SUNG KYUN KWAN UNIVERSITY

설계명세서

SARA

성균관대학교 교내시설 실시간 예약 통합시스템

2013314177 김다솜

SUNG KYUN KWAN UNIVERSITY

Table of Contents

SUNG KYUN KWAN UNIVERSITY

1. Preface

Preface에서는 본 문서의 독자가 누구인지 밝히고 문서의 구조와 각 장의 역할을 기술한다. 또, Version History를 제시하여 문서의 새로운 버전이 만들어질 때마다 변경사항을 서술한다.

1.1 Readership

시스템 앤드유저, 클라이언트 엔지니어, 시스템 아키텍트, 소프트웨어 개발자

설계 명세서의 대상 독자는 다음과 같다. 시스템을 설계하는 시스템 아키텍트, 각 서브 시스템을 구현하는 소프트웨어 개발자 및 클라이언트 엔지니어 및 고객 기술 지원을 위한 서비스 팀도 해당한다.

1.2. Document Structure

각 장의 역할과 전반적인 서술 내용은 다음과 같다.

A. Preface

Preface에서는 본 문서의 예상 독자가 누구인지 밝히고, 문서의 구조, 각 장의 역할과 내용에 대해 기술한다. 또, Version 관리 정책을 제시하고 이에 따른 Version History와 그 변경사항에 대해 기술하였다.

B. Introduction

Introduction에서는 사용자 관점에서의 시스템에 대해 서술한다.

C. System Architecture

System Architecture에서는 전체 시스템에 대한 개요를 서술한다. 시스템의 구조를 Block diagram으로 설명하고 컴포넌트의 관계를 Package diagram, Deployment diagram으로 표현한다. 각 서브 시스템의 자세한 설명은 단원을 나누어 설명한다.

D. Standard

E. Human Resource Management System

F. Reservation Management System

G. Access Control System

H. Protocol Design

I. Database Design

J. Testing Plan

K. Development Environment

L. Index

Index 에서는 주요 용어, 다이어그램, 기능에 대한 인덱스 등이 포함된다.

M. Appendices

Appendices에서는 개발되는 목표 시스템과 관련된 구체적인 정보를 제공한다. 목표 시스템에 사용될 hardware, database등을 설명한다. 목표 시스템 사용에 필요한 설정과 데이터베이스 상에서 결정되는 데이터의 관계 및 구조에 대하여 설명한다.

1.3. Version of the Document

A. Version format

버전 번호는 major number와 minor number로 이루어져 있으며, "(major number).(minor number)"로 표현된다. 본 문서의 버전은 0.1부터 시작한다.

B. Version management policy

본 요구사항 명세서를 수정할 때 마다 버전을 업데이트한다. 다만 변경간의 간격이 1시간 이내일 때에는 버전 번호를 추가로 업데이트 하지 않고 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경할 때에는 minor number 를 증가시키며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 major number 를 증가시킨다.

C. Version Update History

Table 1. Version Update History

Version	Modified date	Explanation
0.1	**2016-05-02**	문서 초안 작성.
1.0	**2016-05-05**	Introduction, Preface, User Requirement Definition, System Architecture 작정.
1.1	**2016-05-07**	User Requirement Definition, System Architecture 수정
2.0	**2016-05-09**	System Requirement Specification 작성
3,0	**2016-05-10**	Scenario 작성.
3.1	**2016-05-12**	System Requirement Specification의 기능적, 비기능적 요구사항 구체화
4.0	**2016-05-14**	System Models, Context diagram 작성
4.1	**2015-05-15**	Context model 수정
5.0	**2016-05-16**	Use case 작성
6.0	**2016-05-18**	Sequence Diagram 작성
6.1	**2016-05-20**	Use case 수정
7.0	**2016-05-21**	Structure models 작성
8.0	**2016-05-23**	Behavioral models, Data-driven modeling 작성
8.1	**2016-05-25**	Data-driven modeling 수정, Event-driven modeling 작성
9.0	**2016-05-27**	System Evolution, Appendices 작성
9.1	**2016-05-29**	Index, contents, appendices 작성, 오류 수정

2. Introduction

이 장에서는 시스템의 배경 및 필요성, 시스템에 포함된 서브시스템들의 간략한 소개와 이 들 사이의 interaction에 대해 서술하고 목표 시스템의 기대효과에 대하여 설명한다.

2.1. 배경 및 필요성

성균관대학교 학생들은 최소 2~3개 이상의 Team Project에 참여하고 있을 것으로 생각된다. 그룹 활동을 하면서 함께 만나서 사용해야 하는 공간이 필요하고 그에 따른 예약과정이 필수적이다. 우리는 현재 교내 공간예약신청 방법에 대해 조사한 후 많은 문제점이 있다고 느꼈다. 우리는 소비자의 니즈를 조사하기 위해 "교내 시설 실시간 예약 시스템" 이라는 이름으로 99명의 성균관대학교 자연과학캠퍼스를 이용하는 학생들을 표본으로 설문조사를 진행하였다.

현 온라인, 오프라인 예약시스템에 관해 몇 가지 불편함이 있었다. 먼저, 70%의 학생들이 세미나실을 대관한 경험이 있었고 온라인으로 세미나실을, 오프라인으로는 세미나실, 운동장, 강당, 공연장 순으로 예약한 횟수가 많았다. 오프라인으로 시설예약을 하는 경우에는 직접 찾아가서 예약하기 귀찮다는 의견이 76.6%로 압도적으로 많았다. 온라인으로 시설예약을 하는 경우에 느낀 불편함은 모바일로 예약이 불가능 하다는 것에 높은 응답률을 보였다. GLS 공간예약신청은 모바일 앱으로 이용할 수 없기 때문이다.

온라인의 경우, 대부분 GLS에서 이루어지지만 예약이행여부, 실시간 예약취소 등이 반영되지 않아 실시간으로 이용가능 여부를 확인할 수 없었고 모바일 환경에서 예약서비스를 이용할 수 없다는 것을 문제점으로 보았다. 일례로 온라인 신청 후 이틀 뒤에 행정실을 방문하였는데 신청했던 예약이 취하되어 강의실을 사용하지 못했던 경험이 있다. 오프라인의 경우에는 빌리려고 하는 장소에 담당하는 행정실의 위치를 찾고 해당행정실의 운영시간 내에 직접 방문하여 예약해야 하는 불편함이 있었다. 이를 개선하기 위해 오프라인 예약을 전부 온라인에서 처리하도록 하여 "교내 시설 실시간 예약 시스템(SARA시스템)"을 개발하고자 한다.

성균관대학교 자연과학캠퍼스에서 학내구성원이 대관 예약 가능한 시설은 다음과 같다.

Table 2. 성균관대학교 자연과학캠퍼스에서 예약가능한 시설

건물번호	호실

제1공학관 22동	22003 22005 (일반강의실)
제1공학관	해동학술정보관 세미나실
제2공학관 26동	26310 (일반강의실)
제2공학관 27동	27515 27511 (일반강의실)
제2공학관	공대 학습실 내 세미나실
제1과학관 31동	31316 31213 (일반강의실)
제2과학관 32동	32362(스터디룸)
화학관	330102(대강의실) 330118 330126 (첨단강의실)
기초학문관 51동	51308 51318 51352B 51213 51356(일반강의실) 51152 (PC실) 51303B (스터디룸)
생명과학관 61동	61307 61209 61208 (일반강의실)
생명과학관 62동	62352 62351 62354 62304 62353 (일반강의실)
제1종합연구동	81B116A (전처리실2) 81719(고분자물리및응용연구실)
제2종합연구동	83307(예비)
DRC관	840207(대표이사실)
N센터	86332(연구교수실) 86137(검사실)
삼성학술정보관	스터디룸 2-1~6 3-1~3 4-1~3
학생회관	소강당 뜨락
	운동장

* 오프라인만 가능

온라인으로 제공되는 예약 시스템의 예약 프로세스는 다음과 같다.

A. 삼성학술정보관 스터디룸

학술정보관 사이트 -> 서비스 -> 시설좌석이용 -> 스터디룸/캐럴 이용안내 -> 신청, 조회

Figure 1. 삼성학술정보관 스터디룸 예약

B. 일반강의실

GLS -> 신청/자격관리 -> 학생생활관련신청 -> 공간예약신청

담당 행정실의 사용 신청 확인은 평일 오전 9-5시 사이에 이루어져야 하며 당일 신청은 행정실 방문예약만 가능하다. 신청취소는 하루 전까지 온라인으로 가능하다.

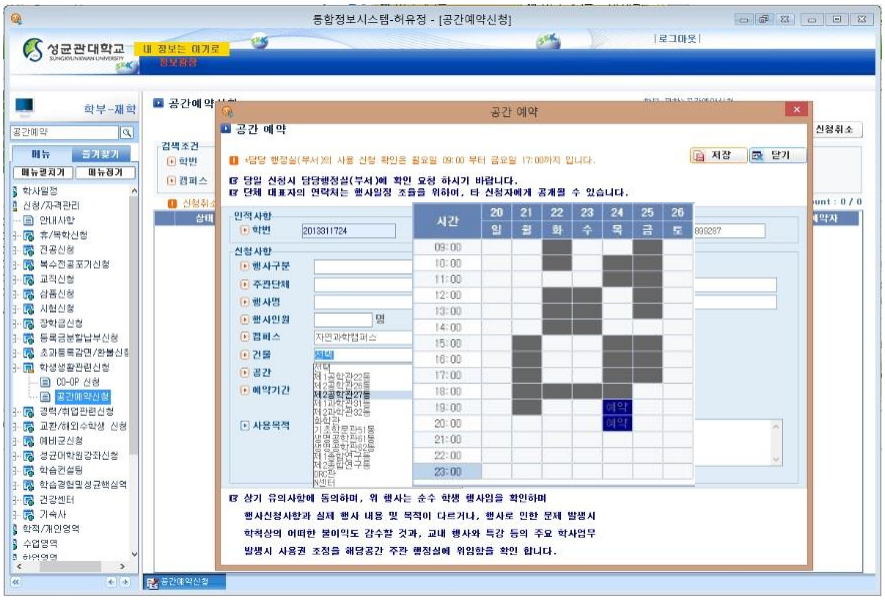


Figure 2. GLS 공간예약 신청

오직 오프라인으로만 예약 가능한 공간은 해동학술정보실 세미나실, 공대 학습실 내 세미나실, 학생회관 소강당, 운동장에 해당한다.

세미나실은 해당 학생회실에 방문하여 예약해야 한다. 소강당은 학생지원팀 홈페이지에서 인터넷 예약 신청 -> 인터넷 신청 후 다음날 까지 소강당 사용신청서 및 준수사항 학생지원팀 방문 제출 -> 간단한 심사 후 승인(교내단체, 적합단체확인 등) 절차로 예약해야 한다. 운동장은 스포츠과학부 학과 사무실에서 예약해야 한다.

C. 뜨락

오프라인으로만 예약이 가능한 뜨락의 경우, 공간 예약은 다음과 같이 동아리 내 카페에서 동

아리 구성원들의 합의 하에 진행되고 있다.

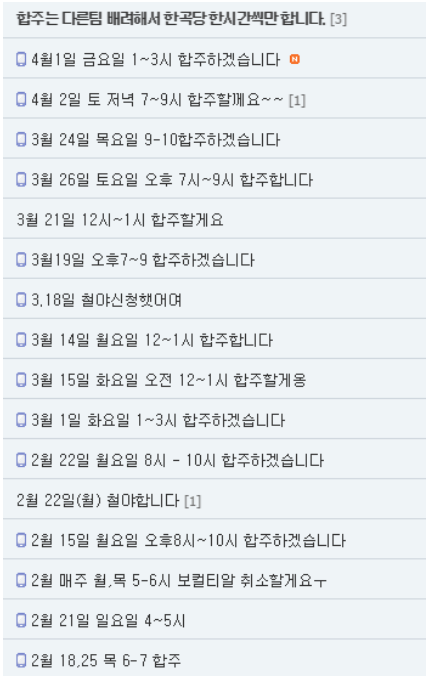


Figure 3. 뜨락 예약 예시

2.2. 기대효과

SARA 시스템은 기존의 교내 공간예약시스템을 하나의 시스템으로 통합하고, 행정실과 같은 오프라인 시설을 거치지 않고 예약을 언제 어디서나 가능하도록 하여 시설 관리실 운영시간 내에 담당 행정실의 승인을 받아야 하는 불편함을 해소하고 시간과 공간의 제약을 극복하게 한다. 예약자가 예약취소를 할 시에 실시간으로 사용현황을 파악해서 자동적으로 예약과 취소를 가능하게 하여 실제 사용되지 않는 시설의 낭비를 방지한다. 또한 벌점 제도로 악의적인 사용자의 사용을 막는다. 또한 사용자에 맞는 맞춤형 서비스를 제공하여 사용자의 편의를 높인다.

2.3. SARA

SARA 시스템은 강의실, 소강당, 대운동장을 비롯한 성균관대 내 총체적인 교내 시설 예약을

웹을 통해 실시간으로 등록 및 확인 가능한 서비스를 제공하는 시스템이다. 이용자들은 웹으로 제공되는 서비스를 통해 실시간으로 시설들을 예약하고 예약된 내역을 확인한다. 예약은 학내 구성원 인증 후에 등록 가능 하며, 예약자는 예약 시간 내 사용 가능한 일회용 비밀번호(One Time Password)로 시설이용권한을 부여 받는다. 예약한 시간 내 시설을 이용하는 경우에는 출입이 허용되고, 이용 가능 시간을 초과한 경우 예약은 자동으로 취소되고 웹에 예약 현황이 실시간으로 반영된다.

SARA는 세 가지 서브 시스템으로 구성되어 있다. 세 가지 서브 시스템에는 인적자원 관리 시스템, 예약 관리 시스템, 출입 관리 시스템이 있다.

인적 자원 관리 시스템은 사용자의 전체적인 정보를 관리하는 시스템으로, 사용자의 정보 관리 및 학내 구성원 인증 관리를 담당하고 있다. 첫번째로 인적 자원 관리 시스템을 통해 회원가입 및 학내 구성원 인증을 완료하고, 예약을 한 후 사용자의 예약 인증 여부를 통해 예약 관리 시스템에서 받은 정보로 사용자의 페널티 정보를 관리한다. 또한 회원가입 후에 사용자는 정보를 수정할 수 있고, 관리자 또한 회원정보를 관리할 수 있다.

예약관리 시스템은 이용자들에게 전면으로 드러나는 시스템으로, 직관적인 UI를 제공하여 사용자의 서비스의 이용을 돕는다. 예약 요청이 들어오면 요청한 예약 정보(예약 시간, 예약 시설, 예약자)가 이미 존재하는 예약 정보인지 데이터베이스를 통해 중복 여부를 확인한다. 중복되었다는 메시지를 받았을 경우에는 웹 UI를 통해 거절 메시지를 보낸다. 중복되지 않았을 경우, 해당하는 예약 정보에 OTP를 생성하여 저장하며 본 시스템은 이 비밀 번호를 이용자에게 전달한다. 이용자가 변심하여 이미 예약한 사항을 취소하였을 경우 데이터 베이스에서 예약 내역을 삭제한다.

출입 관리 시스템은 일종의 IOT 시스템으로, 시설들의 입구에 물리적으로 존재하여 예약 관리 시스템과 메시지를 주고 받아 시설을 열거나 닫는 시스템이다. 사용자로부터 비밀번호를 입력 받아 자신의 시설 정보와 현재 시간 정보를 덧붙여 예약 관리 시스템에 전달하면, 예약 관리 시스템에서 인증 여부를 확인한다. 인증이 성공하면 출입 허용 메시지를 보내게 되고 이 메시지를 받은 출입 관리 시스템은 문을 열게 된다. 인증이 실패할 경우에는 어떤 메시지도 보내주지 않는다.

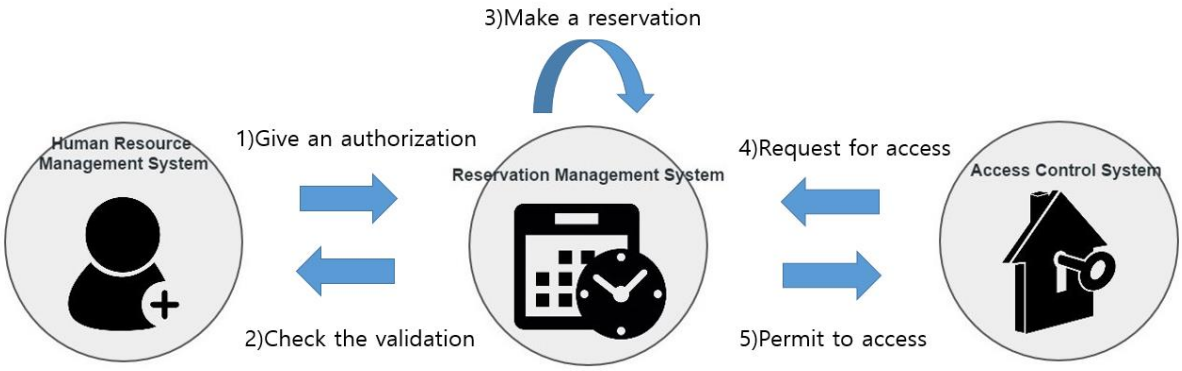
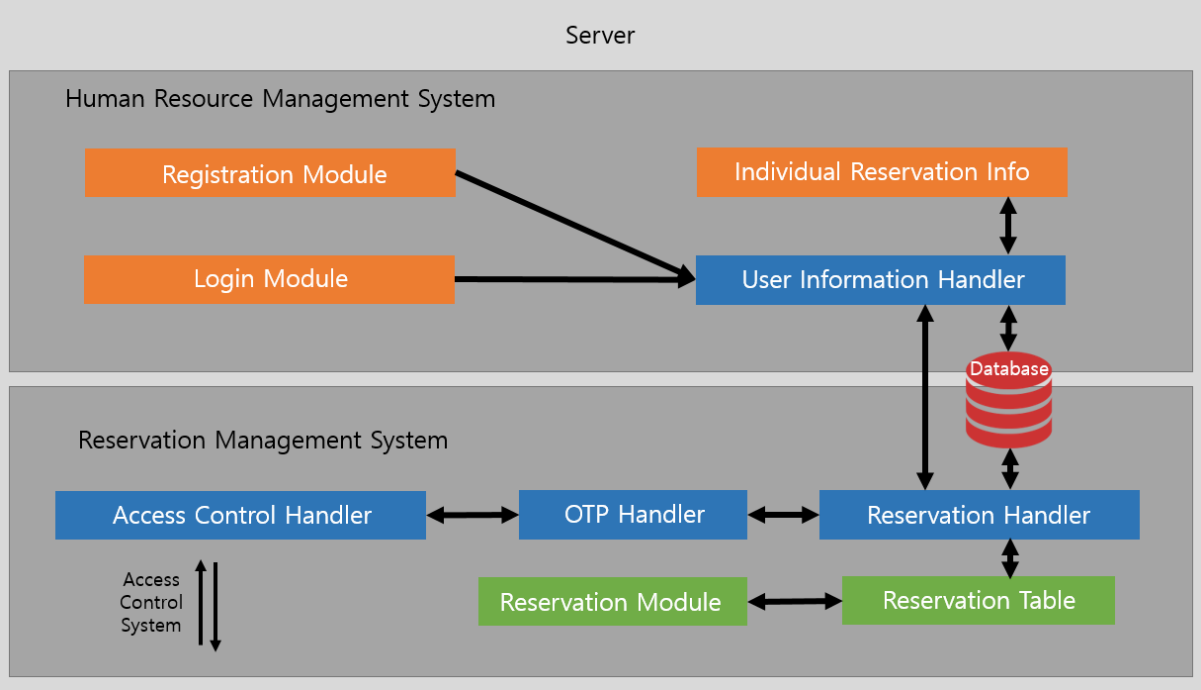


