Smart Lighting With IOT

Software Design Specification

2022. 05. 15.

소프트웨어 공학 TEAM 7

Team leader: 지예준

Team member: 남상욱

Team member: 박현제

Team member: 주소미

목차

그림 목차

1. Preface

이 장에는 본 프로젝트(Smart Lighting With IOT) Software Design Document의 독자 정보, 독자, 범위와 문서의 목적 및 구조가 포함되어 있다.

1.1. Readership

이 문서는 10개의 장과 여러 세부 항목으로 이루어져 있다. 각 장은 시스템의 전반적인 구조(frontend와 backend의 구조, protocols, databases design, testing plan, devlopment plan)에 대해 다루고 있다. 이 문서의 메인 독자는 7조의 팀원과 주된 이용자인 성균관대학교 교수, 학생, 스마트 조명 기기 사용자이다.

1.2. Scope

이 Design Specitication은 사용자의 활동 인식을 통한 조명 조절 프로그램 구현을 디자인하기 위해 Software Engineering과 Software Quality Engineering을 사용한다.

1.3. Objective

이 문서의 주된 목적은 Smart Lighting With IOT 프로젝트의 전반적인 윤곽과, 기술적인 설명을 통해 본 프로젝트의 가이드라인을 제공하기 위함이다. 이 SDS는 이전에 작성된 SRS을 바탕으로 구체화되었으며 프로젝트를 진행하며 바탕이 될 시스템의 구조, frontend와 backend의 구조, protocols, databases design, testing plan, devlopment plan에 대해 설명한다.

1.4. Document Structure

1. Preface: Software Design Document (Smart Lighting With IOT)의 독자 정보, 독자, 범위와 문서의 목적 및 구조에 대해 설명한다.
2. Introduction: 이 문서에서 사용된 도구, 다이어그램에 대해 설명하고 이에 대한 레퍼런스를 담고 있다.
3. Overall System Architecture: 시스템의 전반적인 구조를 context, sequence, use case다이어그램으로 설명한다.
4. System Architecture - Frontend: 시스템의 Frontend 구조를 Class, sequence 다이어그램으로 설명한다.
5. System Architecture - Backend: 시스템의 Backend 구조를 Class, sequence 다이어그램으로 설명한다.
6. Protocol Design: 이 애플리케이션과 서버 사이에서 사용되는 프로토콜의 구조에 대해 설명한다.

7. Database Design: 시스템에서 사용되는 데이터의 구조에 대해 설명하고 데이터베이스에서 어떻게 표현되는지 보여준다.
8. Testing Plan: development testing, release testing, user testing의 계획을 보여준다.
9. Development Plan: 디자인 이후에 진행될 개발에 대한 계획을 보여준다. 개발을 하면 필요한 환경, 사용되는 도구, 인지해야 하는 제한 사항들과 규칙에 대해 설명한다.
10. Supporting Information: document history를 제공한다.

2. Introduction

이 프로젝트의 목적은 조명을 활용하고자 하는 사용자들이 조명과 관련된 다양한 기능을 제공받을 수 있는 애플리케이션을 제작하는 것에 있다. Smart Lighting With IOT의 사용자는 조명 조절에 대한 기본적인 설정을 제공받을 수 있고, 필요한 경우 원하는 조명 상태를 설정하여 활용할 수 있다. 특히 Iot기기를 통한 정보를 받아와 사용자의 활동을 파악하여 사용자에게 맞는 조명을 제공하고자 한다.

이 문서는 설계 문서이며 구현에 사용된 설계에 대한 정보를 포함한다. 또한 이전에 작성된 SRS를 참고하여 작성되었다.

2.1. Objectives

이 장에서는 프로젝트 문서에서 사용된 도구와 다이어그램에 대해 설명한다.

2.2. Applied Diagrams

2.2.1. UML

UML은 요구사항 분석, 시스템 설계, 시스템 구현 등 시스템 개발 과정에서 개발자 간 커뮤니케이션을 용이하게 하기 위한 표준화된 모델링 언어이다. UML은 모델링을 위한 표현력이 강력하고 상대적으로 모순이 적은 논리적 표기법을 가진 언어이다. 따라서 개발자 간 소통이 쉽고, 누락되거나 일관성이 없는 모델링 구조를 지적하기 쉬우며, 개발하고자 하는 시스템의 규모에 관계없이 모든 시스템에 적용할 수 있다. UML은 Use Case Diagram, Class Diagram 등 도표를 기반으로 객체 지향 소프트웨어 개발을 위한 분석 및 설계 장치를 풍부하게 제공하고 있어 향후 상당 기간 산업 표준으로 활용될 것으로 기대된다.

2.2.2. Use Case Diagram

Use Case 다이어그램은 시스템에서 제공하는 Unit 단위에 기능들을 설명하는 다이어그램이다. Use Case 다이어그램이 하는 가장 중요한 역할은, 주요 프로세스에 대한 시스템과 상호 작용 주체 간의 관계뿐만 아니라 다양한 사용 사례 간의 관계를 포함하여 시스템의 기능적 요구사항을 시각화하는 것이다. Use Case 다이어그램은 시스템의 고급 기능이나 시스템의 범위를 설명하는데 사용합니다. 여러 다이어그램중에 가장 기초적이고 간단한 요약이다.

2.2.3. Sequence Diagram

Sequence Diagram은 특정 사용사례 또는 특정 사용사례의 일부를 시간에 따른 흐름으로 자세하게 보여준다. Sequence Diagram은 sequence에 있는 서로 다른 객체 간의 호출 관계를 나타내며, 서로 다른 객체에 대한 호출을 자세히 표시할 수 있다. Sequence Diagram은 2차원으로 표현되며, 세로는 메시지/호출 sequence를 발생 시간 순서로 나타내고, 가로는 메시지가 전송되는 객체 인스턴스를 나타낸다. Sequence Diagram은 매우 간단하며, 클래스 인스턴스(객체)는 다이어그램의 맨 위에 있는 상자에 각 클래스의 인스턴스를 배치하여 분류된다.

2.2.4. Class Diagram

Class Diagram은 서로 다른 객체가 서로 어떻게 연관이 있는지에 대한 관계를 나타낸다. Class Diagram에서는 시스템을 구성하는 객체와 그 객체의 attribute, method, 그리고 객체간의 관계를 보여준다. 일반적으로 Class Diagram은 프로그래머가 구현하는 클래스들을 시각화하는데 사용하고, 클래스들의 실행과 논리적 구조를 표현하는데 사용한다.

2.2.5. Context Diagram

Context Diagram은 시스템 또는 시스템의 일부와 환경 사이의 경계를 정의하는 다이어그램으로, 시스템과 상호 작용하는 엔티티를 보여준다. Context Diagram은 시스템의 데이터 플로우를 표현하는 가장 높은 레벨의 Diagram이다. 따라서 본 Diagram에서는 각 내부의 세부사항을 표시하는 것이 아니라 모든 시스템의 전반적인 상호작용만을 표현하고

있다. Context Diagram의 목표는 전반적인 시스템 요구사항과 개발할 때 제약조건, 고려해야 하는 외부 요인 및 사건에 초점을 맞추는 것이다. Context Diagram은 가장 상위 레벨 Diagram이기 때문에 모든 사용자들이 보고 이해할 수 있을 정도로 쉽게 표시되어야 한다.

2.2.6. Entity Relationship Diagram

엔티티 관계도는 시스템의 데이터베이스 구조를 모델링할 때, 엔티티를 구성하는 고유한 특성을 갖는 엔티티의 특성과 네트워크형 구조에서 이들 사이의 관계를 보여주는 도면이다. 엔티티 관계 다이어그램에서 엔티티 집합은 직사각형으로 표현되고, 속성은 타원으로 표현되며, 엔티티 집합과 그 속성은 선으로 연결되며, 관계 집합은 마름모꼴로 표현되며, 관계 집합의 매핑 형태는 화살표로 표현된다. ER Diagram은 이전에 설명한 기호들과 함께 엔티티들의 역학관계를 그림으로 표현한다. 작성된 ER Diagram은 데이터베이스를 구축하는데 베이스로 사용될 예정이다.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

PowerPoint는 대중들이 발표자료를 만들기 위해서 가장 많이 사용하는 툴이다. 또한 웹에 사람들이 이미 배포해놓은 디자인이 많기 때문에 쉽게 다운받아서 추가/수정 할 수 있기 때문에 확장성이 뛰어나다. PowerPoint로 프레젠테이션을 만들고 다양한 그림을 그리고 수정하는 작업을 수행했다.

2.3.2. Microsoft word Online/Offline

Microsoft word는 문서작업을 할 때, 한글보다 호환성이 좋고 편집도 깔끔하게 구성되어있다. 이 툴은 요구사항 명세서(Software Requirement Specification), 디자인 명세서(Software Design Specification)등을 작성하는데 사용되었다. MS word는 Online/Offline이 모두 가능하여 팀원들과 공유 문서를 만들어서 공동으로 작성을 진행할 수 있었다.

2.3.3. Draw.io

Draw.io는 플로우 차트를 작성할 수 있는 툴로, 자체적으로 UML을 통한 Diagram 작성을 지원하기 때문에 본 프로젝트에서 사용하는 수많은 Diagram을 그리는데 사용하였다. 추가적으로 몇몇 ER Diagram을 그릴때도 본 툴을 사용하였음을 명시한다. Draw.io는 다운받을 필요없는 온라인 툴로 구글 드라이브와 연동이 되기 때문에 활용성이 상당히 좋다.

2.3.4. ERDPlus

ERD Plus는 엔티티 관계 다이어그램, 관계 스키마, 별 스키마 및 SQL DDL 문을 만드는 데이터베이스 모델링 도구이다. ERD Plus는 ERD(Entity Relationship Diagram), 관계도(Relational Diagram), 별도(Dimension Model)를 빠르고 쉽게 만들 수 있는 웹 기반 데이터베이스 모델링 도구이다. ERD Plus를 사용하면 표준 ERD 구성요소(엔티티, 속성, 관계)를 그릴 수 있다.

2.4 Project Scope

스마트 조명 프로젝트는 사람들이 가장 민감하게 작용하는 시각적 요소를 효과적으로 조절하기 위해서 진행된 프로젝트이다. 본 프로젝트에서 조명들의 관리를 최적화할 수 있는 방법을 고안해내고 사용자에게 편리한 인터페이스를 제공하기 위해서 노력할 것이다. 본 서비스의 프론트엔드는 상당히 간단하지만 백엔드 부분에서 사용자들을 위한 효율적인 로컬 데이터베이스 관리가 필요하고 클라우드 내의 데이터를 효율적으로 관리할 수 있게 해줄 데이터베이스 구축에 힘쓸 것이다. 본 프로젝트는 다른 스마트 조명들과는 다르게 활동들을 인식하여 활동마다 조명이 변하는 효과를 줄 것이고, 조명의 각도 또한 조절할 수 있게 하여 실내 인테리어 효과를 가져올 수 있는 것을 목표로 할 것이다.

2.5. Referneces

- Team 7, 2020 Spring. Software Design Document, SKKU.

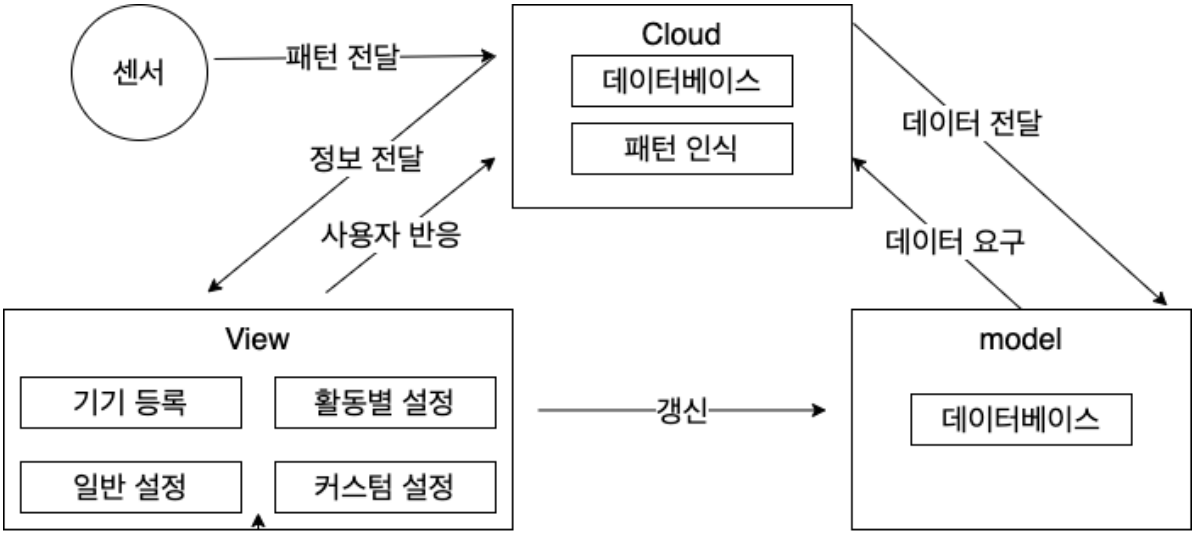
3. System Architecture – Overall

3.1. Objectives

이 장에서는 frontend design, backend design을 포함한 시스템의 전반적인 구조에 대해 다룬다.

