스마트홈 전기관리 시스템

소프트웨어 아키텍처 명세

작성자: 이유훈

Revision History

|  |  |  |
| --- | --- | --- |
| Version | Date | Summary |
| 1 | 8/21 | Interim report 작성 |
| 2 |  |  |
| 3 |  |  |

내용

[1. Business Drivers 5](#_Toc206659953)

[1.1. Business Goals 5](#_Toc206659954)

[1.1.1. Stakeholders List 5](#_Toc206659955)

[1.1.2. Business Goals 6](#_Toc206659956)

[1.2. Business Constraints 8](#_Toc206659957)

[2. System Context 9](#_Toc206659958)

[2.1. System Context Diagram 9](#_Toc206659959)

[2.2. External Entity List 9](#_Toc206659960)

[2.3. External Interface List 11](#_Toc206659961)

[3. Architectural Drivers 13](#_Toc206659962)

[3.1. Use Case Model 13](#_Toc206659963)

[3.1.1. Use Case Diagram 13](#_Toc206659964)

[3.1.2. Actor List 13](#_Toc206659965)

[3.1.3. Use Case List 14](#_Toc206659966)

[3.1.4. UC-01 Title Description 15](#_Toc206659967)

[3.1.5. UC-02 Title Description 16](#_Toc206659968)

[3.1.6. *UC-03 Title* Description 16](#_Toc206659969)

[3.2. Quality Attribute Scenario 16](#_Toc206659970)

[3.2.1. QA Scenario List 16](#_Toc206659971)

[3.2.2. QA-01 Title Scenario 16](#_Toc206659972)

[3.2.3. QA-02 Title Scenario 17](#_Toc206659973)

[3.3. Architectural Constraint 17](#_Toc206659974)

[4. High Level Structure Design Description 19](#_Toc206659975)

[4.1. Domain Modeling 19](#_Toc206659976)

[4.1.1. Conceptual Class List 19](#_Toc206659977)

[4.1.2. Dynamic View 19](#_Toc206659978)

[4.1.3. Static View 20](#_Toc206659979)

[4.2. Quality Driven Architectural Design 21](#_Toc206659980)

[4.2.1. Exploring Architectural Options for *QA-01 Title* 21](#_Toc206659981)

[4.2.2. Exploring Architectural Options for *QA-02 Title* 23](#_Toc206659982)

[4.3. Component & Connector View 23](#_Toc206659983)

[4.3.1. UML Component Diagram 24](#_Toc206659984)

[4.3.2. Component List 24](#_Toc206659985)

[4.3.3. Connector List 25](#_Toc206659986)

[4.4. Deployment View 27](#_Toc206659987)

[4.4.1. Deployment Diagram 27](#_Toc206659988)

[4.4.2. Artifact Definition Diagram 29](#_Toc206659989)

[5. Detailed Component Design Description 31](#_Toc206659990)

[5.1. Design Description for *ComponentName* 31](#_Toc206659991)

[5.1.1. Overview 31](#_Toc206659992)

[5.1.2. Quality Driven Component Design 32](#_Toc206659993)

[5.1.3. Module View 34](#_Toc206659994)

[5.2. Design Description for *Component 2* 36](#_Toc206659995)

[5.3. Design Description for *Component 3* 36](#_Toc206659996)

[6. Architectural Evaluation 37](#_Toc206659997)

[*6.1.* Architectural Evaluation for *QA-01 Title* 37](#_Toc206659998)

[6.1.1. List of Risks 37](#_Toc206659999)

[6.1.2. List of Sensitivities 37](#_Toc206660000)

[6.1.3. List of Tradeoffs 38](#_Toc206660001)

[6.1.4. List of Nonrisks 38](#_Toc206660002)

[7. Appendix 39](#_Toc206660003)

[7.1. Detailed Component Specification for C&C View 39](#_Toc206660004)

[7.1.1. Component 1 Name 39](#_Toc206660005)

[7.1.2. Component 2 Name 40](#_Toc206660006)

[7.2. Interface specifications for C&C View 40](#_Toc206660007)

[7.2.1. *Interface2\_1* Interface Specification 40](#_Toc206660008)

# Business Drivers

본 장에서는 1 가구 1 시스템 보급을 목표로 하는 스마트홈 전기관리 시스템의 이해관계자, 비즈니스 목표, 비즈니스 제약사항에 대해 설명합니다. 본 시스템은 가정 내 전기 사용을 실시간으로 모니터링하고 자동 제어 및 최적화 기능을 통해 에너지 효율성을 높이며 전력 소비를 절감하는 IoT 기반 관리 시스템입니다.

## Business Goals

### Stakeholders List

|  |  |
| --- | --- |
| 이해관계자 | 역할 및 관심사항 |
| 사용자 | **[역할]**  시스템을 통해 가정 내 전력 사용량을 확인하고 가전기기를 제어하며, 에너지 절감 혜택을 받는 최종 주체  **[관심사항]**  전기 요금 절감, 원격 제어를 통한 생활의 편리성 증대, 과부하·누전 등 전기 안전사고 예방, 직관적이고 사용하기 쉬운 애플리케이션 |
| 경영진 | **[역할]**  비즈니스 전략 수립, 수익 모델 개발, 시장 경쟁력 확보 및 투자 유치  **[관심사항]**  시장 점유율 확대, 서비스 유료화 또는 데이터 기반 부가 사업을 통한 수익 창출, 경쟁사 대비 차별화된 기능(AI 최적화 등) 제공, 브랜드 이미지 제고 |
| 개발팀 | **[역할]**  시스템의 설계, 개발, 테스트 및 배포  **[관심사항]**  안정적이고 확장 가능한 시스템 아키텍처 구축, 다양한 IoT 기기 프로토콜 연동의 복잡성 해결, 실시간 데이터 처리 성능 확보, 보안 취약점 없는 견고한 시스템 개발 |
| 운영팀 | **[역할]**  배포된 전체 시스템의 생태계를 원격으로 관리하고 지원  **[관심사항]**  안정적인 OTA(Over-the-Air) 업데이트 제공, (사용자 동의 하에 수집된) 익명화된 데이터 기반의 전체 시스템 문제점 분석, 심각한 기술 문제에 대한 2선 원격 기술 지원 |
| 전력 공급사 | **[역할]**  시스템에 전력 단가, 요금제, 전력 수급 상황 등 외부 정보 제공  **[관심사항]**  정확한 데이터 연동을 통한 요금 예측 정확도 확보, 국가적 에너지 수요 관리 정책과의 연계, 자사 고객에게 부가 서비스 제공 |
| 가전기기 제조사 | **[역할]**  자사 제품(스마트 가전)이 시스템과 호환되도록 기술 규격 준수 및 연동 지원  **[관심사항]**  Matter, Zigbee 등 표준 IoT 프로토콜 지원, 자사 제품의 시스템 연동을 통한 스마트홈 생태계 확장, 연동 안정성 및 보안성 확보 |