Figure 4. 시스템 구성도

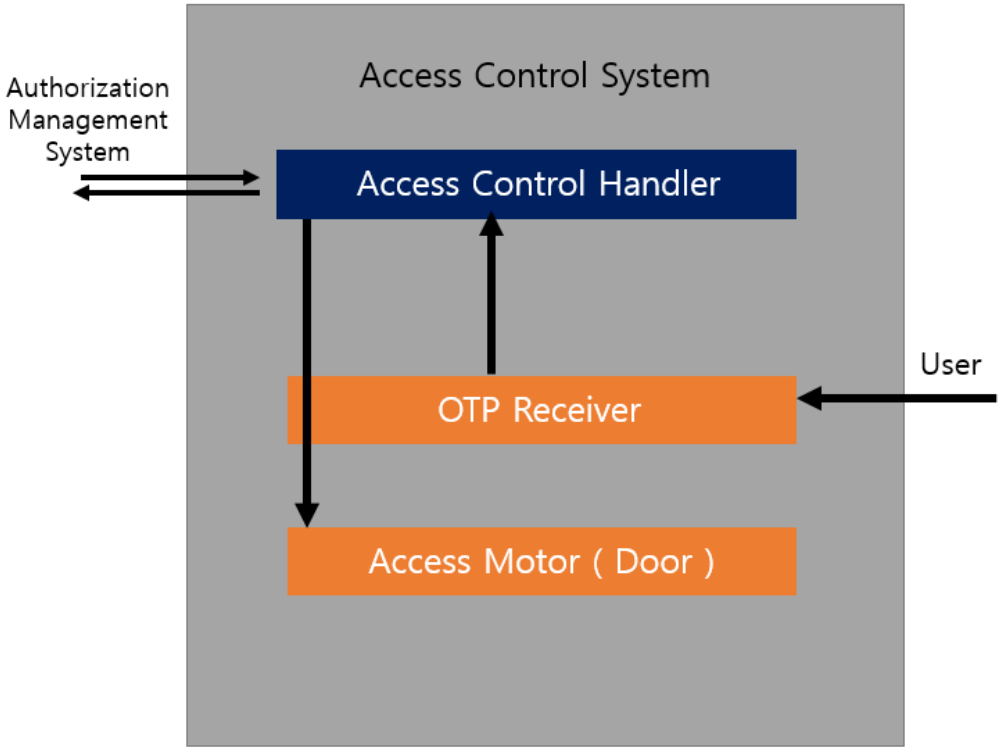
3. System Architecture

3.1 전체 시스템 구조

SARA 시스템은 크게 서버와 출입 관리 시스템으로 구분되며, 서버는 인적 사항 관리 시스템과 예약 관리 시스템으로 구성되어 있다. 전체적인 구조는 다음과 같다.

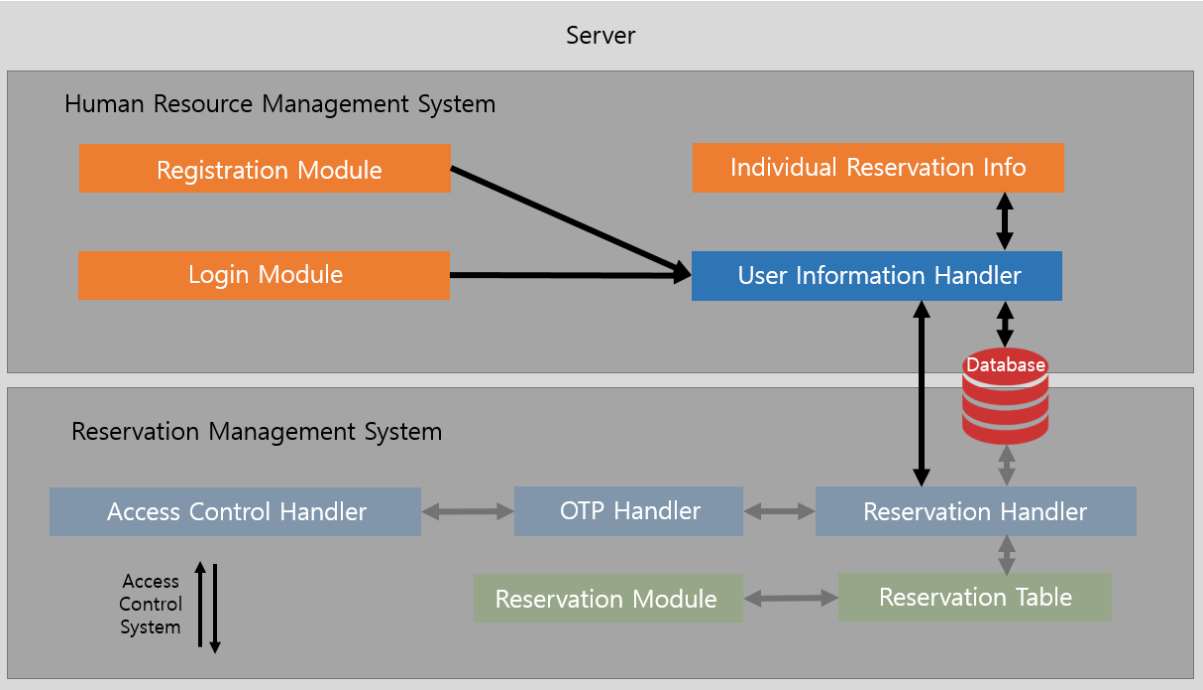


인적 자원 관리 시스템의 주황색 컴포넌트는 사용자가 서비스를 이용하기 전에 호출되어야 하는 모듈들로 사용자의 정보를 저장하고 관리하며 사용자가 예약 서비스를 이용할 때 특정 사용자임을 인증할 수 있게 한다. 예약 관리 시스템의 초록색 컴포넌트는 시스템에서 사용자에게 실질적 예약 서비스를 제공하는 컴포넌트로 사용자와의 상호작용을 통해 작동한다. 시스템 전체의 파란색 컴포넌트는 유저 인터페이스 없이 백그라운드에서 동작하는 서비스로, 주황색 컴포넌트와 초록색 컴포넌트를 통해 작동된다. 주로 데이터베이스와 직접 상호작용하여 정보를 실시간으로 갱신하거나 조회하는 기능을 한다.



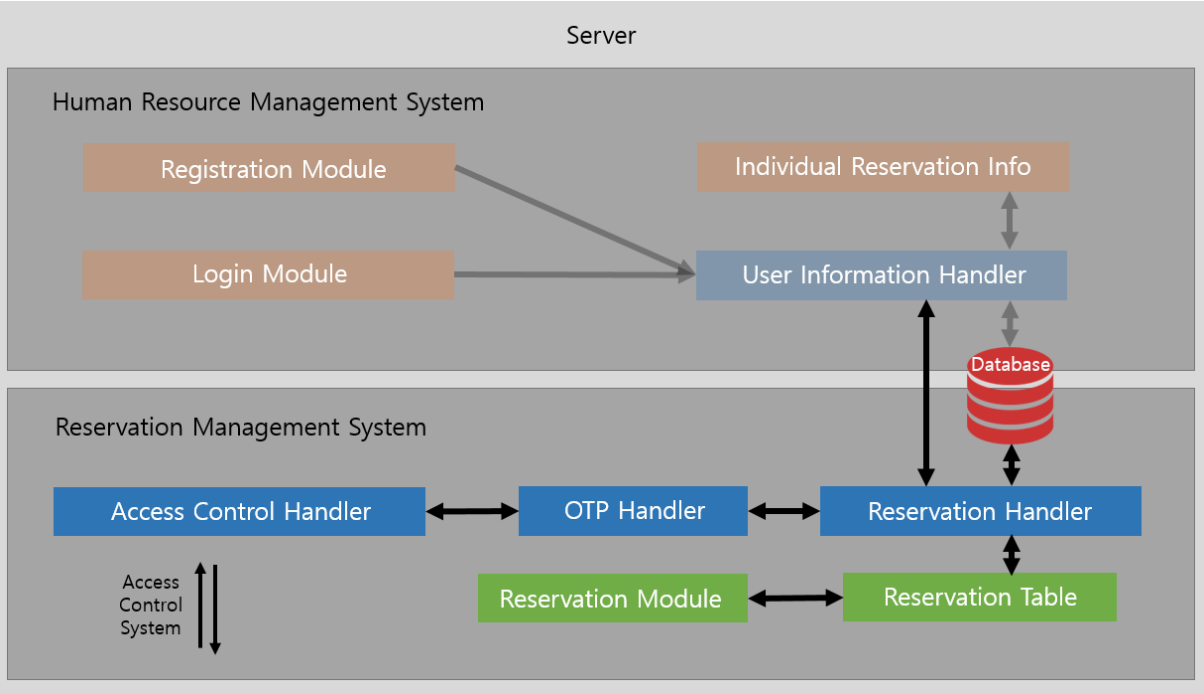
출입 관리 시스템 역시 예약 관리 시스템과 통신하기 위하여 출입 관리 핸들러 컴포넌트가 존재한다. 이 컴포넌트는 OTP Receiver로부터 받은 OTP로 인증을 수행한다. 주황색 컴포넌트는 User에게 직접적으로 보여지는 외부 장치와 연결되어 있으며, OTP Receiver는 User로부터 OTP를 입력 받는 역할, 출입 모터는 문을 열고 닫는 역할을 한다.

3.2 인적 사항 관리 시스템



인적 사항 관리 시스템은 사용자들에게 사용자의 정보를 관리하고 예약 서비스를 이용하였을 경우 해당 예약 정보를 이용한 예약한 사용자에게만 한하여 보여주게 한다. 사용자는 예약 서비스를 이용하기전에 로그인 모듈, 회원 가입 모듈을 작동시키면 사용자 정보 핸들러를 통해 해당 모듈들의 기능을 수행한다. 이 때 사용자 정보 핸들러는 사용자의 ID, 비밀번호 등의 정보들을 조회하거나 갱신한다. 개인 예약 정보 모듈은 개인 정보와 예약 정보들을 사용자 정보 핸들러를 통해 조회하게 되는데, 이 때 사용자 정보 핸들러는 데이터베이스로부터 사용자의 정보를 받아오고 그와 동시에 예약 관리 시스템의 예약 핸들러로부터 해당 사용자의 예약 정보들을 전달받는다.

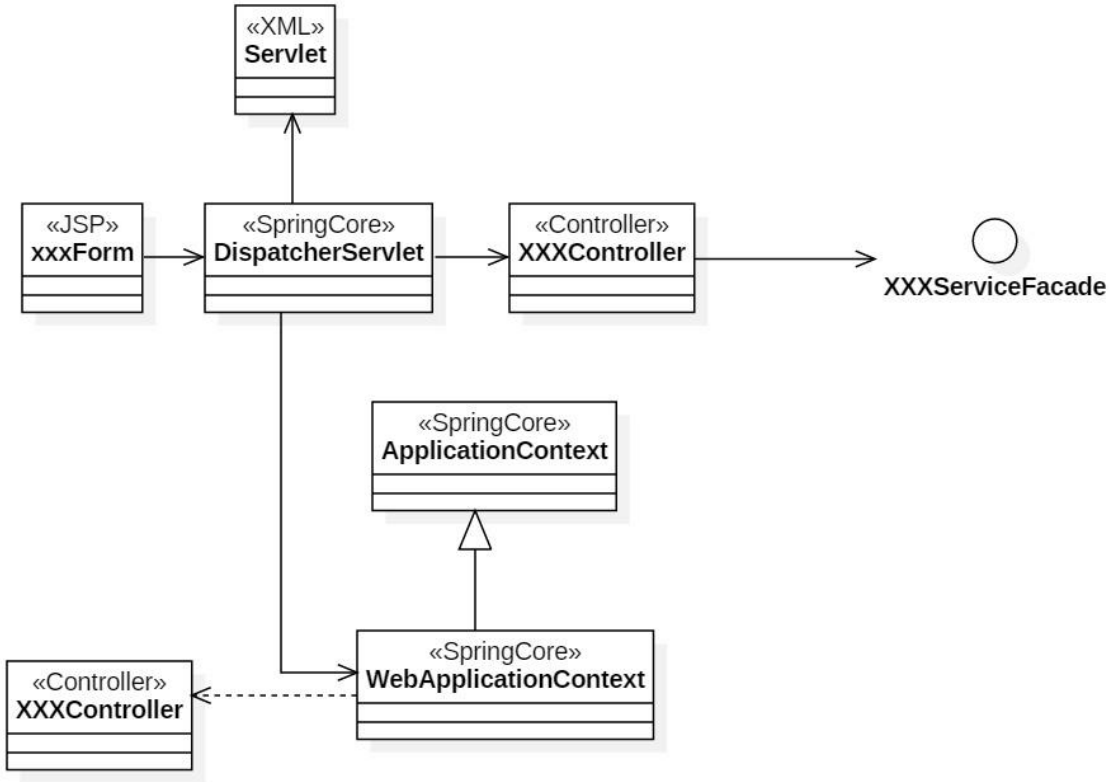
3.3 예약 관리 시스템



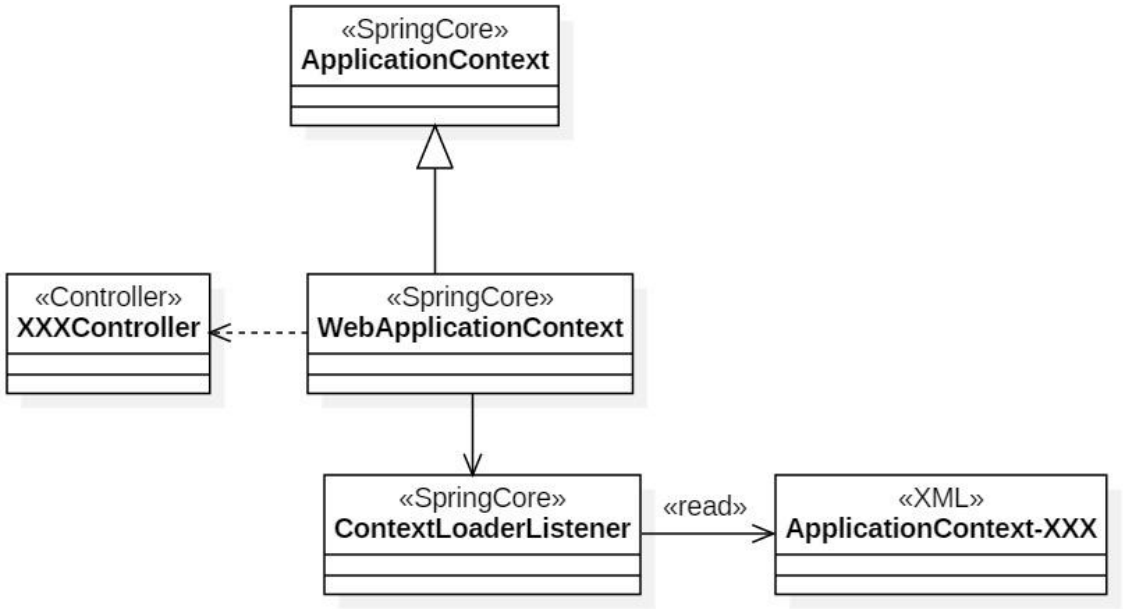
예약 관리 시스템은 사용자에게 실질적인 서비스를 제공하는 시스템이며 데이터 베이스를 통해 예약 정보, 그리고 OTP에 대한 정보들을 관리한다. 예약 모듈은 사용자에게 예약 서비스를 제공한다. 예약 모듈이 작동되면 우선 예약 핸들러가 데이터베이스에서 예약 정보를 조회하여 시간표의 현황을 보여주고 그 내용을 보고 사용자에게 예약 테이블을 통해 예약 정보를 요청한다. 사용자로부터 받은 예약 정보와 사용자 정보를 예약 핸들러에게 전달하고, 예약 핸들러는 사용자 정보 핸들러와 데이터베이스를 조회하여 예약 요청의 유효성을 판단하고 받아들이거나 기각한다. 예약이 유효하다고 판단할 때, 예약 핸들러는 OTP 핸들러를 통해 OTP를 제공받아 데이터베이스에 예약 정보와 연결하여 저장한다. OTP 핸들러는 예약 핸들러에서 OTP 생성 요청을 받으면 OTP를 생성하거나 출입 관리 핸들러로부터 받은 OTP가 유효한 OTP인지를 검사한다. 출입 관리 핸들러는 출입 관리 시스템으로부터 받은 OTP를 OTP Handler를 통해 그 인증 여부를 확인하고 그 인증 여부를 출입 관리 시스템에게 전달해준다.

