포토폴리오

Junior Software Developer

하현숙

이메일 qu3230@gmail.com
연락처 010.4197.4757
깃허브 https://github.com/hyeonsook95
블로그 https://mcheleplant.blogspot.com
새로운 블로그 https://hyeonsook95.github.io

Copyright ⓒ Slug. All right reserved.

※ 목차의 제목과 소제목을 클릭하면 해당 페이지로 이동합니다.

목차

CONTENTS

Copyright ⓒ Slug. All right reserved.

00

자기소개

자기소개

안녕하세요. 신입 개발자 하현숙입니다. :D

신뢰는 쌓이기는 어렵지만, 무너지기는 쉽다고 했습니다.
저는 꾸준한 열정과 노력으로 일 뿐만 아니라
사람사이에서도 신뢰를 쌓는 개발자가 되고 싶습니다!

보유기술




Copyright ⓒ Slug. All right reserved.

01

프로젝트 개요

ASVN

2016.10 – 2016.12

프로젝트 개요

FTP를 통한 **소프트웨어 버전 관리 시스템**입니다. Log와 분산 저장을 통해 버전 관리를 위한 기능을 제공합니다.

담당업무

1. 배포를 위한 Makefile 구현.
2. 관리 디렉토리 분석 설계, 구현.
3. 명령어 처리 알고리즘 구조 설계, 구현.
4. create, delete, mkdir, log 사용자 명령어 설계 및 구현.
5. 로그 형식 설계, 구현.
6. 명령어 별 로그 기록 함수 설계, 구현.

개발환경

개발환경 Ubuntu 16.04, GCC
개발언어 C
프로토콜 FTP

참조 (※ 주소를 클릭하시면 이동합니다!)

자세한 코드는 다음의 GitHub 주소를 참조해주세요.

https://github.com/hyeonsook95/asvn

Copyright ⓒ Slug. All right reserved.

01

분석 및 설계

분석 및 설계

SVN, Git 분석

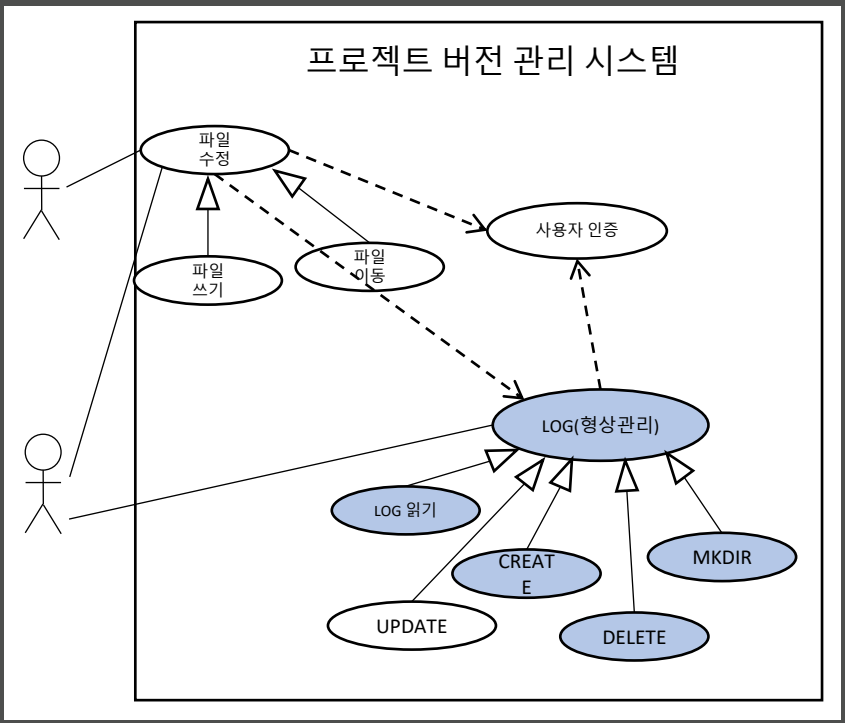
관리 소프트웨어의 구조와 기능에 대해 파악하기 위해 공식 문서와 참조 자료를 공부하였습니다.

- SVN api site
- SVN protocol
- Wiki SubversionBook
- https://git-scm.com/book/ko/v2

전체 기능 설계

저희는 관리 소프트웨어의 핵심기능은 log와 분산 저장이라고 생각하고 핵심기능을 나누어 구현하기로 했습니다.

<구현할 기능 파악을 위한 유스케이스>



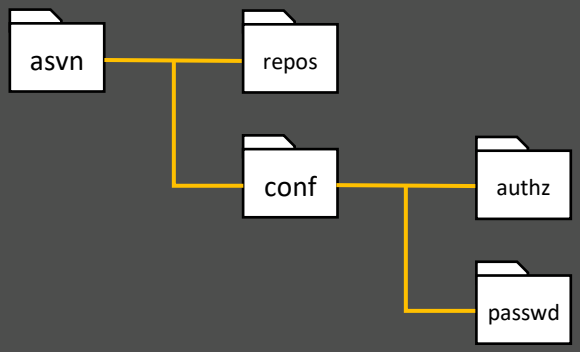
Copyright ⓒ Slug. All right reserved.

관리 디렉토리 설계

사용자는 한 OS에서 여러 저장소를 생성할 수 있으므로 전체적인 저장소를 관리할 디렉토리가 필요하다고 생각하여 글로벌하게 저장소를 관리하고 사용자의 정보를 저장하는 디렉토리를 설계했습니다.

- asvn : 형상관리를 설치할 때, 홈 디렉토리에 생성되는 관리 디렉토리. 저장소에 설치되는 모든 저장소들을 총괄 관리하기 위한 디렉토리

<관리파일 구조1>



- repos : 사용자가 생성한 저장소의 경로들을 기록한 디렉토리
- conf : 사용자들의 정보를 기록하는 디렉토리.

또한, 저장소 별로 **사용한 명령어의 종류, 사용한 날짜, 사용자의 이름**과 같은 정보를 관리하는 디렉토리가 필요하다고 생각하였고, .asvn 디렉토리를 생성하여 각 저장소별로 log를 기록하도록 하였습니다.

- .asvn : 로컬 저장소를 설치할 때, 해당 폴더 내에 생성되는 디렉토리. 로컬 저장소를 관리하기 위한 log를 저장

- log : 실행한 명령어, message, 실행 날짜, 실행한 사용자를 기록한 파일

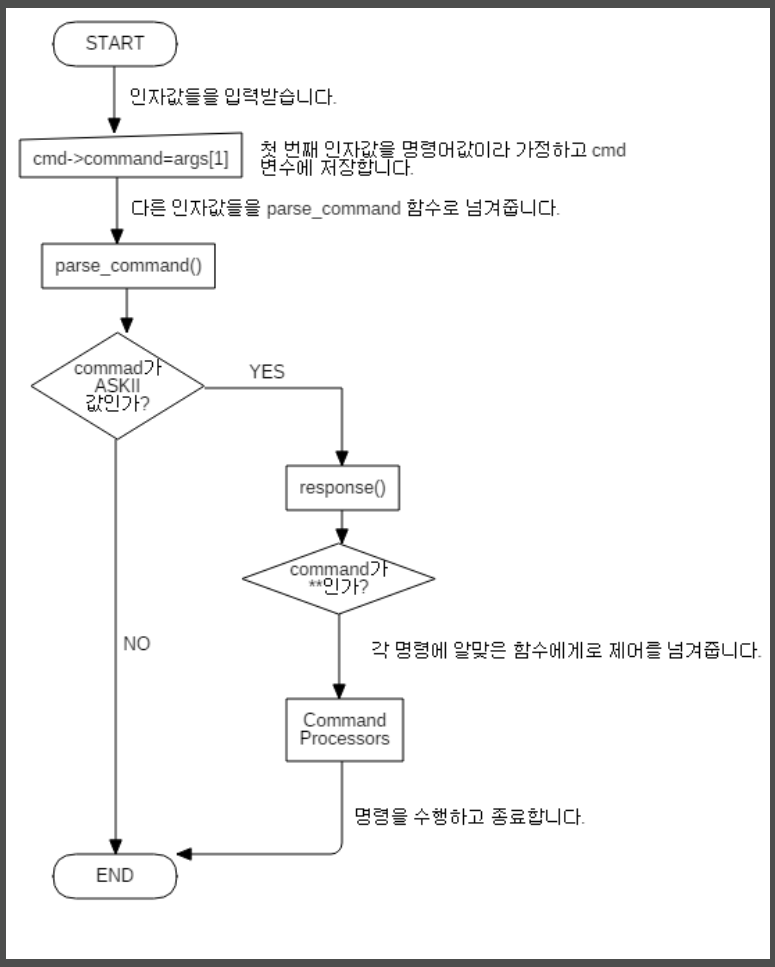
Copyright © Slug. All right reserved.