### Business Goals

|  |  |  |  |
| --- | --- | --- | --- |
| 이해관계자 | 비즈니스 목표 | | |
| ID | 기술 | 중요도\* |
| 사용자 | BG-01 | **[목표]**  AI 기반 전력 최적화 및 원격 제어 기능을 통해, 평균 15%의 전기 요금 절감과 높은 편의성을 제공하여 서비스 만족도 90%를 달성한다.  **[아키텍처와의 연관성]**  사용자의 전력 사용 패턴을 실시간으로 분석하고, 제어 명령을 1초 이내의 지연 시간으로 IoT 기기에 전달할 수 있는 고성능 데이터 처리 및 저지연 메시징 아키텍처가 필요하다.  **[중요도 산정 근거]**  전기 요금 절감과 편의성 증대는 사용자가 서비스를 선택하고 지속적으로 사용하는 가장 핵심적인 동기이다. 사용자의 직접적인 만족도가 서비스 성공의 기반이 되므로 가장 높은 중요도를 부여한다. | 5 |
| 경영진 | BG-02 | **[목표]**  높은 사용자 만족도를 바탕으로 출시 2년 내 스마트홈 에너지 관리 시장 점유율 25%를 달성하고, 이를 이용해 데이터 기반 부가 사업을 통해 출시 3년 차부터 연간 5억 원의 추가 수익을 창출한다.  **[아키텍처와의 연관성]**  대규모 사용자 및 기기 증가에 유연하게 대응할 수 있는 확장 가능한(Scalable) 아키텍처가 요구된다. 또한, 데이터 판매 수익 모델을 위해 대규모 데이터를 안전하게 수집, 저장, 비식별화하여 분석할 수 있는 데이터 플랫폼 아키텍처가 필요하다.  **[중요도 산정 근거]**  시장 점유율 확보와 신규 수익 모델 창출은 기업의 생존과 성장을 위한 최우선 과제이다. 지속 가능한 비즈니스를 구축하는 데 있어 핵심적인 지표이므로 가장 높은 중요도를 부여한다. | 5 |
| 운영팀 | BG-03 | **[목표]**  시스템 장애 발생 시 자동 복구 메커니즘을 통해 1분 이내에 정상화하여, 연간 시스템 가동률 99.5% 이상을 달성함으로써 안정적인 사용자 경험을 제공한다.  **[아키텍처와의 연관성]**  하나의 컴포넌트에서 발생한 장애가 다른 컴포넌트로 전파되지 않도록 격리하는 구조를 가져야 한다. 또한 문제 발생 시 자동으로 복구할 수 있는 능력을 갖추어야 한다.  **[중요도 산정 근거]**  서비스의 안정성은 사용자의 신뢰를 얻고 이탈을 방지하는 기본 조건이다. 24/7 중단 없는 서비스 제공은 운영의 핵심 목표이므로 높은 중요도를 부여한다. | 4 |
| 개발팀 | BG-04 | **[목표]**  표준 프로토콜 지원 및 모듈화 설계를 통해 신규 IoT 기기 연동에 소요되는 개발 공수를 50% 단축하고, 코드 재사용성을 높여 유지보수 비용을 30% 절감한다.  **[아키텍처와의 연관성]**  새로운 통신 프로토콜을 지원하는 모듈을 추가할 때, 기존 시스템의 다른 부분에 미치는 영향을 최소화하는 확장 가능한 구조를 가져야 한다. 각 기능은 독립적으로 개발하고 배포할 수 있도록 모듈화되어야 한다.  **[중요도 산정 근거]**  개발 생산성과 유지보수 효율성은 장기적인 비용 경쟁력과 직결된다. 하지만 서비스 초기에는 시장에 성공적으로 안착하는 것이 더 중요하므로, 다른 목표 대비 상대적으로 낮은 중요도를 부여한다. | 3 |

\* 중요도 << 1~5 또는 (최상, 상, 중, 하, 최하)로 구분 >>

## Business Constraints

|  |  |
| --- | --- |
| 비즈니스 제약사항 | |
| ID | 기술 |
| BC-01 | **[시간 제약]**  최초 버전은 10개월 이내에 개발 및 테스트를 완료하고 시장에 출시해야 한다.  **[아키텍처 영향]**  개발 기간 단축을 위해 검증된 오픈소스 프레임워크와 기술 스택을 우선적으로 채택한다. 초기에는 핵심 기능에 집중하고, 복잡한 AI 모델보다는 규칙 기반의 최적화 로직을 먼저 구현한 후 점진적으로 고도화하는 전략을 고려해야 한다. |
| BC-02 | **[법적 제약]**  사용자의 전력 사용 패턴 데이터는 개인정보보호법에 따라 민감 정보로 취급될 수 있으며, 모든 데이터 수집, 저장, 처리 과정에서 법규를 엄격히 준수해야 한다.  **[아키텍처 영향]**  민감 데이터의 전송 및 저장 시에는 강력한 암호화 메커니즘을 적용해야 한다. 또한, 허가된 사용자만이 역할에 맞는 데이터에 접근할 수 있도록 견고한 인증 및 권한 부여 체계를 아키텍처에 반영해야 한다. |

# System Context

## System Context Diagram

아래 다이어그램은 단일 가구에 설치된 스마트홈 전기 관리 시스템과 상호작용하는 외부 개체를 나타냅니다. 중앙의 '스마트홈 전기관리 시스템'은 각 가정에 설치되어 스마트 미터, IoT 허브 등과 직접 통신하는 물리적인 홈 게이트웨이 시스템을 의미한다. 시스템은 클라우드 플랫폼을 통해 30만명이 사용함을 가정합니다. 이 시스템은 가구 내 사용자 5명을 가정하며, 가정 내에 설치된 100개의 가전기기와 2개의 스마트미터, 3개의 IoT허브와 연동됩니다. 전력 공급자는 단일 기관으로 가정합니다.

\*시스템 사용자 30만명 추산근거 : 국내 IoT 기기 보유자 수 1152만명.

이 중 스마트홈 사용자를 10%로 가정하고, 점유율 25%를 목표로 약 30만명 가정하였습니다.

텍스트, 스크린샷, 포스트잇 노트, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

## External Entity List

|  |  |  |
| --- | --- | --- |
| Name | Description | Related Stakeholder |
| 사용자 | **[역할]**  시스템을 원격으로 제어하고 데이터를 모니터링하며, 설정 변경 및 알림을 수신한다.  **[사용환경]**  인터넷에 연결된 모바일 기기(스마트폰, 태블릿)의 전용 앱 또는 웹 브라우저를 통해 시스템에 접속한다. **[전문성]**  대부분의 사용자는 일반 가정의 구성원으로, IT 기술에 대한 전문 지식이 없으며 스마트폰 앱 사용에 익숙한 수준이다. | 고객 |
| 전력 공급자 | **[역할]**  전력 요금 및 단가 정보, 정전 공지 등을 시스템에 제공하며, 시스템으로부터 데이터를 수신할 수 있다.  **[HW 사양]**  대규모 데이터 처리가 가능한 서버 시스템. **[SW 사양]**   * 데이터 연동 및 전송을 위한 API를 포함한 SW. * 데이터베이스 시스템   **[품질 수준]**   * 정확성 : 요금 정보 및 공지 내용의 정확성 보장. * 가용성 : 데이터 제공 API의 24/7 운영. * 성능 : API 응답시간 500ms 이내. | 전력공급사 |
| 스마트 미터 | **[역할]**  가정 전체의 전력 사용량 등을 실시간으로 측정하고, 이상 전력 패턴을 감지하여 시스템에 전송한다.  **[HW 사양]**   * 정확한 측정이 가능한 계량 센서 * 통신 모듈(Zigbee, Z-Wave 등) * 데이터 처리를 위한 MCU   **[SW 사양]**   * 실시간 측정 및 데이터 전송을 위한 펌웨어. * 데이터 암호화 * 통신 프로토콜 스택 * 전력 데이터 교환 국제 표준(DLMS/COSEM) 준수   **[품질 수준]**   * 정확성 : 측정 오차율 1% 이내. * 가용성 : 연간 가용률 99.5% 이상 * 신뢰성 : 실시간 데이터 전송 성공률 99% 이상. * 성능 : 실시간 데이터 업데이트 주기 : 5초 이내 | 전력공급사 |
| 가전기기 | **[역할]**  시스템의 제어 명령에 따라 작동하고, 현재 상태 및 소비 전력 정보를 시스템으로 전송한다.  **[HW 사양]**   * 전원 제어 모듈 및 통신 모듈이 내장 * 스마트 플러그 등 전력 사용량 측정 가능한 하드웨어를 포함   **[SW 사양]**  시스템의 명령을 수신하고 상태를 전송하는 펌웨어.  **[품질 수준]**   * 상호 운용성 : 다양한 제조사의 IoT 기기들과의 호환성. * 응답성 : 제어 명령 수행 지연 시간 2초 이내. | 고객, 개발팀, 가전기기 제조사 |
| 클라우드 플랫폼 | **[역할]**  사용자와 시스템 간의 원격 통신을 중개하고, 수집된 데이터를 저장하며 알림을 푸시하는 역할을 수행한다.  **[HW 사양]**  높은 가용성과 확장성을 보장하는 분산 서버 인프라.  **[SW 사양]**  사용자 인증 및 API 게이트웨이, 데이터베이스, 메시징 큐 등을 포함한 플랫폼 SW.  **[품질 수준]**   * 성능 : 원격 제어 요청 처리 시간 2초 이내. 30만 사용자의 동시 요청으로 발생하는 초당 최대 30만 건의 트랜적션을 안정적으로 처리 * 가용성 : 연간 시스템 가용률 99.9% 이상. * 신뢰성 : 데이터 유실률 0.001% 미만. | 개발팀, 운영팀 |
| IoT허브 | **[역할]**  시스템의 표준 제어 명령을 수신하여, Matter, Zigbee, Z-Wave 등 다양한 IoT 프로토콜로 변환 후 실제 가전기기에 전달하는 로컬 브릿지 역할을 수행한다. 또한, 각 기기의 상태 정보를 수집하여 시스템에 보고한다.  **[HW 사양]**   * 다양한 IoT 통신 프로토콜을 지원하는 멀티 프로토콜 칩셋 및 안테나를 내장한 임베디드 시스템. * 안정적인 로컬 네트워크 연결을 위한 이더넷 포트를 포함.   **[SW 사양]**   * REST API 서버 * 통신 프로토콜 스택: Matter, Zigbee, Z-Wave 등 물리적 통신을 관리하는 소프트웨어 스택 * OTA 기능 지원   **[품질 수준]**   * 성능 : 명령 수신 후 기기까지 1초 이내 * 가용성 : 연간 시스템 가용률 99.9% 이상. * 신뢰성 : 제어 명령 전달 성공률 99.99% 이상 | 개발팀 |