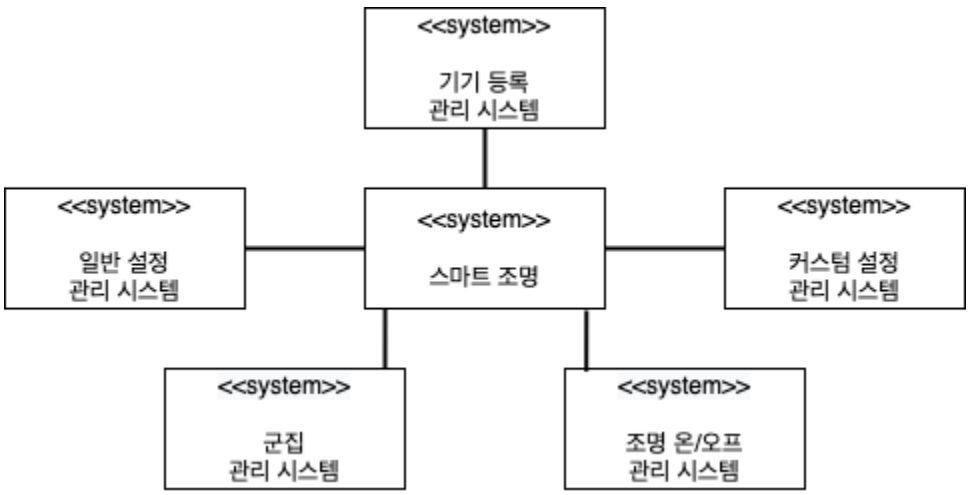
3.2. System Organizations

본 서비스는 사용자가 스마트 조명을 효율적으로 관리하고 사용할 수 있는 서비스를 제공할 것이다. 본 서비스의 프론트 엔드는 사용자와 간단한 상호작용을 하고 사용자가 편리하게 많은 것들이 백엔드에서 작동할 것이다. 프론트엔드 부분에서는 사용자에게 새로운 조명을 등록 받고, 사용자가 등록한 조명들을 군집화하고, 군집화된 조명을 설정하고, 활동에 따라 기존에 설정을 덮어서 사용할 수 있는 추가 설정을 사용자로부터 입력받을 것이다. 백엔드에서는 사용자에게 받은 아주 기본적인 정보들을 바탕으로 애플리케이션을 설정하고 해당 설정을 기기내 데이터베이스에 저장하고, 필요한 설정을 클라우드 서비스에서 불러와 기기내 데이터베이스에 저장할 것이다. 추가적으로 사용자의 행동을 인식하는 센서들과 지속적인 상호작용을 하고 해당 센서가 보내는 신호를 통해 설정을 유동적으로 조절하는 작업을 한다. 간단한 시스템 진행 단계는 사용자가 새로운 조명을 등록할 때, 해당 조명에 대한 메타 정보(기기에 대한 번호, 초기 조명 설정, 군집 등)를 기기내 데이터베이스에 저장한다. 사용자는 군집을 통해서 여러 개의 조명들을 한꺼번에 조정할 수 있다. 이후 사용자가 특정 액티비티를 할 때에는 센서들이 인식해 초기 1번은 사용자의 활동에 맞는 설정을 클라우드로부터 다운받아 조명을 조절하고 이후 사용자가 해당 활동에 대한 설정을 프론트엔드쪽에서 조절할 수 있다. 백엔드에서는 조명의 설정이 이전될 때는 사용자의 IOT기기(스마트폰, 웨어러블 기기)나 다른 스마트홈 기기에 팝업 메시지를 보낸다. 본 애플리케이션은 사용자가 애플리케이션을 켜놓지 않아도 백그라운드에서 지속적으로 활동하고 정보를 수집하고 있을 것이다.



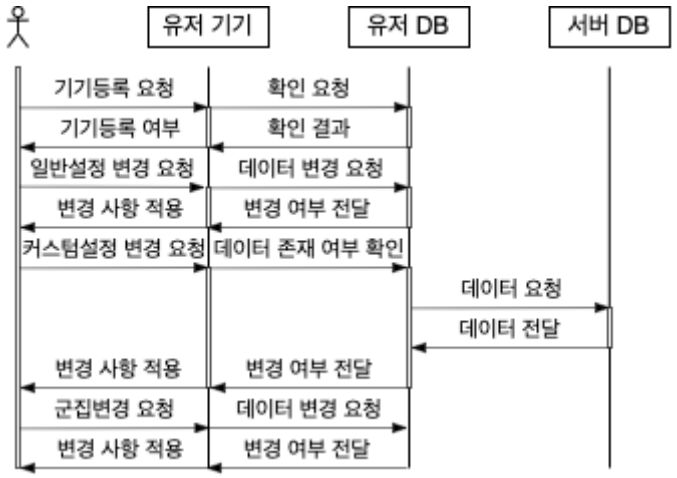
[그림 1] Overall MVC pattern

3.2.1. Context Diagram



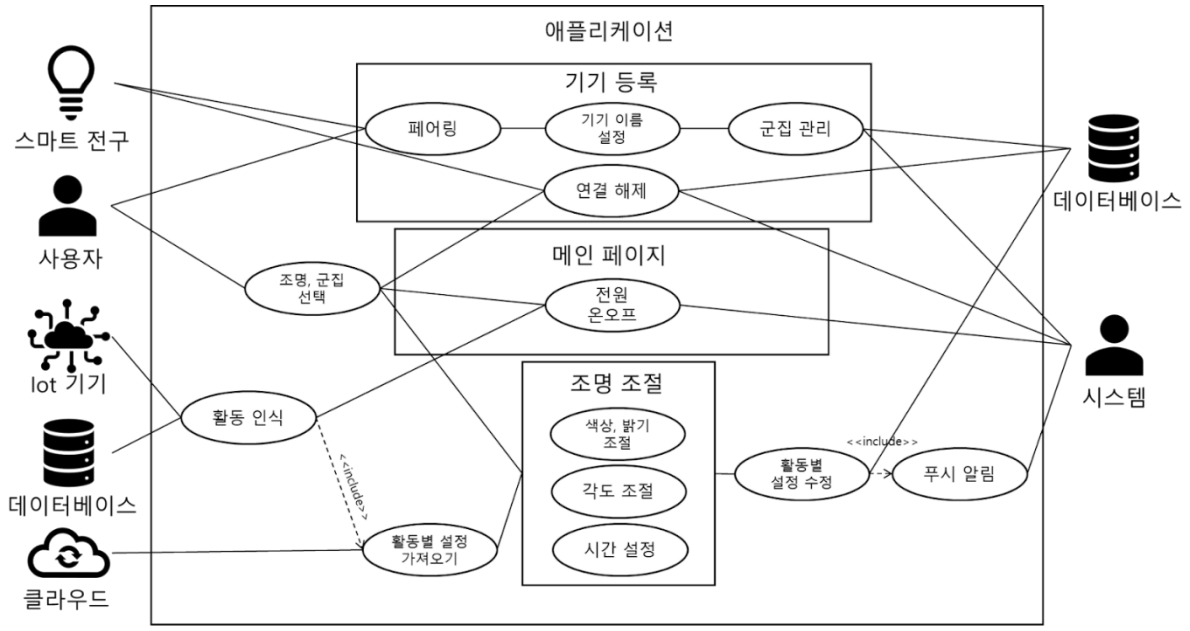
[그림 2] Overall context diagram

3.2.2. Sequence Diagram



[그림 3] overall sequence diagram

3.2.3. Use Case Diagram



[그림 4] overall use case diagram

4. System Architecture - Frontend

4.1. Objectives

이 장에서는 frontend의 구조에 대해 설명한다. 설명에는 각 subcomponent의 속성 및 방법 과 components 사이의 관계를 포함한다.

4.2. Subcomponents

4.2.1. 메인 페이지

메일 페이지 class에서는 전체 조명을 켜고 끌 수 있다.

하나 이상의 군집을 가진 방들이 리스트로 표시된다.

리스트의 요소에 해당하는 방 버튼을 누르면 해당 방에 있는 군집들을 표시하는 팝업창이 실행된다.

방 버튼 내부의 토글 스위치를 누르면 방 전체의 조명을 온/오프 할 수 있다.

4.2.1.1. Attributes

- rooms : 각 방들의 정보를 저장하고 있다

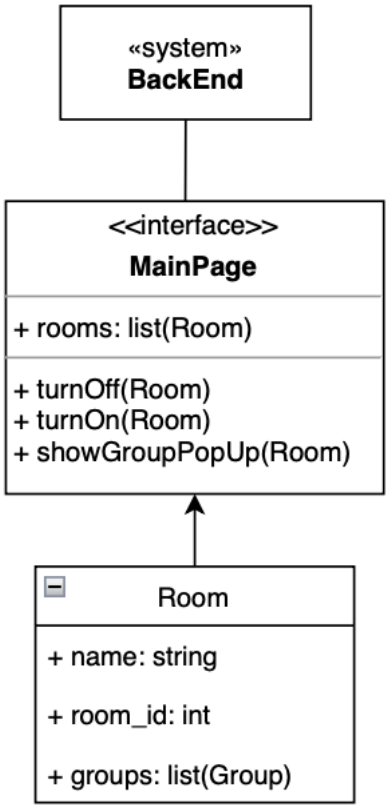
4.2.1.2. Methods

- turnOff() : 선택된 방의 조명을 끈다

- turnOn() : 선택된 방의 조명을 켠다

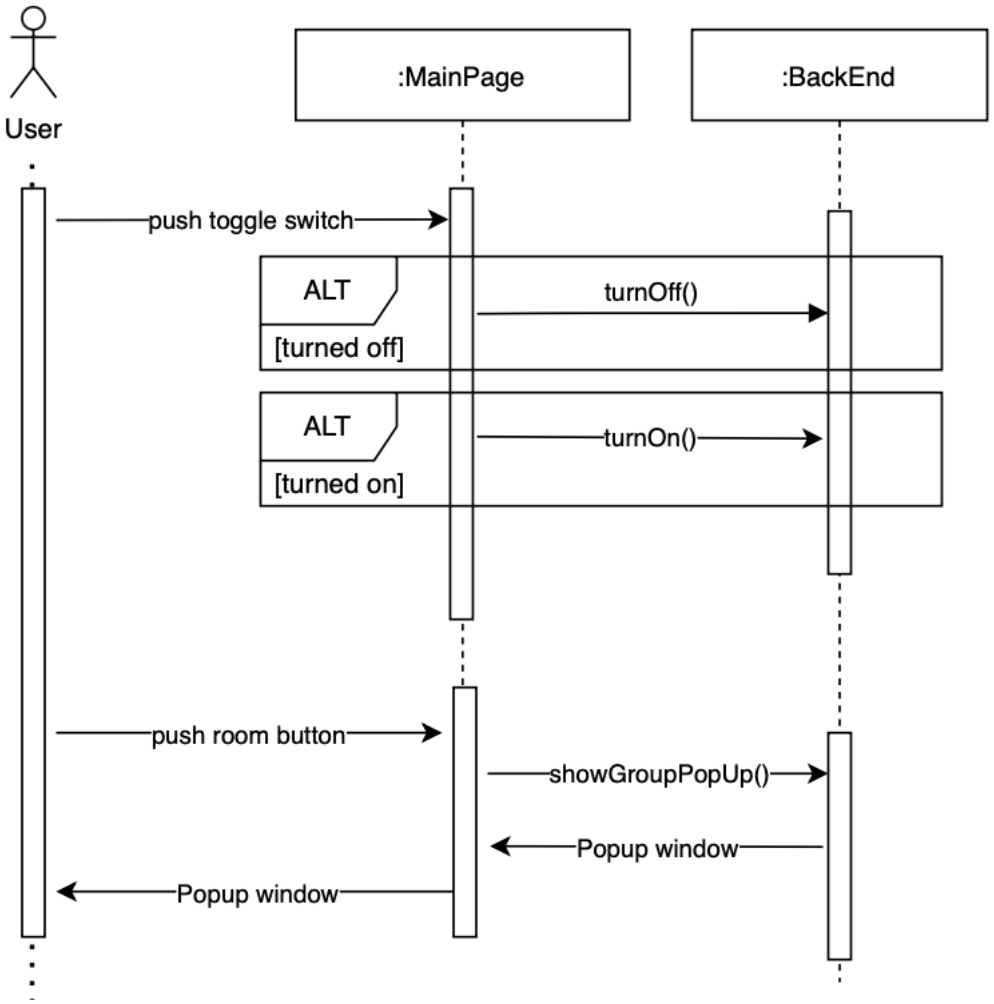
- showGroupPopUp() : 특정 방을 선택했을 때 그 방의 군집별로 조명을 껐다 켰다 할 수 있는 팝업창을 실행한다.