SUNG KYUN KWAN UNIVERSITY

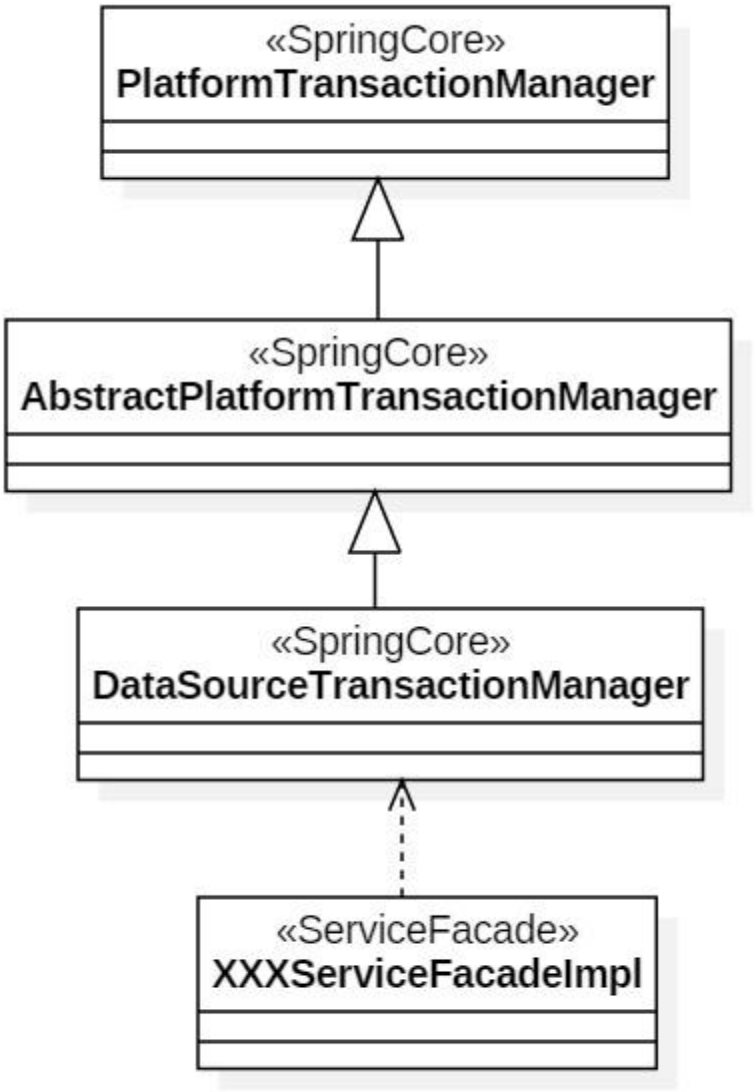
1.Spring4.0-ViewControl



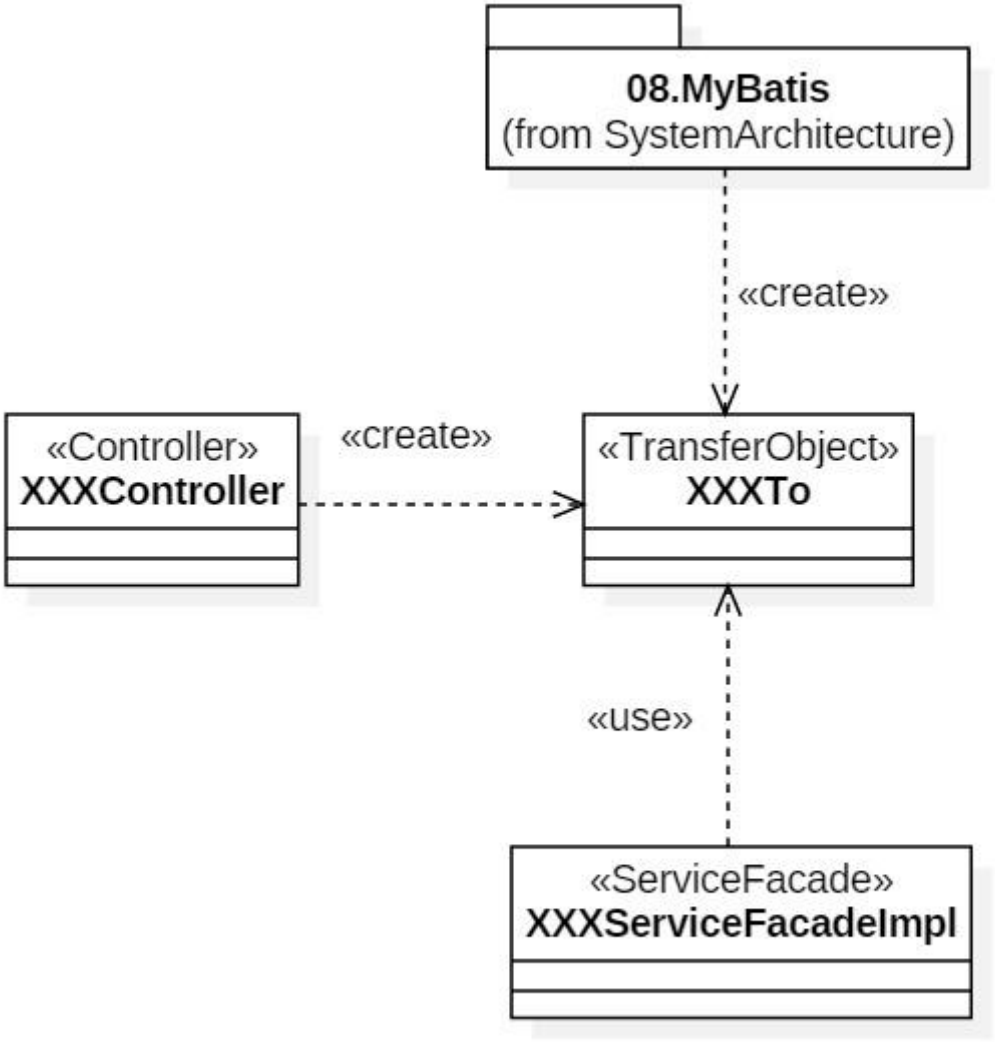
2.Spring4.0-Model



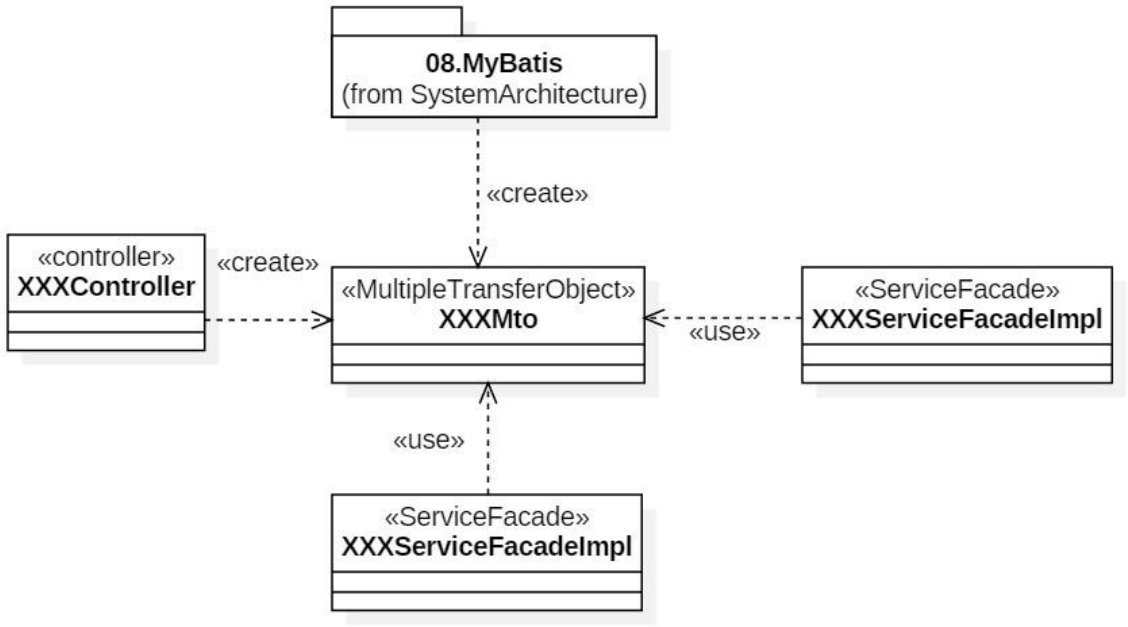
3.Spring4.0-Transaction

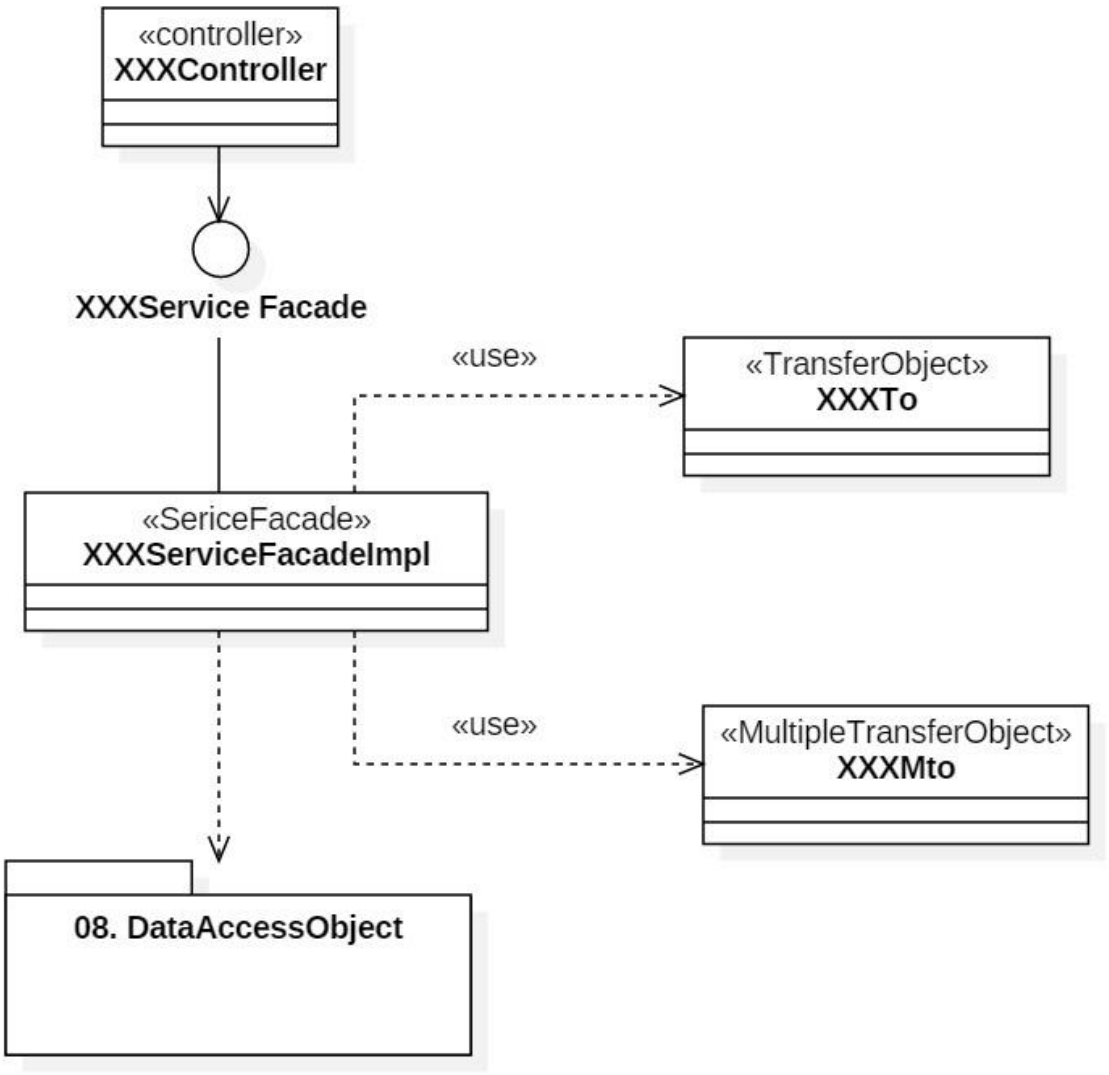


4.TransferObject

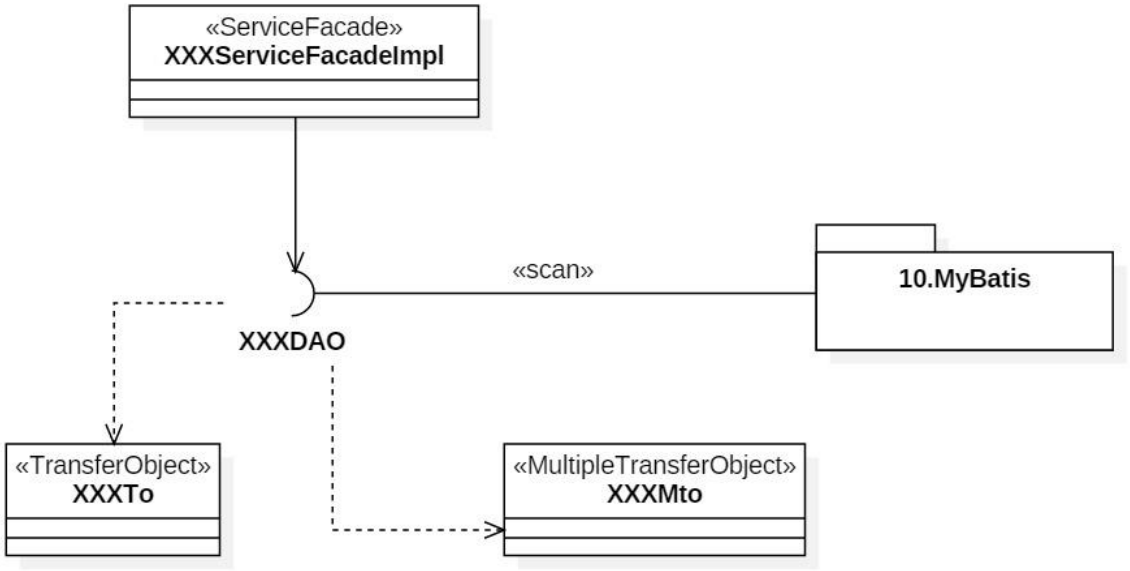


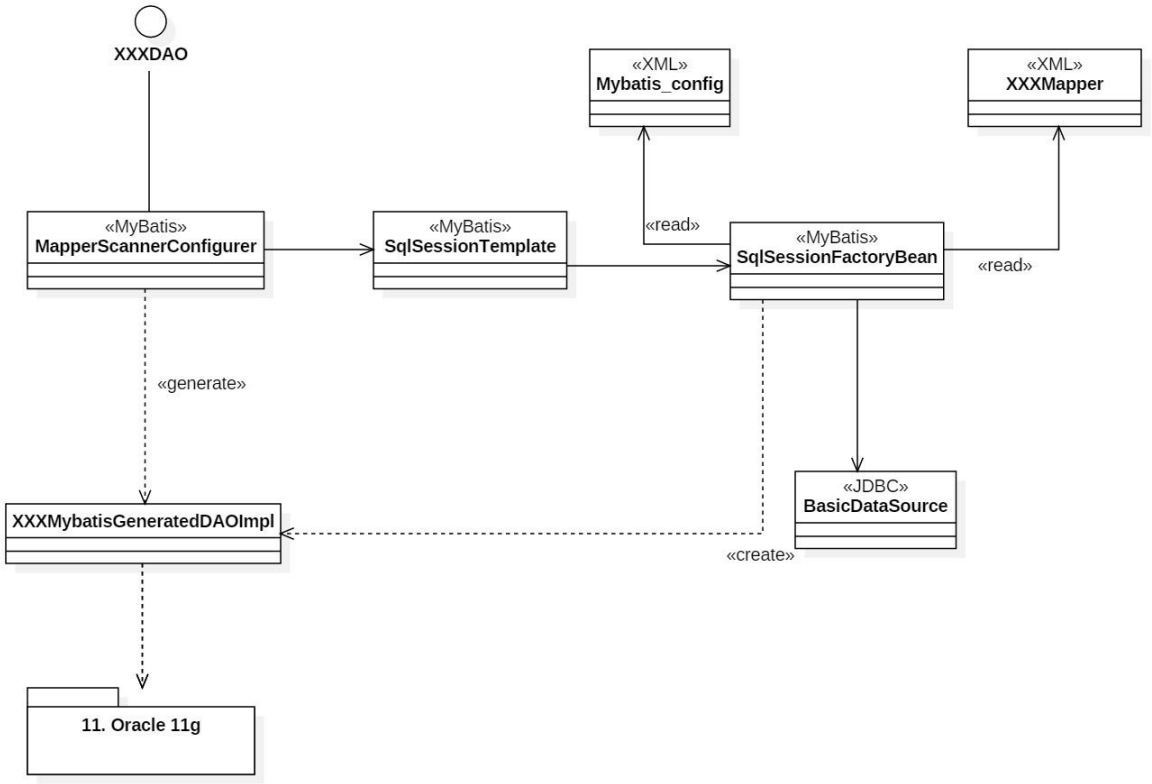
5. Multiple Transfer Object


08.MyBatis
(from SystemArchitecture)
«create»
«controller»
XXXController
«create»
«MultipleTransferObject»
XXXMto
«use»
«ServiceFacade»
XXXServiceFacadeImpl
«use»
«ServiceFacade»
XXXServiceFacadeImpl

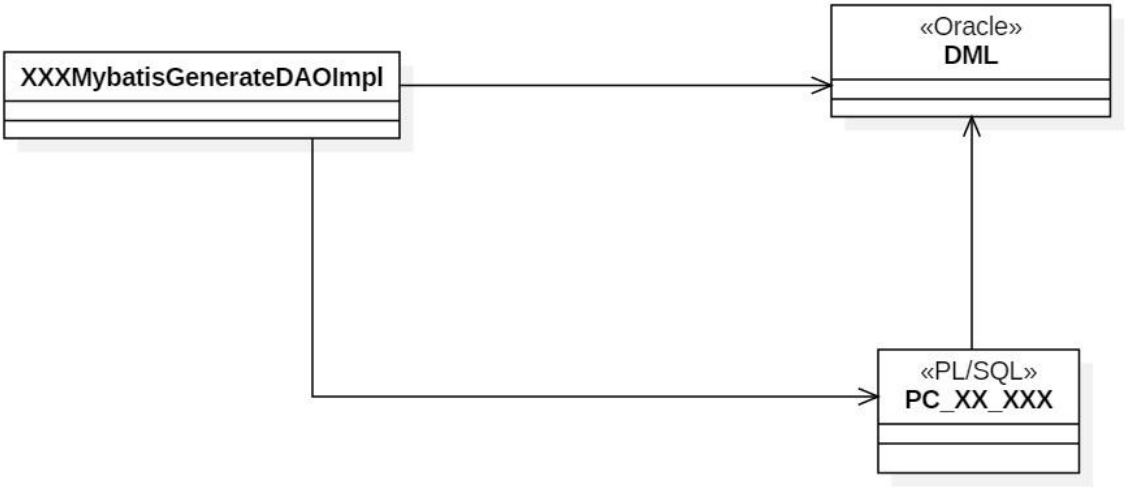


7.Data Access Object

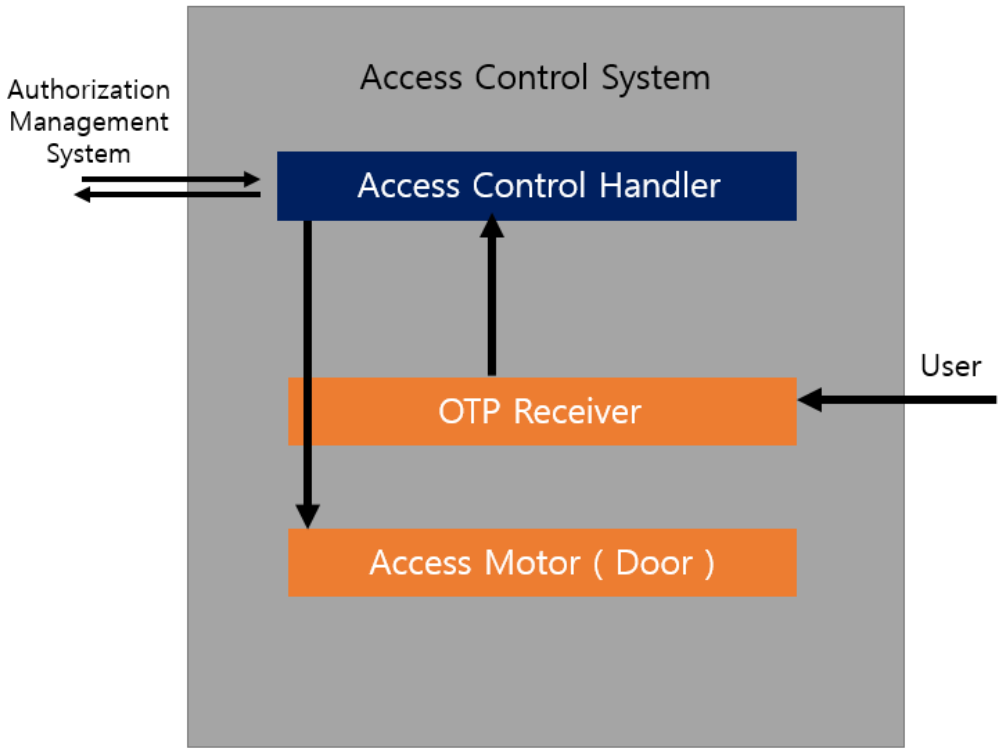

«ServiceFacade»
XXXServiceFacadeImpl
«scan»
10.MyBatis
XXXDAO
«TransferObject»
XXXTo
«MultipleTransferObject»
XXXMto