## External Interface List

|  |  |
| --- | --- |
| Name | Description |
| customerInterface | **[역할]**  사용자에게 전기 사용 현황 조회, 가전기기 원격 제어, 알림 수신 등 시스템의 주요 기능을 제공합니다.  **[공통 특성]**   * 암호화 방식 : SSL/TLS * 인증 방식 : 토큰 기반 인증(JWT 사용)   1. 사용자 인증  **[입출력]:**   * **1단계 인증** * 입력 : 인증 유형(Enum) , ID(String,50B), PW(String,50B) * 출력 : 응답코드, 2단계 임시 토큰 * **2단계 인증** * 입력 : 인증 유형(Enum), 인증 코드(String,6B) * 출력 : 응답코드, 액세스 토큰(JWT)   **[특성]:**   * 주기/빈도 : 초당 15만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시에 로그인 요청 가정) * 응답 시간 : 사용자가 로그인 결과를 받기까지의 응답 시간은 2초 이내여야 함.   **[오류처리 방안]**  - ID/PW 불일치 : 오류 코드 반환 및 메시지 표시  - 인증코드 시간 초과 : 오류 코드 반환 및 메시지 표시, 재인증 유도  2. 전력 사용 현황 조회  **[입출력]:**   * 입력 : 액세스 토큰(JWT), 장치 아이디(Optional, 미지정 시 전체 사용량 조회), 조회 기간 유형(Enum), 시작일, 종료일 * 출력 : 응답 코드, 사용량 데이터(총 사용량, 기간별 상세 데이터 리스트), 전력 관련 정보(전력 요금제, 전력 단가, 공급사 공지)   **[특성]:**   * 입출력 크기 : 입력 최대 500byte / 출력 최대 300byte * 주기/빈도 : 초당 15만 건 (처리량 확보 목표인 300만명 수준의 사용자 중 5%가 동시 요청 가정) * 응답 시간 : 사용자가 결과를 조회하기까지의 응답 시간은 2초 이내여야 함.   **[오류처리 방안]**  - 조회 기간 내 데이터 없음 : 오류 코드 반환 및 메시지 표시  - 잘못된 기간 설정 : 오류 코드 반환 및 메시지 표시, 기간 재설정 요청  - 유효하지 않은 장치 아이디 : 오류 코드 반환 및 메시지 표시   * 5초 이상 응답 지연 시 : 오류 코드 반환 및 메시지 표시, 재시도 유도   - 액세스 토큰 만료/권한 없음 : 오류 코드 반환 및 메시지 표시,재로그인 유도  3. 가전기기 원격 제어 화면  **[입출력]:**   * 입력 : 액세스 토큰(JWT), 장치 아이디, 제어 명령 객체(cmdId, subId, length, values) * 출력 : 응답 코드, 기기 상태(명령 수행 후 변경된 최종상태)   **[특성]:**   * 주기/빈도 : 초당 30만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시 요청 가정) * 응답 시간 : 1초 이내로 시스템에 전달되어야 함   **[오류처리 방안]**  - 기기 오프라인/무응답 : 오류 코드 반환 및 메시지 표시  - 기기에서 명령 처리 실패 : 오류 코드 반환 및 메시지 표시  - 지원하지 않는 명령 : 오류 코드 반환 및 메시지 표시  4. 설정 화면   * **설정 조회** * 입력 : 액세스 토큰(JWT) * 출력 : 응답코드, 설정 객체(앱/문자/푸시 알림, 에너지 절감 모드 On/Off, 다크모드, 글자크기) * **설정 변경** * 입력 : 액세스 토큰(JWT) * 출력 : 응답코드, 설정 객체   **[특성]:**   * 입출력 크기 : 입력 32byte + JWT, 출력 : 32byte * 주기/빈도 : 초당 30만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시 요청 가정) * 응답 시간 : 조회 및 변경 요청 응답은 1초 내 완료되어야 함   **[오류처리 방안]**  - 유효하지 않은 값 입력(변경 시) : 오류 코드 반환 및 메시지 표시, 올바른 값 예시 가이드  - 저장/조회 실패 : 오류 코드 반환 및 메시지 표시  - 토큰 만료/권한 없음 : 오류 코드 반환 및 메시지표시, 재로그인 유도 |
| cloudInterface | **[역할]**  클라우드 서비스가 사용자의 요청을 이행하기 위해 시스템의 데이터나 기능을 호출할 때 사용하는 인터페이스입니다. 클라우드와 시스템 간의 모든 데이터 교환은 이 인터페이스를 통해 이루어집니다.  **[공통 특성]**   * 암호화 방식: SSL/TLS * 인증 방식: API Key 인증 방식   **[공통 오류처리 방안]**   * 5초 이상 응답 지연 시 : 오류 코드 반환 및 로깅 * API Key 불일치 / 권한 없음: 인증 실패 코드 반환 및 로깅   1. 데이터 동기화  **[역할]**  시스템이 수집한 데이터를 주기적으로 클라우드 DB에 동기화(전송)하는 역할.  **[입출력]**   * 입력: API Key, 홈 게이트웨이 식별자, 전력 데이터 배치(Batch). * 출력: 응답 코드   **[특성]**   * 주기/빈도: 초당 6만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시 요청 가정), 가구 기준 5초당 1건,   **[오류처리 방안]**   * 데이터 없음: 오류 코드 반환 및 로깅   2. 기기 제어 명령 전달  **[역할]**  클라우드가 사용자의 '가전기기 원격 제어' 요청을 받아, 시스템에 제어 명령을 전달하고 그 결과를 받아오는 역할을 합니다.  **[입출력]**   * 입력 : 서버 인증용 API 키, 사용자 식별자, 장치 아이디, 제어 명령 객체(customerInterface 입력과 동일) * 출력 : 응답 코드, 기기 상태(명령 수행 후 변경된 최종상태)   **[특성]**   * 주기/빈도 : 초당 30만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시 요청 가정) * 응답 시간 : 시스템에 전달되기까지 1초 이내   **[오류처리 방안]**   * 기기 오프라인 / 명령 실패 / 미지원 명령: 상태에 대한 오류 코드 반환 및 로깅   3. 시스템 설정 관리  **[역할]**  사용자의 설정 중 시스템 레벨에서 저장 및 관리되어야 하는 항목(예: 에너지 절감 모드)을 조회하거나 변경합니다. (다크모드, 글자 크기 등 앱 자체 설정은 제외)  **[입출력]**   * **설정 조회** * 입력 : 서버 인증용 API 키, 사용자 식별자 * 출력 : 응답코드, 설정 객체(에너지 절감 모드 On/Off 등 시스템 관련 설정만 포함) * **설정 변경** * 입력 : 서버 인증용 API 키, 사용자 식별자 * 출력 : 응답코드, 변경된 설정 객체   **[특성]**   * 주기/빈도 : 초당 30만 건 (처리량 확보 목표인 30만명 수준의 사용자가 동시 요청 가정) * 응답 시간 : 조회 및 변경 요청 응답은 1초 내 완료되어야 함   **[오류처리 방안]**   * 유효하지 않은 설정 값: 오류 코드 반환 및 로깅 * DB 저장/조회 실패: 오류 코드 반환 및 로깅 |
| applianceInterface | **[역할]**  시스템이 제어 대상 기기의 종류나 통신 방식(상용 허브, 자체 모듈 등)에 상관없이, 일관된 방식으로 명령을 내리기 위한 인터페이스.  **[공통 특성]**   * 통신 방식 : 내부 함수 호출 * 주기/빈도 : 초당 500 건(처리량 확보 목표인 5명 수준의 사용자가 100개 기기를 동시 제어 요청 시)   **[공통 오류처리 방안]**   * 예외 발생 시 : 오류 코드 반환   1. 기기 제어 함수  **[입출력]**   * 입력: 장치 아이디, 제어 명령 객체 * 출력: 응답코드, 결과 객체(전원 상태)   2. 기기 상태 조회 함수  **[입출력]**   * 입력: 장치 아이디 * 출력: 응답코드, 현재 기기 상태 객체(전원 상태) |
| ioTHubInterface | **[역할]**  ‘applianceInterface’의 표준 규약을 실제로 구현하는 부분으로, 시스템의 표준 명령을 외부 개체인 'IoT 허브'가 알아들을 수 있는 고유한 API 호출(HTTP/REST)로 변환하고 실행하는 역할을 합니다.  **[공통 특성]**   * 프로토콜 : HTTP/REST API * 데이터 포맷 : JSON * 인증 방식 : API Key 또는 인증 토큰   **[공통 오류처리 방안]**   * 5초 이상 지연 시 : 허브 API 응답 지연 시 타임아웃 예외 발생 * HTTP 오류 변환 : 오류코드 반환 및 로깅   1. 기기 제어  **[입출력]**   * 입력: 허브 인증 토큰, 장치 아이디, 제어 명령(JSON Body) * 출력: applianceInterface가 요구하는 표준 결과 객체   2. 기기 상태 조회  **[입출력]**   * 입력: 허브 인증 토큰, 장치 아이디 * 출력: applianceInterface가 요구하는 표준 상태 객체 |
| meterInterface | **[역할]**  시스템이 외부 개체인 스마트 미터로부터 단방향으로 전력 사용량 데이터를 수신하기 위한 표준 통신 규약이다.  **[입출력]**   * 입력 : 데이터 패킷 (계량기 ID, 현재 시각, 누적 사용량(kWh), 현재 소비 전력(W)) * 출력 : 없음. (수신 후 저장)   **[특성]**   * 통신 프로토콜 : DLMS/COSEM 프로토콜 * 데이터 포맷 : Binary 포맷 * 암호화 방식 : SSL/TLS * 주기/빈도 : 5 초당 2 건 (처리량 확보 목표인 2개 수준의 스마트 미터가 동시에 요청 시)   **[오류처리 방안]**   * 데이터 검증 실패 : 데이터 폐기, 오류 로깅 * 데이터 중복 수신 : 무시 및 로깅 |
| supplierInterface | **[역할]**  시스템과 외부 개체인 전력공급사 시스템 간에 전력 사용량 데이터, 요금 정보, 긴급 공지 등을 안전하게 교환하기 위한 B2B연동 규약이다.  1. 전력 사용량 데이터 제출  **[역할]**  시스템이 스마트 미터로부터 수집 및 취합한 전력 사용량 데이터를 주기적으로 전력공급사 시스템에 전송하는 역할을 한다.  **[입출력]**   * **입력**: 인증 토큰(OAuth 2.0), 각 미터기별 사용량 데이터 배치(Batch) (JSON 배열) * **출력**: 응답 코드, 데이터 처리 결과(성공/실패 건수, 배치 ID)   **[특성]**   * 통신 프로토콜: HTTPS/REST API (POST 방식) * 인증 방식: OAuth 2.0 Client Credentials 방식 * 주기/빈도: 매시간 정각, 1시간 분량의 데이터를 배치로 전송.   **[오류처리 방안]**   * 데이터 형식 오류: 오류 코드 반환 * **전송 실패**: 네트워크 또는 서버 문제로 전송 실패 시, 지수 백오프(Exponential Backoff)를 적용하여 최대 3회 재시도한다. |