4.2.1.3. Class Diagram



[그림 5] main page class diagram

4.2.1.4. Sequence Diagram



[그림 6] main page sequence diagram

4.2.2. 그룹별 조명관리

메인 페이지 class의 방 리스트에서 특정 방 버튼을 누르면 실행되는 팝업창이다.

선택된 방에 속한 모든 군집들을 리스트로 표시한다.

리스트에 있는 군집에서 토글 스위치를 누르면 해당 군집의 모든 조명을 온/오프 할 수 있다.

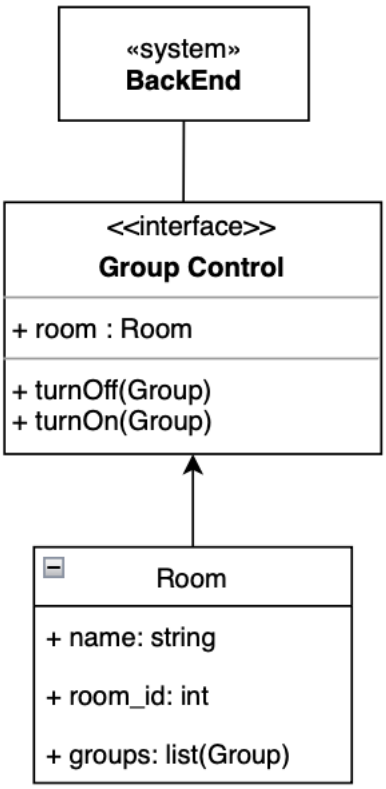
4.2.2.1. Attributes

- room : 방에 대한 정보를 저장한다. 방에 있는 모든 그룹들에 대한 정보가 담겨 있다.

4.2.2.2. Methods

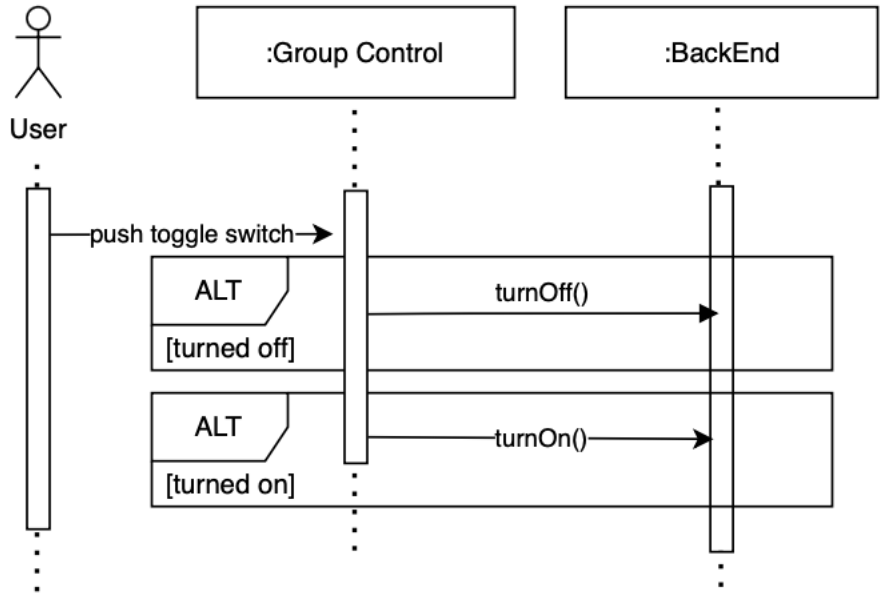
- turnOn()
- turnOff()

4.2.2.3. Class Diagram



[그림 7] group control class diagram

4.2.2.4. Sequence Diagram



[그림 8] group control sequence diagram

4.2.3. 디바이스 관리

디바이스 관리 class는 조명과 군집 정보를 가져온다. 현재 페어링된 조명과 페어링되지 않았지만 신호가 감지된 조명을 보여준다.

페어링되지 않은 조명 카드에서 connect 버튼을 누르면 조명을 페어링하고 조명에 관한 정보를 등록학는 팝업창을 실해한다.

연결되어 있는 조명 카드에서 disconnect 버튼을 누르면 조명을 삭제할 수 있는 팝업창을 실행한다.

4.2.3.1. Attributes

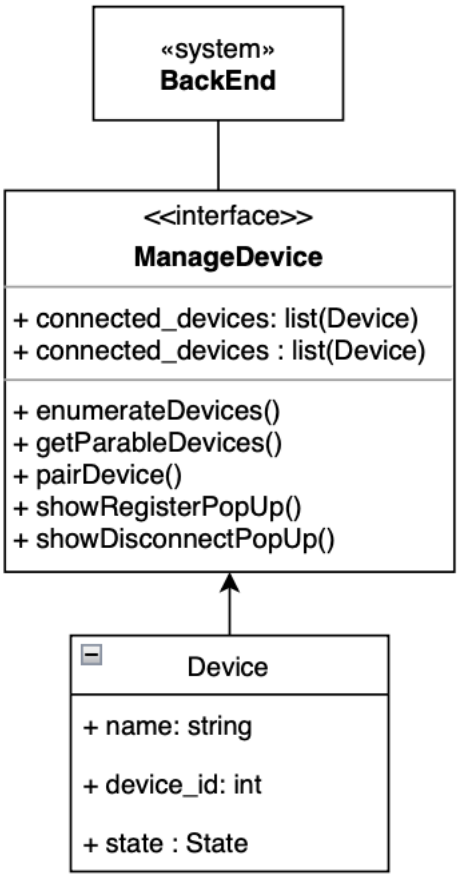
- connected_devices : 등록되어 있는 디바이스들이다

- parable_devices : 등록되어 있지 않지만 연결 가능한 디바이스들이다.

4.2.3.2. Methods

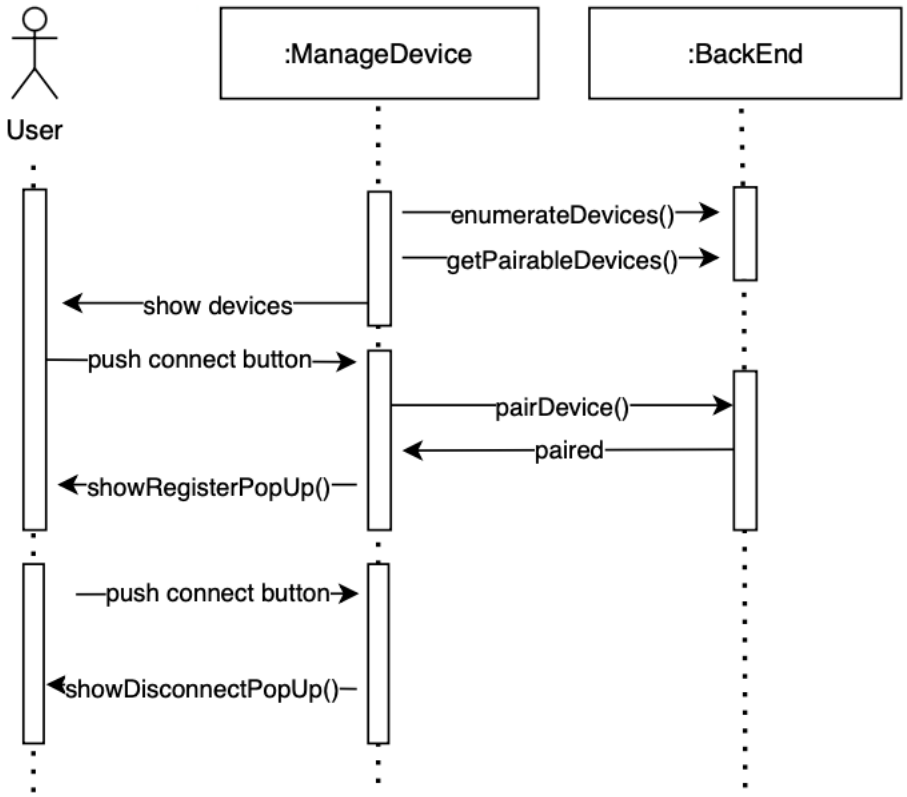
- enumerateDevices() : 현재 연결된 모든 디바이스에 관한 정보를 가져온다.
- getPairableDevices() : 현재 연결 가능한 모든 디바이스에 관한 정보를 가져온다
- pairDevice() : 디바이스와 페어링하고, 조명에 관한 정보를 초기화한다.
- showRegisterPopUp() : 조명에 관한 정보를 입력할 수 있는 팝업창을 띄운다.
- showDisconnectPopUp() : 조명을 삭제할 수 있는 팝업창을 띄운다.

4.2.3.3. Class Diagram



[그림 9] manage device class diagram

4.2.3.4. Sequence Diagram



[그림 10] manage device sequence diagram

4.2.4. 디바이스 등록

페어링된 기기의 이름과 어떤 방에 속한 전구인지를 입력받고, 이를 데이터베이스에 저장하는 팝업창이다.

4.2.4.1. Attributes

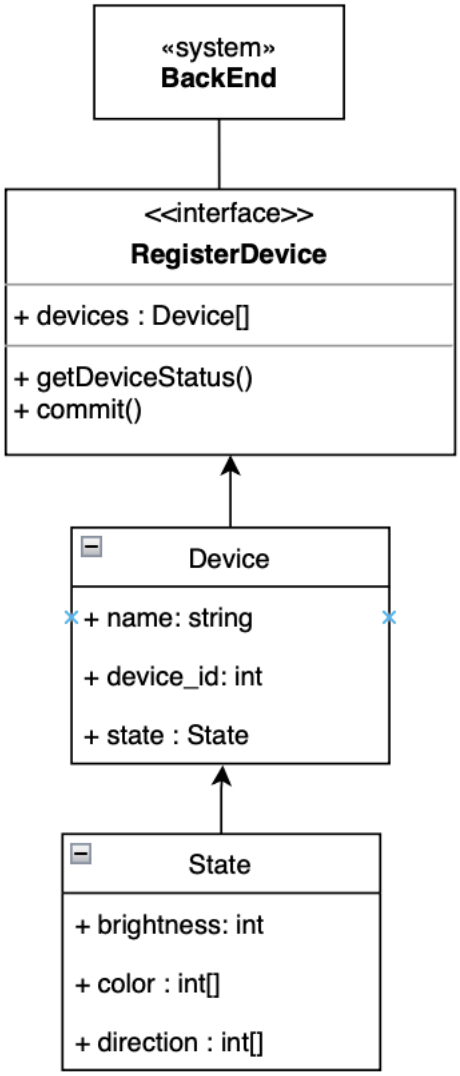
- devices: 조명을 모델링한 클래스 Device 타입의 리스트이다.

4.2.4.2. Methods

- getDeviceStatus() : 집 안에 있는 모든 전구와 군집의 정보를 가져온다.

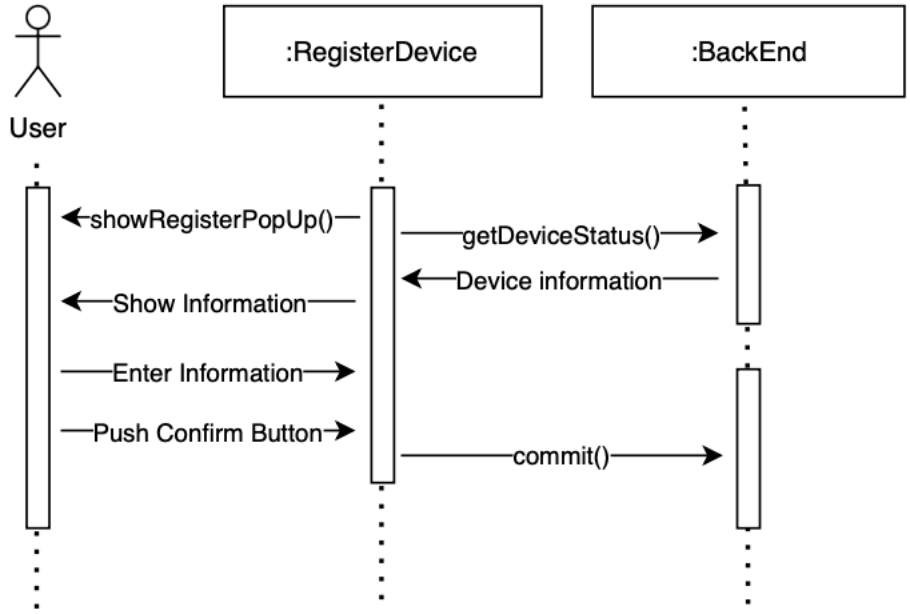
- commit() : 입력된 정보를 데이터베이스에 업데이트한다.