XXXDAO
«XML»
Mybatis_config
«XML»
XXXMapper
«MyBatis»
MapperScannerConfigurer
«MyBatis»
SqlSessionTemplate
«read»
«MyBatis»
SqlSessionFactoryBean
«read»
«generate»
«JDBC»
BasicDataSource
XXXMybatisGeneratedDAOImpl
«create»
11. Oracle 11g

9.Oracle 11g


«Oracle»
DML
XXXMybatisGenerateDAOImpl
«PL/SQL»
PC_XX_XXX

3.4 출입 관리 시스템

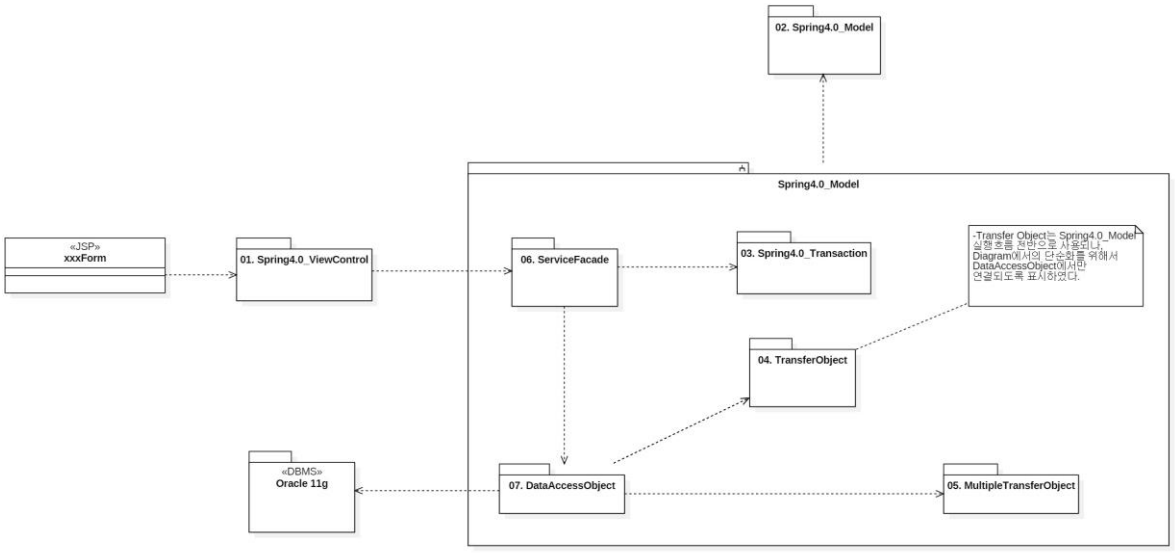


출입 관리 시스템은 사용자에게 최종적으로 서비스를 제공하는 시스템으로 디지털 도어락과 동작 방식이 비슷하며 물리적으로 존재하는 시스템이다. 주황색 컴포넌트은 물리적으로 외부장치와 연결 되어있는 시스템들이다. OTP Receiver는 소형 키보드인 넘버 패드에 연결되어 있으며 이 넘버 패드를 통해 사용자로부터 키 값을 받는다. 그리고 출입 모터는 문을 실제로 열거나 닫는 서보 모터에 연결되어 있다. 사용자로부터 OTP 값을 받은 OTP Receiver는 출입 관리 핸들러에게 그 OTP 값을 전송한다. OTP 값을 전달받은 출입 관리 핸들러는 예약 관리 시스템에게 그 OTP 값을 전송하고 OTP가 유효한지에 대한 대답을 받는다. 대답의 종류에 따라 출입 모터에게 문을 여는 신호를 줘서 문을 열거나 문이 닫힌 상태를 유지하게 한다.

3.5 Deployment Diagram

3.6 Package Diagram

Reservation Management System and Human Resource Management System



SUNG KYUN KWAN UNIVERSITY

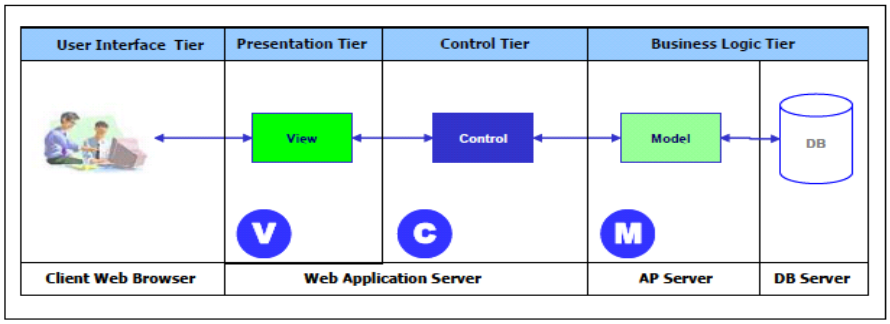
4.Standard

4.1 개발환경 표준

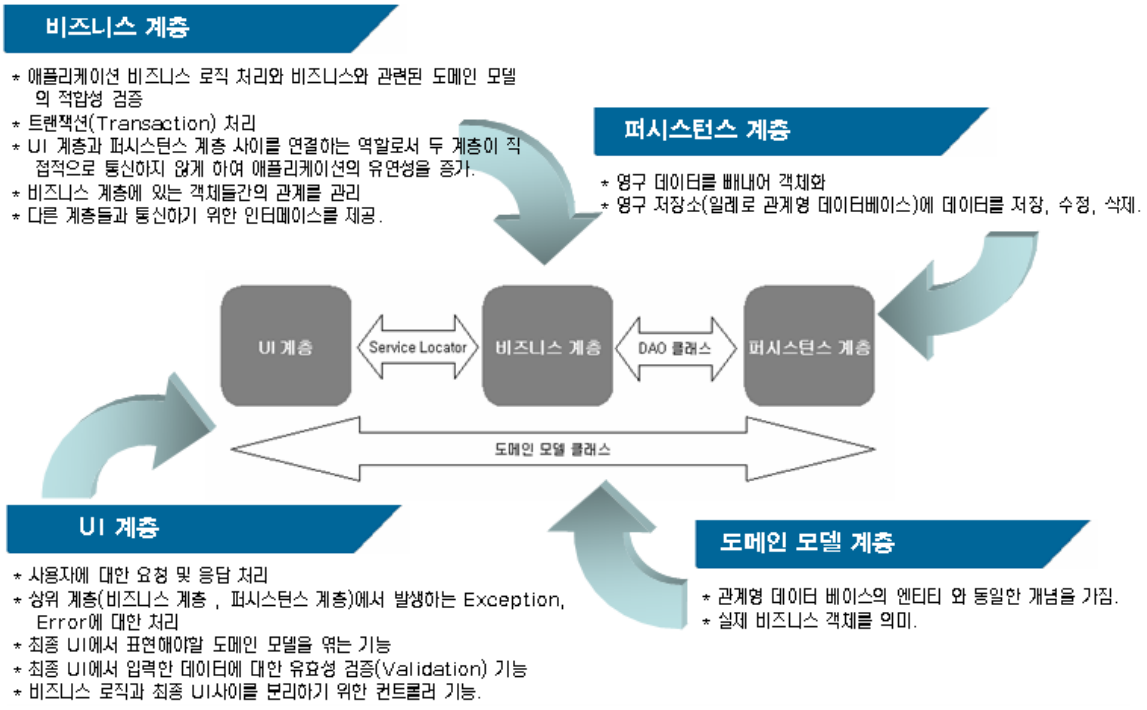
4.1.1 소프트웨어 아키텍처

4.1.1.1 Conceptual Design.

SARA 시스템의 High Level 관점에서 크게 3가지 Tier로 구성되며, 각각 User Interface Tier, Presentation Tier (Control Tier), Business Logic Tier로 나누어 볼 수 있다.



기본 아키텍처



UI 계층

사용자에 대한 요청 및 응답 처리

상위 계층(비즈니스 계층 , 퍼시스턴스 계층)에서 발생하는 Exception, Error에 대한 처리

최종 UI에서 표현해야할 도메인 모델을 엮는 기능

최종 UI에서 입력한 데이터에 대한 유효성 검증(Validation) 기능

비즈니스 로직과 최종 UI사이를 분리하기 위한 컨트롤러 기능.

비즈니스 계층

애플리케이션 비즈니스 로직 처리와 비즈니스와 관련된 도메인 모델의 적합성 검증

트랜잭션(Transaction) 처리

UI 계층과 퍼시스턴스 계층 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 애플리케이션의 유연성을 증가시킨다.

비즈니스 계층에 있는 객체들간의 관계를 관리한다.

다른 계층들과 통신하기 위한 인터페이스를 제공한다.

퍼시스턴스 계층

영구 데이터를 빼내어 객체화 시킨다.

영구 저장소(일레로 관계형 데이터베이스)에 데이터를 저장, 수정, 삭제한다.

도메인 모델 계층

관계형 데이터베이스의 엔티티와 비슷한 개념을 가지는 것으로 실제 비즈니스 객체를 의미한다.

적용 Framework.

Framework는 일종의 클래스들의 집합 혹은 협력으로서 정의될 수 있으며, 관계된 문제들의 집합을 해결하기 위한 추상화된 Design이다. Framework는 반드시 확장이 가능하거나 적용이 가능한 메커니즘과 서브시스템이 포함되어야 한다. Application Framework는 이러한 문제들을 해결하기 위한 추상화된 클래스와 패턴의 협력으로 나타낼 수 있으며 Application domain을 위한 generic software system이다. (R. Johnson)

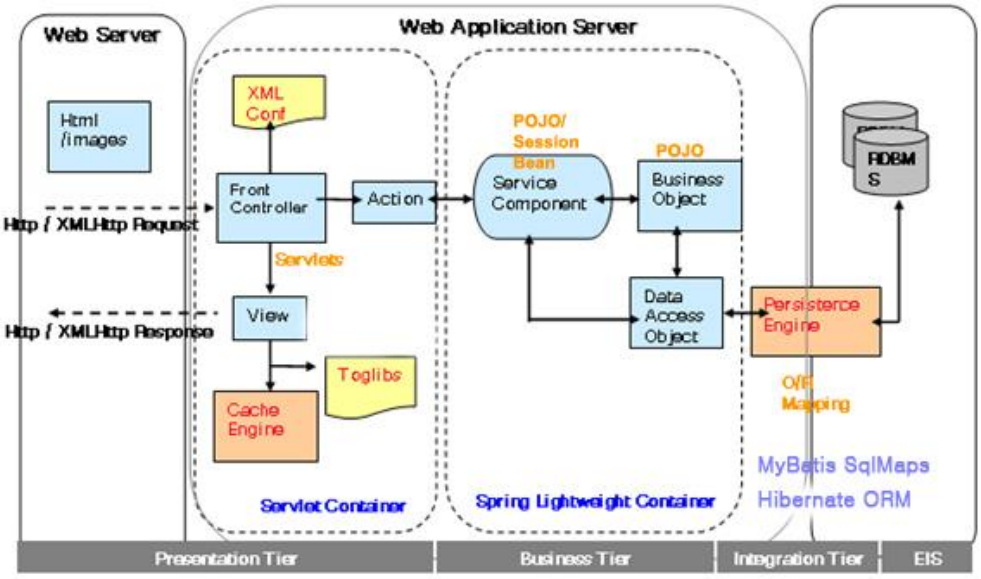
특정상황의 해법을 추상화하고, 해결방법보다는 문제 해결에 집중한다.

디자인 패턴 + 개발 기법 + 개발 가이드를 통해, 이슈를 정의 / 해결하는 목표를 설정, 달성하기 위한 일관성 있는 방법을 제시하기 위함이 목표이다.

Framework = Framework + Framework 의 사상으로 OpenSource + Framework 를 지향한다.

[적용 기술]

Spring Framework 3.2x	MyBATIS	
AJAX	XML	AspectJ
Javascript	LOG4J	



[Presentation Tier]

Spring Framework

Spring framework는 Enterprise Application에서 필요로 하는 기능을 제공하는 오픈 소스 프레임워크이다. Spring framework는 평범한 자바객체(POJO : Plain Old Java Object)를 이용해서 단순하고, 테스트하기 쉬우며, 객체간의 결합이 느슨한 Enterprise Application 개발이 가능하도록 지원한다.

[Business Tier]

TO (Transfer Object)

네트워크 트랙을 줄이고, 적은 원격 호출로 많은 데이터를 전송하며, 코드 중복을 줄여준다.

MTO(Multiple Transfer Object)

하나의 테이블에서 모든 컬럼 정보를 들고 올 필요가 없을 경우, 컬럼들을 여러개

의 TO로 나누어 데이터를 접근한다.

AS (Application Service)

비즈니스 티어의 컴포넌트와 서비스에 흩어져 있는 비즈니스 로직을 중앙 집중화 한다.

DispatcherServlet

사용자의 요청을 받는 단일 진입점의 역할을 수행한다.

Annotation Based

스프링 3.0에서 지원되는 @Annotation으로 Bean을 매핑시킴으로서 공통설정 파일에대한 접근으로 파생되는 갈등상황을 최소화 한다.

MyBatis Framework

java에서 DB를 편하게 핸들링할 수 있게 해주는 framework로서, SQL 실행 결과를 자바빈즈 혹은 Map 객체에 mapping해주는 퍼시스턴스 솔루션으로 SQL을 소스코드가 아닌 XML로 따로 분리해 관리하여 지겨운 JDBC코드로 부터 해방시켜준다. 또한 XML에서 동적으로 SQL 요소를 사용하여 쿼리문장을 프로그래밍 코딩없이 자유롭게 변환 할 수 있게 해준다. MyBatis를 통하여 개발 생산성을 극대화한다.

적용 SW표준

구분	소프트웨어	제품명 및 버전	용도
응용소프트웨어	WebServer	Apache HTTP Server	웹 서버로써 Client 요청을 WAS 에 전송하며 이미지와 같은 정적인 자원에 대한 서비스 제공
	WAS	Tomcat 7.0	웹 어플리케이션 서버로써 서블릿 컨테이너를 제공하며, 비즈니스 호출 및 제어를 담당.
	DBMS	Oracle 11g XE	데이터의 무결성을 보장하며, 사용자의 요청에 의한 데이터 를 가공 및 원하는 데이터를 추출하는 엔진을 제공
개발도구	DB Modeling 도구	ER-Win	논리적 모델 및 물리적 모델을 설계하는 도구로 활용
	IDE(통합개발환경)	Eclipse	JSP 및 어플리케이션 로직을 구현하는 도구로 활용
	UML Tool	StarUML	유스케이스 분석단계부터 설계, 구현을 거쳐 테스트하는 도구로 활용
형상관리도구	문서파일관리 SVN	ToltoiseSVN	Project동기화 및 파일관리 활용
	개발소스 SVN	SVN Team Provider	버전관리 및 개발 이력 관리
	SVN Server	subversion	개발저장소 관리
	Maven	Maven	자동 빌드 및 lib 관리 활용

4.2 J2EE 표준

Naming Convention

목적

이 장에서는 "SARA 시스템" 개발을 위하여 적용되는 Naming Rule에 대한 표준을 제시한다.

Naming Rule 표준은 개발자와 Maintenance담당자간의 교량역할을 담당하고 설계, 구현, 유지 보수에 이르는 프로젝트 전반에 걸쳐 기준을 제시하기 위해 다음과 같은 목표를 가지고 설계하였다.

표준화의 인식 확대

개발의 용이성 제공

프로그램의 표준화 및 일관성 유지

프로그램의 품질 향상

운영, 유지, 보수의 편이성 제공

1.1.1. 업무 Naming Rule

프로젝트 코드

프로젝트 코드는 "SARA System" 프로젝트를 나타내기 위한 가장 큰 개념의 구분이다

SARA는 "SARA System"을 나타내는 이름이다.

시스템 코드

본 프로젝트의 대 분류 성격의 업무를 구분하기 위해 사용한다.

업무 코드 분류

시스템공통: COM (Common System Management)

인적사항 관리: HRS(HUMAN RESOURCE SYSTEM)

예약 RMS(RESERVATION MANAGEMENT SYSTEM)

출입 ACS(ACCESS CONTROL SYSTEM)

1.1.2. 프로젝트 Directory Structure Naming

SARA 시스템 프로젝트 Directory Structure Naming

(1) 프로젝트

sara	**src**	프로젝트에서 발생되는 모든 소스의 루트.
	build	원본 build파일 Directory.
	WebContent	Web Application Directory.
	target	서버배포용 Maven build 파일 Directory.
	pom.xml	자동빌드툴 설정을 위한 Project Object Model 파일
src	**message**	원본 Framwork 기본 메시지 설정 Java Source Directory. * 필수 사항이 아니므로 필요 시 생성 사용.
	test	Test Module Java Source Directory * 필수 사항이 아니므로 필요 시 생성 사용.