# Architectural Drivers

## Use Case Model

### Use Case Diagram

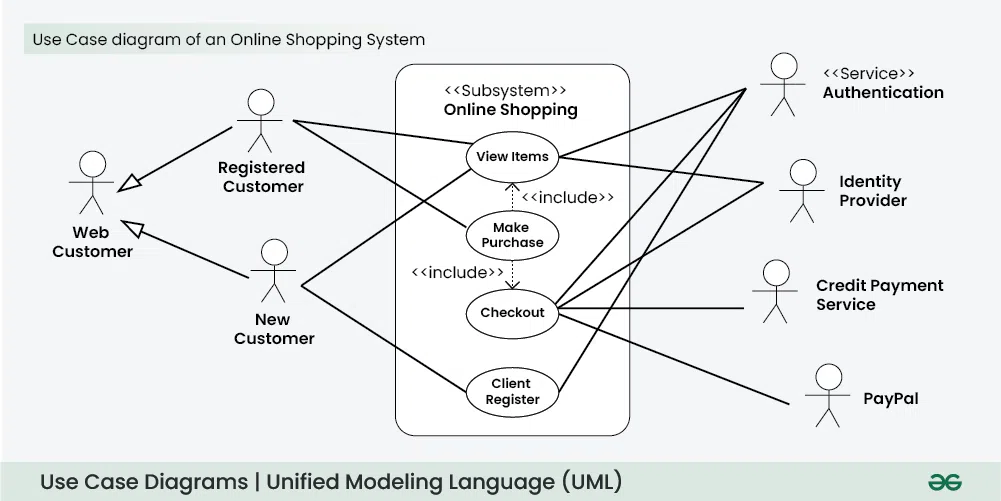
<작성 방법>

* 아키텍처 설계에서 고려할 핵심 기능을 모두 Use case로서 식별한다.
* 시스템과 상호작용하는 주요 대상을 Actor로 식별한다.
* Use case의 이름은 해당 Use case의 결과 즉 Actor에게 제공되는 결과물/서비스 등을 명확하게 기술해야 한다.
* Actor와 Use case 간의 연관 관계는 Actor와 Use Case 사이의 상호작용이 있을 때 표현한다.
* Use case 간의 포함 관계 (include)는 하나의 Use Case가 다른 Use Case를 필수적으로 포함하는 경우에, 확장 관계 (extend)는 하나의 Use Case가 특정 조건 하에서 다른 Use Case의 기능을 확장할 때만 사용해야 한다.
* Actor 간의 일반화 관계는 하나의 Actor가 다른 Actor의 추상화 혹은 일반화된 개념을 나타낼 /대 사용한다.

<점검사항>

* Use Case와 액터 사이의, Use Case 간의, 액터 간의 관계가 올바른가?