4.2.4.3. Class Diagram



[그림 11] register device class diagram

4.2.4.4. Sequence Diagram



[그림 12] register device sequence diagram

4.2.5. 디바이스 삭제

디바이스 관리 class에서 다바이스 삭제 버튼을 눌렀을 때 실행되는 팝업창이다.

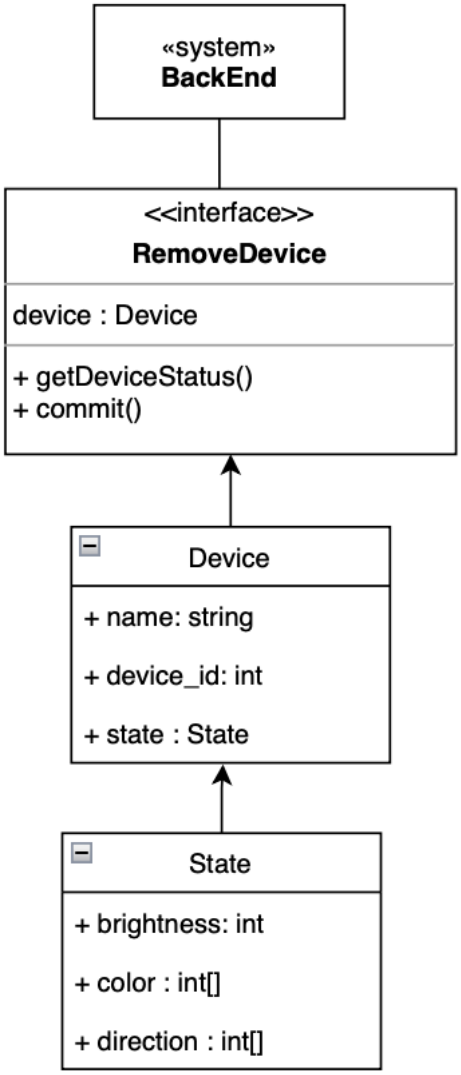
4.2.5.1. Attributes

이미 선택된 디바이스를 삭제하는 것이기 때문에 attribute를 가지지 않는다.

4.2.5.2. Methods

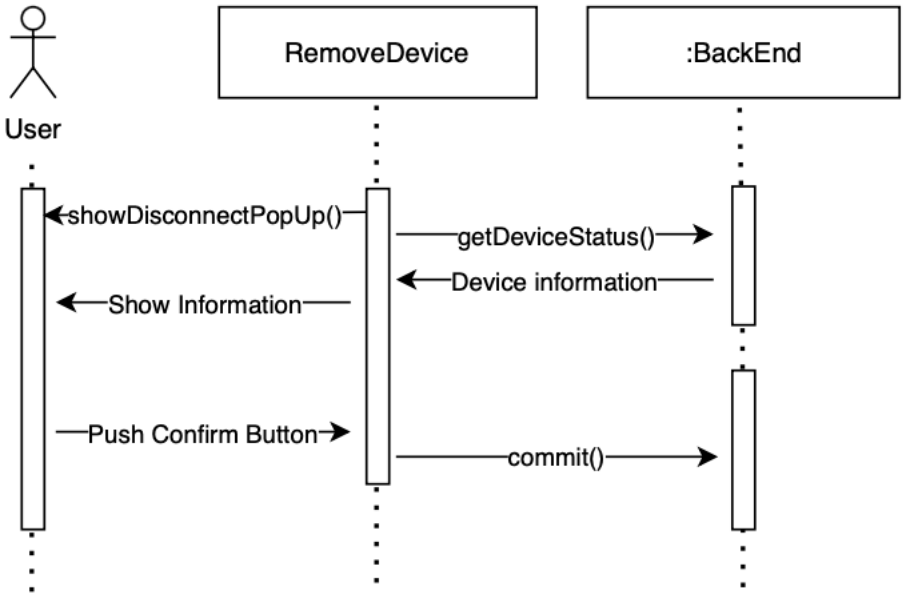
- getDeviceStatus()
- commit() : 조명이 삭제되었다는 것을 데이터베이스에 업데이트한다.

4.2.5.3. Class Diagram



[그림 13] remove device class diagram

4.2.5.4. Sequence Diagram



[그림 14] remove device sequence diagram

4.2.6. 조명 설정

조명을 설정하는 class이다.

4.2.6.1. Attributes

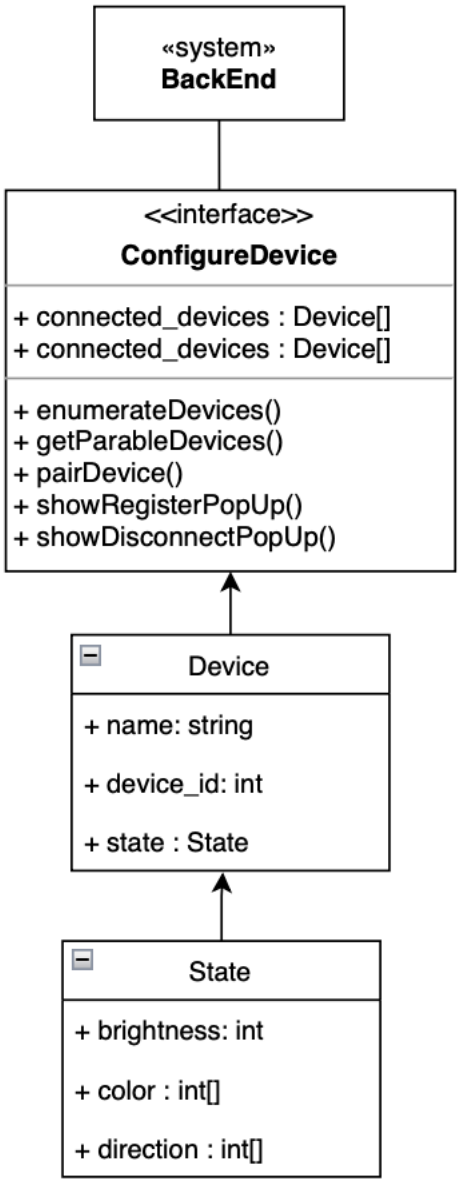
- device_state : 조명의 현재 설정

4.2.6.2. Methods

- getDeviceStatus()
- setBrightness()
- setColor()
- setDirection()

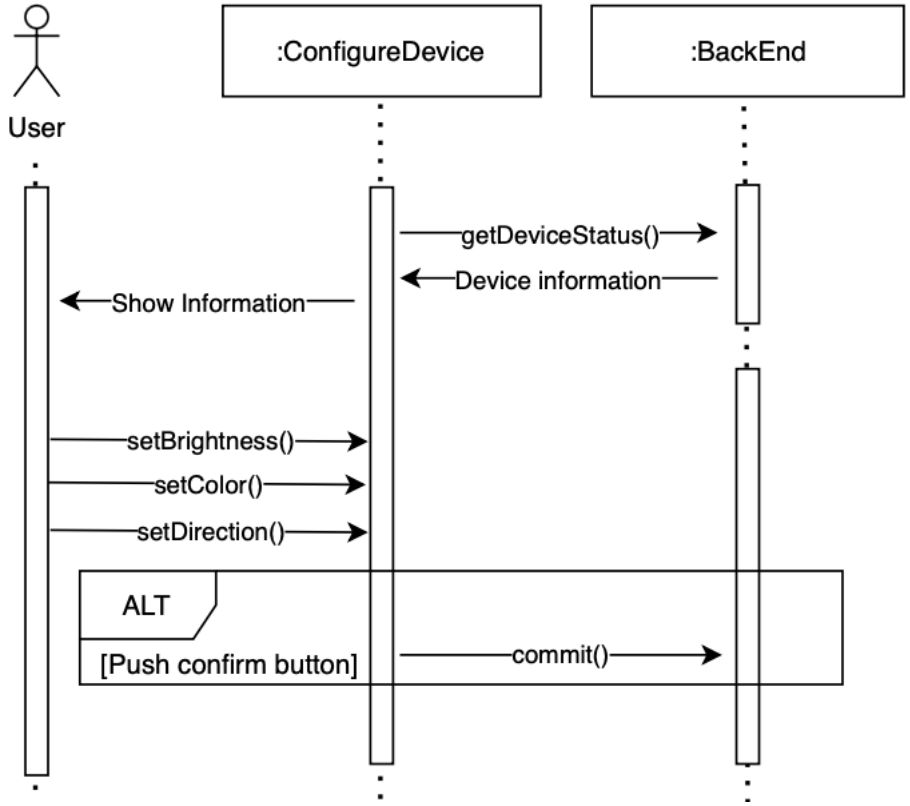
- commit()

4.2.6.3. Class Diagram



[그림 15] configure device class diagram

4.2.6.4. Sequence Diagram



[그림 16] configure device sequence diagram

4.2.7. 활동별 설정 나열

저장된 활동별 설정들을 나열한다.

4.2.7.1. Attributes

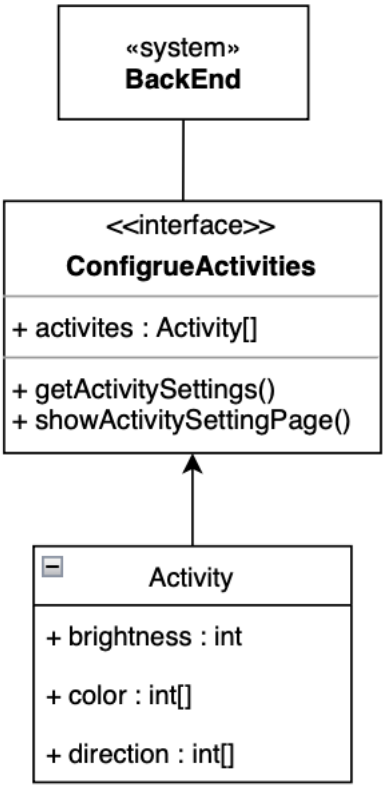
- activities : 특정 활동의 종류와, 해당 활동을 할 때 조명의 상태를 저장한다.

4.2.7.2. Methods

- getActivitySettings()

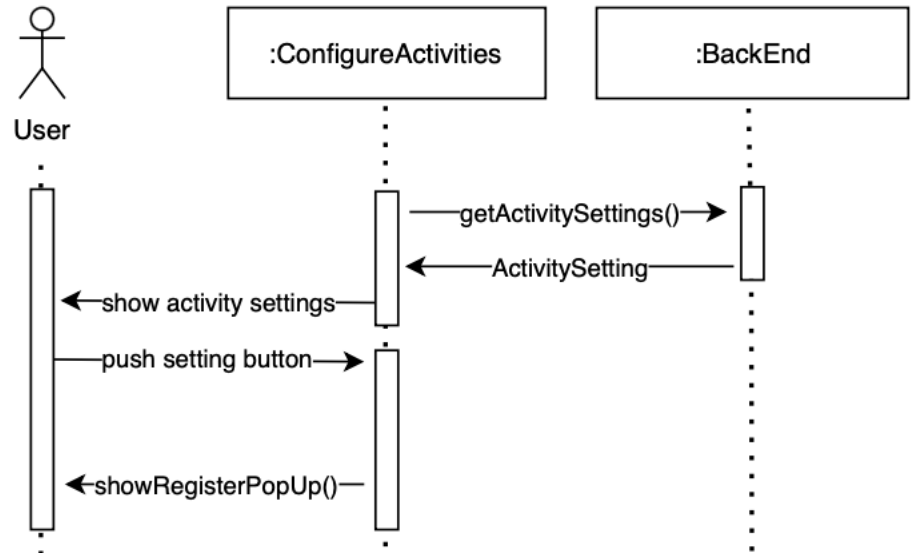
- showActivitySettingPage() : 사용자가 선택한 활동별 설정을 편집할 수 있는 화면으로 이동한다.

