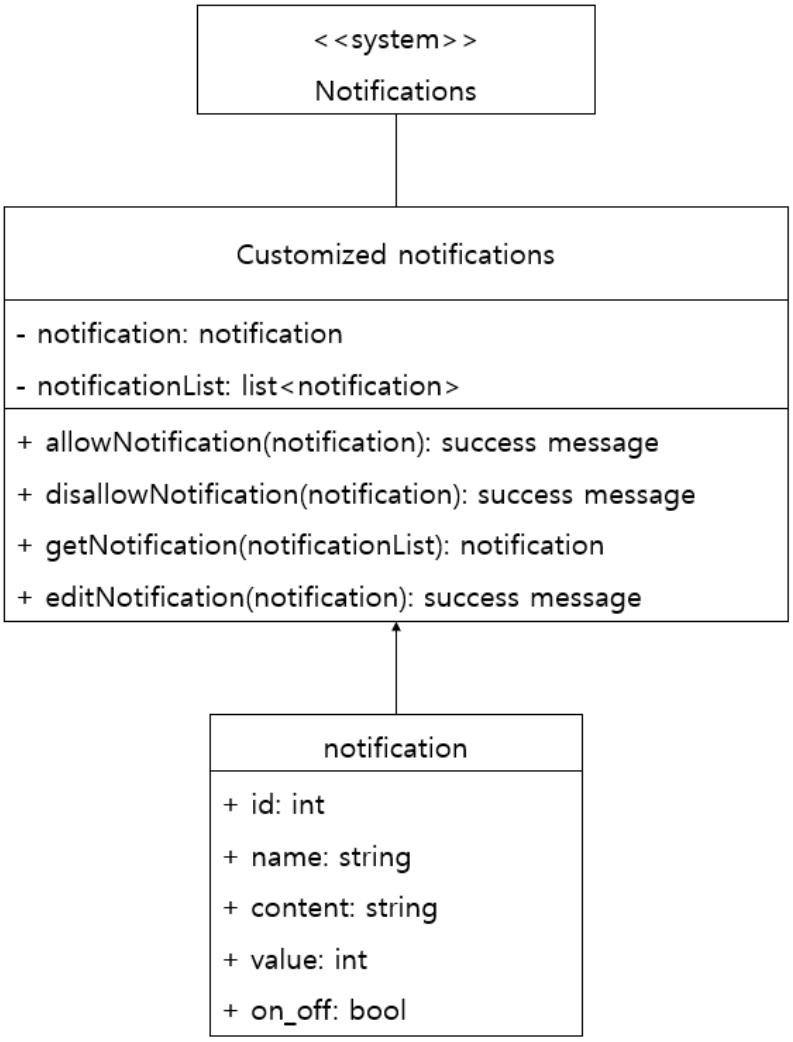
- id: 알림의 id 이다.
- name: 사용자가 설정한 알림의 이름이다.
- content: 알림의 내용이다.
- value: 알림의 내용에서 사용자가 설정한 값이다.
- on_off: 알림 설정 상태이다.

4.2.6.2. Method

해당 Customized notifications object에서의 method는 다음과 같다.

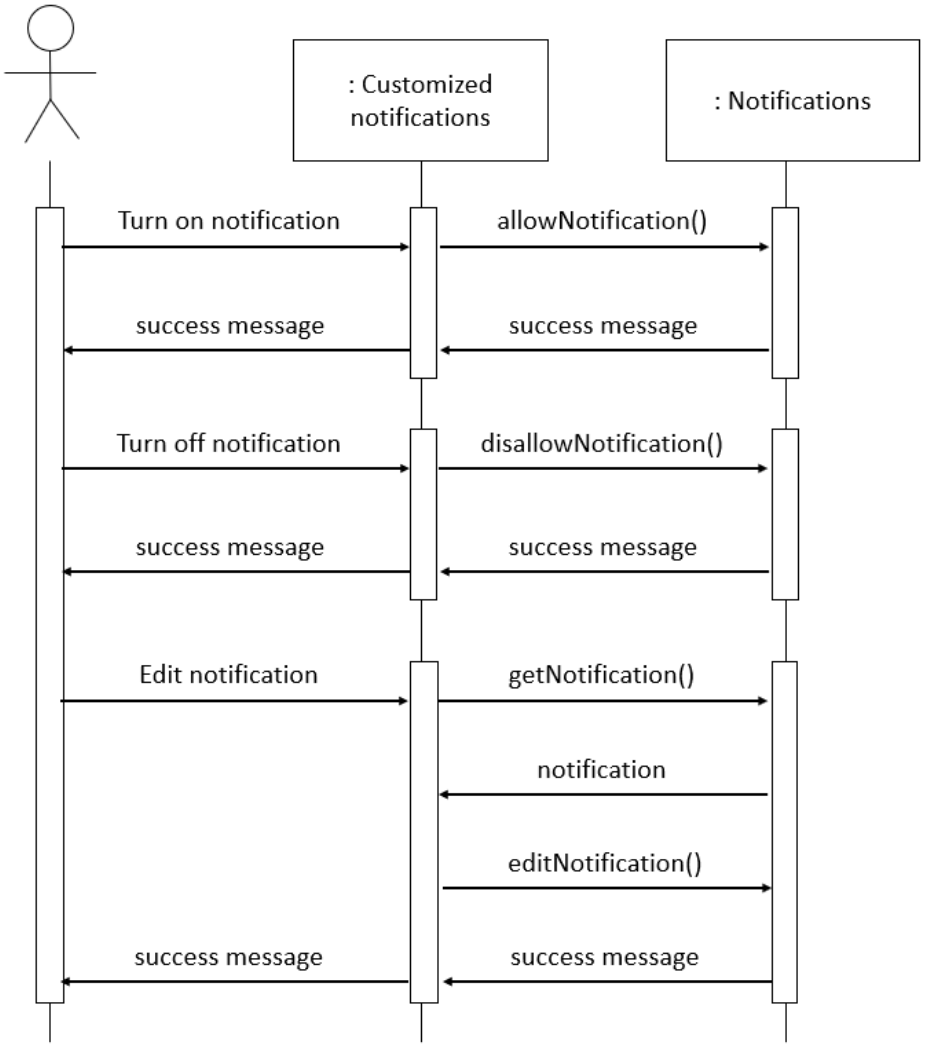
- allowNotification()
- disallowNotification()
- getNotification()
- editNotification()

4.2.6.3. Class Diagram



[그림 15] Class Diagram – 맞춤형 알림 설정

4.2.6.4. Sequence Diagram



[그림 16] Sequence Diagram – 맞춤형 알림 설정

4.2.7. 리포트 확인

4.2.7.1. Attributes

- Report : 요청해서 서버가 반환하는 리포트이다.
- type : 리포트의 유형이 일일 / 주간 / 월간 중 어느 것인지 나타낸다.
- Date : 원하는 리포트의 날짜이다.

- Device : 세부 정보 페이지로 이동할 디바이스이다.
- Schedule : 스케줄 설정 페이지로 이동할 스케줄이다.
- 설정한 값이다.
- on_off: 알림 설정 상태이다.