<예시>



### Actor List

<작성 방법>

* 시스템 컨텍스트 다이어그램의 외부 개체를 모두 Actor로 도출한다.
* Actor에 대한 설명이 시스템 관점에서 바라보았을 때의 Actor의 역할을 파악할 수 있도록 구체적이어야 한다.

|  |  |
| --- | --- |
| Name | Description |
| Actor1 |  |
| Actor2 |  |
| Actor3 |  |

### Use Case List

<작성 방법>

* Use Case Diagram에 표시된 모든 Use Case를 나열한다.
* Use case 이름은 해당 Use Case를 통해서 제공되는 시스템의 기능을 명확하고 구체적으로 표현해야 한다.
* Description은 시스템이 제공할 기능의 개요, BV 값 설정근거, AI 값 설정근거를 명확하게 기술해야 한다.
* BV (Business Value)는 해당 Use Case의 비즈니스 가치를 평가한다.
* AI (Architectural Impact)는 해당 Use Case가 아키텍처에 주는 영향도를 평가한다.
* Business Drivers은 해당 UC가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는지를 나타낸다.

<점검사항>

* 구조에 영향을 미치는 기능 명세가 충분한가?
* 유스케이스들의 우선순위(BV, AI)에 대한 평가가 타당한가?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name | Description | Priority | | Business Drivers |
| BV | AI |
| UC-01 |  | [개요]  [BV 설정근거]  [AI 설정 근거] |  |  |  |
| UC-02 |  |  |  |  |  |
| UC-03 |  |  |  |  |  |

\* BV: Business Value, AI: Architectural Impact << 최상 상 중 하 최하로 구분 >>

### UC-01 Title Description

<Pre Condition 작성 방법>

* Basic Flow 동작이 시작되기 위하여 필요한 시스템 상태를 Pre Condition으로 기술

<Post Condition 작성 방법>

* Basic Flow 동작이 종료되었을 때 충족해야 할 시스템 상태를 Post Condition으로 기술

<Basic Flow 작성 방법>

* 시스템이 제공할 기본 기능에 해당하는 동작만 기술한다.
* 각 스텝은 하나의 Actor 또는 시스템의 동작을 기술한다.
* 각 스텝은 Actor 또는 시스템을 주어로 하는 능동태 문장으로 기술한다.

<Alternative Flow 작성 방법>

* 시스템이 제공할 추가 기능 및 예외 상황 처리에 해당하는 동작을 기술한다.

<점검사항>

* 시나리오 명세가 구체적이고 명확한가?
* 아키텍처 측면에서 고려할 이슈를 포함하는 대안 시나리오가 식별되었는가?
* 해당 시나리오에 관련된 모든 Actor가 시나리오에서 언급되었는가?

|  |  |
| --- | --- |
| Pre Condition |  |
| Post Condition |  |
| Basic Flow |  |
| Alternative Flow |  |

### UC-02 Title Description

### *UC-03 Title* Description

## Quality Attribute Scenario

### QA Scenario List

<작성 방법>

* 비즈니스 목표 달성 위하여 필요한 모든 QA를 식별한다.
* Description은 품질 요구사항의 개요, BV 값 설정근거, AI 값 설정근거를 명확하게 기술해야 한다.
* BV (Business Value)는 해당 QA 시나리오의 비즈니스 가치를 평가한다.
* AI (Architectural Impact)는 해당 QA 시나리오가 아키텍처에 주는 영향도를 평가한다.
* Business Drivers은 해당 QA가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는 지를 나타낸다.

<점검사항>

* 비즈니스 목표를 실현하는데 필요한 품질요구사항이 충분한가?
* 품질요구사항의 우선순위(BV, AI)에 대한 평가가 타당한가?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Type | Priority | | Business Drivers |
| BV | AI |
| QA-01 | [개요]  [BV 설정근거]  [AI 설정근거] |  |  |  |  |
| QA-02 |  |  |  |  |  |
|  |  |  |  |  |  |

\* 아키텍처 드라이버를 구성하는 QA 6개 이상 정의 (성능 QAS 1개 이상, 가용성 QAS 1개 이상, 변경용이성 QAS 1개 이상이 포함되어야 함)

### QA-01 Title Scenario

<작성 방법/점검사항>

* Description이 해당 QA Scenario에 대한 간략한 설명을 명확하게 기술하였는가?
* Source 및 Stimulus가 구체적인가? 즉 아키텍처 평가를 수행할 때 실제 어떤 입력이 사용되는 명확하게 판단할 수 있는가?
  + 나쁜 예) 기능의 추가
  + 좋은 예) 새로운 스케줄링 방법의 추가
* Artifact 및 Environment가 구체적인가? 즉 아키텍처 평가를 수행할 때 평가 대상인 시스템과 평가 환경을 명확하게 판단할 수 있는가? Environment는 Stimulus에 대한 Response를 관찰하며 Measure의 수치가 기대될 수 있는 구체적인 환경이 기술되어야 한다. 예) Stimulus 수신 당시 시스템의 부하
* Response가 구체적인가? 즉 해당 QA에 대한 평가를 수행할 때 Measure의 대상이 되는 동작/결과를 구체적으로 기술해야 한다.
* Response Measure가 Response와 일치하며 구체적인가? 즉 Measure의 유형 및 수치가 Response와 일치해야 하며 QA의 충족 여부 및 충족 수준을 판단할 수 있도록 Measure의 수치가 구체적인가?

|  |  |
| --- | --- |
| QA Type |  |
| Description |  |
| Source of Stimulus |  |
| Stimulus |  |
| Artifact |  |
| Environment |  |
| Response |  |
| Response Measure |  |

### QA-02 Title Scenario

## Architectural Constraint

<작성 방법>

* 앞서 도출한 비즈니스 제약 사항 중에서 아키텍처 설계 결정에 영향을 미칠 수 있는 요소를 모두 도출한다.

<점검사항>

* Title이 해당 architectural constraint의 핵심 사항을 명확하게 표현하고 있는가?
* Description이 해당 architectural constraint의 내용을 구체적으로 명확하게 기술하고 있는가?
* Business Drivers은 해당 architectural contraint가 어떤 비즈니스 드라이버 (비즈니스 목표, 비즈니스 제약사항)으로부터 도출되었는 지를 나타낸다.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Description | Business Drivers |
| AC-01 |  |  |  |
| AC-02 |  |  |  |
|  |  |  |  |

# High Level Structure Design Description

## Domain Modeling

\* 우선순위가 높은 UC 3개 이상에 대해 작성함. Basic Flow 시나리오는 기본적으로 기술하고 아키텍처적으로 중요한 Alternative Flow 시나리오도 포함하여 기술해야 함.

### Conceptual Class List

|  |  |  |  |
| --- | --- | --- | --- |
| Conceptual Class | | | Relevant Use Cases |
| ID | Name | Type |
| CC-01 | UserData | entity | UC-01, UC-02 |
|  |  |  |  |

<작성 방법>

* Use case별로 해당 기능을 실현하기 위하여 필요한 Conceptual Class 목록을 도출한다.
  + Conceptual Class의 도출은 한번에 이루어기보다는 개별 Use Case를 기반으로 도출을 반복하면서 정제된다.
* Conceptual Class 카테고리에 따라서 entity, control, boundary, application logic 등의 스테레오타입 정보를 Type란에 기술한다.

<점검 사항>

* Use Case 시나리오를 실현하기 위해 필요한 개념적 클래스가 충분히 도출되었는가?
* 구현 수준의 클래스를 포함하고 있지 않은가?

### Dynamic View

#### *UC-01 Title* Use Case Dynamic Domain Model

<작성 방법>

* UC 시나리오를 UML Communication Diagram을 이용하여 conceptual class 인스턴스 간의 Message 흐름으로 표현한다.
* Messgae label은 **ReturnData := MessageName (argument-list)** 의 형식으로 표현한다.

<점검 사항>

* 인스턴스 간의 메시지가 메시지 형식에 맞춰서 작성되었는가?
* 인스턴스 간의 메시지 상호작용이 구체적이고 명확한가?
* 유스케이스 시나리오와 일관성이 있는가?

텍스트, 도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

#### *UC-02 Title* Use Case Dynamic Domain Model

### Static View

텍스트, 도표, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

<작성 방법>

* 도출된 conceptual class간의 association을 추가하여 클래스 다이어그램을 작성한다.
* Association에 multiplicity 정보를 추가한다.
* Entity 클래스에는 속성을 추가한다.