명령어 처리 구조 설계

사용자의 명령어를 받아 처리할 때, 각기 다른 길이를 가진 명령어를 어떻게 처리해야 할지 고민했습니다.

큰 길이의 char 배열 값으로 받아 처리를 하는 알고리즘을 짜보았지만, 메모리상 비효율적이라고 생각하여 다른 방법을 찾기로 했습니다.

그리고 cmd로 입력을 받는 다른 서버에서는 어떻게 명령어를 처리하는지 검색하면 그 방법을 사용할 수 있을 것이라 생각하고 간단히 구현된 서버를 github에서 검색하여 오픈소스의 코드를 참조하였습니다.



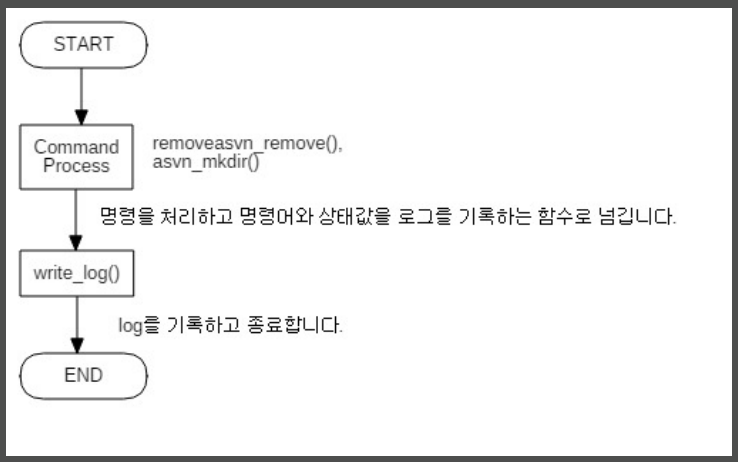
< 오픈소스의 명령어 처리 흐름도 >

Copyright © Slug. All right reserved.

서버와 관련 없는 사용자 함수 설계

버전 관리를 위한 사용자 함수는 사용자가 실행한 명령어가 사용한 사용자의 이름, 날짜, 명령어가 log에 저장되어 관리되는 것이 중요 로직이라 생각하여 **명령어 실행부분과 그 명령어를 log에 기록하는 부분으로 나누어 설계**했습니다.

<사용자 함수 흐름도>

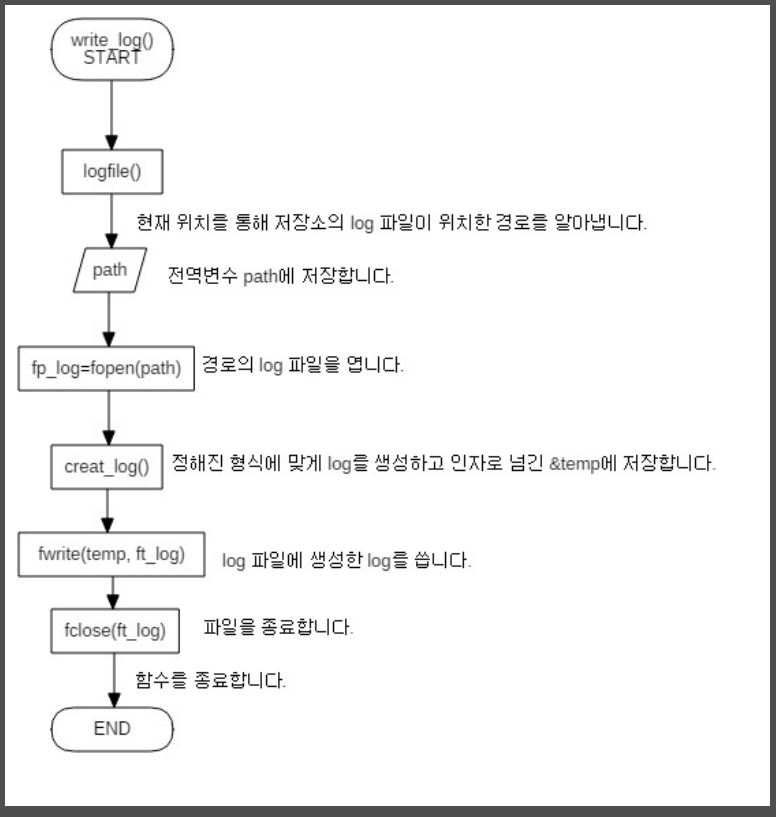


Copyright ⓒ Slug. All right reserved.

로그를 위한 함수 설계

로그를 위한 함수는 각 부분 함수의 재사용을 위해 3부분으로 나누어 설계했습니다.

- write_log() : 각 부분함수를 활용하여 실행된 명령어의 log를 기록하는 함수
- logfile() : 현재 저장소의 log 파일의 경로를 전역변수 path에 저장하는 함수
- creat_log() : 형식에 맞게 로그를 생성하는 함수



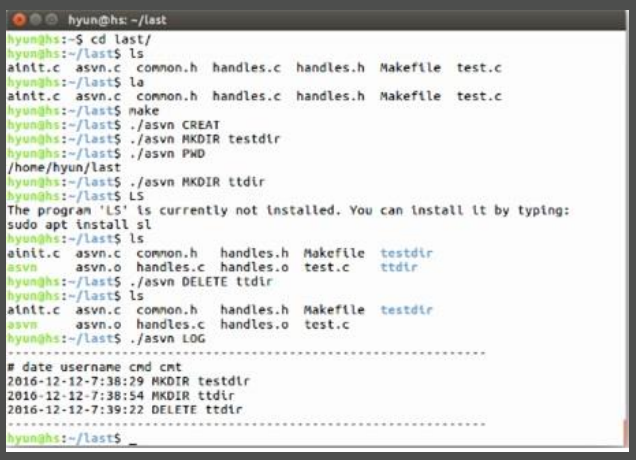
<write_log() 함수의 흐름도>

Copyright © Slug. All right reserved.

01

실행화면

실행화면



Copyright © Slug. All right reserved.

02

프로젝트 개요

Webmail System

2018.03 - 2018.06

프로젝트 개요

기존에 구현된 웹 메일 시스템을 유지보수화 과정을 통해
소프트웨어의 생명을 연장시키는 목표를 가진 프로젝트입니다.

담당업무

1. 사용자 관리 기능 추가 (완전화 유지보수) 설계 및 구현
2. 사용자 기능 '주소록' 추가 (완전화 유지보수) 설계 및 구현
3. 비밀번호 보안 강화 (예방 유지보수) 설계 및 구현

개발환경

개발환경 Ubuntu 16.05
개발언어 Java, JSP
개발기술 MySQL, Apache James Server

참조 (※ 주소를 클릭하시면 이동합니다!)

자세한 코드는 다음의 GitHub 주소를 참조해주세요.

https://github.com/hyeonsook95/webmail-system

Copyright ⓒ Slug. All right reserved.

02

분석 및 설계

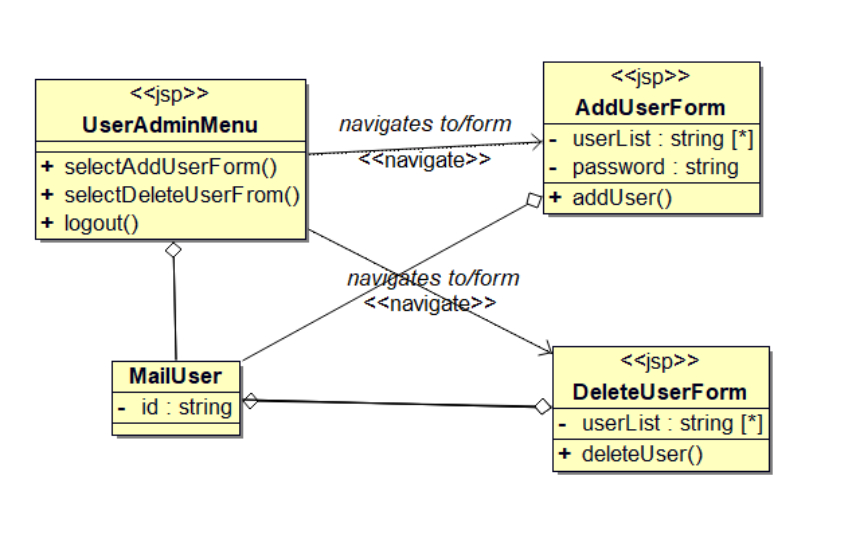
분석 및 설계

사용자 관리(완전화 유지보수) 설계

기존의 프로그램은 관리자 계정이 제공되었지만 관리자가 시스템을 위해 사용자를 관리하는 기능이 없었습니다.