SUNG KYUN KWAN UNIVERSITY

	javadocs	Javadocs 용 Source Directory * 필수사항이 아니므로 필요시 생성 사용

SUNG KYUN KWAN UNIVERSITY

(2-1) 프로젝트 -> Java Source

src/com	**sara.common**	Framwork 공통 Java Source Directory.
	sara	각 업무에 관련된 application Java Source Directory. ** 실제 개발 디렉토리*

(2-2) 프로젝트 -> Java Source -> 시스템 영역

com.sara.common	**advice**	aop 관련 Java Source Directory.
	filter	공통 메소드 관련 Java Source Directory. 필요시 생성 사용.
	to	BaseBean 관련 Java Source Directory.
	util	Utilitily Base Java Source Directory 공통 Utilitily 메소드 Java Source Directory.
	exception	공통 exception 관련 Java Source Directory.

(2-3) 프로젝트 -> Java Source -> 업무 프로그램 (=> 업무 서브시스템별 프로그램) 영역

com.sara	**com**	서브시스템중 공통 시스템 관련 프로그램 Java Source Directory.
	hrs	서브시스템중 인적사항 관련 프로그램 Java Source Directory.
	rms	서브시스템중 예약 관련 프로그램 Java Source Directory.
	acs	서브시스템중 출입 관련 프로그램 Java Source Directory.

(2-4) 프로젝트 -> Java Source -> 공통시스템 업무 프로그램 영역 -> 최종 Source Directory

com.xxx (중분류약자 3~6자)	**controller**	Controller Java Source Directory.
	dao	Dao Java Source Directory.
	dao.mapper	MyBatis SqlMap XML Source Directory.
	exception	Exception Java Source Directory.
	service	Service Java Source Directory.
	application Service	Appliction Service Java Source Directory
	to	Transfer Object Java Source Directory.

SUNG KYUN KWAN UNIVERSITY

(3-1) 프로젝트 -> WebContent

WebContent	**WEB-INF**	최종 Application Output Directory

(3-2) 프로젝트 -> WebContent-> WEB-INF

WEB-INF	**config**	**context**	Framwork 설정 파일. contextConfigLocation 과 servlet action 정보의 설정 File ● applicationContext-*.xml
		servlet	servlet-*.xml
		reports	IREPORT 관련 파일 업무코드+기능을설명하는단어.jrxml
		batis	MyBatis 설정 파일 ● mybatis-config.xml
		log4j	Log4j 설정 파일 ● log4j.xml
		property	JDBC 설정파일 ●jdbc.properties resourceBundleView 설정파일 ●resourceBundle.properties 사진 경로파일 ●imageFileLocation.properties
	classes	서버 배포 컴파일 2진 파일	
	● web.xml		

1.1.3. 프로젝트 Program File Naming

일반적으로 웹서비스에서는 단어와 단어 사이를 대소문자로 구분하는 방법인 카멜 표기법 (Camel Notation)과 단어의 첫 번째 글자를 대문자로 사용하는 파스칼 표기법을 사용하며,

모든 이름은 카멜 표기법 (Camel Notation)과 파스칼 표기법(PascalCasing Naming Convention)을 따르는 것을 원칙으로 한다.

1.모든 Java 파일은 첫글자가 대문자로 시작 (xml은 소문자) - 파스칼 표기법

2.복합 단어는 대문자로 구분 (e.g., "itemEdit.xml", "RegEmp.java) - 카멜표기법

3.밑줄("_")이나 대시("-")는 파일이름에 사용하지 않음.

4.파일이름에 공백을 사용하지 않음

5.예외로 DAO(Data Access Object),AS(Application Service)만 대문자로 표기한다.

ex> empDAO.java , empAS.java.

1.1.4. Class Naming Rule

분류	대상	비고
Business Service (SOURCE)	Model Class, Controller Class Name, Dao Class Name, Service Class Name	JAVA File에 대해서만 기술.

(1) Business Service

□ □ □ □ □ □ **클래스구분 . java** 기능을 의미하는 단어(~10) + postfix

postfix : 클래스 구분 (알파벳 대문자로 시작)

Controller	DAO	DAOImpl	ServiceFacade	ServiceFacadeImpl	AS	ASImpl	Bean
Class	Interface	Class	Interface	Class	Interface	Class	Class

1.1.5. 프로젝트 Method Naming

Method는 동사로 시작되며, 첫 글자는 소문자로 시작한다.

"동사+명사"를 원칙으로 하고, 동사의 시작은 소문자로, 명사의 시작은 대문자로 한다

Prefix 동사형의 종류

구분	Controller	Service/BD/AS	Dao
등록	register~	register~	insert~
수정	modify~	modify~	update~
삭제	remove~	remove~	delete~
상세 조회	find~Detail	find~Detail	select~Detail
목록 조회	find~List	find~List	select~List
일괄처리	batch~Process	batch~Process	batch~Process

멤버변수(Attribute)

여러 단어가 조합될 경우 조합되는 단어의 첫 번째 문자는 대문자로 시작되어야 한다.

변수명은 짧으면서도 충분히 의미가 느껴지도록 한다.

집합형일 경우에는 List로 표현된다.

예) Customer[] mCustomerList = new Customer[MAX_CUSTOMERLIST];

Method Parameter

생성자(constructor)나 set 메서드에서 멤버 필드에 값을 지정할 때, 파라미터 이름은 멤버 필드 이름과 동일하게 부여한다.

첫문자는 소문자로 시작한다.

Local Variables

라인당 하나의 변수만 선언한다.

로컬 변수마다 "//"로 설명을 달아준다.

로컬 변수는 하나의 목적으로만 사용한다.

Constant

클래스 상수 또는 일반상수로 정의된 변수명은 단어간에 밑줄 문자("_")로 분리되는 대문자를 사용한다.

static final int MIN_WIDTH = 4;

static final int MAX_WIDTH = 999;

static final int PAGE_LENGTH = 10;

분석단계

스트레오 타입	설명
businessworker	액터가 아닌 Business Domain에서 실제업무를 담담하는 업무담당자를 말한다.
usecaseRealization	유즈케이스가 어떻게 실현 될 것인지 구체화 되는 모델표현이다.
boundary	액터가 시스템에 접근하는 화면UI를 말한다.
control	액터의 요청에 따라 요청을 해당하는 액션으로 구분하여 처리하는 액션이다.
entity	Model을 처리하는 영속성 계층의 개체이다.
bo	Model과 Control을 연계하여주는 수송자이다.
list	조회를 하였을 때 조회 대상항목정보를 갖고 있다.
in	사용자가 직접 입력하여야 하는 입력정보이다.
out	시스템이 화면에 보여주는 출력정보이다.
auto_in	시스템이 자동으로 계산하거나 입력하여 주는 입력정보이다.
auto	시스템이 자동으로 계산하여 출력하여 주는 출력정보이다.
select	입력 정보 선택 시 보기에서 선택하는 동작을 나타낸다.
search	검색조건으로 들어가는 입력정보이다.
hidden	화면에 출력 되지 않는 숨겨진 정보이다.

1.1. UML 스트레오 타입

설계단계

스트레오 타입	설명
Server Programming	
useCaseRealization	
Controller	
ServiceFacade	
ApplicationService	
DataAccessObject	
xml	
to	

UI(User Intefrace)

search	검색할 조건을 속성으로 가지는 Collection
[i_][o_][io_]edit	개발자가 입출력하는 속성을 가진 Edit Component 또는 타입
dialog	Modal Dialog를 띄워서 사용자가 부모 폼의 Component에 사용자가 선택한 값을 전달
popup	부모 Form과 해당 dialog사이의 **관계**에 사용
button	마우스의 Click을 통해 지정한 Script를 수행
calendar	날짜와 시간을 입력받기위한 Component
[i_][o_][io_]checkBox	선택된 상태/선택되지 않은 상태를 Switch 하면서 체크표시
[i_][o_][io_]combo	여러 후보값중에서 1개의 값을 편리하게 선택하기위한 용도로 사용
div	하나의 화면에 여러개의 부분화면을 구성할 때 사용
dataset	데이타를 Table 형태로 저장하는 Component
binds	데이터셋과 다른 컴포넌트간에 바인딩할 때 사용
innerBinds	바인딩된 데이터셋을 바인딩하는데 사용
fileDownload	FileDownload 를 수행하는 Component
imageView	ImageViewer는 Image를 화면에 출력하기위해 사용하는 Component
JS	화면을 구성하는 기본 단위로써 내부적으로 각 컴포넌트들을 배치하여 Business 화면을 구성 Form Component
grid	Dataset의 내용을 격자 모양으로 표현
groupBox	border 영역에 title을 입력하여 그룹 형태를 나타냄
[i_][o_][io_]listBox	ListBox는 여러개의 후보값중에서 1개 또는 여러개의 값을 선택하기위해 사용
maskEdit	정해진 규격에 따라 입력될 필요가 있는 자료가 있을 때 사용
menu	개발자가 MenuBar를 구성하거나 PopupMenu를 구성
radio	동그란 모양의 여러가지 선택사항 중에서 하나를 선택할 수 있도록 하는 Button의 일종
spin	일정범위 내의 숫자값을 입력하기 위해 사용
tab	여러개의 Tab Page를 추가해서 보여주는 Component
tabpage	각각의 Tab을 구성하는 TabPage

textArea	여러줄로된 문자열을 입력받기위해 사용하는 Component
tree	데이터셋의 내용을 레벨값을 이용하여 트리형태로 표현
[i_][o_][io_]cell	grid의 cell을 나타냄
기 타(etc)	
Client	Front End 사용자를 말한다.

1.2.

1.3. Java Code Convention

Java Code Conventions이 필요한 이유는 다음과 같다.

Maintenance의 편리성

소프트웨어 소스코드의 가독성(Readability) 향상

소스코드의 이식성(Portability) 및 재사용성(Reusability)에 도움을 준다.

개발자가 새로운 코드를 더욱 빠르고 완전하게 이해하는데 도움을 준다.

수정이 용이하고(Verifiability) 효율성(Efficiency)이 증대된다.

1.4. File Names

File Suffixes

File Type	Suffix
Java Source	.java
Java Bytecode	.class
Property File	.properties
JavaScript Source	.js
HTML Source	.html
Xml Source	.xml
CSS Source	.css

Common File Name

File Name	Use
Java Source file	Class/Interface 명과 동일하게 하여야 한다. Class/Interface Naming Rule은 Class Naming Rule에 기술한 내용에 따른다.
Xml Source	Class Naming Rule에 기술한 내용에 따른다.

Property file	카멜표기법을 따른다.

1.5. Class and Interface Declarations

각종 변수 선언은 public , protected , private 순으로 한다. 메소드 선언은 접근 권한에 의한 것보다 기능성 위주로 순서를 정하기로 한다. 이것은 코드를 이해하기 쉽게 하기 위해서다.

1.6. File Organization

File Organization의 내용은 아래와 같다.

Package and Import statements

Beginning comments

Class and interface declarations

1.7. Package and Import Statement

패키지를 선언하거나 import할 때의 문장이다.

package com.smarket.xxx

import javax.servlet.http.xxx;

1.8. Beginning Comment

소스 파일의 처음부분에 기술하여야 한다.

Package and Import Statement아래에 위치한다.

Indentation

Tab size를 4, Indent size를 4로 맞춘다.

1.9. Line Length

가급적 한 줄에 80자 이상의 코딩은 하지 않는다.

JAVA Script와 HTML은 예외로 80컬럼 이상을 허용한다.

많은 툴과 터미널에서 Program Coding을 보기 쉽게 하기 위함이다.

1.10. Wrapping Lines

한 줄에 80자 이상의 코딩이 넘어갈 때는 다음의 예시에 준하여 코딩한다.

메소드의 인자로 인하여 80컬럼의 범위를 벗어날 때

```
function(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);
```

메소드의 인자에 메소드가 선언되어서 80컬럼의 범위를 벗어날 때

```
var = function1(longExpression1,
                function2(longExpression2,
                          longExpression3));
```

메소드에 관련된 키워드로 인하여80컬럼의 범위를 벗어날 때

```
private static synchronized horkingLongMethodName(int anArg,
    Object anotherArg, String yetAnotherArg,
    Object andStillAnother) {
      ...
}
```

연산 문장이 80컬럼의 범위를 벗어날 때

```
longName1 = longName2 * (longName3 + longName4 - longName5) +
            4 * longname6;

if ((condition1 && condition2) || (condition3 && condition4) ||

(condition5 && condition6)) {

doSomethingAboutIt();
}
```

조건 비교 문장이 80컬럼의 범위를 벗어날 때

1.11. Comment

Block Comment

comment 를 block으로 작성함

```
/*
 * Here is a block comment with some very special
 * formatting that I want indent(1) to ignore.
 *
 *      one
 *              two
 *                  three
 */
```

Single-Line Comments

한줄의 comment는 C++과 동일하게 '//' 를 이용한다

```
if (condition) {
    // Handle the condition.
    ...
}
```

Trailing Comment

문장 뒤에 오는 comment역시 C++과 동일하게 "//"를 사용한다.

```
if (a == 2) {
    return TRUE;      // special case
} else {
    return isprime(a);      // works only for odd a
}
```

1.12. Declarations

Number Per Line

주석처리를 하기 위하여 한 라인당 한번의 선언을 한다.

```
int level;          // indentation level
```

```
int size;               // size of table
```

Class and Interface Declarations

메소드 이름과 "(" 문자사이의 공백은 없게 한다.

메소드선언후 공백한칸을 띄운후 "{" 가 오게 한다.

```
class Sample extends Object {
    int ivar1;
    int ivar2;
    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }
    int emptyMethod() {}
...
}
```

1.13. Statement

Simple Statements

한 라인에는 반드시 한 개의 statement만 허용한다.

```
String STATUS_FILE_DEV
="/home/weblogic/bea617/wlserver6.1/lib/config/paid_service_status";
```

if, if-else, if-else-if-else Statements

if 조건문

```
if (condition) {
    statements;
}
```

```
if (condition) statements;
```

Note: {}를 사용함이 원칙, if 블록이 한 문장으로 끝나는 경우에도 {}를 생략하지 않는다.

if-else 조건문

```
if (condition) {
    statements;
```

```
} else {
    statements;
}
```

for Statements

for statement의 initialization, condition, update는 한 라인에 표현을 하며 한 라인으로 기술하지 못 할 경우 라인 Skip의 Rule를 따른다.

```
for (initialization; condition; update) {
    statements;
}
```

while Statements

while statement 는 for statements와 동일한 Rule를 따른다.

```
while (condition) {
    statements;
}
```

```
while (condition);
```

단 "{}"안의 내용이 없을 경우는 "{}"를 생략 한다.

do-while Statements

```
do {
    statements;
} while (condition);
```

switch Statements

switch statement에서 case문과 statements는 반드시 라인을 바꾸어서 기술한다.

```
switch (condition) {
    case ABC:
        statements;
```

```
    case DEF:
        statements;
        break;
    default:
        statements;
        break;
}
```

try-catch Statements

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}finally {
    statements;
}
```

1.14. White Space

Blank Spaces

키워드 다음에 괄호가 나오는 경우 키워드와 괄호 사이에 반드시 공백을 둔다.

```
while (true) {
...
}
```

메소드 이름과 그 다음에 오는 괄호사이에는 공백을 두지 않는다. 이것은 메소드 Call과 키워드를 구분하기가 쉽게 한다.

인자 리스트에서 콤마(,) 뒤에는 반드시 공백을 둔다.

연산자와 변수 사이에는 반드시 공백을 둔다.

```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++) {
    n++;
```

}

for 구문 예제

for (expr1; expr2; expr3)

myMethod((byte) aNum, (Object) x);

myFunc((int) (cp + 5), ((int) (i + 3)) + 1);

Casting 예제

1.15. Miscellaneous Practices

Parentheses

```
if (a == b && c == d) // AVOID!

if ((a == b) && (c == d)) // RIGHT
```

Returning Values

Return Values를 결정함에 있어서 보다 간결한 코드를 지향한다.

예제1)

```
if ( booleanExpression ) {
        return TRUE;
} else {                                =>      return booleanExpression;
        return FALSE;
}
```

예제 2)

```
if (condition) {
    return x;                           =>      return (condition ? x: y);
}
return y;
```

1.16. JavaDoc에서 제공하는 Tags

Tag	사용법	사용가능한 block
@author	@author *name* ex) @author skcc	class, interface
@version	@version *version* ex)version 1.33, 04/22/98	class, interface
@param	@param *name* *description* ex)@param s String	메쏘드, 생성자
@return	@return *description* ex)return 참이면 true, 거짓이면 false	메쏘드
@exception	@exception *classname* *description* ex)@exception NumberFormatException 스트링이 분석되지 못하는 integer를 가진 경우	메쏘드
@see	@see *name* ex)@see java.lang.Long#valueOf.	모두사용

Tag	사용법	사용가능한 block
@since	@since *sinec-text* ex)since JDK1.0	모두사용
@deprecated	@deprecated *deprecated-text* *ex)@* deprecated As of JDK1.1, replaced by serBounds @see #setBounds(int,int,int,int)	deprecation된 API에 @see tag와 함께 사용

javadoc에서 제공하는 tags를 통해서 formated된 문서를 생성한다. 각 tag는 @로 시작 (ex>@see)

Option	Description
-public	public classes,members를 보여준다.
-protected	protected/public classes,members를 보여준다. (default)
-package	package/protected/public classes, members를 보여준다.
-private	모든 classes,members를 보여준다.
-J *flag*	flag를 runtime system에 전달한다.
-encoding *name*	*name* : source file encoding name(ex> EUCJIS₩SJIS)
-docencoding *name*	*name* : output HTML file encoding name
-version	@version tag를 포함한다.(default에서는 생략)
-author	@author tag를 포함한다. .(default에서는 생략)
-noindex	package index를 생략한다.(AllNames.html이 생성 않됨).
-d *directory*	*directory* : output file이 생성될 디렉토리 지정
-sourcepath *path*	*path* : source file의 path ex)
-classpath *path*	*path* : .class files의 path (환경변수<CLASSPATH>로 잡혀있어서 사용할 필요 없음)
-nodeprecated	@deprecated tag를 생략한다.

1.17. OPTIONS

5 메시지 표준

1.18. 메시지 분류 기준

메시지 단계(Message level)

애플리케이션 단계 (Application level)

최종 사용자가 볼 수 있는 메시지로 구성

presentation level

아키텍쳐 단계 (Architecture level)

프로그램 개발 시 발생할 수 있는 에러 메시지로 구성

common library에서 발생

Error handling 영역

DB Vendor Error 영역

Web Application Server 및 OS 영역

1.19. 분류배경 및 특징

에러 메시지를 두 단계로 분리하여 프리젠테이션 단계(Application level)의 메시지는 사용자가 처리 가능한 메시지로 구성하고 아키텍쳐 단계에서 발생하는 (에러) 메시지는 별도의 에러 핸들링을 통해 대처해야 한다.

1.20. 메시지 처리

메시지는 간단 명료하고, 어미는 ~하십시오. ~주십시오. ~없습니다. ~입니다. ~합니다. 등 경어체를 사용한다.

Bussiness Layer 및 Persistance Layer에서의 처리결과를 나타내주는 애플리케이션 단계의 메시지는 아래파일에 '메시지코드=메시지설명'의 형태로 등록하여 사용한다.

src/message/업무구분(3자리소문자)+Message.properties

메시지코드는 다음과 같이 구성한다.

구 조	업무구분(3자리대문자) + 연번(3)
규 칙	연번 : 001~999 사이의 숫자를 동일 명칭으로 하나씩 증가시키면서 부여
적용예	WES001=조회된 자료가 없습니다. WES002=자료가 저장되었습니다.

1.21. 로그 처리

로그의 처리는 프레임워크에서 제시한 log4j 를 사용하여 로그처리를 하도록 한다.

System.out.println()는 사용하지 않는다.

AOP를 사용해 적용한다.

log4j.properties 예시