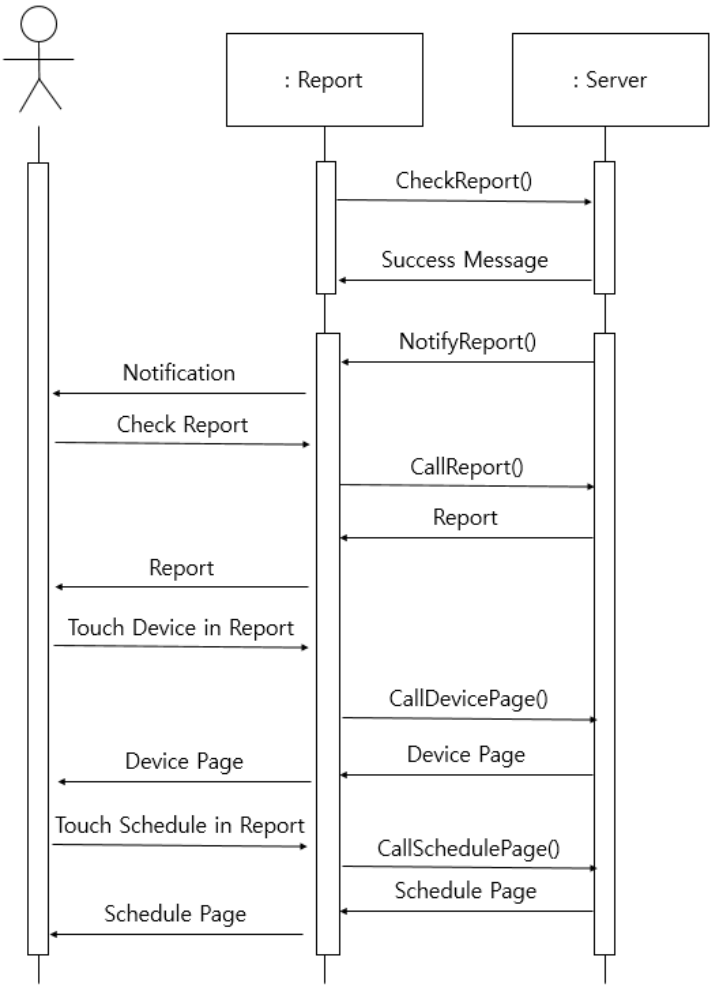
4.2.6.2. Methods

- CheckReport()
- NotifyReport()
- CallReport()
- CallDevicePage()
- CallSchedulePage()

4.2.6.3. Class Diagram

[그림 17] Class Diagram – 리포트 확인

4.2.6.4. Sequence Diagram



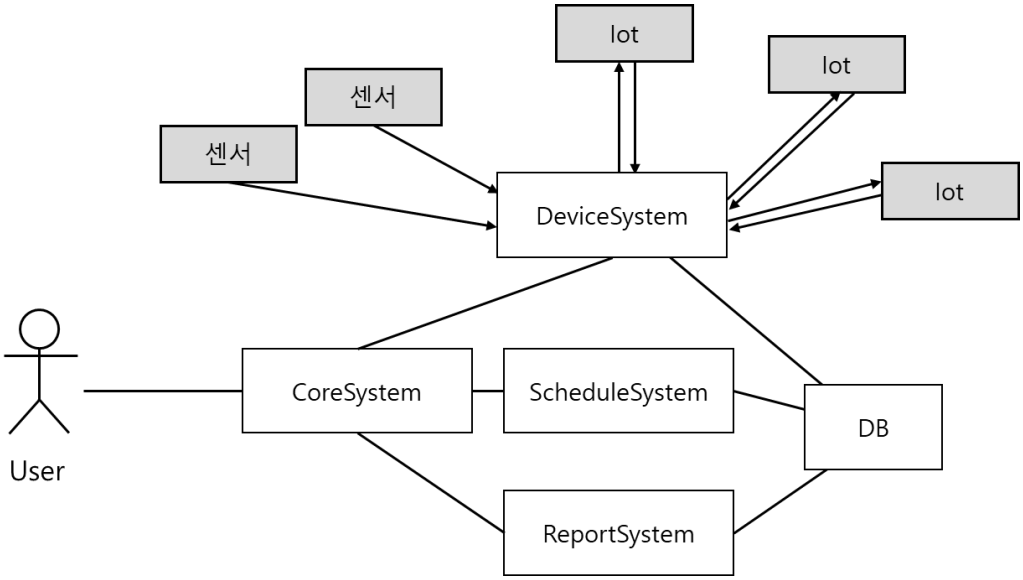
[그림 18] Sequence Diagram – 리포트 확인

5. System Architecture Backend

5.1. Objective

이 챕터에서는 back-end 시스템의 구조와 상호작용을 기술한다.

5.2. Overall Architecture



[그림 19] Overall architecture

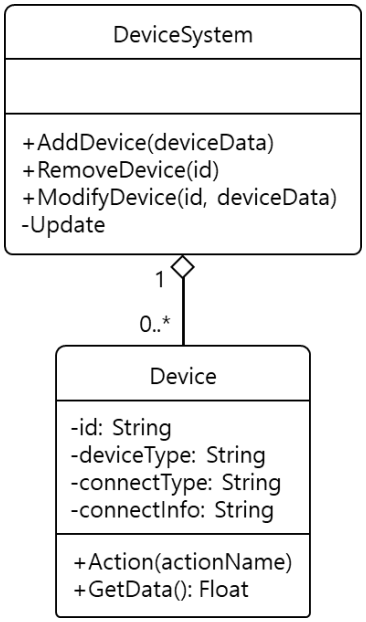
Smart Energy Management System의 전체적인 구조는 위와 같다.

모든 요청은 먼저 CoreSystem을 거친다. CoreSystem은 외부로부터 들어오는 요청들을 받아서 요청에 필요한 system들을 호출하고, 적절한 응답을 반환한다.

5.3. Subcomponents

5.3.1. DeviceSystem

5.3.1.1. Class Diagram



[그림 20] Class diagram – Device System

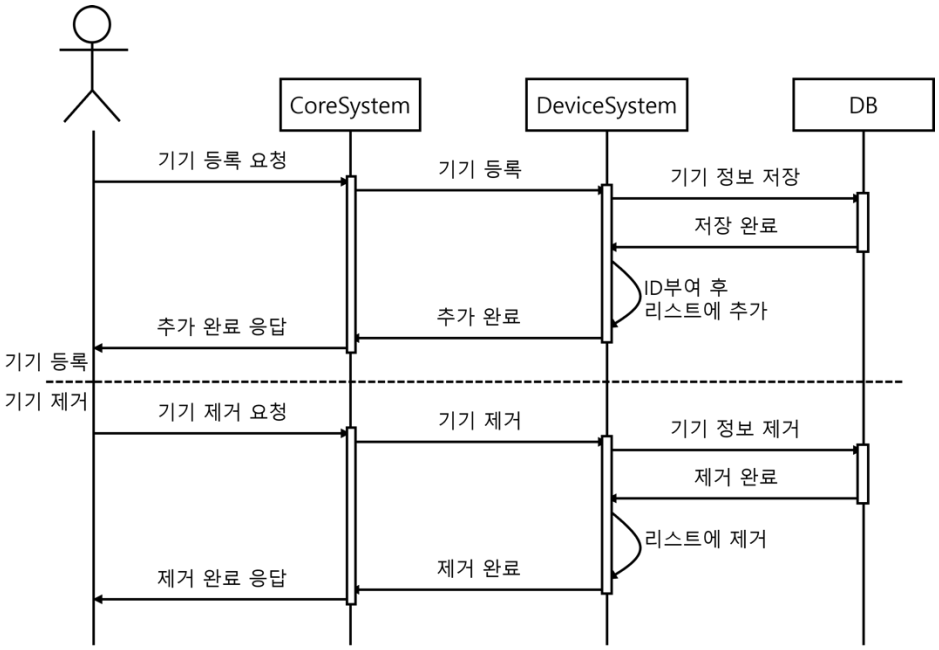
- DeviceSystem

  연결된 장치들을 관리하는 클래스이다. 장치들을 추가하거나 제거하고, 주기적으로 장치로부터 정보를 얻는다.

- Device

  장치의 정보를 담고 있는 클래스다. Action 메서드로 장치를 조작할 수 있고 GetData 메서드로 소비전력이나 온도 등 정보를 얻을 수 있다.