관리자가 시스템을 쉽게 관리하여 시스템을 개선할 수 있도록 관리자가 사용자를 강제적으로 추가, 삭제할 수 있는 기능을 추가했습니다.

<사용자 관리 클래스 다이어그램>



Copyright © Slug. All right reserved.

사용자 추가

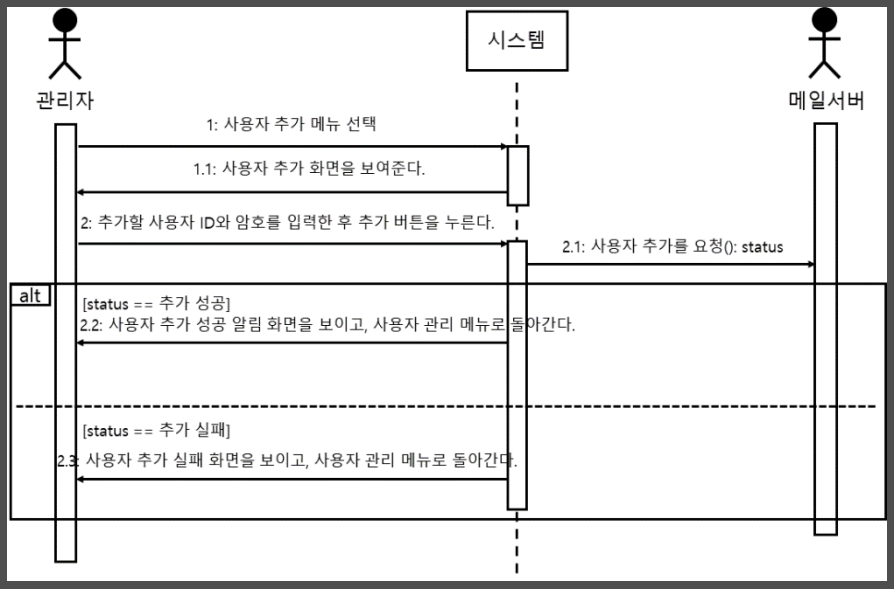
추가할 기능을 명확히 하기 위해 오퍼레이션의 책임, 사전조건, 사후조건을 정리하고 순차 다이어그램으로 함수의 흐름을 나타냈습니다.

책임은 함수의 핵심기능, 사전조건은 오퍼레이션이 실행되기 전 시스템이나 모델 객체 상태,사후조건은 완료된 후 모델 상태입니다.

^사용자 추가 시스템 오퍼레이션^

오퍼레이션	addUser(String userId, String password)
책임	서버에 신규 사용자의 ID와 암호를 등록한다.
사전조건	• 시스템의 서버가 연결되어 있어야 한다.
사후조건	• 비밀번호가 조건에 맞다면 서버에 신규 사용자의 정보를 저장한다.

^사용자 추가 순차 다이어그램^



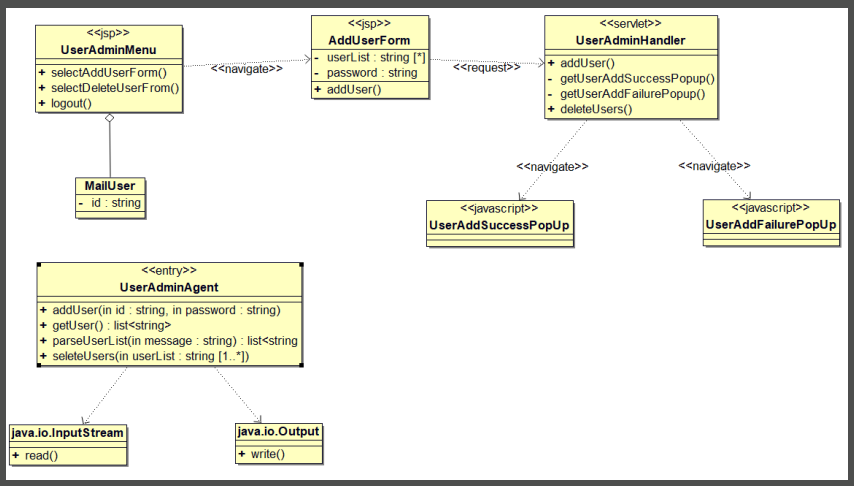
Copyright ⓒ Slug. All right reserved.

02

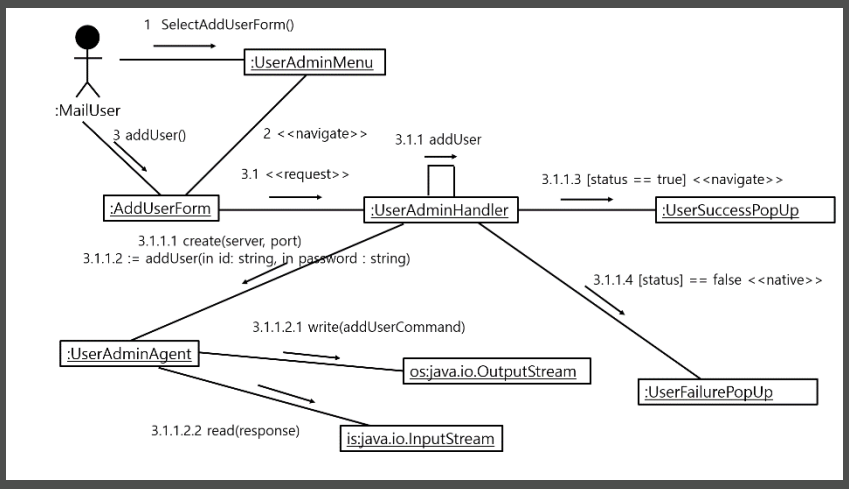
분석 및 설계

추가될 클래스와 기존에 구현된 시스템과의 상호작용과 충돌을 방지하기 위해 클래스 다이어그램과 통신 다이어그램을 작성하였습니다.

^사용자 추가 클래스 다이어그램



^사용자 추가 통신 다이어그램



Copyright © Slug. All right reserved.

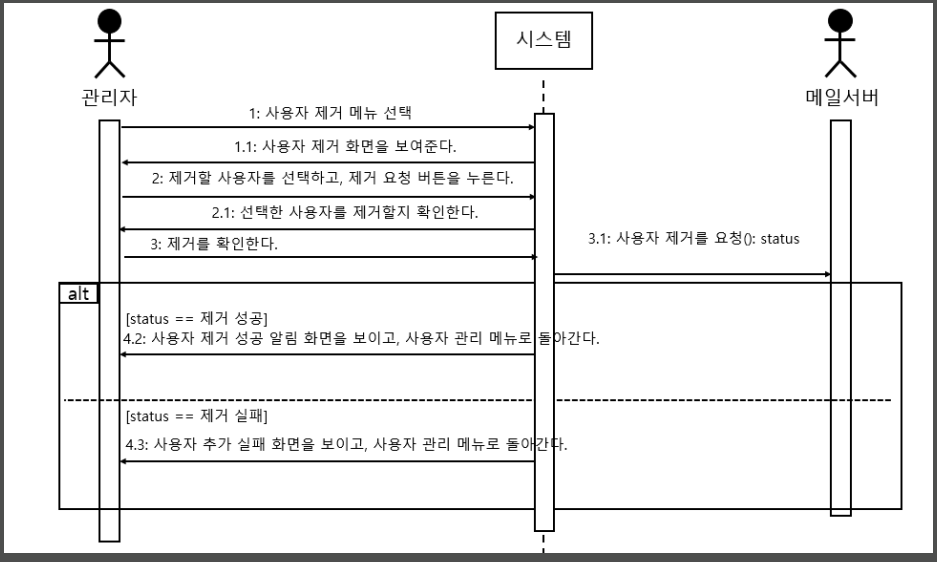
사용자 삭제

다음의 다이어그램과 오퍼레이션 정의는 위와 같은 목적을 가지고 작성되었습니다.