```
# Root Logger
log4j.rootLogger=DEBUG, Console, File

#log4j.category.gov.paid=ERROR, Console, File
#log4j.category.gov.paid=WARN, Console, File
#log4j.category.gov.paid=INFO, Console, File
#log4j.category.gov.paid=DEBUG, Console, File

# Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss}] %-5p %c{10}.%M(%3L) - %m%n

# File
log4j.appender.File=org.apache.log4j.DailyRollingFileAppender
log4j.appender.File.File=paid.log
log4j.appender.File.layout=org.apache.log4j.PatternLayout
log4j.appender.File.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss}] %-5p %c{10}.%M(%3L) - %m%n
```

log수록 예시

```
package com.erp.test;
```

```
import java.util.TimeZone;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class TimeZoneTest{
        private static final Log LOG= LogFactory.getLog(TimeZoneTest.class);
        public static void main(String args[]){
        String[] str = TimeZone.getAvailableIDs();

        for(int i=0 ; i<str.length ; i++){
            LOG.debug("debug test");
            TimeZone tz = TimeZone.getTimeZone(str[i]);
            LOG.warn("warning test");
            Calendar cal = Calendar.getInstance();
            LOG.error("error test");
            Date date = cal.getTime();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd - HH:mm:ss");
            sdf.setTimeZone(tz);
            LOG.info(tz.getDisplayName() + "₩t" + str[i] + " time :" +
sdf.format(date));
        }
        }
}
```

6 DBMS 표준

1.22. Naming Convention

1.1.6. Table

1.1.7. XXX : 의미있는 명칭 약어

1.1.8. Column

추가되는 필드에 한해서도 범용 스펙을 준수한다.

Column명 앞에 중분류를 위한 prefix를 붙이지 않는다.

Column명 앞에 테이블명을 prefix를 붙이지 않는다.

XXX_XXX : 의미있는 명칭 약어

1.1.9. DataType

Column 종류	데이터타입	길이	예	비고
날짜	VARCHAR2	10	VARCHAR2(10)	"YYYY-MM-DD" 사용
개수, 수량, 금액	NUMBER			
설명	VARCHAR2	100	VARCHAR2(100)	
코드타입	VARCHAR2	코드길이		
그 외	VARCHAR2	10, 20, 30 (10단위를 원칙으로 함)	VARCHAR2(10) VARCHAR2(20)	

1.23. PL/SQL

1.24. Comment

●Beginning Comment

개발자는 다음의 양식처럼 스크립트별 정보를 주석으로 작성하여야한다.

```
/*******************************************************************************
   개체이름 : 개체명

   작 성 일 : YYYY-MM-DD
   작 성 자 : 작성자명

   개     요 : 함수 및 프로시저 목적 및 개요

   내     용 : 1. 함수 및 프로시저 내용
       2. 함수 및 프로시저 내용
            3. 함수 및 프로시저 내용
*******************************************************************************/
```

함수 및 프로시저 CREATE OR REPLACE Statement 아랫줄에 위 주석을 추가한다.

ex)

```
CREATE OR REPLACE PROCEDURE PC_SD_ACCEPT_INORDER_PATNO(
/*****************************************************************************
    개체이름 : PC_SD_ACCEPT_INORDER_PATNO

    작 성 일 : 2009-06-03
    작 성 자 : 홍길동

    개     요 : 환자의 특정 처방만 약처방을 접수한다.

    내     용 : 1. 환자정보를 변경 및 추가.
                2. 마스터정보를 변경 및 추가.
               3. 환자의 특정 처방만 약처방 추가.
*****************************************************************************/
   i_dosdate      in   varchar2,       --투약일자
   i_keyin        in   varchar2,       --환자번호
   i_procstat     in   varchar2,       --처방구분 (N : 정규처방, A: 추가처방,   E: 응급처방)
   i_editid       in   varchar2,       --작업자