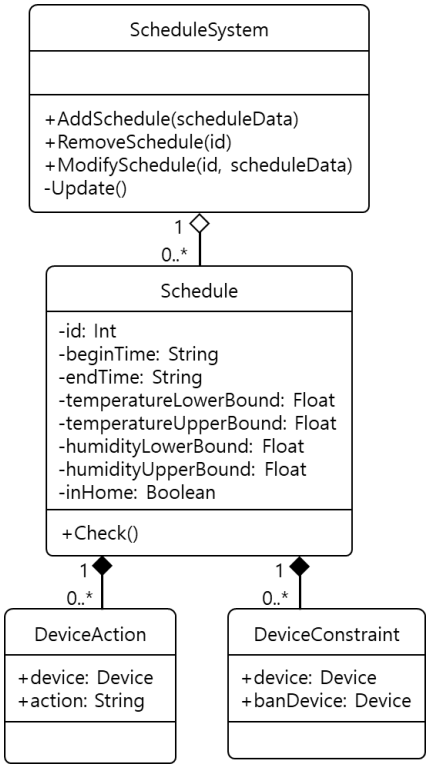
5.3.1.2. Sequence Diagram



[그림 21] Sequence diagram – Device System

5.3.2. ScheduleSystem

5.3.2.1. Class Diagram



[그림 22] Class diagram – Schedule System

- ScheduleSystem

  등록된 스케쥴들을 관리하는 클래스다. 스케쥴들을 추가하거나 제거하고, 주기적으로 스케쥴이 만족하는지 확인한다.

- Schedule

  스케쥴의 정보를 담고 있는 클래스다.

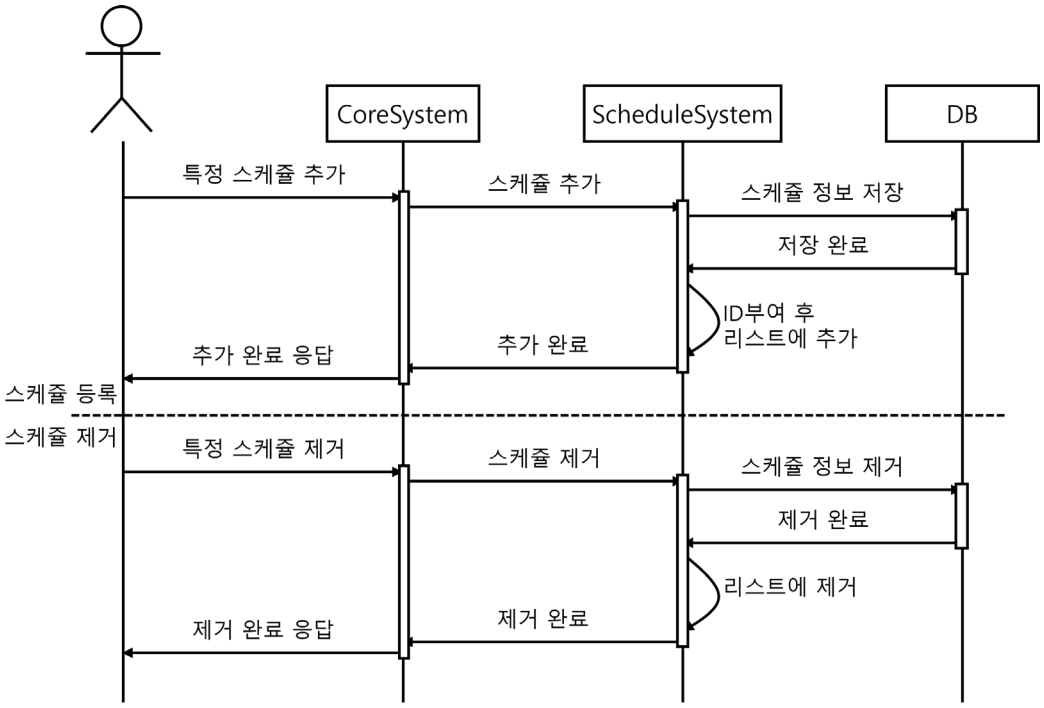
- DeviceAction

  스케쥴의 조건을 만족할 시 동작할 장치와 동작 내용을 담은 클래스다.

- DeviceConstraint

  특정 디바이스가 실행되면 특정 디바이스를 실행하지 못하도록 제한하는 조건 클래스다.

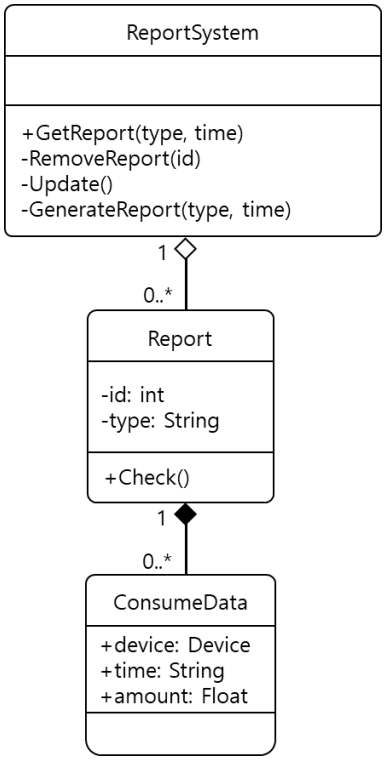
5.3.2.2. Sequence Diagram



[그림 23] Sequence diagram – Schedule System

5.3.3. ReportSystem

5.3.3.1. Class Diagram



[그림 24] Class diagram – Report System

- ReportSystem

  레포트들을 관리하는 클래스다. 요청 시 필요한 레포트를 반환하고, 주기적으로 레포트를 생성한다.

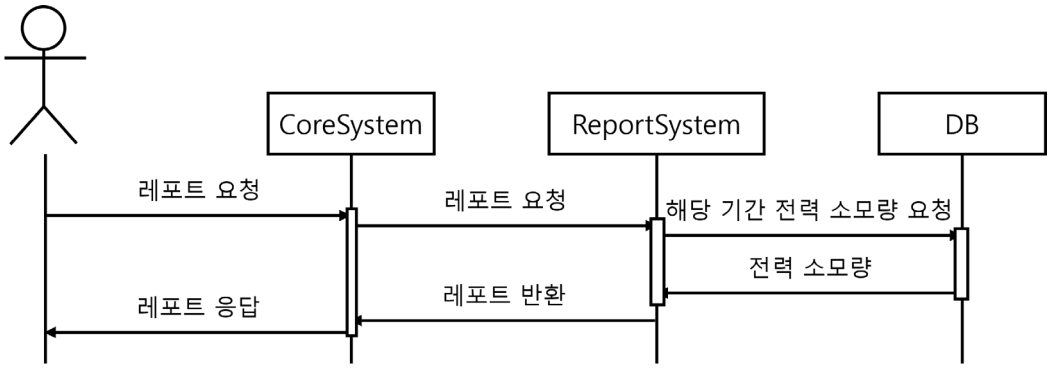
- Report

  레포트의 정보를 담은 클래스다.

- ConsumeData

  레포트 정보에서 장치마다 소비한 전력 데이터를 담은 클래스다.

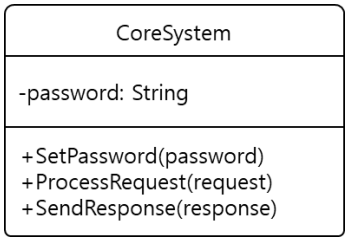
5.3.3.2. Sequence Diagram



[그림 25] Sequence diagram – Report System

5.3.4. CoreSystem

5.3.4.1. Class Diagram



[그림 26] Class diagram – Core System

- CoreSystem

  서버의 메인 system이다. 모든 외부로부터 오는 요청은 이 system을 거친다. 들어온 요청에 맞는 다른 system들을 호출해 요청을 처리하고, 응답을 보낸다.