^사용자 삭제 시스템 오퍼레이션^

오퍼레이션	deleteUser(String[] userList)
책임	서버에서 사용자의 정보를 제거한다.
사전조건	• 시스템의 서버가 연결되어 있어야 한다. • 서버에 등록된 사용자여야 한다. • 사용자를 삭제할 수 있는 권한을 가진 사용자여야 한다.
사후조건	• 선택한 사용자가 제거된다.

^사용자 삭제 순차 다이어그램^

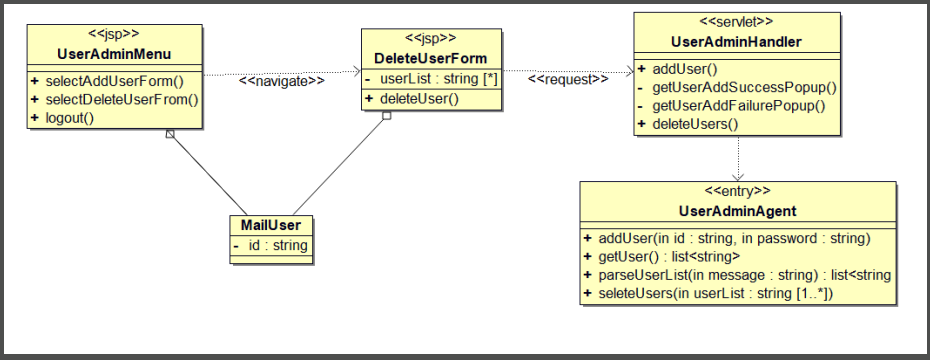


Copyright ⓒ Slug. All right reserved.

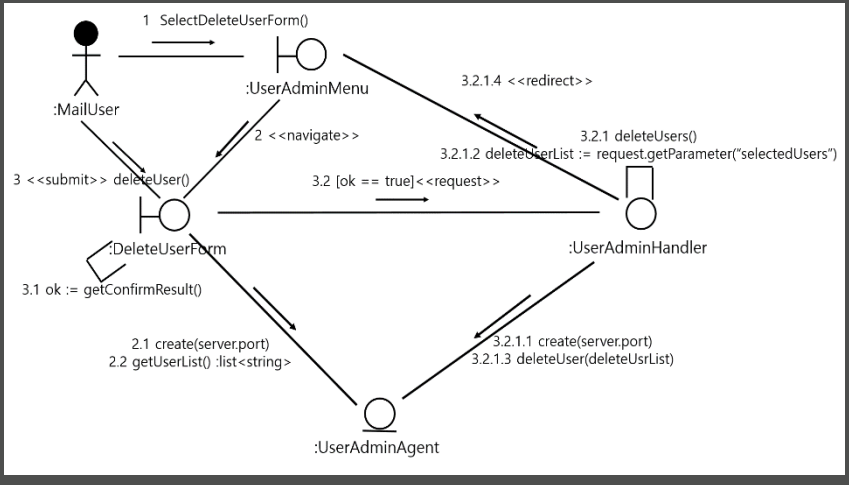
02

분석 및 설계

^사용자 삭제 클래스 다이어그램



^사용자 제거 통신 다이어그램



Copyright ⓒ Slug. All right reserved.

사용자 기능 추가 (완전화 유지보수) 설계

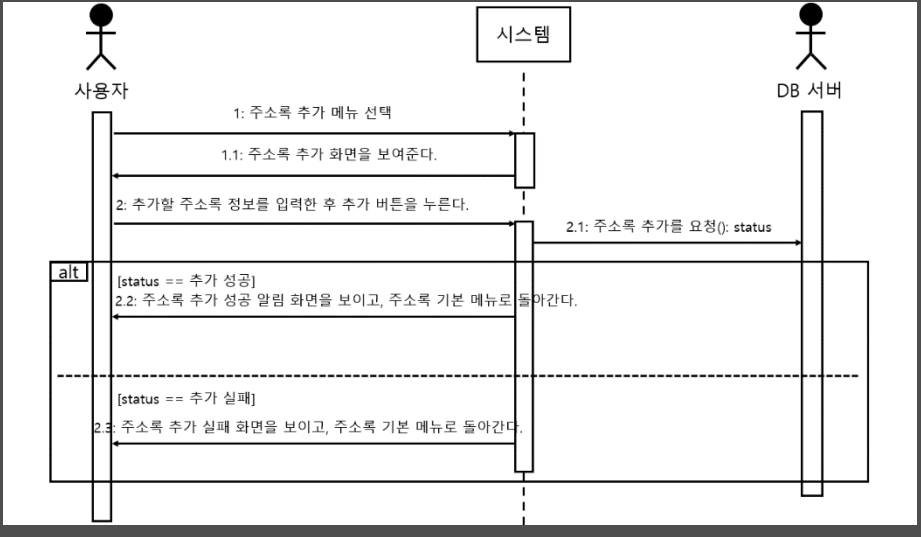
기존의 웹 메일 시스템에서 제공하지 않던 기능인 **주소록** 기능 제공을 통해 사용자가 더 편하게 시스템을 활용할 수 있도록 하였습니다.

주소록 추가

^주소록 추가 시스템 오퍼레이션^

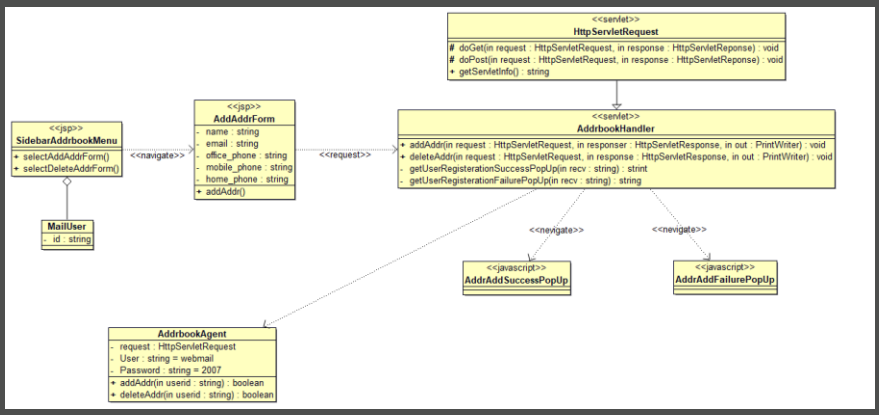
오퍼레이션	addAddr(String userId)
책임	입력한 주소록의 정보를 DB에 저장한다.
사전조건	• 로그인된 상태여야 한다.
사후조건	• 입력된 주소록이 DB에 저장된다.

^주소록 추가 순차 다이어그램^

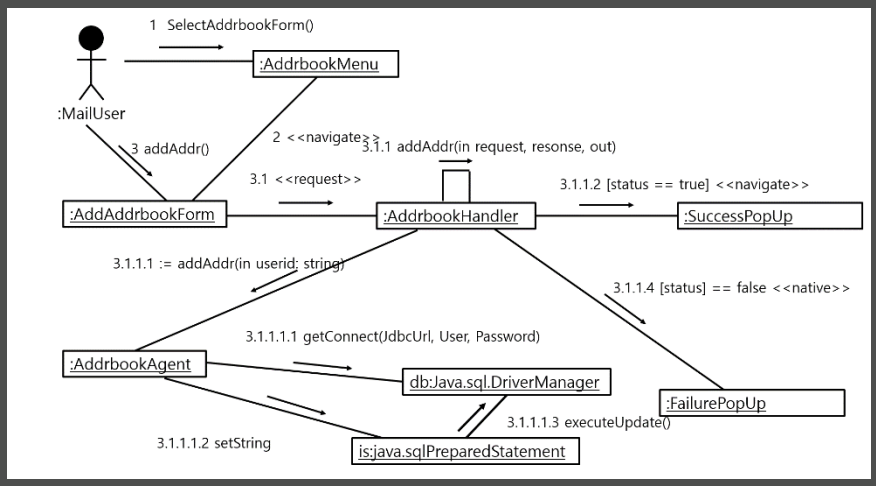


Copyright ⓒ Slug. All right reserved.