4.2.7.3. Class Diagram



[그림 17] configure activities class diagram

4.2.7.4. Sequence Diagram



[그림 18] configure activities sequence diagram

4.2.8. Configure Activity Setting

활동별 설정 나열 class에서 선택된 특정 활동별 설정을 설정한다.

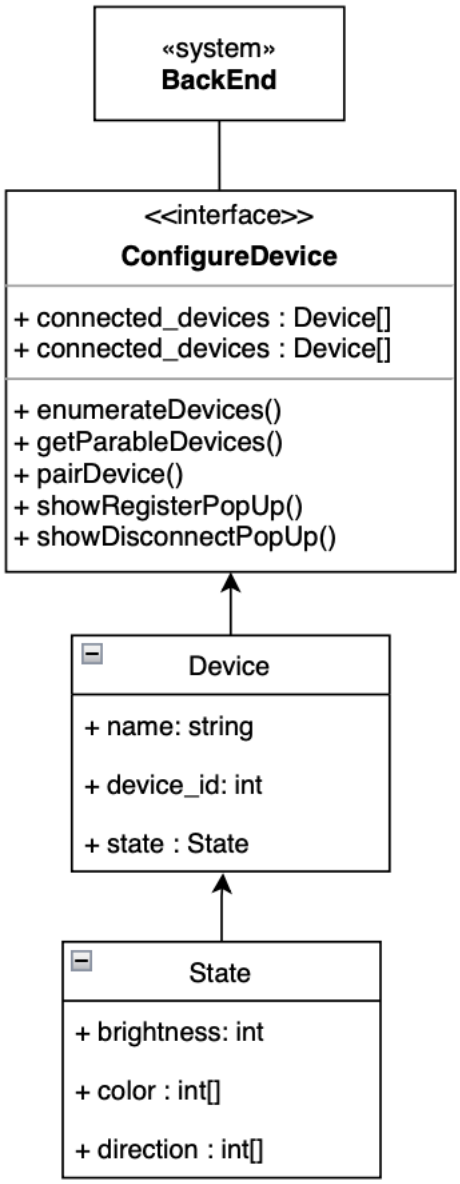
4.2.8.1. Attributes

device_states : 군집에 있는 모든 조명들의 설정을 저장한다.

4.2.8.2. Methods

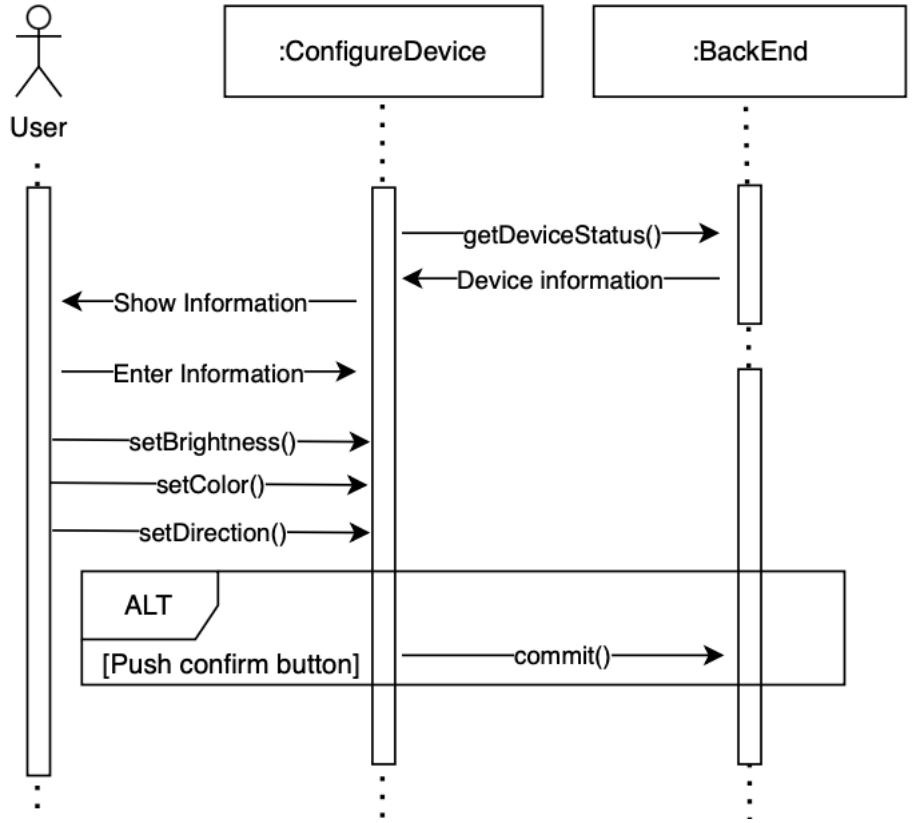
- setBrightness()
- setColor()
- setDirection()
- commit()

4.2.8.3. Class Diagram



[그림 19] configure device class diagram

4.2.8.4. Sequence Diagram



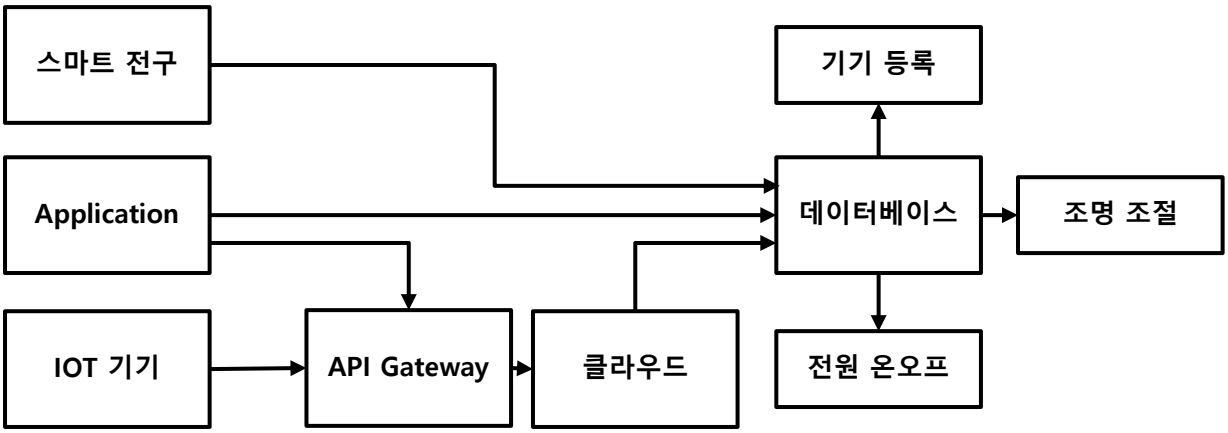
[그림 20] configure device sequence diagram

5. System Architecture – Backend

5.1. Objctives

이 장에서는 backend의 구조에 대해 설명한다.

5.2. Overall Architecture

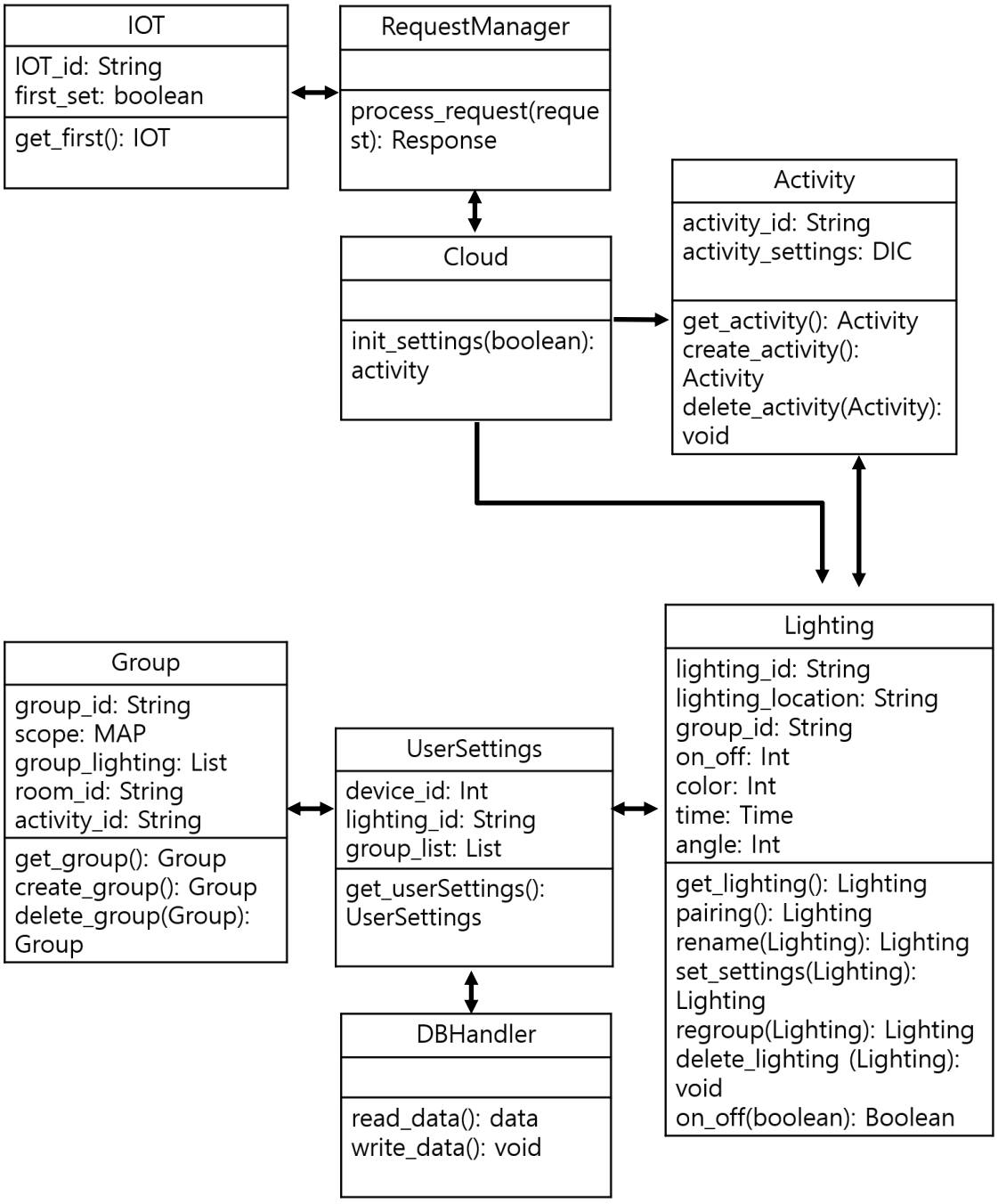


[그림 21] overall system architecture

전반적인 시스템의 구조는 위 그림과 같다. 이 시스템은 크게 두가지의 구조로 이루어져 있다. 먼저 초기에 API Gateway를 통해 IOT 기기와 Application의 요청을 받아 클라우드로부터 정보를 받아오고 이를 다시 디바이스의 데이터베이스에 저장하는 구조가 있다. 두번째는 frontend로부터 요청을 받아 데이터베이스로 전달되는 구조이다. 이렇게 데이터베이스에 저장된 정보는 시스템이 필요로 하는 형태로 사용자에게 전달된다.

