2017.1.10 메일파트

메일 시스템 개편 프로젝트 중간 리뷰

CONTENTS

프로젝트 목적

STRENGTH

- 1997년 5월 국내 최초 무료 웹메일
- MAU 1400만 명
- 월간 발송 메일 6000만 통
- 월간 수신 메일 40억 통

THREATS

- 커뮤니케이션 채널의 변화 – 모바일 메신저
- 정보 구독 채널의 변화 – SNS
- 스팸 메일

WEAKNESS

- 20년 간 누적된 레거시 환경
- 시스템 단일 장애점
- C 언어 개발자 구인 어려움
- 기능 변경 비용 큼
- 오랜 시간 서비스 혁신이 없었음

OPPORTUNITIES

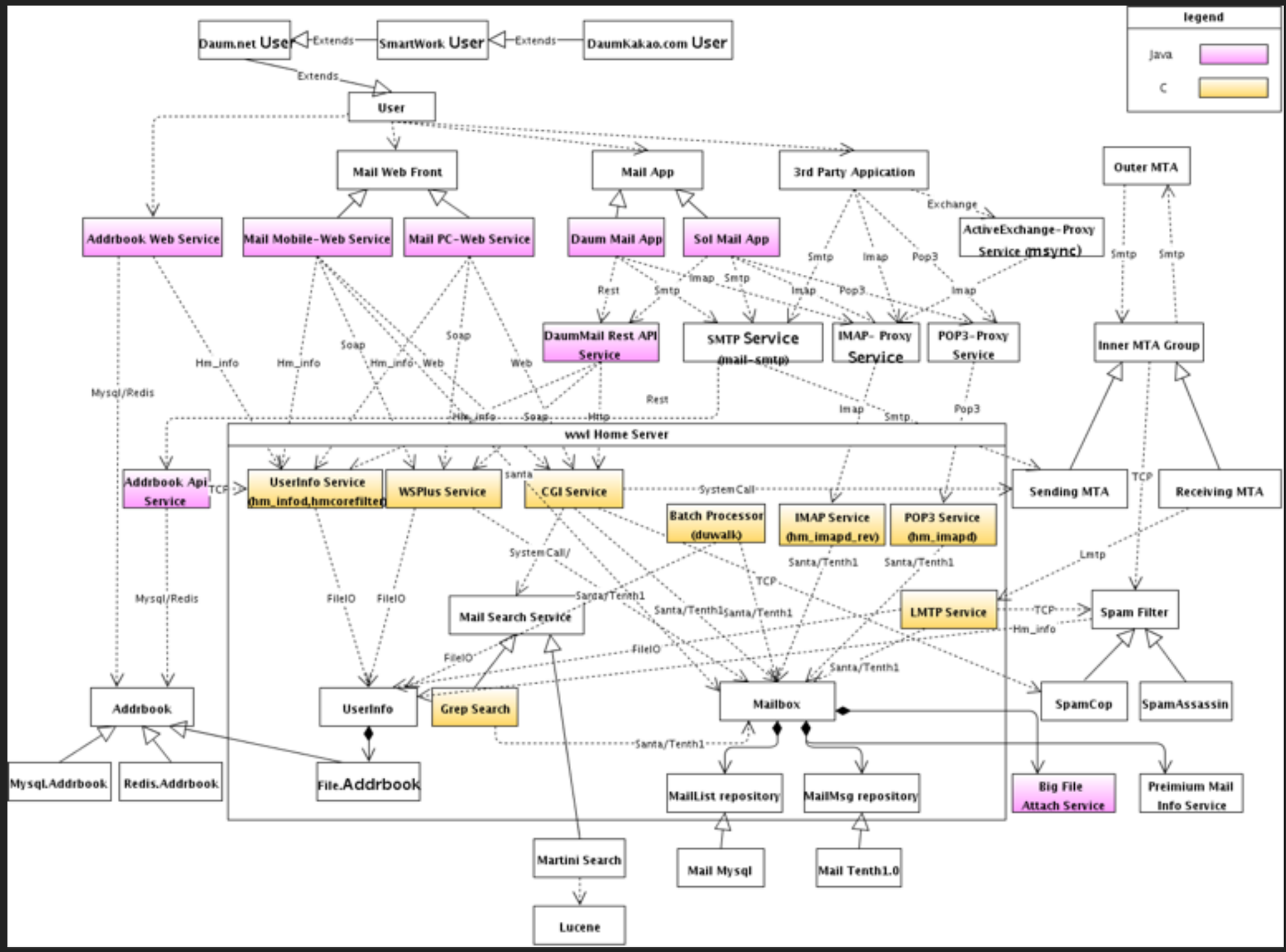
- KakaoTalk 에서 one-click으로 메일 계정을 생성 가능하다면?
- 메일 수신, 오픈 이력을 분석해 광고 타게팅에 활용 한다면?
- SNS 와 연동되는 소셜주소록으로 이메일 주소를 몰라도 지인에게 메일을 보낼 수 있다면?
- 옵트인 방식으로 신뢰있는 메일만 수신 허용한다면?
- 앱스토어 같은 뉴스레터 구독, 리뷰, 랭킹 플랫폼을 제공한다면?