^주소록 추가 클래스 다이어그램



^주소록 추가 통신 다이어그램



주소록 삭제

^주소록 추가 시스템 오퍼레이션

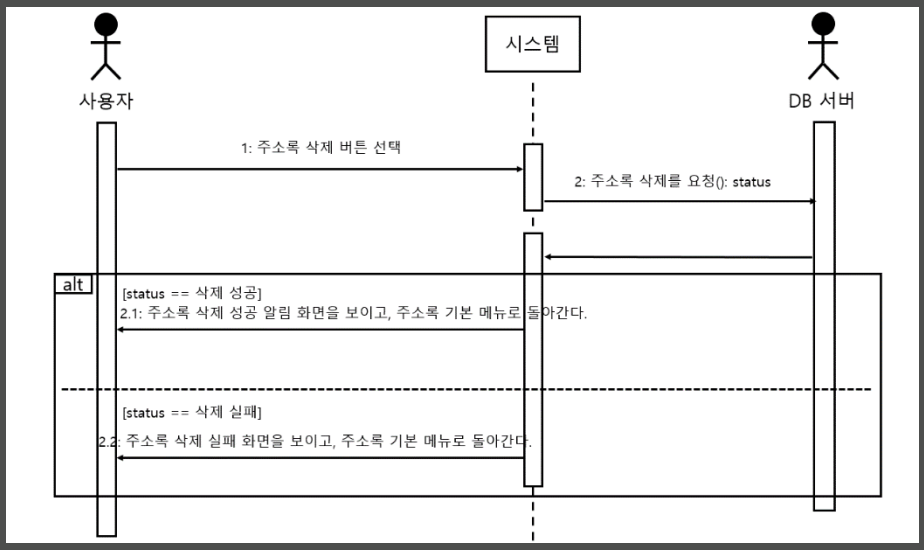
오퍼레이션	deleteAddr()
책임	선택한 주소록의 정보를 DB에서 삭제한다.
사전조건	• 로그인된 상태여야 한다. • 삭제할 주소록이 등록된 상태여야 한다.
사후조건	• 선택된 주소록이 DB에서 삭제된다.

Copyright © Slug. All right reserved.

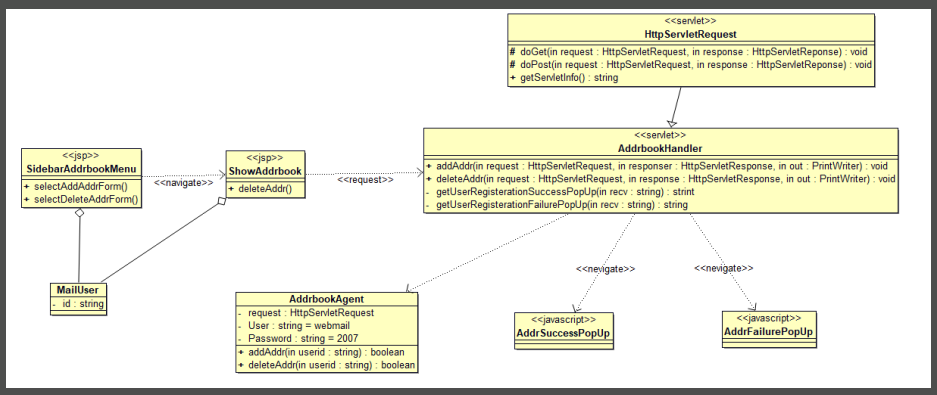
02

분석 및 설계

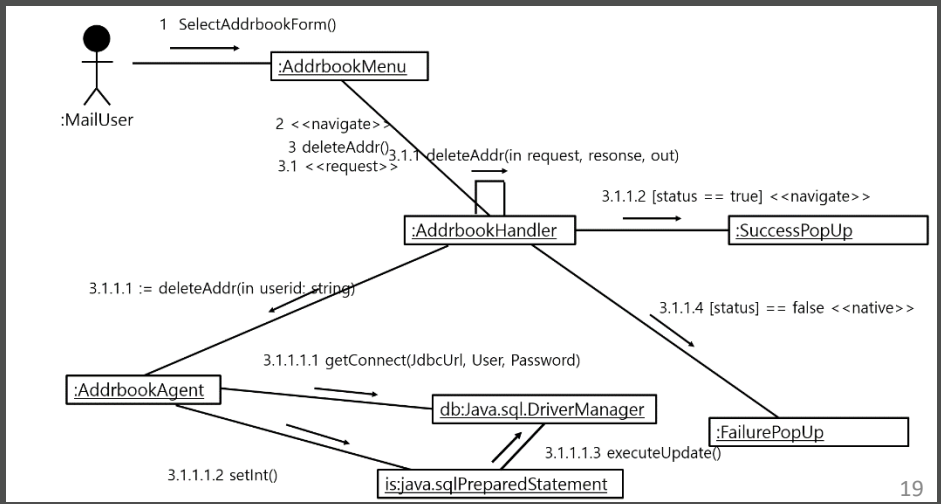
^주소록 삭제 순차 다이어그램



^주소록 추가 클래스 다이어그램



^주소록 삭제 통신 다이어그램



Copyright © Slug. All right reserved.

사용자 기능 추가 (완전화 유지보수) 설계

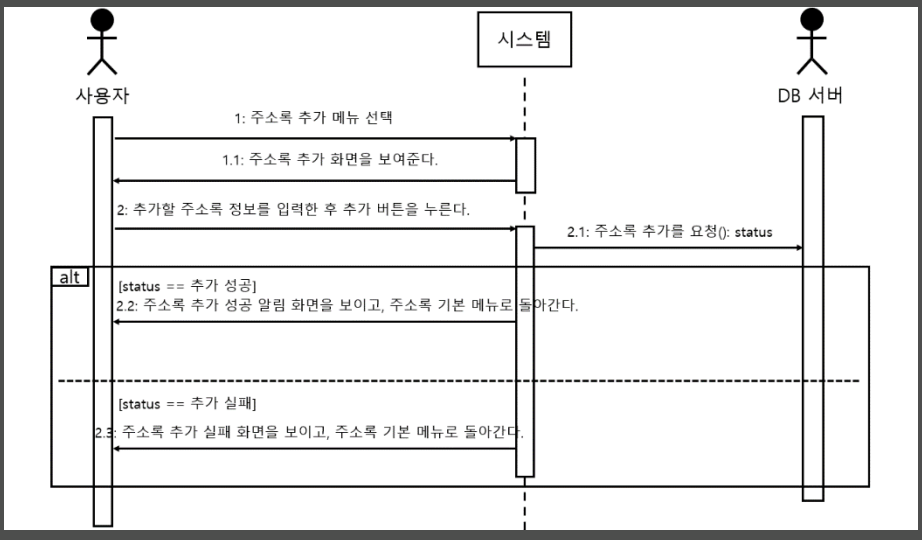
기존의 웹 메일 시스템에서 제공하지 않던 기능인 **주소록** 기능 제공을 통해 사용자가 더 편하게 시스템을 활용할 수 있도록 하였습니다.

주소록 추가

^주소록 추가 시스템 오퍼레이션^

오퍼레이션	addAddr(String userId)
책임	입력한 주소록의 정보를 DB에 저장한다.
사전조건	• 로그인된 상태여야 한다.
사후조건	• 입력된 주소록이 DB에 저장된다.

^주소록 추가 순차 다이어그램^



Copyright ⓒ Slug. All right reserved.

02

실행화면

실행화면

실행 영상은 아래의 주소를 클릭하시면 시청하실 수 있습니다.

https://youtu.be/f_HfVW31H1A

Copyright ⓒ Slug. All right reserved.

03

프로젝트 개요

BusanIn

2019.11 - 2020.01

프로젝트 설명

부산의 관광지와 맛집을 소개하고, 댓글과 메시지를 통해 사용자간 소통할 수 있는 웹 서비스입니다.

담당업무

1인 개발

1. 게시글 CRUD, 사업체 CRUD, 댓글 CRUD, 사용자 CRUD 설계, 구현.
2. 사용자간 대화 기능 설계, 구현.
3. Keyword를 통한 post, business 검색 구현.
4. Post와 business 형식 별 검색 구현.
5. UserPassesTesMixin을 활용한 기능별 접근 제한 구현.
6. Social Login(Kakao, Github) 구현.
7. Mailgun 서비스를 통한 회원가입 인증 서비스 구현.
8. AWS EB와 AWS S3를 통해 서비스 배포.
9. AWS Route53를 통해 https://busanin.be로 서비스 배포(현재는 내렸습니다.).
10. TailwindCSS, HTML을 통한 Front End 화면 설계, 구현.

Copyright ⓒ Slug. All right reserved.

개발환경

개발환경 Windows10
개발언어 Python 3
개발기술 Django 2.2, Tailwind CSS, HTML5, AWS EB, AWS S3, ~~Route53(현재 내렸습니다.)~~
기타 Mailgun, 소셜 로그인(KakaoTalk, Google)

참조 (※ 주소를 클릭하시면 이동합니다!)