5.3. Subcomponents

5.3.1. Class Diagram



[그림 22] backend class diagram

1. IOT
    - IOT_id: IOT 디바이스의 이름
    - first_set: IOT 디바이스가 처음 연결된 것인지 boolean 값으로 나타냄
    - get_first: IOT 디바이스 연결 여부를 나타내는 함수

2. RequestManager
    - process_request: API gateway로 부터 받은 요청을 처리하는 함수

3. Cloud
    - init_settings: 디바이스가 처음 연결되었을 때 패턴을 인식해서 기본 설정을 불러오는 함수

4. Activity
    - activity_id: 활동의 이름
    - activity_settings: 활동별 조명의 설정
    - get_activity: 활동에 대한 정보를 불러오는 함수
    - create_activity: 새로운 활동을 생성하는 함수
    - delete_activity: 기존의 활동을 삭제하는 함수

5. Group
    - group_id: 군집 이름
    - scope: 군집 지도
    - group_lighting: 군집내 속하는 전구 리스트

- room_id: 군집이 속해 있는 방 명
- activity_id: 활동의 이름
- get_group: 군집의 정보를 불러오는 함수
- create_group: 새로운 군집을 생성하는 함수
- delete_group: 군집을 삭제하는 함수

6. Lighting
    - lighting_id: 전구 이름
    - lighting_location: 전구가 있는 방 이름
    - group_id: 전구가 속해 있는 군집 이름
    - on_off: 조명의 온오프 상태
    - color: 조명의 색상
    - time: 조명의 타이머
    - angle: 조명의 각도
    - get_lighting: 전구의 정보를 불러오는 함수
    - pairing: 전구를 애플리케이션과 연결하는 함수
    - rename: 전구의 이름을 바꾸는 함수
    - set_settings: 조명을 설정하는 함수
    - regroup: 새로운 군집으로 이동하는 함수
    - delete_lighting: 전구 연결을 해제하는 함수
    - on_off: 전구를 켜고 끄는 함수

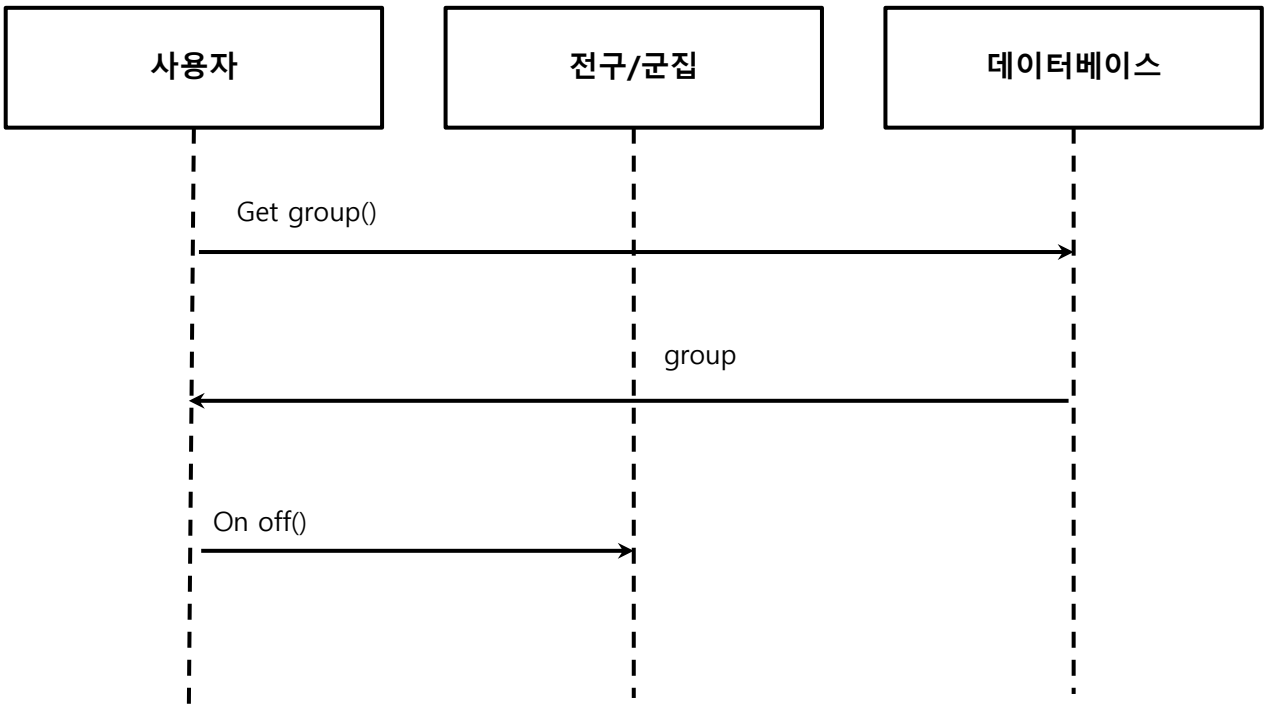
7. UserSettings

- device_id: 사용자 디바이스의 고유번호
- lighting_id: 전구 이름
- group_list: 군집들의 집합
- get_userSettings: 사용자 설정사항을 불러오는 함수

8. DBHandler
   - read_data: 데이터를 불러오는 함수
   - write_data: 데이터를 작성하는 함수

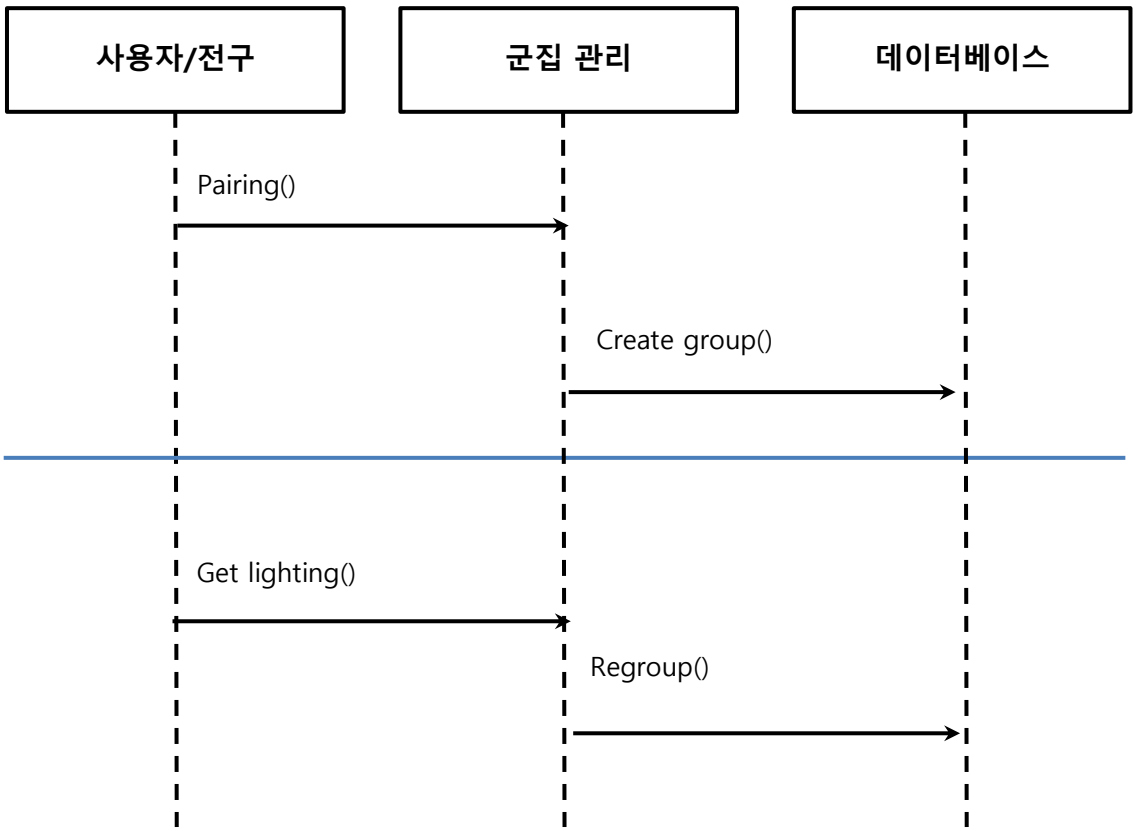
5.3.2. Sequence Diagram

1. 조명 온오프



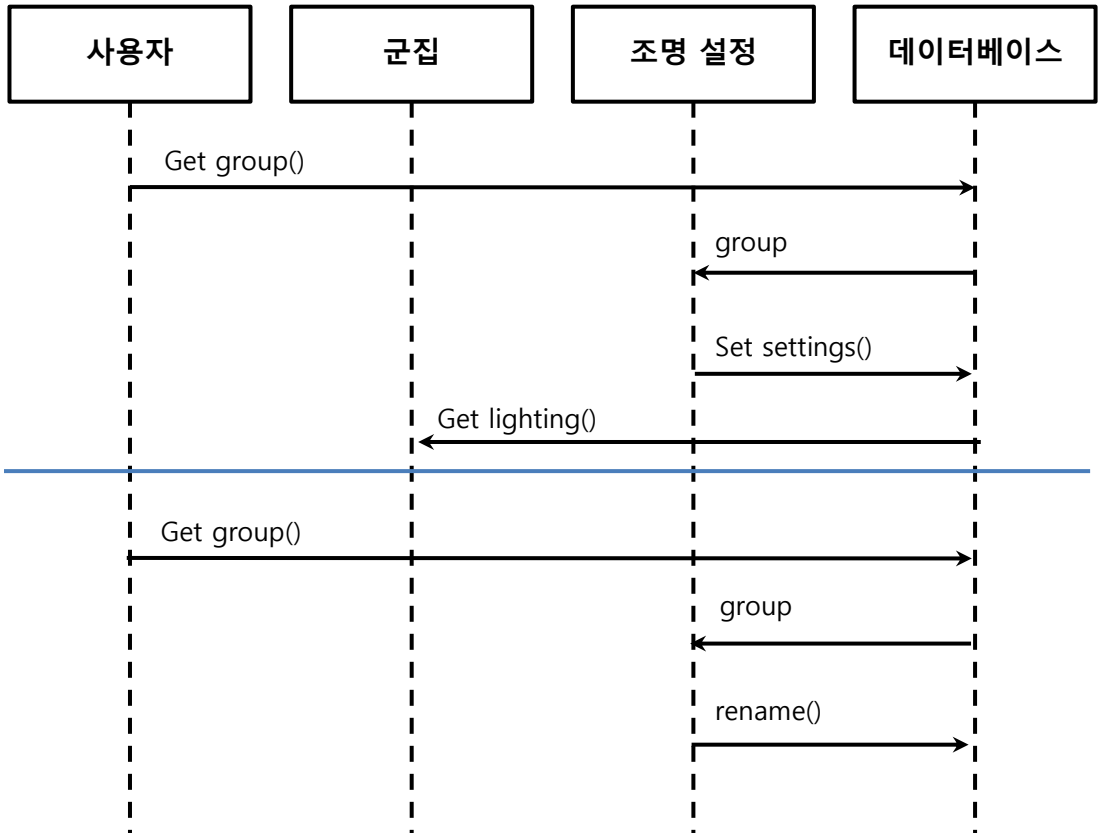
[그림 23] on/off sequence diagram

2. 군집관리



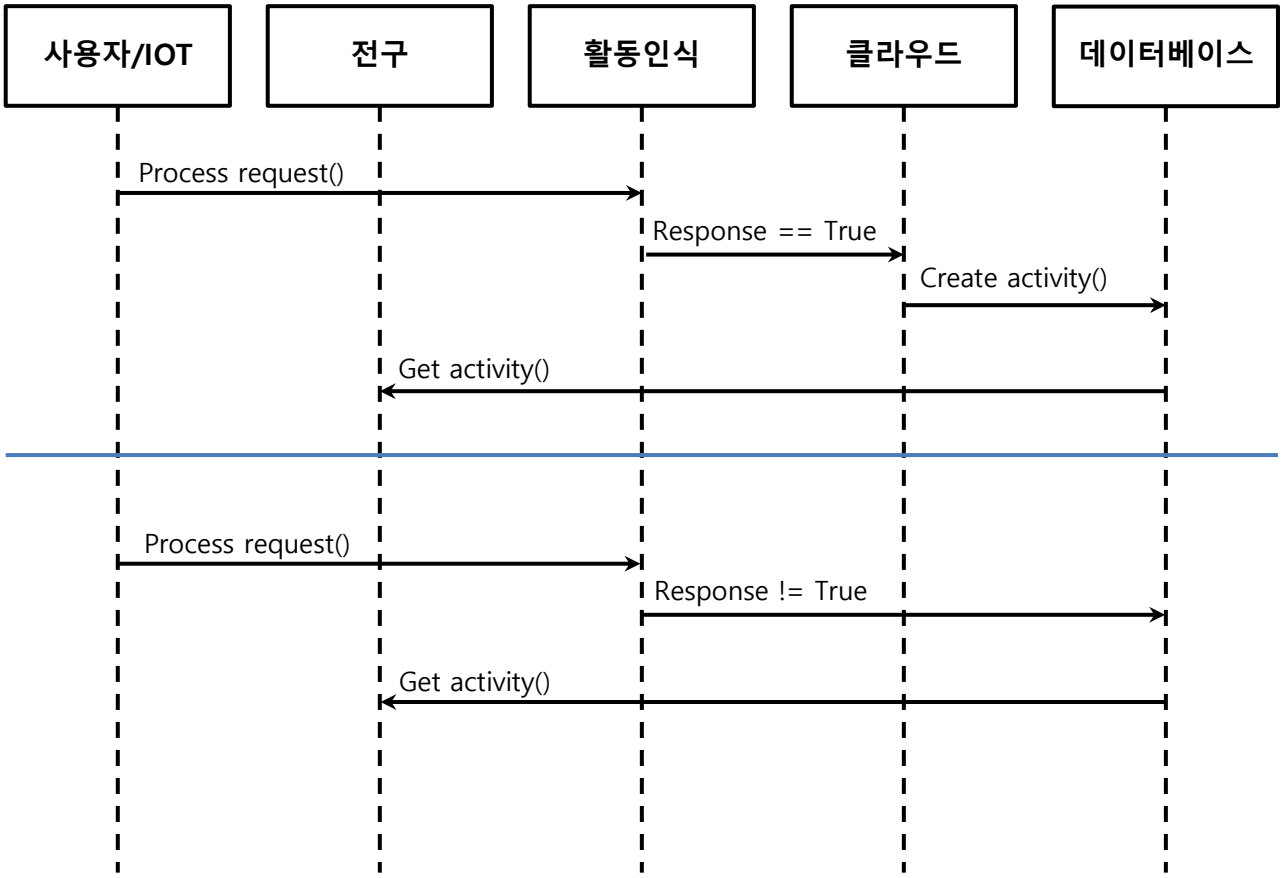
[그림 24] group manage sequence diagram

3. 조명 설정



[그림 25] light settings sequence diagram

4. 활동 인식



[그림 26] get activity sequence diagram

6. Protocol Design

6.1. Objectives

이 장에서는 애플리케이션과 서버 간의 통신이 어떻게 이루어지는지, 통신이 이루어질때 어떤 데이터를 주고 받는지에 대해 정의하고 설명한다.