   o_logmsg       out varchar2,       --결과코드
   o_successyn    out varchar2        --결과메세지
)
is ...
```

Single Line Comment : 두개의 대시(-)를 주석 앞에 사용한다.

Multi Line Comment : /*과 */ 사이에 주석을 기록한다.

1.25. 오라클 Function & Procedure Naming Role

1.26. FN_업무구분2자_함수명

1.27. PC_업무구분2자_함수명

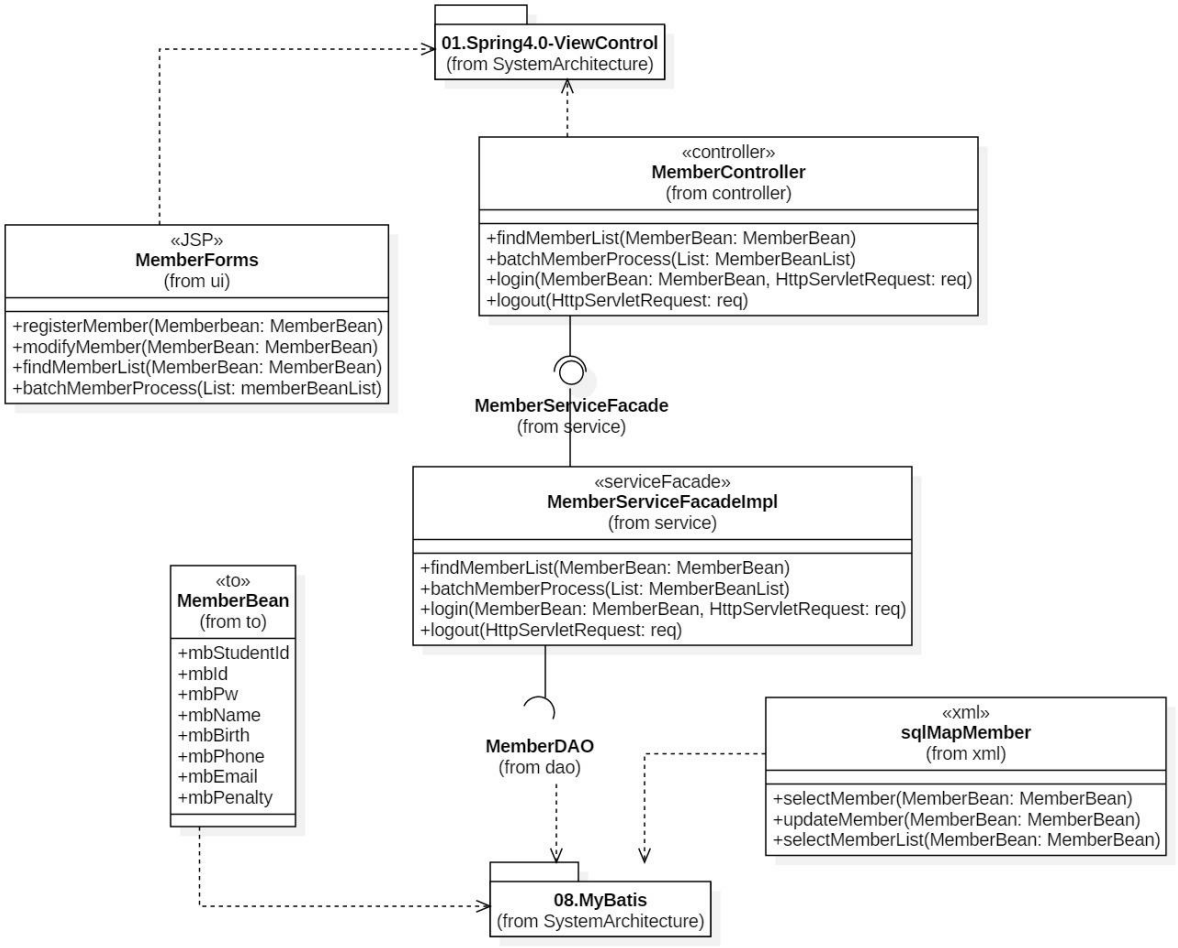
SUNG KYUN KWAN UNIVERSITY

5.Human Resource Management System

5.1 Objectives

본 SARA 시스템을 구성하는 서브시스템 중 하나인 인적 사항 관리 시스템을 통하여 회원의 정보를 관리하고 학내 구성원 인증 프로세스를 설명한다. Class diagram, Sequence diagram, State diagram을 통하여 컴포넌트 내부를 표현하고 이에 대해 설명한다.

5.2 Class Diagram



UI

MemberForms: 회원의 인적 정보를 관리하는 User interface

Function RegisterMember(MemberBean): 화면단에서 새로운 회원을 등록하는 메서드

Function ModifyMember(MemberBean): 화면단에서 회원의 정보를 수정하기 위한 메서드

Function FindMember(MemberBean): 화면단에서 해당 회원의 정보를 찾기 위한 메서드

Function BatchMemberProcess(MemberBeanList): 화면단에서 회원의 정보를 일괄처리하기 위한 메서드

Controller

MemberController: 회원 관리의 중앙집중화를 담당하는 부분

ModelAndView FindMemberList(MemberBean): 해당 회원정보에 해당하는 회원 리스트를 찾는 메서드

ModelAndView BatchMemberProcess(MemberBeanList): 회원리스트 정보를 일괄처리하기 위한 메서드

ModelAndView Login(MemberBean,HttpServletRequest): 회원이 로그인 하기 위한 메서드

ModelAndView Logout(HttpServletRequest): 로그아웃을 위한 메서드

Service

MemberServiceFacadeImpl: 회원관리의 비즈니스 계층에 있는 처리 로직을 담당한다.

List FindMemberList(MemberBean): 해당 회원정보에 해당하는 회원 리스트를 찾는 메서드

Void BatchMemberProcess(MemberBeanList): 회원리스트 정보를 일괄처리하기 위한 메서드

Void Login(MemberBean,HttpServletRequest): 회원이 로그인 하기 위한 메서드

Void Logout(HttpServletRequest): 로그아웃을 위한 메서드

DAO

MemberDAO: 실제 회원정보 엔티티를 담당하는 부분

MemberBean SelectMember(MemberBean): 해당 회원정보에 해당하는 회원을 선택한다.

Void UpdateMember(MemberBean): 회원의 정보를 수정하는 메서드

List SelectMemberList(MemberBean): 해당 회원정보에 해당하는 회원 리스트를 불러오는 메서드

XML

SqlMapMember: 회원관리부분의 DB를 핸들링 하는 XML 부분

SelectMember(MemberBean): 해당 회원 정보에 해당하는 회원을 찾는다.

UpdateMember(MemberBean): 해당 회원 정보를 수정한다.

SelectMemberList(MemberBean): 해당 회원 정보에 해당하는 회원 리스트를 반환한다.

To

MemberBean: 회원 정보를 전달하기 위한 객체

MbStudentId : 학번

MbId: 회원아이디

MbPw: 회원비밀번호

MbName: 회원이름

MbBirth: 회원생년월일

MbPhone: 회원 전화번호

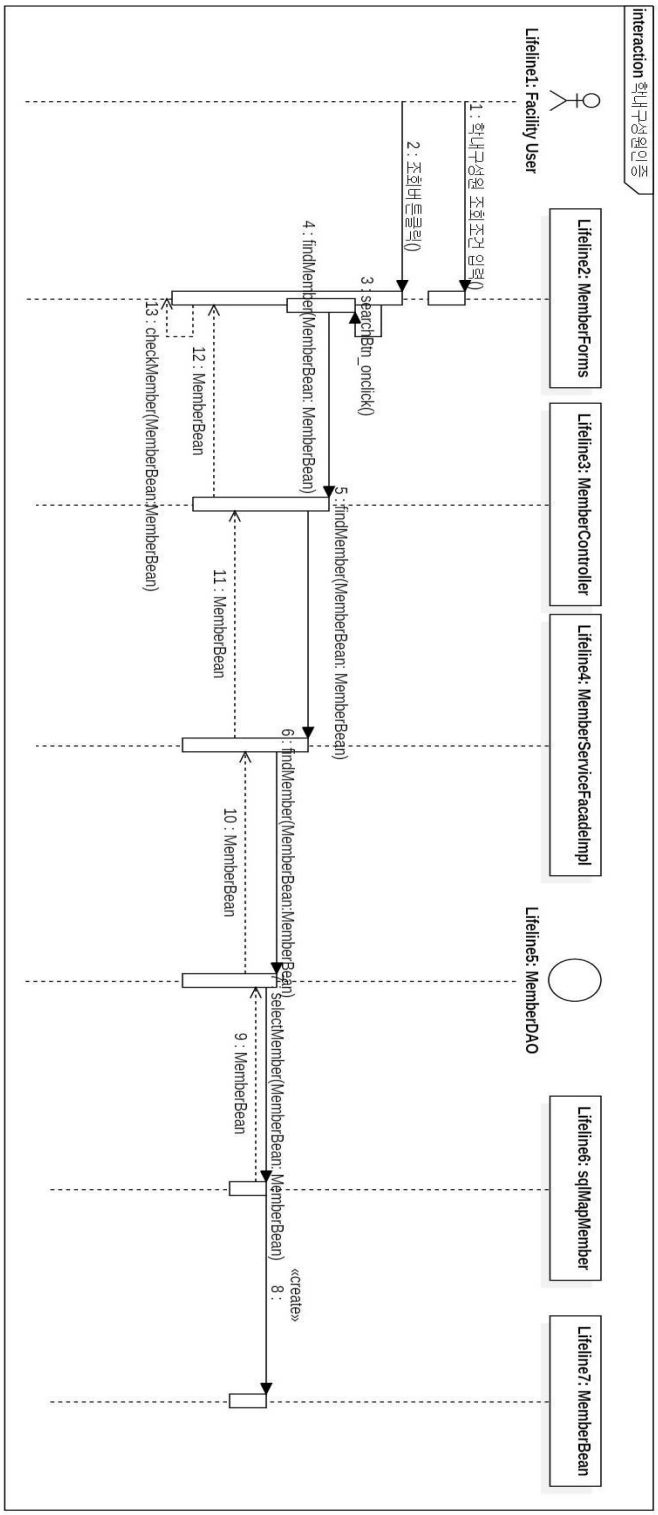
MbEmail: 회원 이메일

MbPenalty: 회원 페널티점수

5.3 Sequence Diagram

5.3.1 학내 구성원 인증

SUNG KYUN KWAN UNIVERSITY

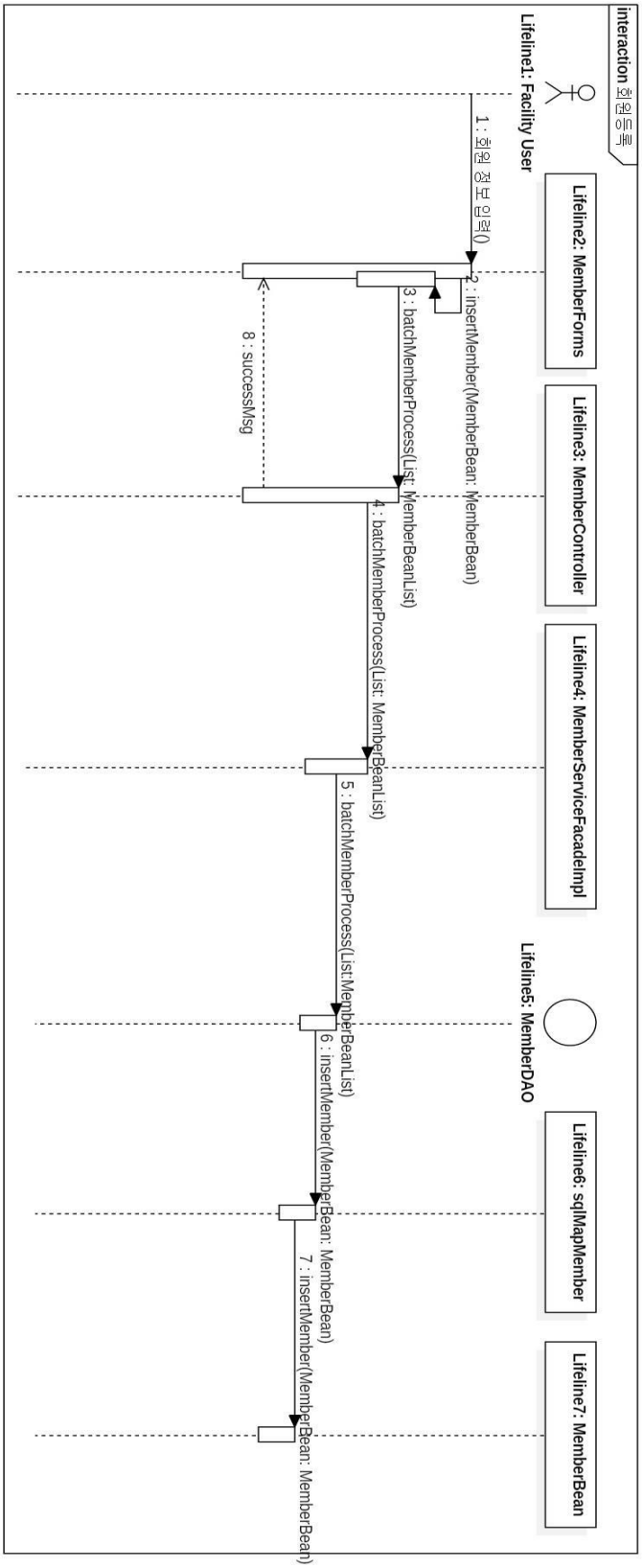


Messages	Description
(1) 학내구성원 조회조건 입력()	로그인 후 [학번] 입력

(2) 조회버튼 클릭()	학내 구성원 조회 버튼 클릭
(3) SearchBtn_onclick()	SearchBtn_onclick 메서드 실행
(4) findMember(MemberBean member)	회원 정보 조회
(5) findMember(MemberBean member)	회원 정보 조회
(6) findMember(MemberBean member)	회원 정보 조회
(7) selectMember(MemberBean member)	회원 정보 조회
(8) <<create>>	해당 MemberBean 생성
(9) MemberBean	MemberBean 객체 전달
(10) MemberBean	MemberBean 객체 전달
(11) MemberBean	MemberBean 객체 전달
(12) MemberBean	MemberBean 객체 전달
(13) CheckMember(MemberBean member)	반환된 Member가 학내 구성원인지 여부 확인 후 True/False 반환

5.3.2 회원 등록

SUNG KYUN KWAN UNIVERSITY

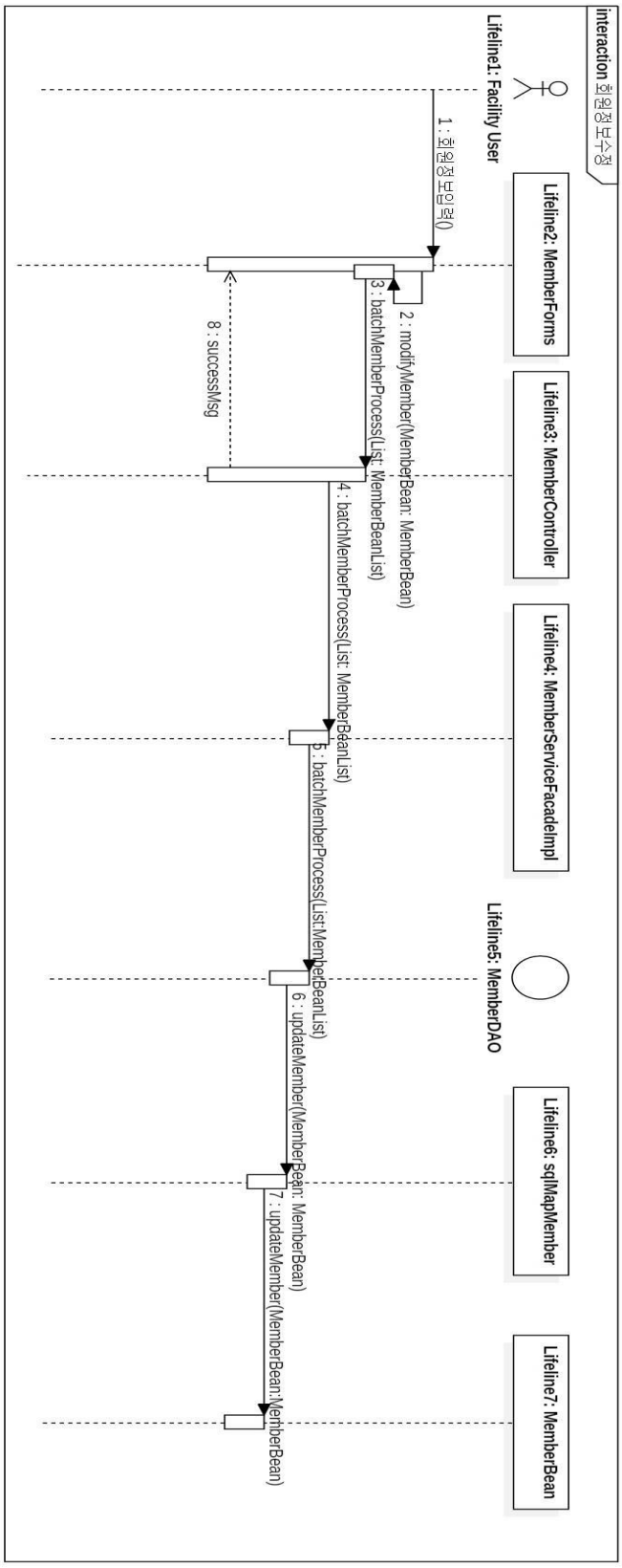


Messages	Description

(1) 회원 정보 입력	가입할 회원의 정보를 입력한다. [이름,아이디,비밀번호,생년월일,휴대폰,이메일,소속학부대학,동아리]
(2) insertMember(MemberBean:MemberBean)	회원 정보 등록
(3) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(4) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(5) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(6) insertMember(MemberBean: MemberBean)	회원 정보 등록
(7) insertMember(MemberBean: MemberBean)	회원 정보 등록
(8) successMsg	일괄 처리 성공/실패 메시지

5.3.3 회원 정보 수정

SUNG KYUN KWAN UNIVERSITY



Messages	Description

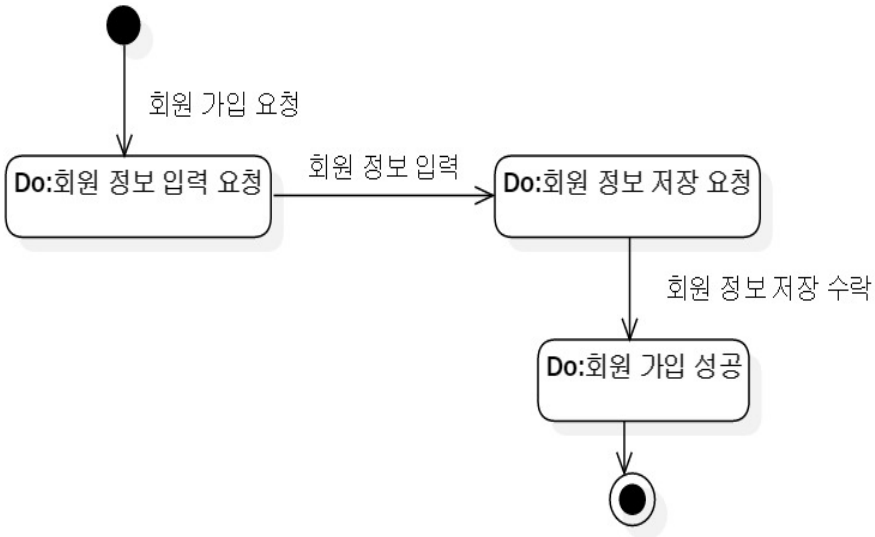
(1) 회원 정보 입력	수정할 회원 정보를 입력한다.
(2) modifyMember(MemberBean: MemberBean)	회원 정보를 수정한다.
(3) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(4) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(5) batchMemberProcess(List: MemberBeanList)	회원 정보를 담은 리스트의 회원 정보를 일괄 처리한다.
(6) updateMember(MemberBean: MemberBean)	해당 회원의 정보를 수정한다.
(7) updateMember(MemberBean: MemberBean)	해당 회원의 정보를 수정한다.
(8) successMsg	일괄 처리 성공/실패 메시지

5.4 State Diagram

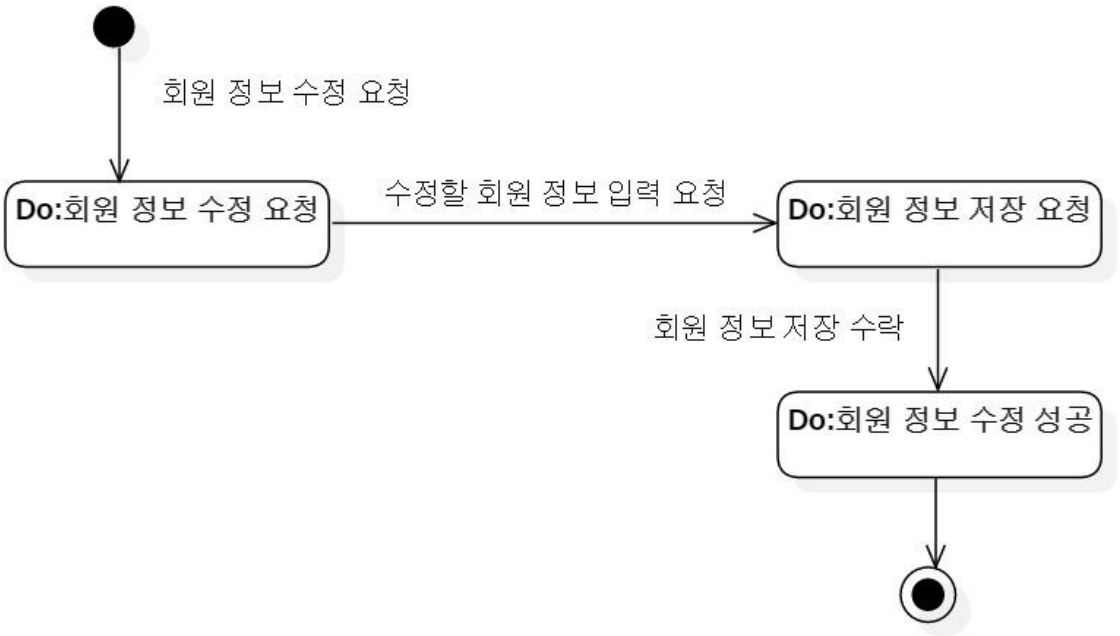
SUNG KYUN KWAN UNIVERSITY

학내 구성원 인증

회원 가입



회원 정보 수정



6. Reservation Management System

SUNG KYUN KWAN UNIVERSITY

6.1 Objectives

6.2 Class Diagram

6.3 Sequence Diagram

6.4 State Diagram

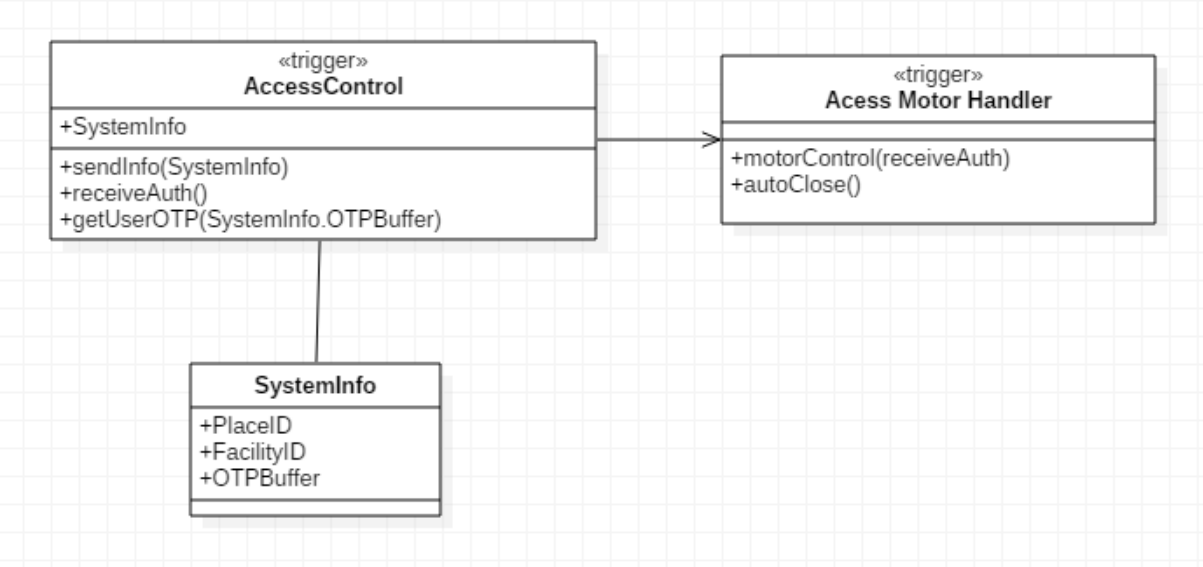
7. Access Control System

7.1 Objectives

이 섹션에서는 사용자의 OTP 값을 입력 받은 후 그 값이 실제 사용자가 예약하고 제공받은 유효한 값인지를 예약 관리 시스템을 통해 파악하여 그 여부에 따라 출입문을 개폐하는 출입 관리 서브시스템을 Class diagram, Sequence diagram, State diagram을 통해 설명한다.

출입 관리 시스템의 사용자 입력을 받는 메소드는 상시 작동 중이며 사용자가 미리 부착된 외부 입력 장치로 비밀번호 값을 입력했을 때 다른 메소드들을 호출하기 시작한다. 사용자의 비밀번호 입력 값을 예약 관리 시스템으로 전송하고, 유효한 OTP인지에 대한 결과 값을 받는다. 인증 성공 값을 받을 경우 출입문을 열고 인증 실패 값을 받았을 경우 출입문을 닫거나 닫은 상태를 유지한다.

7.2 Class Diagram