자세한 코드는 다음의 GitHub 주소를 참조해주세요.

https://github.com/hyeonsook95/busanin

Copyright ⓒ Slug. All right reserved.

03

설계 및 구현

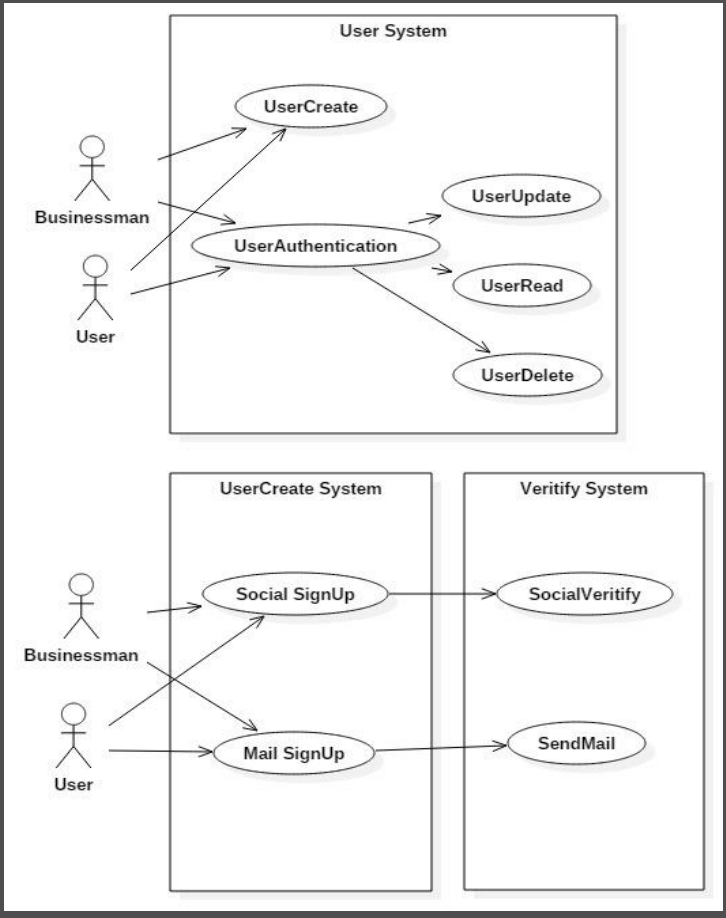
Django는 MVC(MTV) 패턴으로 이루어진 Framework입니다.

Model을 통해 움직일 데이터를 정의하고, View를 통해 Model의 데이터를 어떻게 움직일지 구현한 후 Template로 보여줍니다.

기능 분석

사용자 기능 분석

- 웹 서비스에서 제공하는 유저는 관리자, 일반사용자, 사업자가 있습니다.
- 사용자들은 email과 social을 통해 회원가입 할 수 있습니다.



<User 유스케이스>

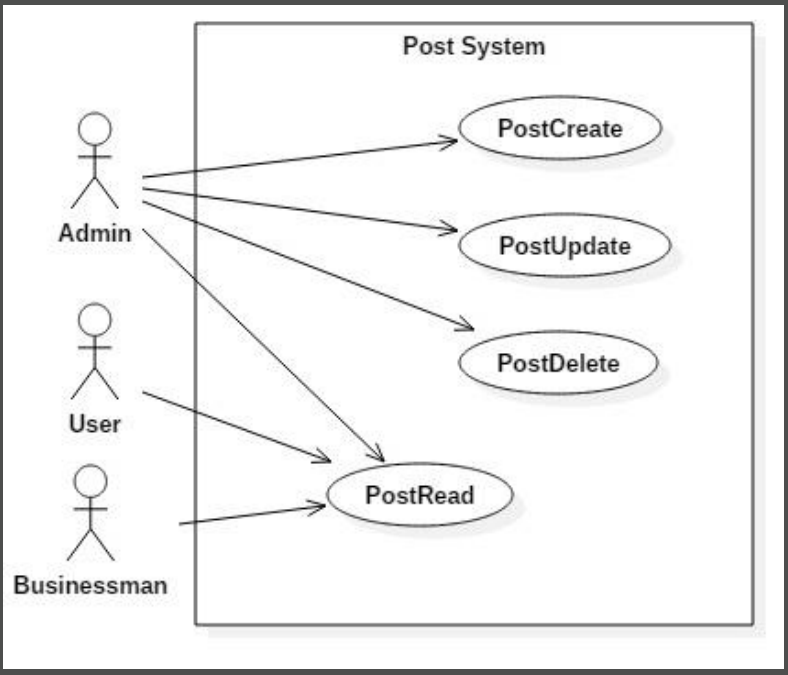
Copyright © Slug. All right reserved.

03

설계 및 구현

게시글 기능 분석

- 관리자는 게시글을 CRUD할 수 있습니다.
- 일반사용자와 사업자는 게시글을 읽을 수 있습니다.



<Post 유스케이스>

Copyright © Slug. All right reserved.

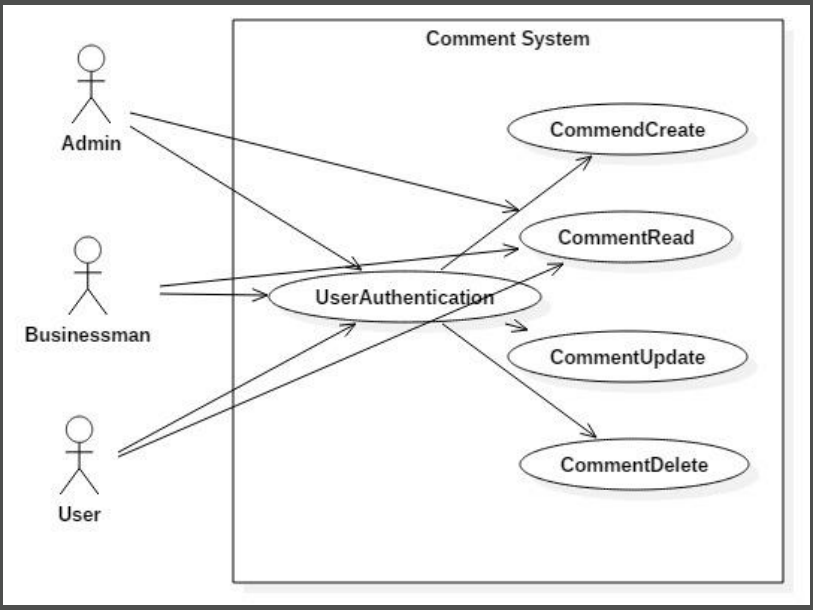
03

설계 및 구현

댓글 기능 분석

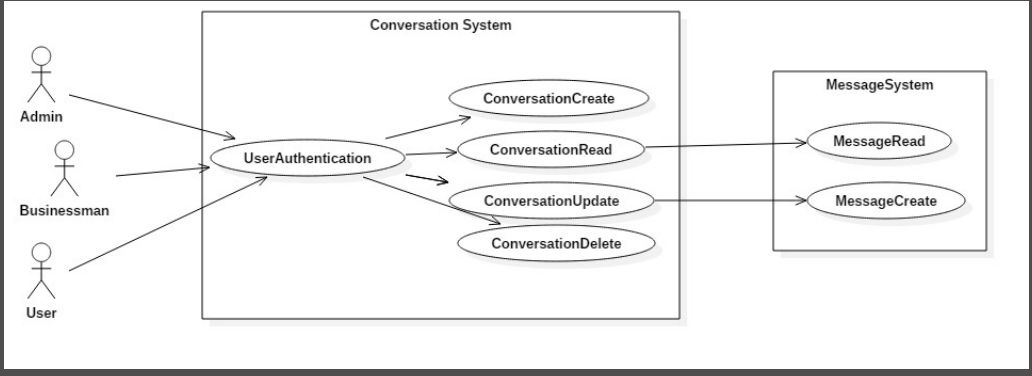
- 사용자들은 댓글 기능을 사용하기 위해 인증해야 합니다.
- 사용자들은 댓글을 CRUD할 수 있습니다.

<Comment 유스케이스>



대화 기능 분석

- 사용자들은 대화를 CRUD할 수 있습니다.
- 사용자들은 Message를 생성할 수 있습니다.