6.2. 전달 방식

6.2.1. JSON

JSON은 데이터를 저장하거나 가벼운 데이터를 전송하는데 쓰이는 자바스크립트 객체 표기법의 약어이다. 흔히 자바스크립트에서 오브젝트를 만들 때 사용하는 표현이다. JSON 표현은 사람과 기계 모두 이해하기 쉽고 용량이 작기 때문에 최근에는 JSON이 XML을 대체하여 데이터 전송에 널리 사용되고 있다. JSON은 프로그래밍 언어가 아니라 단순히 데이터를 표현하는 방법이다.

6.2.2. HTTP

HTTP는 Hyper Text Transfer Protocol의 약자로, 인터넷에서 데이터를 주고받을 수 있는 프로토콜입니다. HTTP는 TCP/IP 단으로 구성되어 있으며, 요청과 응답으로 통신을 한다. 애플리케이션과 같은 클라이언트가 형식에 맞춰서 요청을 하면, 서버는 이에 알맞은 응답을 한다. 클라이너트가 요청할 때 형식은 General, Headers, Body로 이루어져있다. General에는 주소, Request 종류, Client 상태, 규칙 등이 들어간다. Header에는 요청에 대한 정보를 담고 있고 Body에는 Header에 맞게 정보를 보낼 수 있다. 여기서 전송하는 데이터는 주로 JSON형태로 되어있다.

본 프로젝트에서는, HTTP로 일반적인 통신을 다룬다.

6.2.3. OAuth

OAuth는 인터넷 사용자들이 비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보에 대해 애플리케이션의 접근 권한을 부여할 수 있는 공통적인 수단으로서 사용되는,

개방형 표준이다. 이 메커니즘은 아마존, 구글, 페이스북, 마이크로소프트, 트위터와 같은 여러 회사에서 사용되며, 사용자들이 제3자 응용 프로그램이나 웹 사이트의 계정에 대한 정보를 공유할 수 있게 한다. OAuth를 사용하기 전에는 인증방식에 대한 기준이 없어 기존 기본 인증 ID와 비밀번호를 사용했는데, 이는 보안 측면에서 취약한 구조이다. Non-basic 인증의 경우 각 애플리케이션이 자체 개발한 업체의 방식에 따라 사용자를 확인했다. OAuth는 표준화된 인증 방법이다. OAuth를 사용하면 이 인증을 공유하는 응용 프로그램에 별도의 인증이 필요하지 않다. 따라서 여러 애플리케이션을 통합하고 사용할 수 있게 된다.

6.3. Authentication

6.3.1. 기기등록

- Request

Attribute	Detail	
Protocol	OAuth	
Request body	Device_id	Light device id
	Request token	Token for OAuth
	User	User's device information

- Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 401 (unauthorized)	
	HTTP 404 (Not found)	
	HTTP 500 (Not found)	
Success response body	Access Token	Token for register light
	Message	Message: "Link success"

Failure response body	Message	Message: "Link fail"

6.3.2. 일반 설정 변경하기

- Request

Attribute	Detail	
URI	/user/:IOT_id/setting	
Method	POST	
Parameter	Udrt	User's device information
	Interest	User interest / purpose
Header	Authorization	User authentication

- Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for change setting
	Message	Message: "Change success"
Failure response body	Message	Message: "Change fail"

6.3.3. 커스텀 설정 변경하기

- Request

Attribute	Detail	
URI	/user/:IOT_id/custion_setting	
Method	POST	
Parameter	Udrt	User's device information
	Interest	User interest / purpose

Header	Authorization	User authentication

- Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for change custom setting
	Message	Message: "Change success"
Failure response body	Message	Message: "Change fail"

6.3.4. 커스텀 초기 설정 불러오기

- Request

Attribute	Detail	
URI	/server/database/init_setting/action_id	
Method	Get	
Parameter	User	User's device information
	Action_id	User's action
Header	Authorization	User authentication

- Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 404 (Not found)	
Success response body	Access Token	Token for access
	Map view	View the close range of map

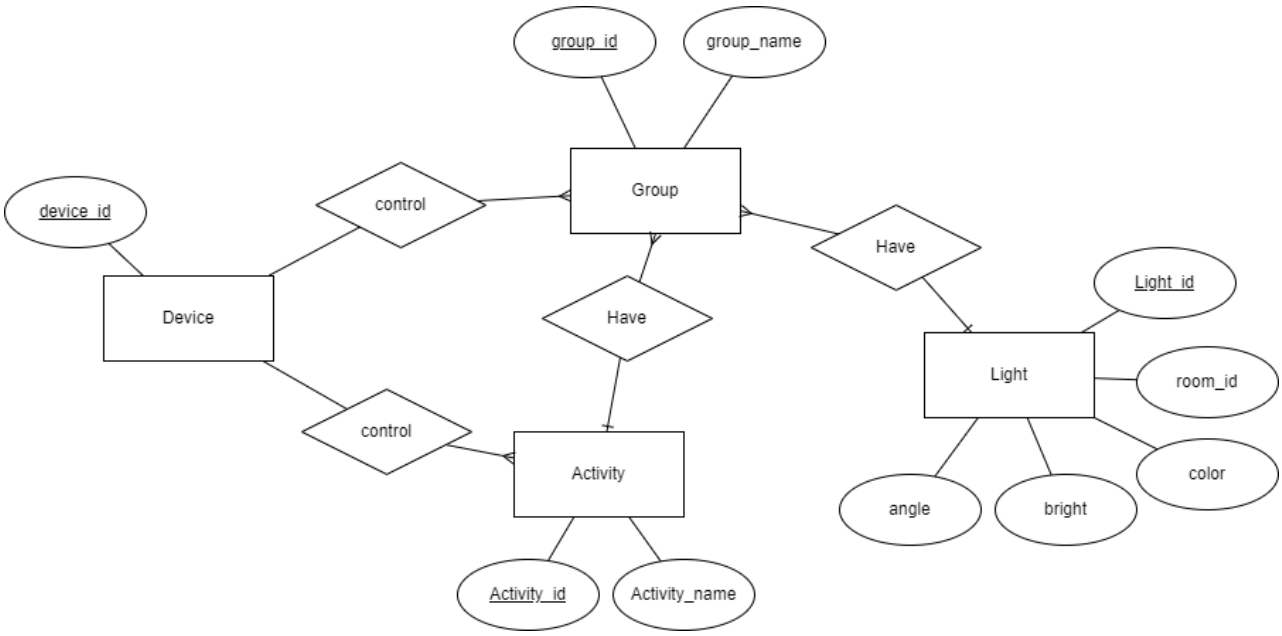
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

7. Database Design

7.1. Objectives

이 장에서는 시스템 데이터 구조와 그 데이터들이 데이터베이스에서 어떻게 표현되는지 설명한다. 데이터베이스의 ER diagram과 Entites, Relat, SQL DDL을 포함한다.

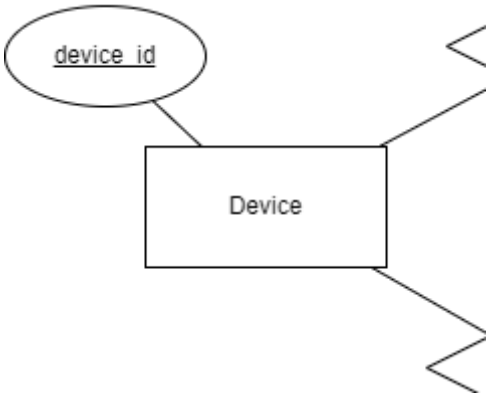
7.2. ER diagram



[그림 27] ER diagram

7.2.1 Entities

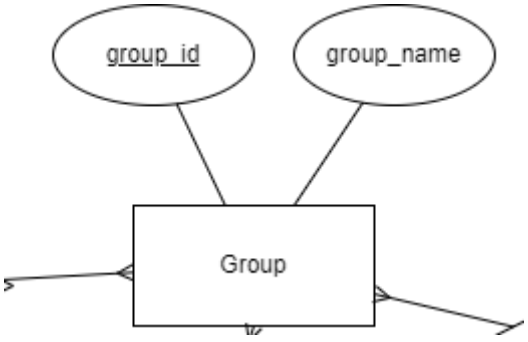
7.2.1.1. Device



[그림 28] device entity

이 entity는 디바이스에 관 정보를 포함한다. 기본키로 divice_id가 있으며, Group, Activity와 일대다 관계를 맺고 있다.

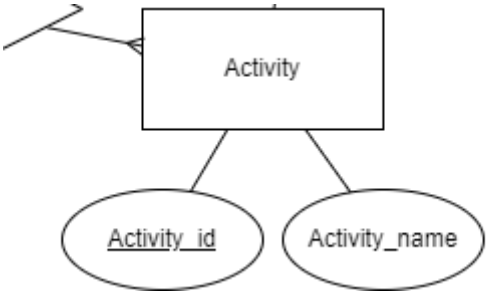
7.2.1.2. Group



[그림 29] group entity

이 entity는 군집에 관한 정보를 포함한다. 기본키로 group_id가 있고, 이름에 대한 정보가 있다. Device, Activity, Light와 다대일 관계를 맺고 있다.

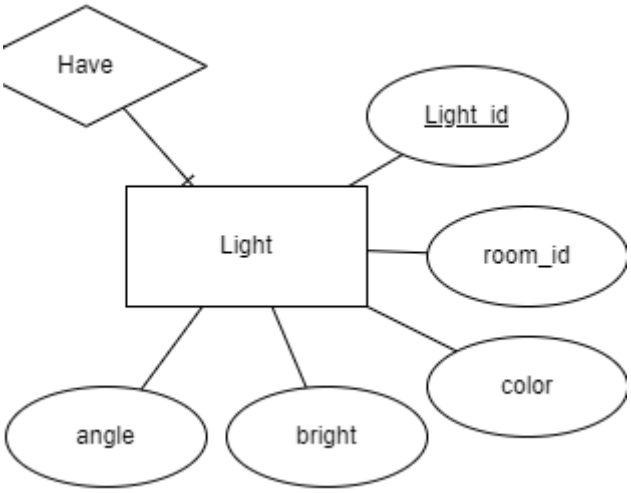
7.2.1.3. Activity



[그림 30] activity entity

이 entity는 활동에 관한 정보를 포함한다. 기본키로 Activity_id가 있고, 이름에 대한 정보가 있다. Device와 다대일 관계를 맺고 있고, Group과 일대다 관계를 맺고 있다.