<점검 사항>

* Static View의 Class Diagram과 Dynamic View의 Communication Diagram간에 일관성이 있는가?

## Quality Driven Architectural Design

<작성 방법>

* 3장에서 식별된 각 QA를 위한 Architectural Design이 제시되어야 한다.
* 우선순위가 높은 QA 3개 이상에 대해 작성함.

### Exploring Architectural Options for *QA-01 Title*

#### Quality Driven Architectural Decision Structure



<작성 방법>

* 해당 QA를 만족시키기 위해서 설계 시 고려해야할 설계이슈(Design Issue)와 관련된 설계 옵션 (Design Option) 간의 의사결정 구조를 도식화 한다.
* 고려해야할 설계 이슈로는 다음과 같은 것을 고려해 볼 수 있다.
  + 컴포넌트 단위로 시스템 구조화 방안
  + 컴포넌트 제어 방안
  + 컴포넌트 간의 상호작용 방안
  + 데이터 접근 및 관리 방안
  + 하드웨어 리소스 관리 방안
  + 컴포넌트 배포 방안
* 해당 Design Issue가 다른 Design Option에서 추가적으로 파생된 문제인 경우에는 이들을 연결시켜서 표현한다.

<점검 사항>

* 품질 요구사항에 대한 아키텍처 설계 결정 구조가 적절한가?

#### Design Decisions for *설계 이슈1*

##### ***설계 이슈1***

*설계 이슈1*에 대한 설명 (배경 및 가정, 이슈 정의) 기술

|  |  |
| --- | --- |
| 배경 및 가정 |  |
| 이슈 정의 |  |

<작성 방법>

* 이슈가 발생하는 상황과 그 배경을 구체적으로 기술하고, 이슈와 관련된 가정된 조건이 있다면 이를 구체적으로 명시한다.
* 기술된 배경 및 가정하에서 제기된 이슈(문제)가 무엇인지 명확하게 정의하고, 이 문제가 왜 중요한지 설명한다.

<점검 사항>

* 설계 이슈가 다루는 문제가 구체적이고 명확한가?

##### **Design Options for *설계 이슈1***

|  |  |
| --- | --- |
| Design Options | |
| Title | Description |
|  |  |
|  |  |
|  |  |

<작성 방법>

* 해당 설계이슈와 관련된 모든 Design Option을 기술한다.
* Design Option의 Title이 설명하고 있는 Description과 일관성이 있어야 한다.
* Design Option의 Description이 실질적이고 세부적인 design이 제시되어야 한다.
  + 적용하고 있는 설계 솔루션(패턴, 택틱 등)에 대한 일반적인 설명이 아니라 해당 솔루션을 이 시스템에 적용할 때의 아키텍처 설계 결정을 구체적으로 제시해야 한다.
  + Layer patter n 적용의 경우: Layer의 수 및 각 layer의 역할, Layer interface 설계 등에 대한 decision이 제시될 필요가 있음
* 가능하면 제시된 설계 옵션들을 명확히 비교해 볼 수 있도록 구체적인 설계결정을 반영한 설계 뷰를 제시하는 것이 좋음

<점검 사항>

* 설계 옵션에 대한 설명이 구체적인가?

##### **Decision and Rationale for *설계 이슈1***

|  |  |  |  |
| --- | --- | --- | --- |
| Design Option Title | Pros | Cons | Rationale |
| 마이크로커널 스타일 적용  (selected) |  |  |  |
| 마이크로서비스 스타일 적용 |  |  |  |

<작성 방법>

* 각 Design Option별로 3장에서 제시된 모든 관련된 QA 측면 및 필요한 관심사(concerns) 측면에서 장/단점을 제시한다.
  + Pros/cons는 QA와 관심사 관점에서 구체적으로 장단점을 기술해야 한다. 즉 각 QA 관점에서 Response Measure에 대한 유/불리 와 Constraint 충족 여부 등이 명시적으로 제시될 필요가 있음
* 제시된 Design Option들 중에서 가장 적합한 Design Option을 선정하는 근거를 기술한다.

<점검 사항>

* 설계 옵션에 대한 평가와 선택 근거가 타당한가?

#### Design Decisions for *설계 이슈2*

#### Design Decisions for *설계 이슈3*

### Exploring Architectural Options for *QA-02 Title*

#### Quality Driven Architectural Decision Structure

#### Design Decisions for *설계 이슈1*

#### Design Decisions for *설계 이슈2*

## Component & Connector View

* 4.2절에서 기술된 아키텍처 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the C&C View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 C&C View에 반영된 부분이 있으면 C&C View의 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

* 아키텍처 설계 결정이 구체적으로 올바로 반영되어 있는가?

### UML Component Diagram



### Component List

<작성 방법>

* Component Name은 컴포넌트가 제공하는 역할(responsibility)를 명확하게 나타낼 수 있도록 정의한다.
* Component Kind은 컴포넌트의 종류로서 다음과 같은 값을 가질 수 있다.
  + 분산 환경에서의 서브시스템 요소로서 client, server, data store
  + 동시 수행 요소로서 process (또는 task), thread
* Property Description에는 **해당 컴포넌트에 요구되는 기능 및 품질 요구사항, 제약사항 등** 아키텍처 드라이버 관점에서 기술한다.
  + 품질요구사항 및 제약사항에 대해서는 *완전만족/부분만족/미충족* 여부를 표시한다.
* Relevant ADs는 해당 컴포넌트와 관련된 모든 기능 및 품질요구사항, 제약사항을 나열한다.

<점검 사항>

* Property Description에 기술된 내용이 구체적이고 관련된 Architectural Drivers와의 일관성이 존재하는가?

|  |  |  |  |
| --- | --- | --- | --- |
| Component  Name | Component  Kind | Property Description | Relevant ADs |
| Component1 | process |  | UC-02  QA-01  QA-03 |
| … |  |  |  |

### Connector List

<작성 방법>

* Connector Name은 커넥터가 제공하는 컴포넌트 간의 상호작용을 명확하게 나타낼 수 있도록 명명한다. 잘 알려진 정의된 커넥터의 경우에는 해당 이름을 사용할 수 있다. (예, RPC, Message Queue, Message Broker 등)
* Property Description에는 해당 커넥터의 프로토콜(행위) 및 데이터 포맷 등이 어떠한 품질속성 및 기능과 관련 있는지를 기술한다.
* Relevant ADs는 해당 커넥터와 관련된 모든 기능 및 품질요구사항, 제약사항을 나열한다.

<점검 사항>

* Property Description에 기술된 내용이 구체적이고 관련된 Architectural Drivers와 일관성이 존재하는가?

|  |  |  |
| --- | --- | --- |
| Connector  Name | Property Description | Relevant ADs |
|  |  | UC-02  QA-01  QA-03 |
| … |  |  |

## Deployment View

* 4.2절에서 기술된 아키텍처 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the Deployment View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 Deployment View에 반영된 부분이 있으면 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