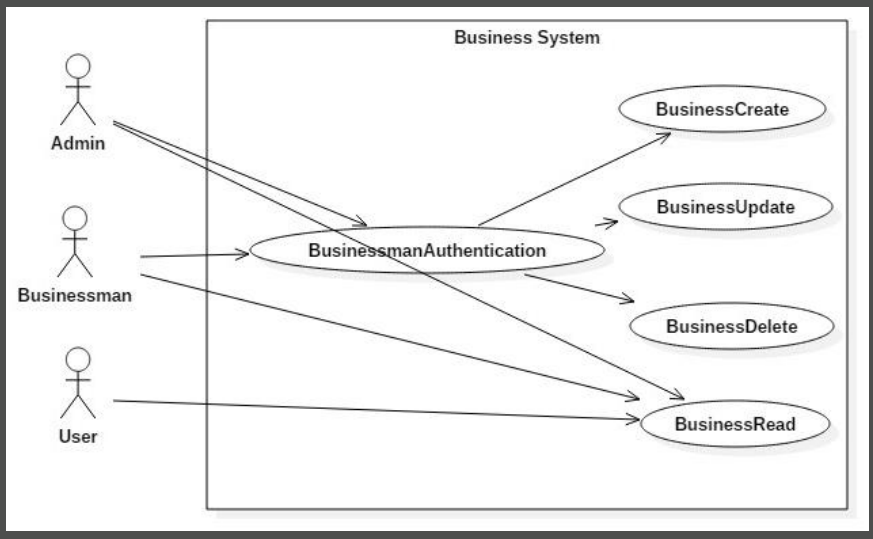


Copyright ⓒ Slug. All right reserved.

사업체 기능 분석

- 관리자와 사업자는 사업체를 CRUD할 수 있습니다.
- 사용자는 사업체를 읽을 수 있습니다.

<Business 유스케이스>



Copyright ⓒ Slug. All right reserved.

03

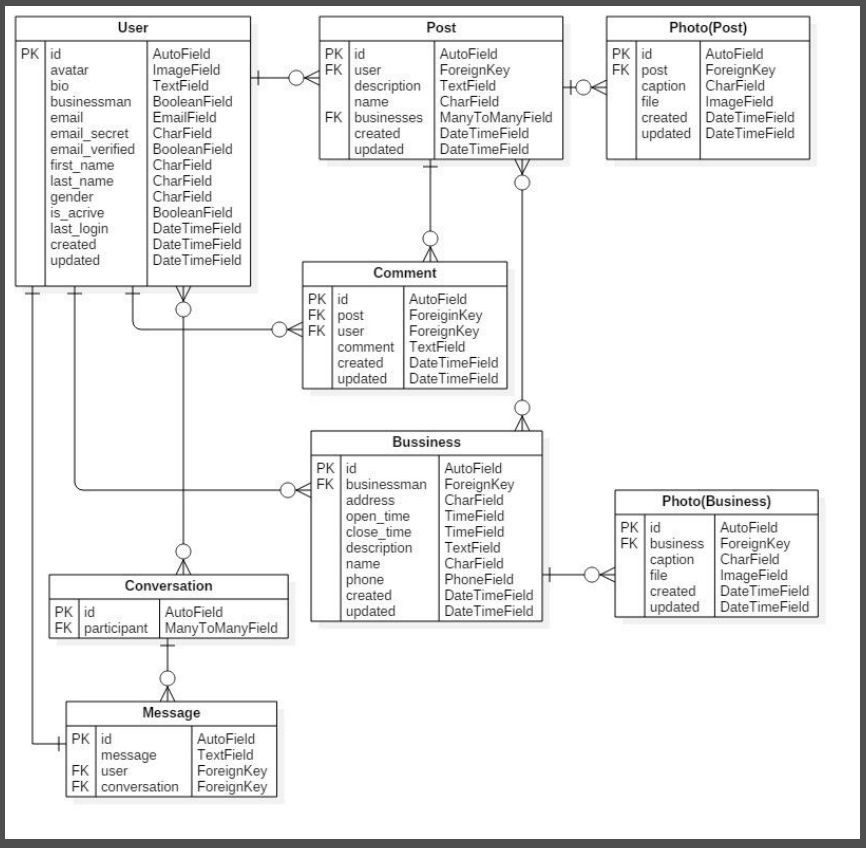
설계 및 구현

Model 설계

구현할 기능을 분석하여 설계할 때 Model 설계에 가장 많은 시간을 할애했습니다.

App과 Model을 분리하고 생성하는 기준을 어떻게 해야 할지, Python의 클래스 개념을 Model에 더 잘 활용할 수 있을지에 대해 고민하면서 설계했습니다.

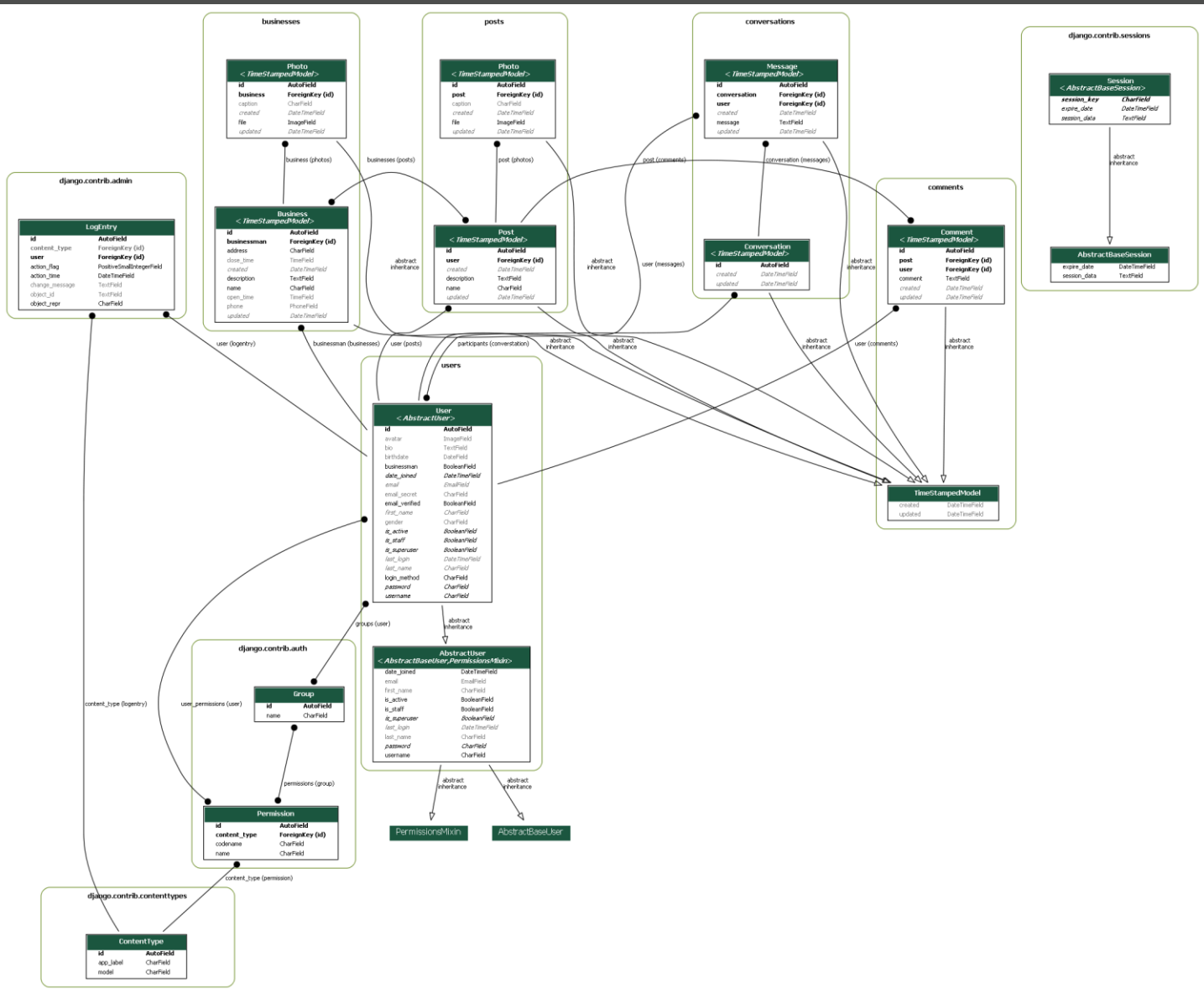
그래서 2개 이상의 모델들에 공통으로 들어가고 앞으로 업데이트를 할 때, 들어갈 계획이 있는 조건들은 Abstract 클래스 기능을 활용하여 Model을 설계했습니다.