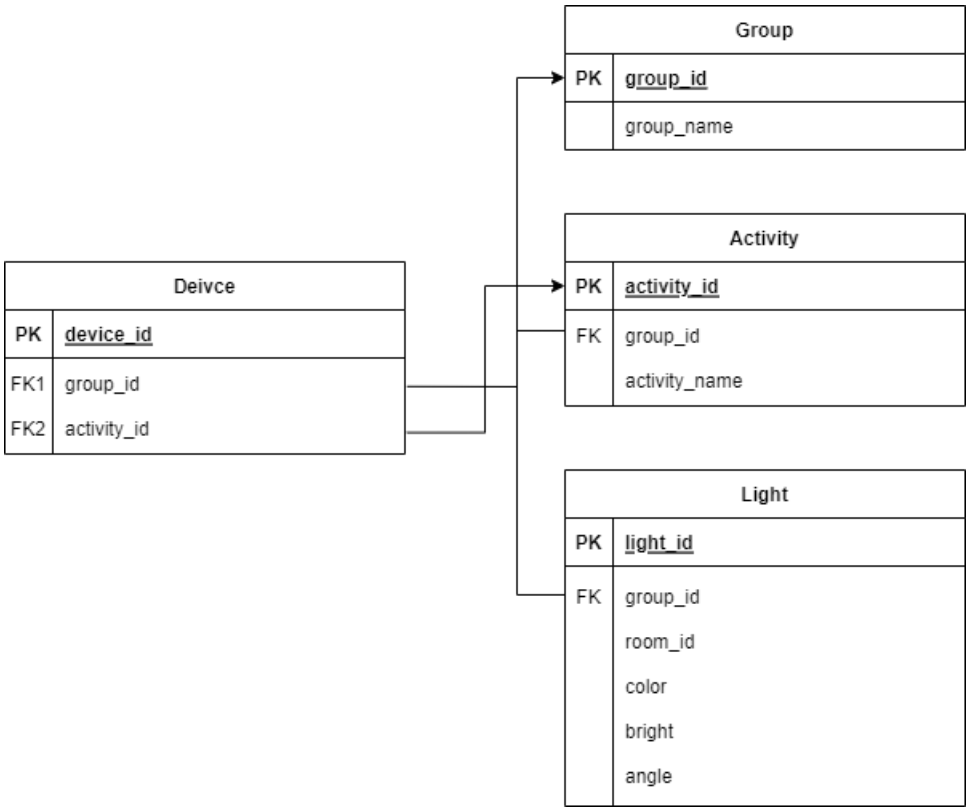
7.2.1.4. Light



[그림 31] light entity

이 entity는 조명에 관한 정보를 포함한다. 기본키로 light_id가 있고, 속한 방의 이름, 색상, 밝기, 각도 정보가 포함되어 있다. Group과 일대다 관계를 맺고 있다.

7.3. Relational schema



[그림 32] relational schema

7.4. SQL DDL

7.4.1. Device

```
CREATE TABLE Device
{
        device_id INT NOT NULL
        group_id INT NOT NULL
        activity_id INT NOT NULL
        PRIMARY KEY(device_id)
        FOREIGN KEY(group_id) REFERENCES Group(group_id)
```

```
        FOREIGN KEY(activity_id) REFERENCES Activity(activity_id)
};
```

7.4.2. Group

```
CREATE TABLE Group
{
        group_id INT NOT NULL
        group_name VARCHAR NOT NULL
        PRIMARY KEY(group_id)
};
```

7.4.3. Activity

```
CREATE TABLE Activity
{
        activity_id INT NOT NULL
        group_id INT NOT NULL
        activity_name VARCHAR NOT NULL
        PRIMARY KEY(activity_id)
        FOREIGN KEY(group_id) REFERENCES Group(group_id)
};
```

7.4.4. Light

```
CREATE TABLE Light
{
```

```
        light_id INT NOT NULL
        group_id INT NOT NULL
        room_id INT NOT NULL
        color INT NOT NULL
        bright INT NOT NULL
        angle INT NOT NULL
        PRIMARY KEY(light_id)
        FOREIGN KEY(group_id) REFERENCES Group(group_id)
};
```

8. Testing Plan

8.1. Objectives

이 장에서는 test 계획을 development testing, release testing, user testing으로 설명하고 있다. 이러한 test들은 잠재적인 오류와 결함을 줄이는데 도움을 주고 사용자에게 안정성 있는 기능성을 제공하게 해준다.

8.2. Testing Policy

8.2.1. Development Testing

Development Testing은 개발 단계에서 수행되는 소프트웨어 개발 접근 방식으로 주요 목적은 프로그램 안정화이다. 이 단계에서는 static code 분석, data flow 분석, peer code 리뷰, unit 테스팅이 이루어진다. 이를 통해 performance, reliability, security을 달성하고자 한다.

이러한 Development Testing은 소프트웨어 개발 수명 주기(SDLC)에서 더 높은 수준의 효율성을 가능하게 하고 소프트웨어 오류의 영향을 줄일 수 있다. 이는 고객에게 새로운 기능과 버그 수정을 신속하게 제공할 수 있게 해준다.

1. Performance

   Smart Lighting With IOT 시스템은 사용자에게 일상생활에 편의성을 제공하기 위한 애플리케이션이다. 따라서 기기 간의 연결이나 기타 여러 기능들이 실행되는데 오랜 시간이 걸리지 않도록 해야한다. 이를 위해 기기연결에 대한 응답 시간은 10초이대로 둘 것이고 기타 정보 저장 및 조명 온오프와 같은 기능들은 2초로 두어 각 기능들이 기대 시간 이내에 실행되는지에 대해 Testing 할 예정이다.

2. Reliability

   Smart Lighting With IOT 시스템은 다양한 IOT 기기들 간의 연결을 통해 사용자의 상황을 받아오는 애플리케이션이다. 따라서 각 기기들간 연결에 누락 및 오류가 없는지 Testing해야 한다.

3. Security

   Smart Lighting With IOT 시스템은 사용자의 주거지, 생활공간에 대한 정도를 담고 있다. 따라서 이러한 정보가 노출되지 않도록 불필요한 외부의 접근이 없어야 한다. 또한 사용자 정보의 불필요한 사용이 없는지 확인해야 한다.

8.2.2. Release Testing

release testing은 고객이나 사용자에게 공개할 시스템의 버전을 테스트하는 과정으로 주요 목표는 시스템이 사용하기 적합한지 확인하는 것이다. 따라서 시스템은 명시된 기능을 수행하고 적절한 성능과 안정성을 가지고 있으며 오작동하지 않아야 한다.

8.2.3. User Testing

user testing은 실제 사용자들과 함께 실제 상황에서 발생하는 문제들을 Testing하는 과정이다. 이 프로젝트에서는 20명의 사용자들의 의견과 조언을 받아 Smart Lighting With IOT에 대한 Testing을 진행할 예정이다.

8.2.4. Testing Case

testing case는 앞에서 언급한 내용을 바탕으로 확인한다. 또한 기능적으로 잘 작동하는지에 대한 검사도 함께 진행한다. 다음은 이에 대한 예시이다.

Item	Action	Test Case	Result
응답 시간	1. 조명 온오프 2. 조명 설정 및 저장 3. 군집 설정 및 저장	2초 이내이다.	SUCCESS
		2초 초과이다.	FAIL
응답 시간	1. IOT 기기 연결 2. 기기간 페어링	10초 이내이다.	SUCCESS
		10초 초과이다.	FAIL
기기 연결	1. 기기의 상태를 수신한다. 2. 기기와 애플리케이션 간의 연결을 시도한다.	기기 연결 성공	SUCCESS
		기기 오류 및 반응 없음	FAIL
클라우드 접속	IOT 기기가 처음 연결을 시도했다	True	클라우드 연결
		False	데이터베이스 연결
저장	1. 조명 온오프 2. 조명 설정 및 저장 3. 군집 설정 및 저장 4. IOT 정보 저장 5. 사용자 정보 저장	저장된다.	SUCCESS
		누락된다.	FAIL
보안	1. 클라우드 접속 2. 기기간 연결	불필요한 접근이 없다.	SUCCESS
		불필요한 접근이 인식된다.	FAIL

9. Development Plan

9.1. Objectives

이 장에서는 개발의 필요한 기술과 환경에 대해 다룬다.

9.2. Frontend Environement

9.2.1 Adobe Photoshop (UI/UX Design)

[그림 33] Adobe Photoshop logo

어도비 포토샵은 어도비 시스템즈사에서 개발한 레스터 그래픽 편집기이다. 픽셀을 기본단위로 하는 비트맵 방식의 툴이다. 해당 툴로 사진 편집, 그래픽 디자인, 합성, 디자인 및 페인팅을 간단히 수행할 수 있다.

9.2.2 Adobe Xd (UI/UX Design)

[그림 34] Adobe Xd logo

애플리케이션 개발할때 필요한 UI/UX의 모든 해결책이 있는 프로그램이다. 해당 프로그램은 다양한 프로그램들의 프로토 타입을 제공하고 해당 제품들에 대한 의사소통 창구 또는 피드백의 장소로 사용된다.

9.2.3 Android Studio (Application)

[그림 35] Android Studio logo

안드로이드 스튜디오는 구글 플레이스토어에 등록할 애플리케이션을 만드는 프레임워크이다. 일반적으로 Kotlin이나 Java를 사용하여 앱을 개발할 수 있다. 애플리케이션을 개발할 때 인터페이스나 커뮤니티가 상당히 잘되어 있어서 진입장벽이 상대적으로 낮은 편이고 웹 애플리케이션으로의 적용도 잘되기 때문에 많이 사용하고 있다.

9.3. Backend Environement

9.3.1 Github (Open Source)

[그림 36] Github logo

깃허브는 오픈 소스가 모여있는 플랫폼으로 개인이 어떤 프로그램을 개발할 때 효율적으로 버전관리를 할 수 있게 도와주는 툴이다. 또한 협업으로 개발하는데 있어서 변경사항을 한눈에 볼 수 있게 표시해주고 겹치는 사항에 대해서 잘 처리할 수 있게 도와주는 툴이다.

9.3.2 Firebase (Data Base)

[그림 37] Firebase logo

파이어베이스는 클라우드 스토리지, 실시간 데이터베이스, 머신러닝 키트 등 다양한 기능을 제공해 모바일과 웹 애플리케이션 개발을 도와준다. 그 중 우리는 활동별 설정에 대한 초기 데이터를 저장하기 위한 데이터 베이스와 패턴을 인식하는 알고리즘을 위한 머신러닝 키트를 사용할 것이다. 모든 클라이언트가 데이터베이스에 접근할 수 있습니다.

9.3.3 SQLite Database (Data Base)

[그림 38] SQLite logo

MYSql 및 Postgre 와 같은 데이터베이스 관리 시스템입니다. SQL이지만 서버가 아닌 응용프로그램에서 사용하는 가벼운 데이터베이스이기 때문에 중소형 작업에서 장점을 보입니다.

9.3.4 Android Studio (Application)

[그림 39] Android Studio logo

애플리케이션을 외부 API에 연결하여 기능을 추가하고 Json 파일을 애플리케이션 활동에 연결할 수 있다. 또한 데이터 제어를 위해 애플리케이션을 Device 내부에 있는 데이터베이스에 연결한다.

9.4. Constraints

소프트웨어는 이 문서에 언급된 내용을 기반으로 설계 및 구현된다. 기타 세부 사항은 개발 시 개발자가 선호하는 방향으로 수행되나 다음 사항을 준수해야 한다.
이미 증명된 기술들을 사용한다.

- 필요한 기술의 license를 준수한다.
- 비용이 드는 기술들을 사용하는 것을 지양한다.
- 유저가 편하게 사용할 수 있는 방향으로 개발한다.
- 지속적인 유지를 위해 안정성 있는 개발을 한다.
- 소프트웨어 성능의 향상을 도모하는 방향으로 디자인을 결정한다.
- 소프트웨어 코드를 최적화해야 한다.
- 후에 차후 지속적인 개발을 위해 소스 코드를 명확하게 작성하고 이에 대한 코멘트들을 적어 놓아야 한다.

- 개발환경을 고려한 개발을 한다.

9.5. Assumptions and Dependencies

이 애플리케이션은 Android, IOS 기반 모바일 기기를 대상으로 한다. 또한 기기는 최소 1GB의 RAM 용량과 1Hz 프로세서를 내장하고 있어야 한다. 그 이외에 상황에서는 지원이 불가능함을 명시한다.

10. Supporting Information

10.1. Software Design Specification

이 요구사항 명세서는 IEEE Recommendation에 따라 작성되었다.

10.2. Document History

Date	Version	Description	Writer
2021/05/02	0.1	Style and overview	전체
2021/05/03	1.0	Addition of 1	주소미
2021/05/04	2.0	Addition of 2,3	지예준
2021/05/06	3.0	Addition of 4	박현제
2021/05/07	4.0	Addition of 5	주소미
2021/05/08	5.0	Addition of 6	지예준

2021/05/09	6.0	Addition of 7	남상욱
2021/05/12	7.0	Addition of 8, 9, 10	남상욱
2021/05/13	8.0	Revision of 1, 2, 3, 8, 9, 10	박현제
2021/05/14	9.0	Revision of 4, 5, 6, 7	남상욱
2021/05/15	10.0	Revision of Sructure	전체