6. Protocol Design

6.1. Objective

프로토콜 디자인 파트는 애플리케이션과 서버 간의 통신이 어떻게 이루어지고, 통신이 이루어질 때 전달되는 방식과 형식에 대해서 설명한다.

6.2. 전달 형식

6.2.1. HTTP

HTTP는 인터넷에서 데이터를 주고받을 수 있는 프로토콜이다. HTTP는 TCP/IP 단으로 구성되어 있으며, 요청(Request)와 응답(Response)로 통신을 한다. Client가 형식에 맞춰서 요청을 하면, Server는 알맞은 응답을 하는 형식이다. 요청을 할 때는 General, Headers, Body 로 이루어져 있다. General에는 주소, Request 종류, Client 상태, 규칙 등이 들어간다. 헤더에는 요청에 대한 정보를 담고 있다. 그리고 BODY에는 Header에 맞게 정보를 보낼 수 있다. 이는 JSON 형태로 데이터를 주로 보낸다. 본 프로젝트에서는 HTTP로 일반적인 통신을 다룬다.

6.3. 로그인 및 유저 정보 등록 / 수정

6.3.1. 유저 정보 등록

- Request

Attribute	Detail	
URL	/user/registration	
Method	POST	
Parameter	User_id	아이디
	pwd	비밀번호 (string)
	Login_ip	ip주소(string)

[표 1] 유저 등록 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "등록되었습니다"
Failure response body	Message	Message: "등록에 실패했습니다"

[표 2] 유저 등록 response

6.3.2. 로그인

- Request

Attribute	Detail	
URL	/user/login	
Method	POST	
Parameter	User_id	아이디
	pwd	비밀번호 (string)
	Login_ip	ip주소(string)

[표 3] 로그인 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "로그인 성공"
Failure response body	Message	Message: "로그인 실패"

[표 4] 로그인 response

6.3.3. 유저 정보 수정

- Request

Attribute	Detail	
URL	/user/registration	
Method	POST	
Parameter	Pwd	과거 비밀번호
	newPwd	새 비밀번호

[표 5] 유저 정보 수정 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "수정 성공"
Failure response body	Message	Message: "수정 실패"

[표 6] 유저 정보 수정 response

6.4. 기기 등록 및 수정, 삭제

6.4.1. 기기 등록

- Request

Attribute	Detail	
URL	/user/type/plug or /user/type/light	
Method	POST	
Parameter	type	등록하려는 기기의 종류

[표 7] 기기 등록 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "성공적으로 등록되었습니다"
Failure response body	Message	Message: "등록에 실패했습니다"

[표 8] 기기 등록 response

6.4.2. 기기 등록 정보 수정

- Request

Attribute	Detail	
URL	/user/type/plug or /user/type/light	
Method	PUT	
Parameter	plug / light	수정하려는 기기

[표 9] 기기 등록 정보 수정 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "저장되었습니다"
Failure response body	Message	Message: "실패했습니다"

[표 10] 기기 등록 정보 수정 response

6.4.3. 기기 등록 정보 삭제

- Request

Attribute	Detail	
URL	/user/type/plug or /user/type/light	
Method	DELETE	
Parameter	plug / light	삭제하려는 기기

[표 11] 기기 등록 정보 삭제 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "삭제되었습니다"
Failure response body	Message	Message: "실패했습니다"

[표 12] 기기 등록 정보 삭제 response

6.5. 사용량 데이터 확인

6.5.1. 플러그 선택

- Request

Attribute	Detail	
URL	/user/type/plug/list	
Method	GET	
Parameter	Plug	사용량 데이터 확인하려는 플러그

[표 13] 플러그 선택 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	Id	플러그 id

		Name	플러그 name
		Device	플러그에 연결된 기기
Failure response body	Message	Message: "가져오는데 실패했습니다"	

[표 14] 플러그 선택 response

6.5.2. 기기 사용량 정보 불러오기

- Request

Attribute	Detail	
URL	/user/device/loadusagedata	
Method	GET	
Parameter	Device	사용량 데이터 확인하려는 기기
	Date	사용량 데이터 확인하려는 날짜
	Time	사용량 데이터 확인하려는 시각

[표 15] 사용량 정보 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	UsageDataList	사용량 정보 리스트
Failure response body	Message	Message: "가져오는데 실패했습니다"	

[표 16] 사용량 정보 response

6.6. 스케줄 / 동작 설정

6.6.1. 기기 선택

- Request

Attribute	Detail	
URL	/user/type/plug/list or user/type/light/list	
Method	GET	
Parameter	Plug	스케줄 / 동작 설정하려는 플러그
	Light	스케줄 / 동작 설정하려는 전구

[표 17] 기기 선택 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	Device	플러그에 연결된 기기
		Id_light	전구 id
Failure response body	Message	Message: "가져오는데 실패했습니다"	

[표 18] 기기 선택 response

6.6.2. 스케줄 목록 불러오기

- Request

Attribute	Detail	
URL	/user/device/schedule/list	
Method	GET	
Parameter	Device	스케줄 설정하려는 기기

[표 19] 스케줄 목록 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	ScheduleList	스케줄 리스트
Failure response body	Message	Message: "가져오는데 실패했습니다"	

[표 20] 스케줄 목록 response

6.6.3. 스케줄 선택

- Request

Attribute	Detail	
URL	/user/device/loadschedule	
Method	GET	
Parameter	scheduleID	설정하려는 스케줄

[표 21] 스케줄 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	Schedule	스케줄 정보
Failure response body	Message	Message: "가져오는데 실패했습니다"	

[표 22] 스케줄 response

6.6.4. 스케줄 추가 / 수정

- Request

Attribute	Detail	
URL	/user/device/uploadschedule	
Method	GET	
Parameter	Date	날짜 조건
	Time	시간 조건
	Temperature	기온 조건
	Humidity	습도 조건
	Action	동작
	AnotherDevice	연동 기기 조건

[표 23] 스케줄 추가 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "등록 / 수정 성공"
Failure response body	Message	Message: "실패했습니다"

[표 24] 스케줄 추가 response

6.6.5. 스케줄 삭제

- Request

Attribute	Detail	
URL	/user/device/uploadschedule	
Method	DEL	
Parameter	ScheduleID	삭제하려는 스케줄

[표 25] 스케줄 삭제 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "삭제 성공"
Failure response body	Message	Message: " 실패했습니다"

[표 26] 스케줄 삭제 response

6.7. IoT 기기 원격 제어

- Request

Attribute	Detail	
URL	/user/control	
Method	GET	
Parameter	control	제어하려는 기기

[표 27] IoT 기기 원격 제어 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "정상적으로 제어했습니다"
Failure response body	Message	Message: "제어에 실패했습니다"

[표 28] IoT 기기 원격 제어 response

6.8. 맞춤형 알림 설정

6.8.1. 알림 활성화

- Request

Attribute	Detail	
URL	/user/notification	
Method	GET	
Parameter	notification	알림

[표 29] 알림 활성화 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "활성화되었습니다"
Failure response body	Message	Message: "실패했습니다"

[표 30] 알림 활성화 response

6.8.2. 알림 비활성화

- Request

Attribute	Detail	
URL	/user/notification	
Method	GET	
Parameter	notification	알림

[표 31] 알림 비활성화 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "비활성화되었습니다"
Failure response body	Message	Message: "실패했습니다"

[표 32] 알림 비활성화 response

6.8.3. 알림 내용 수정

- Request

Attribute	Detail	
URL	/user/notification	
Method	PUT	
Parameter	notification	알림

[표 33] 알림 내용 수정 request

- Response

Attribute	Detail	
Success code	HTTP 200 (OK)	
Failure code	HTTP 400 (Bad request)	
Success response body	Message	Message: "저장되었습니다"
Failure response body	Message	Message: "실패했습니다"

[표 34] 알림 내용 수정 response

6.9. 리포트 확인

6.9.1. 리포트 목록 가져오기

- Request

Attribute	Detail	
URL	/user/schedule/list	
Method	GET	
Parameter	User_id	유저의 ID

[표 35] 리포트 목록 request

- Response

Attribute	Detail		
Success code	HTTP 200 (OK)		
Failure code	HTTP 400 (Bad request)		
Success response body	Parameter	ReportList	리포트 리스트
Failure response body	Message	Message: "가져올 리포트가 없습니다."	