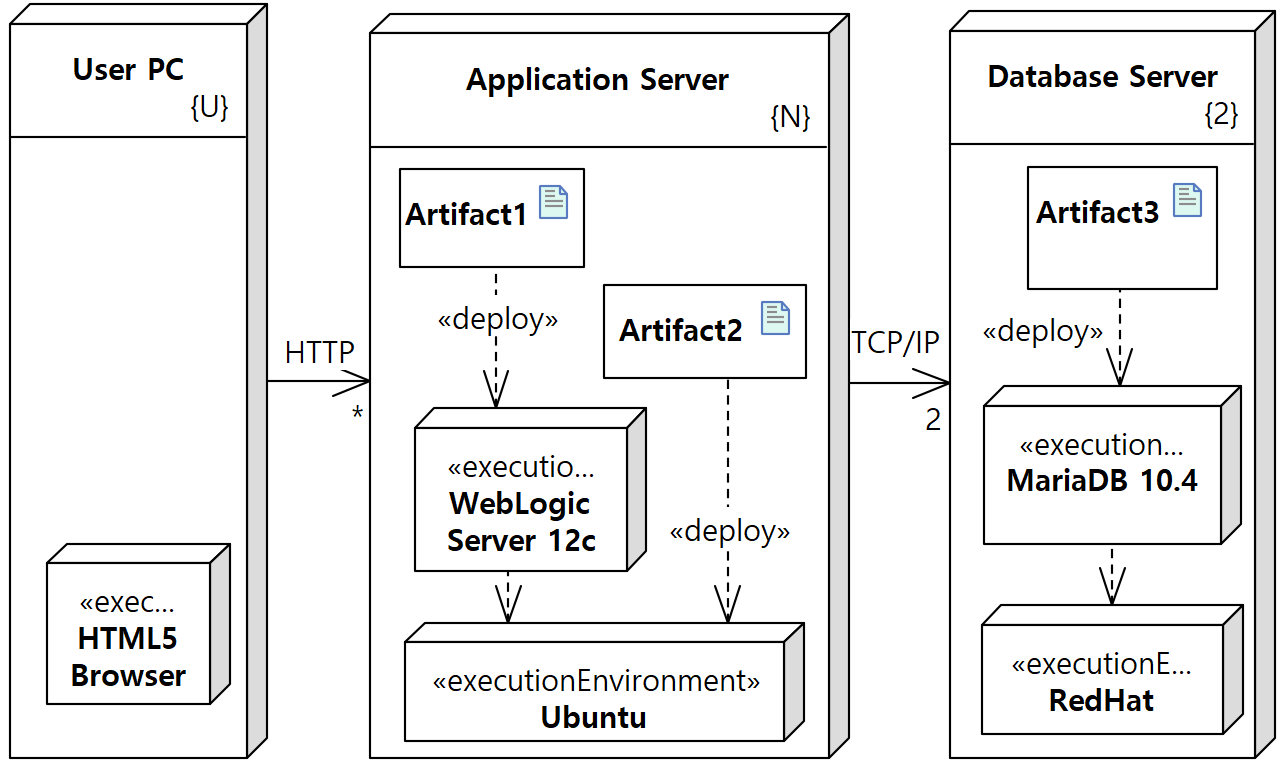
* 아키텍처 설계 결정이 구체적으로 Deployment View에 올바로 반영되어 있는가?

### Deployment Diagram

<작성 방법>

* 개발된 시스템 운용에 필요한 노드 및 노드 간의 통신과 노드의 실행 환경을 표현한다.
  + 노드의 유형과 역할(기능/QA)에 일치하는 구체적인 이름을 기술한다.
  + SW 컴포넌트의 실행에 필요한 SW 환경(플랫폼)을 기술할 필요가 있다면 <<execution environment>>로써 식별한다.
  + 각 노드 간의 필요한 모든 통신 경로를 식별하고 통신 방법을 구체적으로 기술한다.

<예시1>



#### Node Specification

<작성 방법>

* Deployment diagram에서 표시된 각 노드 별로 기술한다.
* 즉 각 노드가 시스템의 기능/QA 측면에서 어떤 역할을 하는지 설명이 필요하다. 복수 개 Multiplicity 인 경우 각 Instance의 역할을 명확히 기술한다. AD로 선택된 QAS의 만족을 보이기 위해 노드의 특성(CPU, Memory, HDD 등 HW 사양) 정보가 필요하다면 해당 정보를 기술한다.

<점검 사항>

* 각 노드의 설명이 구체적인가?

|  |  |
| --- | --- |
| Name | Description |
| User PC |  |
| Application Server |  |
| Database Server |  |

#### Execution Environment Specification

<작성 방법>

* Deployment diagram에서 <<execution environment>> 을 사용했다면 해당 항목별로 기술한다.
* Architectural driver(QA 등)의 만족을 위해 필요한 <<execution environment>> 특성(SW 제품 및 버전 등 SW 사양)과 결정 근거를 기술한다.

<점검 사항>

* 각 Execution Environment의 설명이 구체적인가?

|  |  |  |
| --- | --- | --- |
| Node | Name | Description |
|  | HTML5 Browser |  |
|  | WebLogic Server12c |  |
|  |  |  |

#### Communication Path Specification

<작성 방법>

* 각 통신 경로의 특성(통신을 위한 HW 및 SW 특성)과 그러한 특성을 결정한 근거를 3장에서 식별한 Architectural driver(QA 등)의 충족과 관련하여 정당화한다

<점검 사항>

* 각 통신 경로의 설명이 구체적인가?

|  |  |
| --- | --- |
| Path | Description |
| User PC – Application Server |  |
| Application Server – Database Server |  |

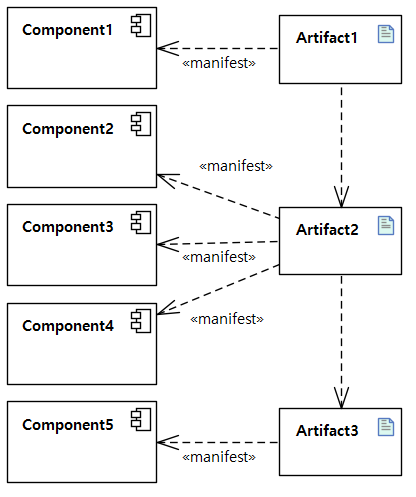
### Artifact Definition Diagram

<작성 방법>

* Deployment diagram의 Node에 설치될 각 파일들을 Artifact로서 식별한다.

<점검 사항>

* C&C View의 각 Component는 1개 이상의 Artifact에 포함되었는가?



#### Artifact Definition Specification

<작성 방법>

* Artifact definition diagram과 일치하여 Artifact, Manifested Components, Depends on을 기술한다.

<점검 사항>

* 각 Artifact의 기능/역할이 명확히 설명되었는가?.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Manifested Component | Depends on | Description and Rationale |
| Artifact1 | Component1 | Artifact2 |  |
| Artifact2 | Component2  Component3  Component4 | Artifact3 |  |
| Artifact3 | Component5 | None |  |

# Detailed Component Design Description

\* 아키텍처적으로 중요한 컴포넌트 3개 이상에 대해 작성하고, 컴포넌트의 대표적 Behavior를 시퀀스 다이어그램으로 기술해야 함.

## Design Description for *ComponentName*

### Overview



<작성 방법>

|  |  |
| --- | --- |
| 항목 | 설명 |
| 개요 | 이 컴포넌트의 역할을 한/두 줄로 간략하면서 명확하게 서술한다. |
| 컴포넌트 기능 요구사항 | 이 컴포넌트에 부여된 기능 요구사항을 서술한다. Provided Interface를 중심으로 구체적으로 설명한다. |
| 컴포넌트 품질 요구사항  및 제약사항 | 이 컴포넌트가 만족해야 할 품질 요구사항 및 제약사항을 서술한다.   * 시스템의 QA를 충족시키기 위하여 이 컴포넌트에서 만족시켜야하는 QA를 구체적으로 서술한다. * 시스템의 제약사항 중에서 이 컴포넌트에서 만족시켜야 하는 제약사항이 있다면 이를 구체적으로 서술한다. |

<점검 사항>

* 컴포넌트의 요구사항이 명확하게 정의되었는가?

### Quality Driven Component Design

#### Exploring Design Options for *QA-01 Title*

* + - * 1. **Quality Driven Component Design Structure**



<작성 방법>

* 해당 QA를 만족시키기 위해서 설계 시 고려해야할 설계이슈(Design Issue)와 관련된 설계 옵션 (Design Option) 간의 의사결정 구조를 도식화 한다.
* 고려해야할 설계 이슈로는 다음과 같은 것을 고려해 볼 수 있다.
  + 컴포넌트의 모듈 관점의 분해
  + 컴포넌트 간의 상호작용의 모듈 설계
* 해당 Design Issue가 다른 Design Option에서 추가적으로 파생된 문제인 경우에는 이들을 연결시켜서 표현한다.

<점검 사항>

* 품질 요구사항에 대한 컴포넌트 설계 결정 구조가 적절한가?