A. SystemInfo

A.1. Attributes

PlaceID : 해당 출입 관리 시스템이 가지고 있는 장소 번호이다.

FacilityID : 해당 출입 관리 시스템이 가지고 있는 시설(건물) 번호이다.

OTPBuffer : 사용자로부터 OTP를 입력받는 버퍼이다.

A.2. Operations

없음

B. AccessControl

B.1. Attributes

SystemInfo : 해당 시스템이 가지고 있는 고유 정보들을 담고있는 클래스 인스턴스이다.

B.2. Operations

sendInfo(SystemInfo) : 예약 관리 시스템에게 본 시스템의 시설 정보와 OTP의 정보를 전송한다.

receiveAuth() : 예약 관리 시스템으로부터 전송한 시설 정보와 OTP가 유효한지에 대한 인증 값을 받는다.

getUserOTP(SystemInfo.OTPBuffer) : 사용자로부터 OTP 값을 입력받아 OTPBuffer에 저장한다.

C. AccessMotorHandler

C.1. Attributes

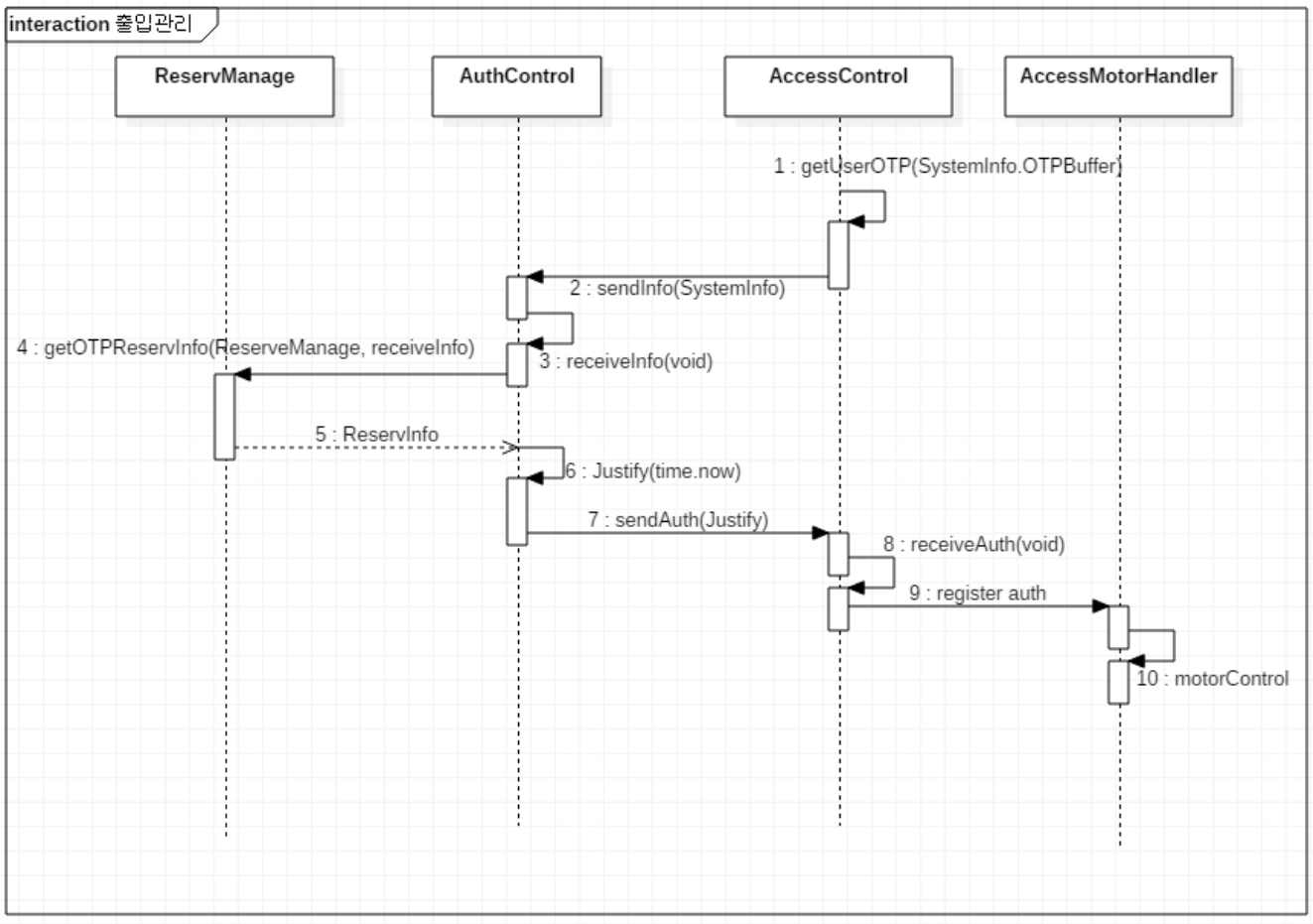
없음

C.2. Operations

motorControl(receiveAuth) : 받은 인증 값이 유효하면 모터를 조절해 출입문을 개방하고 아니면 닫는 상태를 유지시킨다.

autoClose() : 가장 최근 motorControl 함수가 실행된 이후 수 초가 지나면 출입문을 자동으로 닫는다.

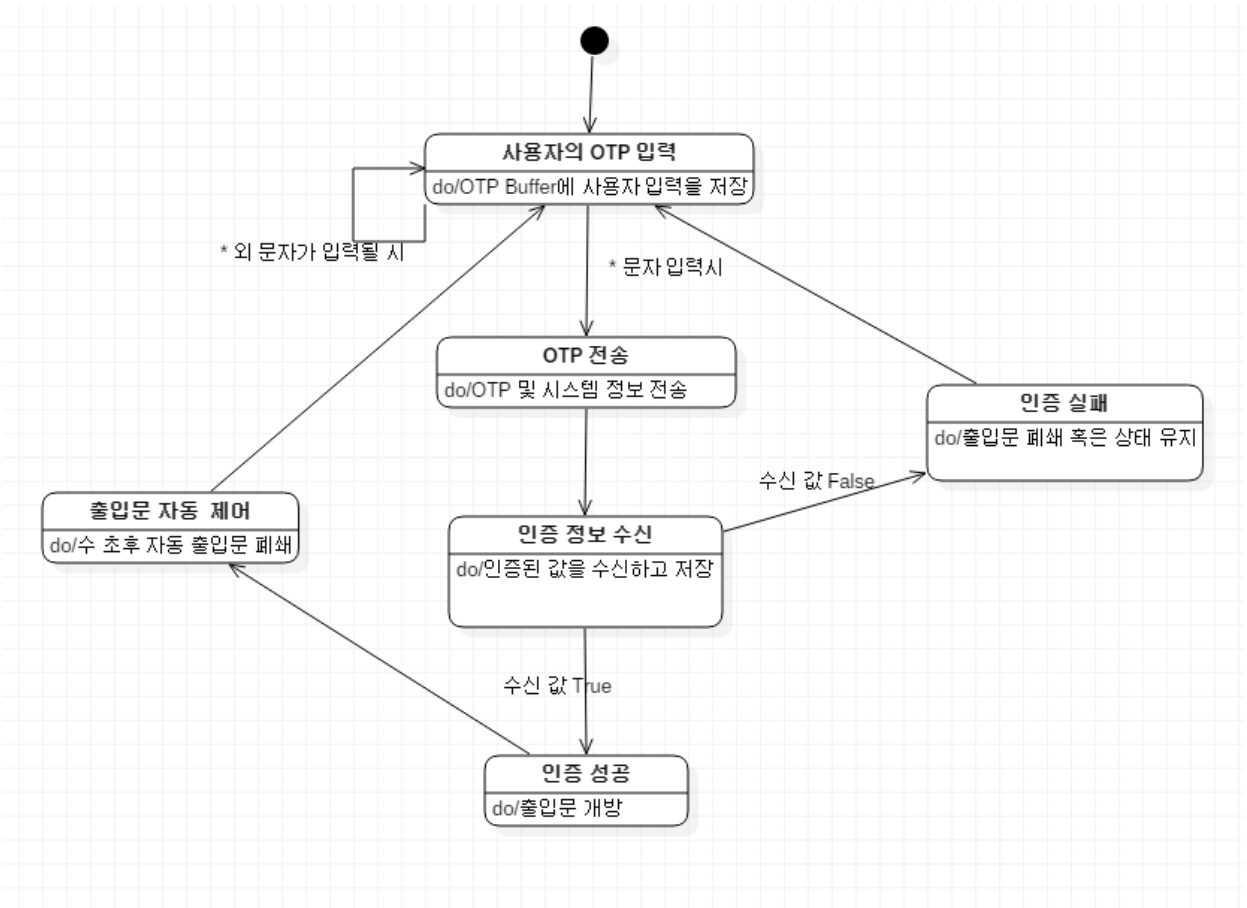
7.3 Sequence Diagram



Messages	Description
(9) getUserOTP	OTPBuffer에 사용자의 입력 값을 저장
(10) sendInfo	OTP와 시스템 정보를 AuthControl에 전달
(11) receiveInfo	AuthControl의 정보 수신
(12) getOTPReservInfo	해당 OTP가 등록된 예약 정보 요청
(13) ReservInfo	예약 정보 전달
(14) Justify	현재 시간과 비교하여 OTP 유효성 인증
(15) sendAuth	OTP의 인증 결과 전달
(16) receiveAuth	인증 결과 수신
(17) register auth	AccessMotorHandler에 인증 결과 등록

(18) motorControl	인증 결과를 기반으로 출입문 제어

7.4 State Diagram



SUNG KYUN KWAN UNIVERSITY

8.Protocol Design

SUNG KYUN KWAN UNIVERSITY

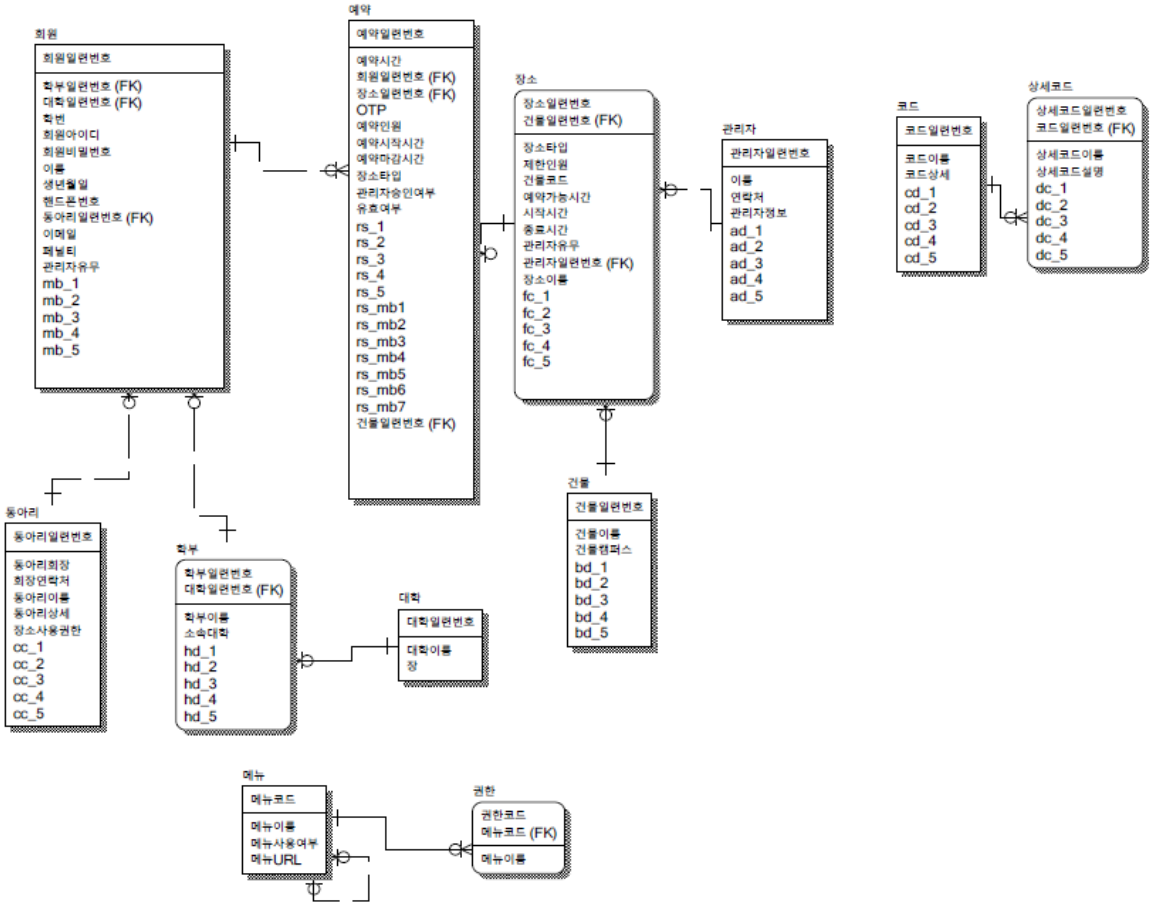
9 . Database Design

9.1 Objectives

Database Design에서는 서버의 데이터 모델을 설명한다. 요구 사항 명세서에 서술된 요구사항을 바탕으로 Data Architect를 설계하고 , Data Object 간의 관계를 설명한다. 정보를 주제별 테이블로 구분하여 중복된 데이터를 줄이고 정보의 정확도 및 무결성을 지원하고 보장하는 데 도움이 되는 디자인을 지향한다.

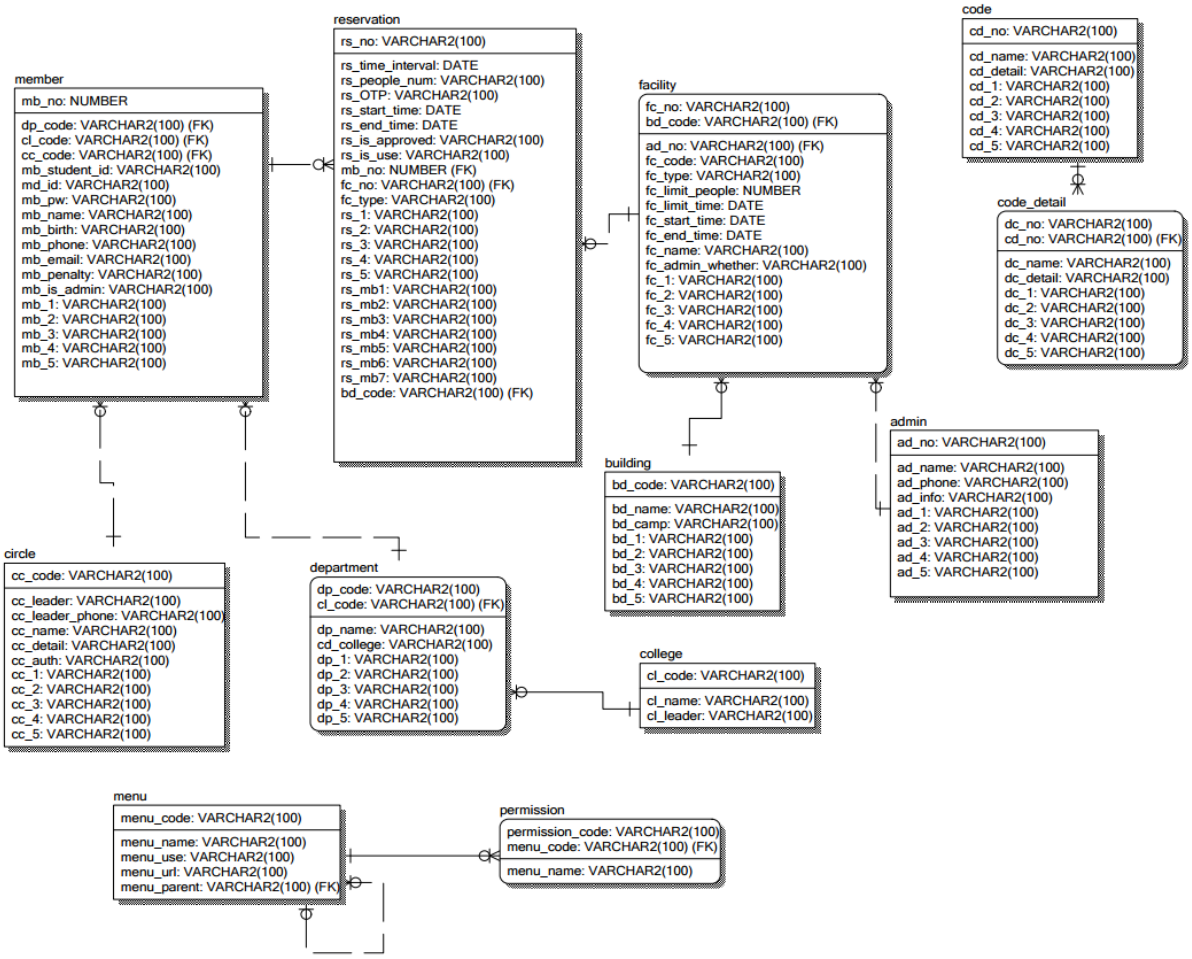
9.2 ER Diagram

9.3 Logical Model



SUNG KYUN KWAN UNIVERSITY

Physical Model



Entity

Relationship

SUNG KYUN KWAN UNIVERSITY

Relation Schema

SQL DDL

10.Testing Plan

SUNG KYUN KWAN UNIVERSITY

11.Development Environment

SARA 시스템 개발 환경과 version 관리 도구를 어떻게 사용할 것인가에 대해 서술한다.

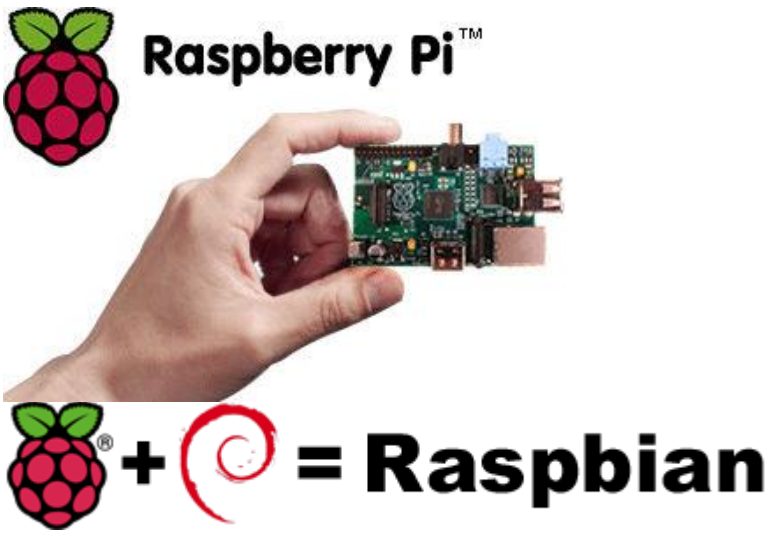
11.1. operating systems

windows 10

개발 환경의 운영체제로는 빠른 개발과 친숙한 환경을 위해 팀원 모두 사용해오던 windows 10을 사용하게 되었다. window만의 장점은 바로 '호환성'이다. 예전 응용 프로그램(legacy app)을 최신 window로도 실행할 수 있다. OS X, Linux 등은 흉내내지 못하는 강점이다.

11.2. Running Environment

Raspbian & Raspberry Pi , Ubuntu 14.04 x64



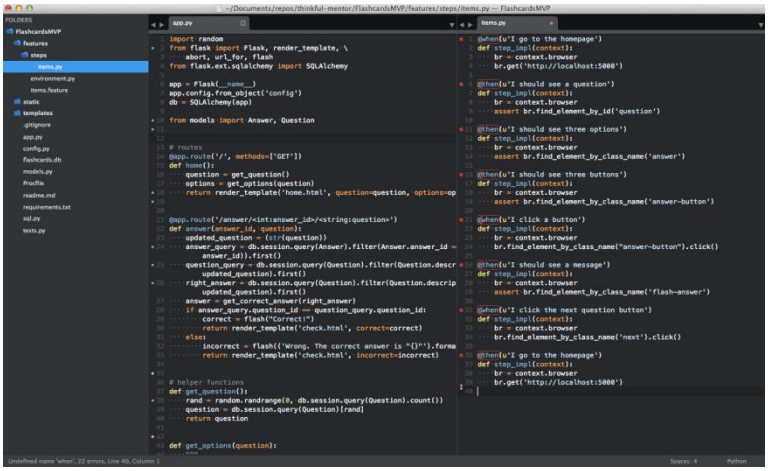
라즈베리 파이 재단은 Debian과 아치 리눅스 ARM 배포판의 다운로드를 제공하고, 주요 프로그래밍 언어로 python의 사용을 권장하기 때문에 적합하다. 라즈베리 파이는 리눅스 커널 기반 운영 체제를 사용한다. Raspbian이라는 라즈베리 파이에 최적화된 데비안 계열의 자유 운영 체제가 현재로서는 가장 권장되는 시스템이다. 리눅스 운영체제 중 가장 많이 통용되어 사용하는 debian 계열의 Ubuntu 14.04 x64를 사용하기로 하였다.

11.3. IDE

11.3.1. Eclipse

이클립스는 이클립스 재단에서 개발하고 배포하는 JAVA 개발 통합 환경 제공 프로그램이다. 이클립스는 다양한 플랫폼에서 쓸 수 있으며, 자바를 비롯한 다양한 언어를 지원하는 프로그래밍 통합 개발 환경을 목적으로 시작하였으나, 현재는 범용 응용 소프트웨어 플랫폼으로 진화하였다.

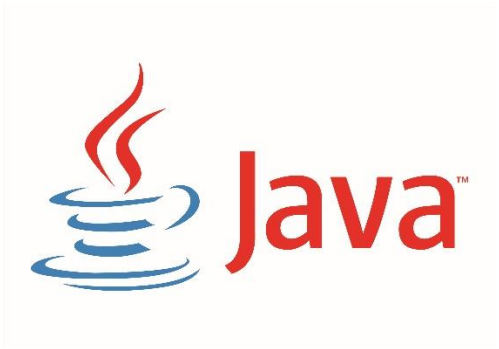
11.3.2. Sublime Text 3.0



sublime text의 경우 가장 가벼우면서도 강력한 개발 tool 중의 하나로, 다양한 언어에 맞게 syntax 조정 및 highlighting이 가능하다는 것이 특징이다. 특히 웹 개발을 할 경우 관련된 Addon과 함께 강한 performance를 낼 수 있다. editor 프로그램으로 유용한 플러그인을 제공한다. sublime text를 활용하여 제한된 시간에 완성도 높은 웹 페이지를 제작하는데 도움을 줄 것으로 보인다. 또한, 운영체제에 독립적인 장점을 가지고 있다.

11.4 Programming language

java



JAVA는 객체 지향적 프로그래밍 언어이며, 썬 마이크로시스템즈에서 무료로 제공하고 있다. 현재 버전 9까지 출시했다. 자바의 개발자들은 Unix 기반의 배경을 가지고 있었기 때문에 문법적인 특성은 Pascal이 아닌 C++의 조상인 C 언어와 비슷하다. 자바를 다른 컴파일언어와 구분짓는 가장 큰 특징은 컴파일된 코드가 플랫폼 독립적이라는 점이다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환한다. 바이트코드를 실행하기 위해서는 JVM(자바 가상 머신, Java Virtual Machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자바 바이트코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU나 운영 체제의 종류에 관계없이 JVM을

설치할 수 있는 시스템에서는 어디서나 실행할 수 있으며, 이 점이 웹 애플리케이션의 특성과 맞아떨어져 널리 사용되고 있다.

python 3.5.1

Python은 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬은 초보자부터 전문가까지 사용자층을 보유하고 있다. 동적 타이핑(dynamic typing) 범용 프로그래밍 언어로, 펄 및 루비와 자주 비교된다. 다양한 플랫폼에서 쓸 수 있고, 라이브러리(모듈)가 풍부하여, 대학을 비롯한 여러 교육 기관, 연구 기관 및 산업계에서 이용이 증가하고 있다. 또 파이썬은 순수한 프로그램 언어로서의 기능 외에도 다른 언어로 쓰인 모듈들을 연결하는 glue 언어로써 자주 이용된다. 실제 파이썬은 많은 상용 응용 프로그램에서 스크립트 언어로 채용되고 있다. 도움말 문서도 정리가 잘 되어 있으며, 유니코드 문자열을 지원해서 다양한 언어의 문자 처리에도 능하다. 파이썬은 기본적으로 해석기(인터프리터) 위에서 실행될 것을 염두에 두고 설계되었다.

파이썬의 주요한 특징으로는 다음 3가지가 있다. 동적 타이핑(dynamic typing)을 하여 실행 시간에 자료형을 검사한다. 객체의 멤버에 무제한으로 접근할 수 있어 속성이나 전용의 method 훅을 만들어 제한할 수는 있다. 모듈, 클래스, 객체와 같은 언어의 요소가 내부에서 접근할 수 있고, reflection을 이용한 기술을 쓸 수 있다.

11.5. Version Control

Git

SUNG KYUN KWAN UNIVERSITY

Git은 프로그램 등의 소스 코드 관리를 위한 분산 버전 관리 시스템이다. 빠른 수행 속도에 중점을 두고 있는 것이 특징이다. Git의 작업 폴더는 모두, 전체 기록과 각 기록을 추적할 수 있는 정보를 포함하고 있으며, 완전한 형태의 저장소이다. 네트워크에 접근하거나 중앙 서버에 의존하지 않는다.

GitHub는 분산 버전 관리 툴인 Git을 사용하는 프로젝트를 지원하는 웹호스팅 서비스이다. GitHub는 영리적인 서비스와 오픈소스를 위한 무상 서비스를 모두 제공한다. Git이 텍스트 명령어 입력 방식인데 반해, Github는 화려한 그래픽 유저 인터페이스(GUI)를 제공한다.

12.Index

13.Appendices