[표 36] 리포트 목록 response

6.9.2. 리포트 불러오기

- Request

Attribute	Detail	
URL	/user/report/loadReport	
Method	GET	
Parameter	Type	리포트의 종류
	date	리포트의 날짜

[표 37] 리포트 선택 request

- Response

Attribute	Detail
Success code	HTTP 200 (OK)
Failure code	HTTP 400 (Bad request)

Success response body	Parameter	Report	해당하는 리포트
Failure response body	Message	Message: "가져오는데 실패했습니다"	

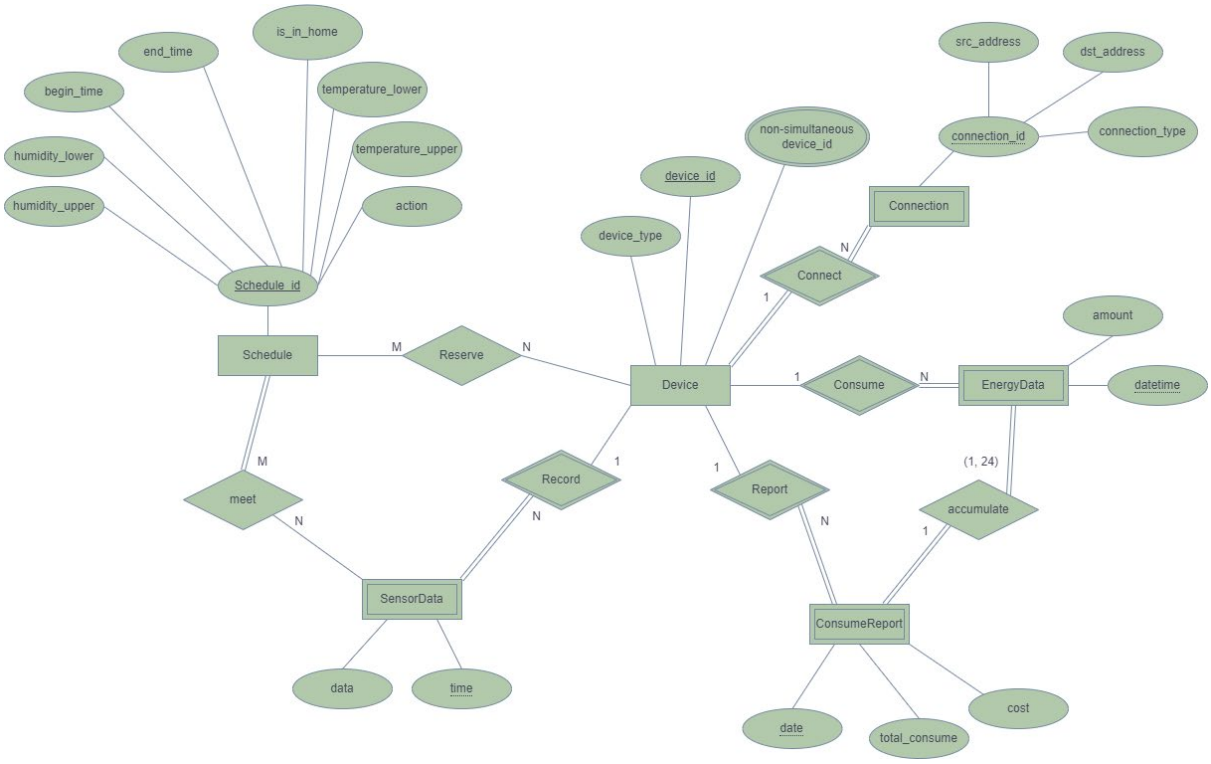
[표 38] 리포트 선택 response

7. Database Design

7.1. Objectives

이 챕터에서는 시스템의 데이터 구조와 이 구조가 데이터 베이스에 어떻게 구현되는지 설명한다. 먼저 객체와 그들의 관계를 중심으로 ER-Diagram을 알아본다. 그 후에 릴레이션 스키마에 대해서 알아보고, 각각의 데이터 베이스 테이블을 SQL로 어떻게 구현할 것인지에 대해서 알아본다.

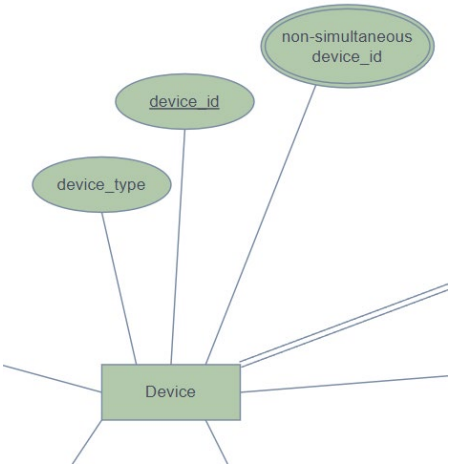
7.2. ER Diagram



[그림 27] ER Diagram

7.2.1. Entities

7.2.1.1. Device

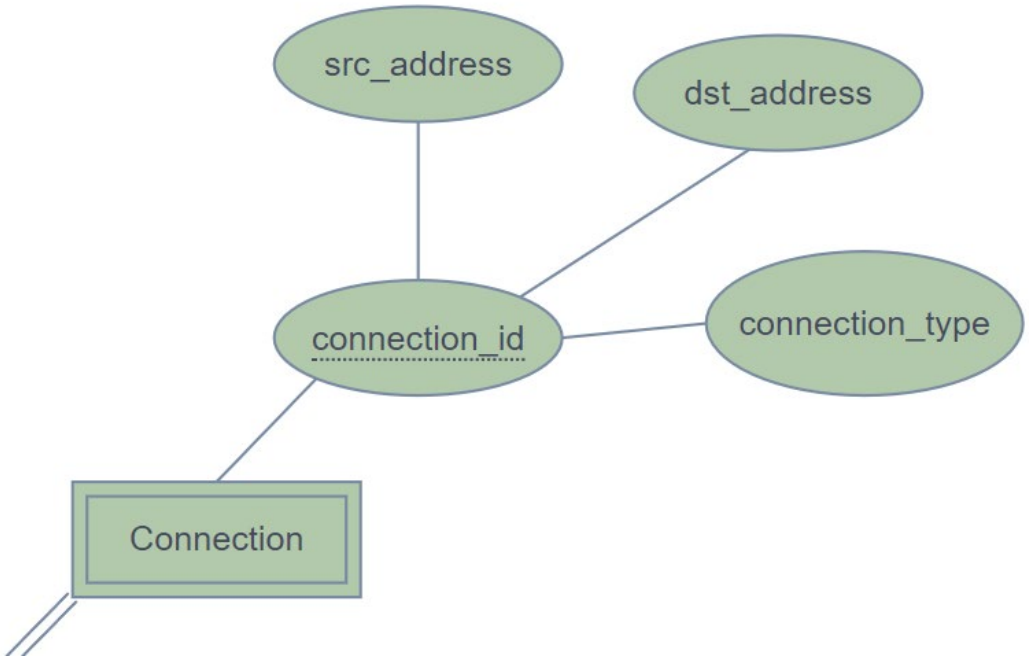


[그림 28] ER Diagram - Device

Device 객체는 스마트 전구, 스마트 플러그, 센서 중 하나를 나타낸다. Device는 device_type, device_id, non-simultaneous_device_id를 attribute로 가지며, primary key는 device_id이다.

non-simultaneous_device_id는 이 Device와 동시에 작동되지 않아야 하는 device들을 말한다.