<DataDiagram 설계>

Copyright © Slug. All right reserved.

Django-extensions로 뽑아낸 실제 구현된 Model입니다.

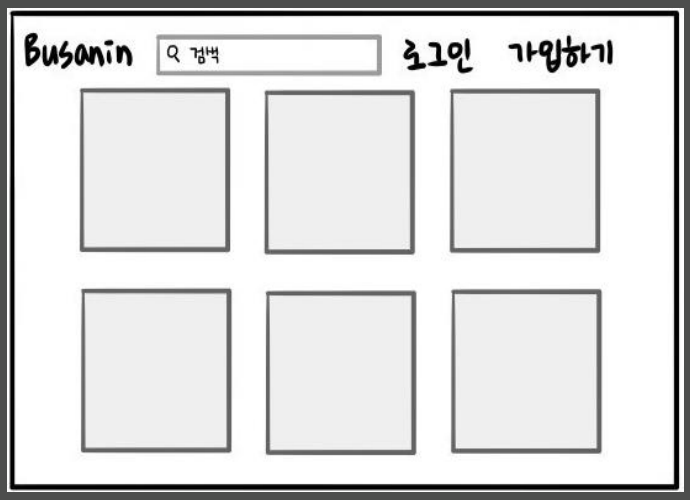


Copyright © Slug. All right reserved.

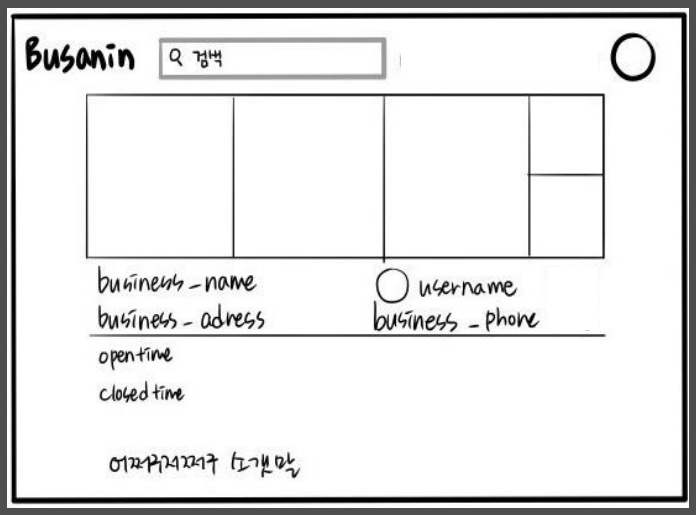
Template 설계

디자인은 Airbnb과 인스타그램의 디자인을 참조했습니다.

각 Template의 설계는 기능별로 가장 상세한 부분만 설계했습니다.



메인 화면

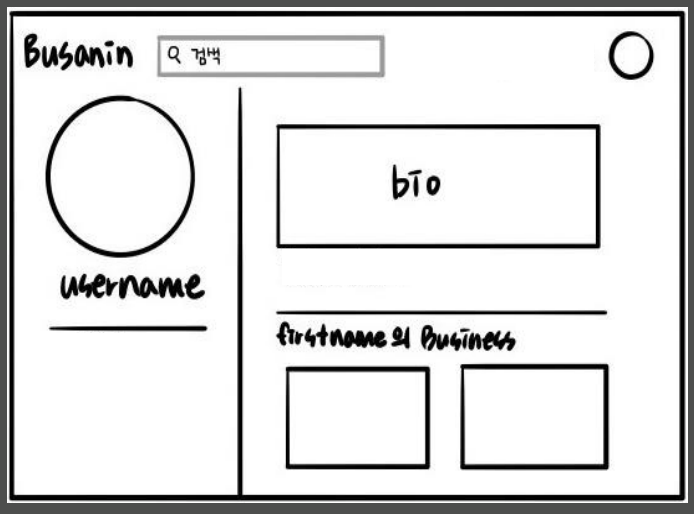


사업체 상세 화면

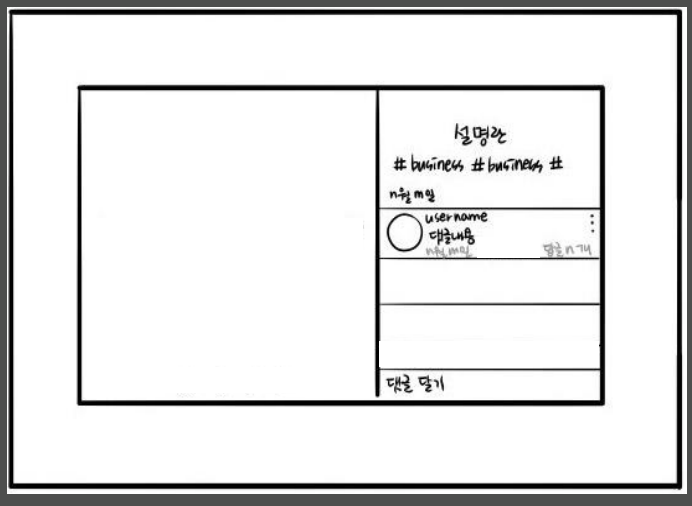
Copyright © Slug. All right reserved.

03

설계 및 구현



사용자 정보 보기 화면



게시 글 상세 화면

Copyright © Slug. All right reserved.

View 구현

Core App

Core 앱에서는 여러 앱에서 공통적으로 사용되는 모델이나 상속 모델과 기능들을 구현하였습니다.

View를 구현하면서 코드의 확장성을 최대한 늘이기 위해 고민했습니다. 처음에는 많은 View들을 함수형으로 구현하였는데 Django를 더 공부하면서 확장성을 위해 클래스형으로 리팩토링 하고, Django에 구현되어 있는 Mixin을 참고하여 적용시켰습니다.

#models.py

TimeStampedModel	• 여러 모델에서 공통으로 사용하는 created, updated 속성을 선언해 놓은 Abstract 클래스입니다.

#managers.py

CustomModelManager	• get_or_none 함수(objec를 찾을 때, 없으면 none을 반환)를 선언하여 사용하기 위한 클래스입니다. • 확장성을 위해 Custom Manager를 따로 선언하고 필요한 모델 Manager가 상속받는 형식으로 만들었습니다.

#mixins.py

EmailLoginOnlyMixin	• 사용자가 Email User 일 때만 접근 가능하도록 제한하기 위한 Mixin 클래스입니다.
LoggedOutOnlyMixin	• 사용자가 Log Out 상태일 때만 접근 가능하도록 제한하는 Mixin 클래스입니다.
LoggedInOnlyMixin	• 사용자가 Log In 상태일 때만 접근 가능하도록 제한하여, Log In 페이지로 넘겨주는 Mixin 클래스입니다.

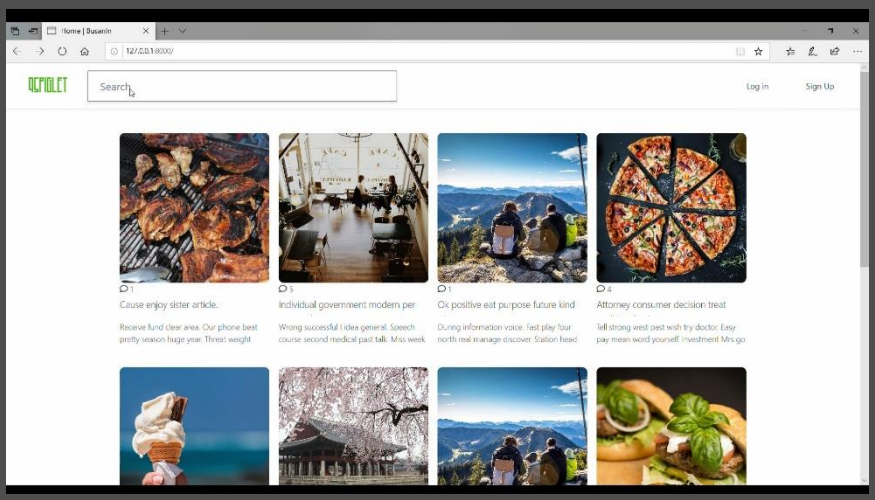
Copyright © Slug. All right reserved.

03

실행화면

실행 영상은 아래의 주소를 클릭하시면 시청하실 수 있습니다.

https://youtu.be/nTb7JKrVl7Q



Copyright © Slug. All right reserved.

감사합니다.

THANK YOU.

Copyright ⓒ Slug. All right reserved.