#### Design Decisions for *설계이슈1*

###### Definition for *설계이슈1*

설계 이슈(*설계이슈1*)에 대한 설명 기술

<작성 방법>

* 이슈가 발생하는 상황과 그 배경을 구체적으로 기술하고, 이슈와 관련된 가정된 조건이 있다면 이를 구체적으로 명시한다.
* 기술된 배경 및 가정하에서 제기된 이슈(문제)가 무엇인지 명확하게 정의하고, 이 문제가 왜 중요한지 설명한다.

###### Design Options for *설계이슈1*

|  |  |
| --- | --- |
| Design Option | |
| Title | Description |
|  |  |
|  |  |
|  |  |

<작성 방법>

* 해당 설계이슈와 관련된 모든 Design Option을 기술한다.
* Design Option의 Title이 설명하고 있는 Description과 일관성이 있어야 한다.
* Design Option의 Description이 실질적이고 세부적인 design이 제시되어야 한다.
  + 적용하고 있는 설계 솔루션(패턴, 택틱 등)에 대한 일반적인 설명이 아니라 해당 솔루션을 이 시스템에 적용할 때의 아키텍처 설계 결정을 구체적으로 제시해야 한다.
  + Layer patter n 적용의 경우: Layer의 수 및 각 layer의 역할, Layer interface 설계 등에 대한 decision이 제시될 필요가 있음
* 가능하면 제시된 설계 옵션들을 명확히 비교해 볼 수 있도록 구체적인 설계결정을 반영한 설계 뷰를 제시하는 것이 좋음

<점검 사항>

* 설계 이슈와 설계 옵션에 대한 설명이 구체적인가?

##### Decision and Rationale for *설계이슈1*

|  |  |  |  |
| --- | --- | --- | --- |
| Design Option Title | Pros | Cons | Rationale |
| Design Option Title1 |  |  |  |
| Design Option Title2 |  |  |  |

<작성 방법>

* 각 Design Option별로 5.1절에서 제시된 모든 관련된 QA 측면 및 필요한 관심사(concerns) 측면에서 장/단점을 제시한다.
  + Pros/cons는 QA와 관심사 관점에서 구체적으로 장단점을 기술해야 한다. 즉 각 QA 관점에서 Response Measure에 대한 유/불리 와 Constraint 충족 여부 등이 명시적으로 제시될 필요가 있음
* 제시된 Design Option들 중에서 가장 적합한 Design Option을 선정하는 근거를 기술한다.

<점검 사항>

* 설계 옵션에 대한 평가와 선택 근거가 타당한가?

#### Exploring Design Options for *QA-02 Title*

### Module View

* 5.1.2절에서 기술된 컴포넌트 설계 결정 위주로 표현한다.

|  |  |  |
| --- | --- | --- |
| QA Scenarios ID | (Selected) Design Option Title | Reflection in the Module View |
| QA-01 | Design Option Title1 |  |
| Design Option Title3 |  |
| QA-02 |  |  |
|  |  |

<작성 방법>

* 4.2절에서 기술한 QA에 대한 아키텍처 설계결정 중에서 Module View에 반영된 부분이 있으면 어느 부분에 어떻게 반영되어 표현되었는 지를 기술한다.

<점검 사항>

* 컴포넌트 설계 결정이 모듈 뷰에 구체적으로 반영되어 있는가?

#### Class Diagram

<작성 방법>

* Component를 구현하기 위하여 필요한 Class 및 Interface들을 모두 식별한다.
* Class 간의 관계(generalization, association, composition/aggregation, dependency)를 표시한다.
* Class의 attribute 및 operation이 구체적으로 명시한다.
* 많은 수의 Class 및 Interface가 식별된 경우 Package Cohesion/Coupling을 고려하여 Packaging한다.

<점검 사항>

* Class Diagram이 구체적이고 올바로 작성되어 있는가?



##### Element List

<작성 방법>

* 위의 Class diagram에 표현된 모든 요소(package, class, interface)에 대한 역할을 설명한다.
* 각 요소가 Component에 부여된 요구사항(기능, 품질, 제약사항 등)에 어떻게 기여하는 지 측면에서 설명한다.

<점검 사항>

* Element의 설명이 구체적인가?

|  |  |
| --- | --- |
| Name | Description |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### Sequence Diagram

<작성 방법>

* 컴포넌트가 제공하는 인터페이스의 각 오퍼레이션이 어떻게 내부 모듈 인스턴스 간의 상호작용으로 실현되는가를 시퀀스 다이어그램을 이용해서 기술하시기 바랍니다.
* 시작은 컴포넌트 provided IF가 커넥터 모듈로부터 호출되는 방식으로 기술하시기 바랍니다.

<점검 사항>

* Sequence Diagram이 구체적이고 올바로 작성되어 있는가?



## Design Description for *Component 2*

## Design Description for *Component 3*

# Architectural Evaluation

* 우선순위가 높은 QA 3개 이상에 대해 이와 선정된 Architectural Decision에 대한 분석서를 작성한다.

## Architectural Evaluation for *QA-01 Title*

<작성 방법>

* 제시된 양식에 맞춰서 작성한다.

<점검 사항>

* 아키텍처 결정의 민감점, 절충점, 위험요소, 비위험요소 분석이 타당한가?
* 아키텍처 결정에 대한 Reasoning이 타당한가?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QA 시나리오 |  | | | |
| 품질속성 |  | | | |
| 자극 소스 |  | | | |
| 자극 |  | | | |
| 환경 |  | | | |
| 응답 |  | | | |
| 측정 |  | | | |
| 아키텍처 결정 | 위험요소 | 민감점 | 절충점 | 비위험요소 |
|  | R1 | S1 | T1 |  |
|  |  |  |  | NR1 |
|  |  |  |  |  |
| Reasoning | <<아키텍처 결정들이 시나리오가 표현하는 품질속성을 달성하는데 어떻게 공헌하는지에 대한 근거를 설명>> | | | |

### List of Risks

* R1
* …

### List of Sensitivities

* S1
* …

### List of Tradeoffs

* T1
* …

### List of Nonrisks

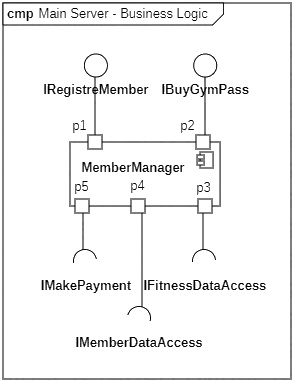
* NR1
* …

Appendix

## Detailed Component Specification for C&C View

### Component 1 Name

<< 아래 그림과 유사하게 UML 컴포넌트 다이어그램으로 컴포넌트의 인터페이스를 모델링하기 바랍니다.>>



##### Interface List

<점검 사항>

* 해당 Component의 모든 interface가 정의되었는가?
* 각 interface의 유형(Provided/Required)가 UML Component Diagram과 부합하는가?
* 각 interface의 역할/기능을 명확하게 설명하였는가?

|  |  |  |
| --- | --- | --- |
| Interface Name | Kind | Description |
| IRegisterMember | Provided | 회원 가입을 요청하는 interface |
| IBuyGymPass | Provided | 이용권 구매를 요청하는 interface |
| IMakePayment | Required | 결제를 요청하는 interface |
| IMemberDataAccess | Required | DB에 저장된 회원 정보에 접근하는 interface |
| IFitnessDataAccess | Required | DB에 저장된 피트니스 정보에 접근하는 interface |

### Component 2 Name

## Interface specifications for C&C View

### *Interface2\_1* Interface Specification

<작성 방법>

* 인터페이스를 구성하는 각 operation 별로 구체적인 인자/반환 타입을 기술한다.

<점검 사항>

* Component에 할당된 모든 기능이 Operation으로 할당되었는가?
* Operation의 이름이 제공되는 기능을 명확히 나타내는가?
* Operation의 인자/반환 타입이 올바르게 정의되었는가?

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| Operation | Responsibility |
| op1() |  |
| op2() |  |