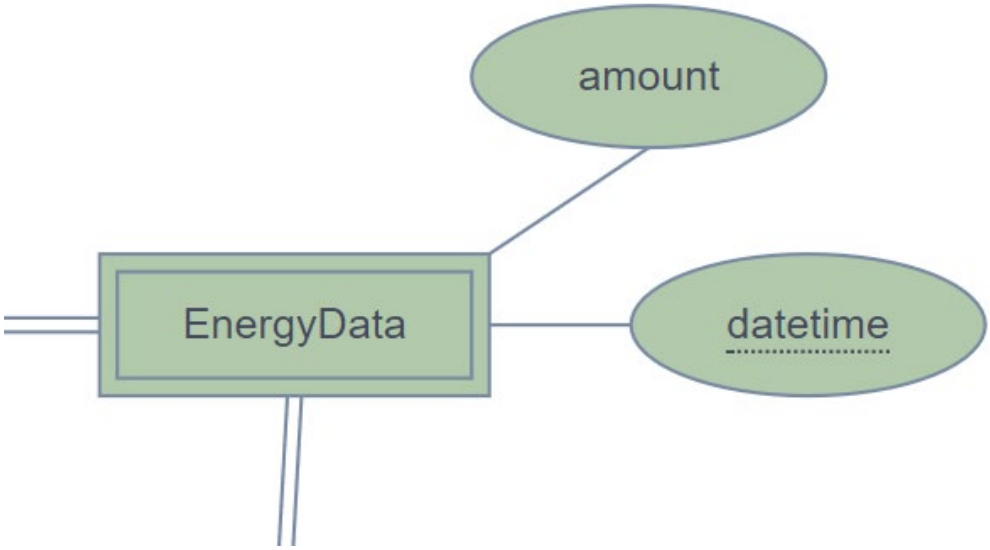
7.2.1.2. Connection



[그림 29] ER Diagram - Connection

Connection 객체는 어떤 한 기기가 가지고 있는 connection의 들을 저장을 하는 테이블이다. Device는 서버와 블루투스, ipv4, ipv6, wifi등 여러 방법을 통해 서버와 통신을 할 수 있으므로, 이러한 데이터 들을 이 Table에 저장을 할 수 있다.

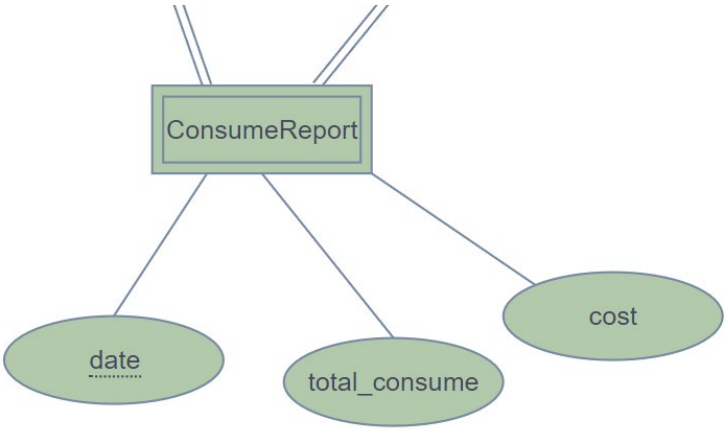
7.2.1.3. EnergyData



[그림 30] ER Diagram - EnergyData

EnergyData 시간 당 에너지 소모량을 기록하는 테이블이다. 이 테이블은 시간을 기록하는 datetime과 에너지 소모량을 기록하는 amount를 attribute로 갖는다.

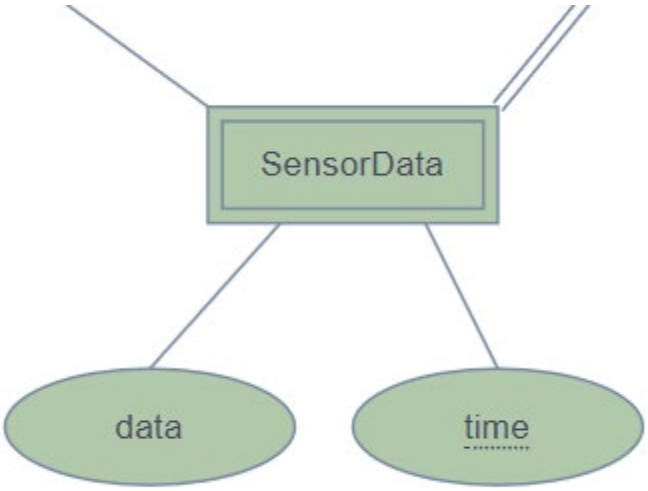
7.2.1.4. ConsumeReport



[그림 31] ER Diagram - ConsumeReport

ConsumeReport는 하루의 에너지 소모 보고서를 기록하는 테이블이다. Date는 날짜를 저장하고, total_consume은 하루의 총 에너지 소모량을 저장한다. 또 cost는 하루의 에너지 비용을 저장한다.

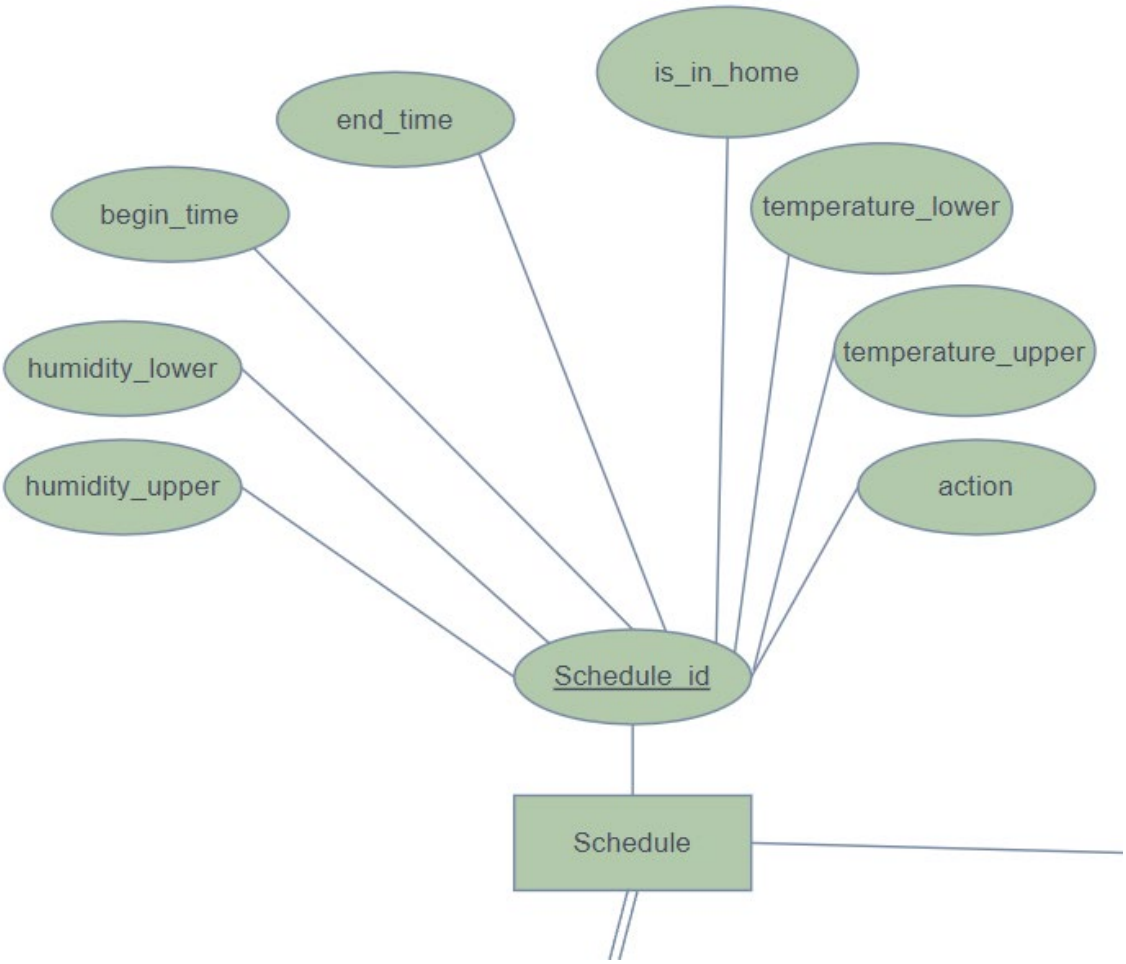
7.2.1.5. SensorData



[그림 32] ER Diagram - SensorData

SensorData는 시간 별로 센서(GPS, 온도 센서, 습도 센서, 움직임 감지 센서) 등의 데이터를 기록하는 테이블이다.

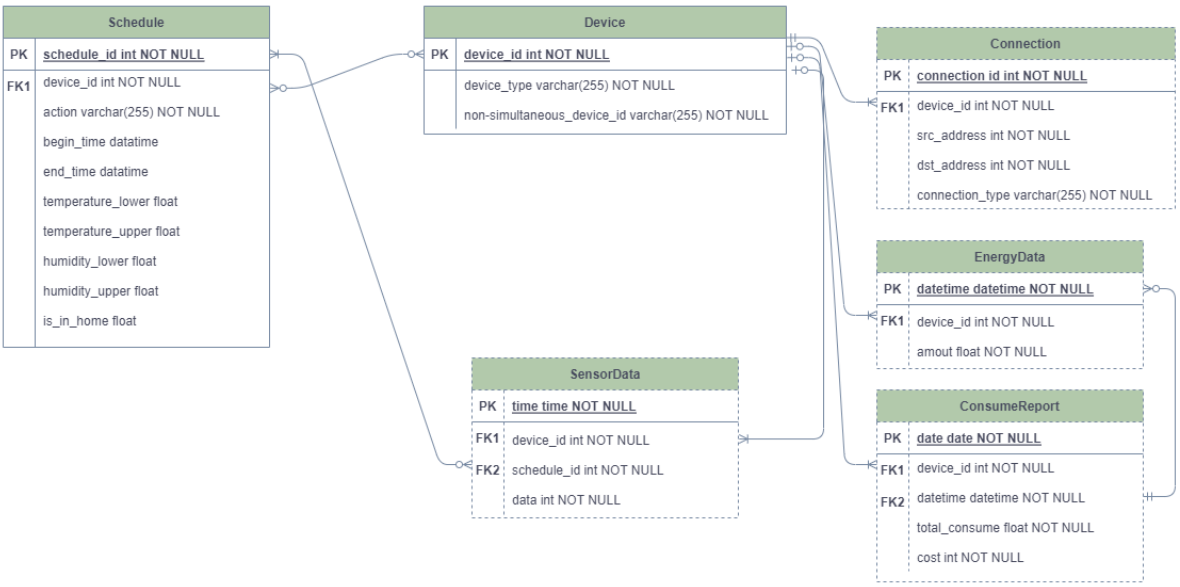
7.2.1.6. Schedule



[그림 33] ER Diagram - Schedule

Schedule은 사용자가 사전에 특정 조건을 만족하였을 때, 자동으로 기기가 특정한 행동을 하도록 예약한 내용을 저장하는 테이블이다. 온도, 습도, 시간에 따라 단일/복합 조건에 따라 특정 그룹의 기기들을 제어하는 예약을 할 수 있다.

7.3. Relational Schema



[그림 34] Relational Schema

7.4. SQL DDL

7.4.1. Device

```
CREATE TABLE Device
{
        device_id INT NOT NULL
        device_type VARCHAR(255) NOT NULL
        non-simultaneous_device_id VARCHAR (255) NOT NULL
        PRIMARY KEY (device_id)
};
```

7.4.2 Connection

```
CREATE TABLE Connection
{
        device_id INT NOT NULL
        connection_id INT NOT NULL
        src_address INT NOT NULL
        dst_address INT NOT NULL
        connection_type INT NOT NULL
        FOREIGN_KEY (device_id) REFERENCES Device(devicd_id)
};
```

7.4.3. EnergyData

```
CREATE TABLE EnergyData
{
        datetime DATETIME NOT NULL
        device_id INT NOT NULL
        amount FLOAT NOT NULL
        FOREIGN_KEY (device_id) REFERENCES Device(devicd_id)
};
```

7.4.4 ConsumeReport

```
CREATE TABLE ConsumeReport
{
        date DATE NOT NULL
        device_id INT NOT NULL
        total_consume FLOAT NOT NULL
        const INT NOT NULL
        FOREIGN_KEY (device_id) REFERENCES Device(devicd_id)
};
```

7.4.5 SensorData

```
CREATE TABLE SensorData
{
        time TIME NOT NULL
        device_id INT NOT NULL
        schedule_id INT NOT NULL
        data INT NOT NULL
        FOREIGN_KEY (device_id) REFERENCES Device(devicd_id)
        FOREIGN_KEY (schedule_id) REFERENCES Schedule(schedule _id)
};
```

7.4.6 Schedule

```
CREATE TABLE Schedule
{
        schedule_id INT NOT NULL
        device_id INT NOT NULL
        action VARCHAR(255) NOT NULL
        begine_time DATETIME
        end_time DATETIME
        temperature_lower FLOAT
        temperature_upper FLOAT
        humidity_lower FLOAT
        humidity_upper FLOAT
        is_in_home BOOLEAN
        PRIMARY KEY (schedule_id)
        FOREIGN_KEY (device_id) REFERENCES Device(devicd_id)
};
```

8. Testing Plan

8.1. Objectives

Testing plan은 시스템의 의도한대로 제대로 작동하고 있는지, 사용자의 요구사항을 제대로 충족하고 있는지, 시스템이 안전한지 등을 검사하기 위한 테스트 계획을 기술하는 챕터이다. Test는 Development Test, Release Test, User Test, 이 세가지로 구성된다

8.2. Testing Policy

8.2.1. Development Test

Development test는 시스템을 개발하는 단계에서 진행되는 Test이다. 이 test는 미래에 생길 수 있는 여러 잠재적인 위협과 문제사항을 발견 및 예방하여 시간적, 금전적 비용을 경감시키기 위한 전략으로 진행된다. 이 Test 단계에서는 충분한 테스트가 이루어지지 않아 시스템이 불안정하고 구성요소들간 충돌이 일어날 수 있다. 그러한 이유로 각각의 요소들이 의도대로 기능하는지, 요소들을 통합시킬 경우 요소들간 충돌하며 생기는 문제가 있는지를 확인하고 해결하여야 한다. 이를 통해 우리는 결과물의 성능, 신뢰도, 보안성을 보다 높은 수준으로 끌어올릴 수 있다.

8.2.1.1. Performance

실시간 전력사용량 확인 및 해당 데이터 기반 리포트 출력을 위한 데이터 처리작업이 필요하므로 해당 작업이 지나치게 오래 걸려 사용자에게 불편함을 주는 일이 없도록 하여야 한다.

통신상태에 문제가 없다는 가정하에 어플리케이션 실행 이후 IoT기기목록, 조건 설정 등의 서버에 저장된 데이터를 불러와 화면에 출력하기까지의 시간은 2초 이상이 걸리면 안 된다. 또한 리포트의 경우 많은 양의 데이터를 불러와 월간, 주간, 일일 관 같은 기간선택에 따른 시각화 처리를 해야 하는데 이 경우에도 리포트를 띄우기까지 4초보다 많은 시간이 걸리면 안 된다. 특정 날짜를 클릭하여 기기 별 전력사용량을 띄우는 화면을 출력하는 것은 1초 이내에 진행되어야 할 것이다. 이러한 요구 사항들은 어플리케이션의 최적화 및 기기의 사양에 영향을 받는 부분이므로 테스트는 기기의 최소요구사항인 1GB의 RAM과 1GHz 싱글 코어 CPU, 그리고 Android 6.0 또는 iOS 10의 조건을 만족하는 기기에서 이루어져야 할 것이다.

8.2.1.2. Reliability

이 프로그램의 경우 여러 IoT기기와 서버와 연결되어 IoT기기에서 데이터를 보내오면 그를 토대로 리포트를 만들어 내고 알림을 보내오고 스케줄을 추천하는 식으로 이루어진다. 그 때문에 IoT기기로부터 전력사용량 데이터가 누락되는 것 없이 제대로 전달되어 오는지 확인하여야 할 것이다. 또한 사용자가 서버를 통해 IoT기기를 제어하기도 하므로 요소간 충돌과 같은 문제로 제어가 제대로 이루어 지지 않거나 하는 문제가 없도록 오류를 확인하여야 할 것이다. 그리고 새로운 IoT기기가 추가되거나 기존 IoT기기 정보가 수정될 때 마다 역시 확인하여야 할 것이다.

8.2.1.3. Security

사용자는 등록된 IoT기기에 대한 제어 권한을 가지고 있고 서버에는 사용자의 생활패턴 및 시간대별 집안 내 사람의 유무 등을 유추할 수 있는 기기 별 전력사용량 등의 정보가 저장되어 있다. 그러므로 약속된 사용자가 아닌 외부인이 무단으로 권한을 취득해 IoT기기 제어를 하거나 서버내 저장된 데이터에 접근할 수 있는 루트가 있는지 확인하여야 할 것이다.

8.2.2. Release Test

Release test란 기본적인 구현이 완료된 시점에 진행되며 배포전에 시행하는 Test이다. 이 테스트의 경우 개발에 참여한 팀을 제외한 다른 팀에서 진행하게 되는데 그 목적은 사용자가 프로그램을 목적에 맞게 계획대로 사용될 수 있는지를 확인하고 배포하기에 문제가 없는지 확인하는 것이다. 이 테스트의 경우 개발에 참여한 인원을 제외한 다른 인원을 통해 진행될 것이다.

8.2.3. User Test

User test의 경우 실제 사용자들에게 배포하여 테스트를 진행하는 것으로 그들이 직접 체험 후 개선점과 의견 등의 피드백을 받는 단계이다. 이 프로그램의 경우 무작위로 일반 사용자 20명을 모집하여 진행될 것이다.

8.2.4. Test Case

Test case의 경우 performance, reliability, security 세 가지측면에서 각각 10가지의 테스트 케이스를 만들어서 어플리케이션을 통해 제대로 작동하는지 테스트를 하고 이를 기반으로 평가서를 작성할 것이다.

9. Development Plan

9.1. Objectives

이 챕터에서는 개발에 필요한 여러 외부 기술 및 프로그램에 대하여 기술한다.

9.2. Frontend Environment

9.2.1. Adobe Photoshop

어플리케이션의 아이콘 및 디자인 작업에 활용하기 위한 프로그램이다. 사진 편집기능이 주를 이루지만 기본적인 드로잉 툴도 포함되어 있어 그림을 그리는 작업 역시 가능하다. Windows와 macOS에서 사용 가능하다. 이 프로그램을 통해서 Flutter에서 기본적으로 제공되는 UI디자인 이외의 아이콘과 이미지들을 만들어서 가시성과 접근성을 향상시킬 것이다.

9.2.2. Flutter

구글에서 출시한 모바일/웹/데스크톱 크로스 플랫폼 GUI SDK이다. 하나의 코드 베이스로 Android OS와 iOS, 두 OS환경에서 모두 동작하는 앱을 개발할 수 있다. 또한 Flutter는 웹 브라우저, Windows, Mac등 여러 환경에서 사용가능하기 때문에 여러 개발자가 함께 개발하는 것에 대하여

이점을 가질 수 있다.

9.3. Backend Environment

9.3.1. GitHub

GitHub는 소프트웨어 개발 버전을 관리하여 다수의 개발자들 간의 협업을 지원하는 코드 호스팅 플랫폼이다. 이 플랫폼을 통하여 다수의 개발진이 프로젝트를 동시에 함께 개발하고 취합하며 개발과정에 도움이 된다.

9.3.2. MySQL

세계에서 가장 많이 쓰이는 오픈 소스의 RDBMS이다. 다중 스레드, 다중 사용자 형식의 구조질의어 형식의 DBMS로 오라클에서 관리 및 지원하고 있다. 이를 통해 서버내에 저장되는 데이터를 관리한다.

9.3.3. Node.js

Node.js는 오픈 소스 JavaScript 엔진인 크롬 V8에 비동기 이벤트 처리 라이브러리인 libuv를 결합한 플랫폼이다. Windows와 macOS버전 둘 다 존재한다. IoT기기를 통해 실시간으로 송수신되는 데이터 처리를 위한 서버는 필수적이다. 이러한 서버를 구축하기 위하여 Node.js를 활용한다.

9.4. Constraints

소프트웨어 개발은 해당 문서 내에 상세화 된 내용을 기반으로 진행될 것이다. 이외의 세부적인 구현이나 기반적인 제약은 아래 사항을 준수하며 진행될 것이다.

-이미 범용적으로 쓰이고 증명된 기술들을 사용하다.

-가능한 부분은 오픈소스를 활용한다.

-정확한 데이터 수집 및 처리가 중요한 소프트웨어이므로 신뢰성을 중점으로 개발한다.

-소프트웨어의 성능을 향상시키는 방향으로 개발한다.

-코드를 최적화하고 차후 코드의 유지보수 및 진화가 원활하도록 보기 쉽게 코드를 작성하고 주석 작성을 충분하게 한다.

-UI 설계 시 사용자가 설명서 없이도 간편하게 활용할 수 있도록 직관적이게 설계한다

9.5. Assumptions and Dependencies

이 어플 구동을 위한 기기의 최소 사양은 1GB의 RAM과 1GHz 싱글 코어 CPU로 테스트 역시 해당 사양의 기기로 진행될 것이다.

또한 OS의 경우 iOS, Android OS 두 운영체제에 지원되는 앱을 개발할 것이다. iOS의 경우 10.0, Android OS의 경우 6.0버전 이상을 타겟으로 개발을 할 것이다.

10. Supporting Information

10.1. Software Design Specification

본 문서는 IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830). 서식을 따라 제작되었다.

10.2. Document History

날짜	버전	설명 (편집 파트)	참가자
5/11	0.0.0	문서 작업 시작	전체
5/12	0.1.0	1, 2	이예송
5/13	0.2.0	8, 9	송유호
5/13	0.3.0	4, 6	김서정
5/14	0.4.0	5	임형진
5/14	0.5.0	7	김주원
5/14	1.0.0	목차 정리	이예송
5/14	1.0.1	1, 2 수정	이예송
5/15	1.1.0	4 추가	김현중
5/15	1.1.1	8의 목차 명 수정	송유호
5/15	1.2.0	6 추가	김현중
5/15	1.2.1	3	임형진
5/15	1.2.2	7 추가	김주원
5/15	1.2.3	2 내용 추가	이예송
5/15	1.3.0	표 그림 정리/ 캡션 수정	이예송