레거시 시스템 리팩토링 전략

LEGACY PAIN POINT

- C 언어로 작성된 코드
- Data 에 직접 접근 하는 다수의 바이너리
- 오래된 서비스 정책, 부족한 문서
- 복잡한 상호 Dependency
- 커스텀 프로토콜(UDP, TCP)

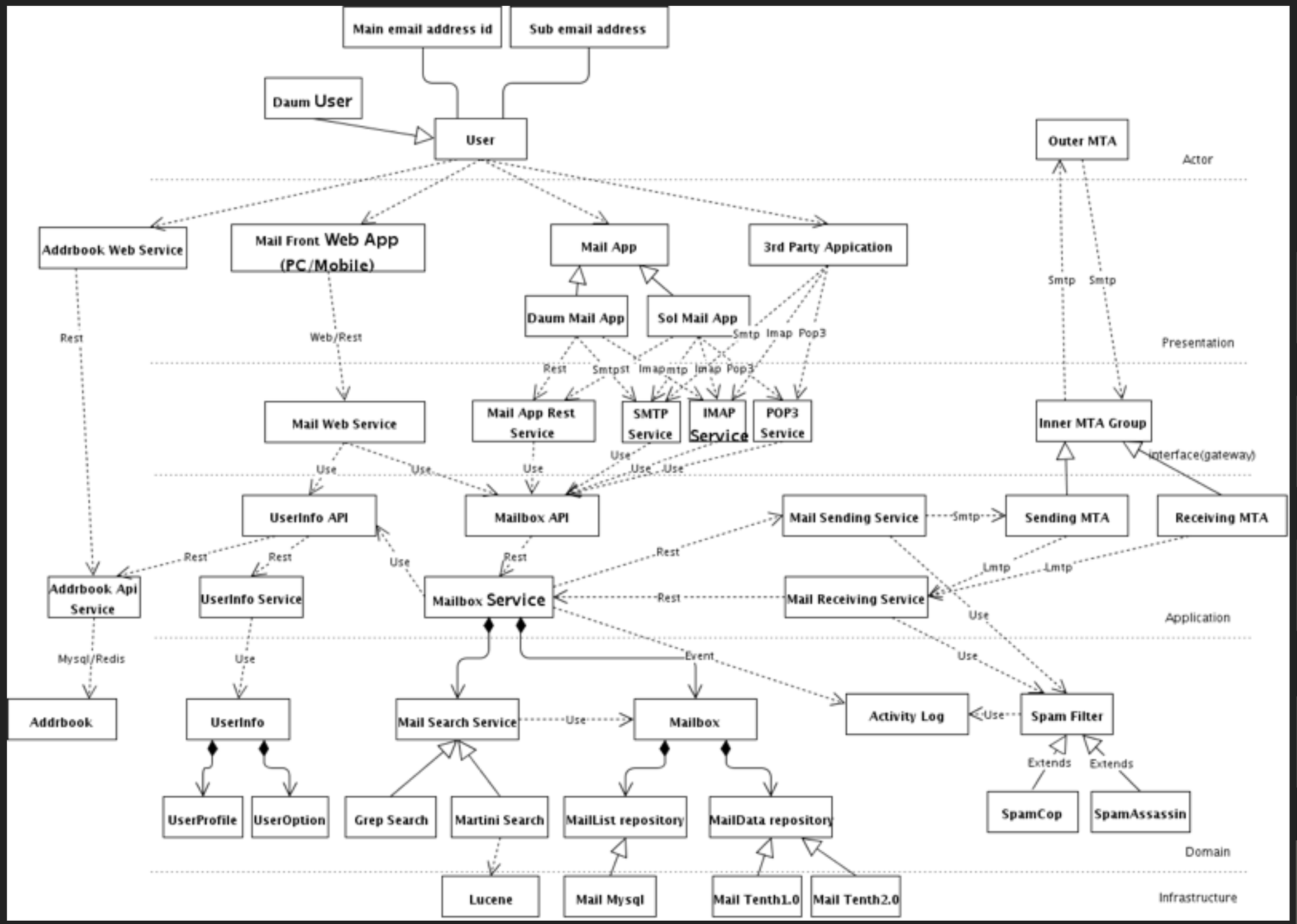
SYSTEM DOMAIN MODEL AS-IS



리팩토링 전략

- 현대적인 생산성 높은 언어
- 마이크로 서비스 아키텍쳐
- 점진적인 전환
- 충실한 문서(DSDM)
- TDD, CI, Pair
- 데이터 분석 환경

SYSTEM DOMAIN MODEL TO-BE (LAYERED VIEW)



ACTOR

PRESENTATION

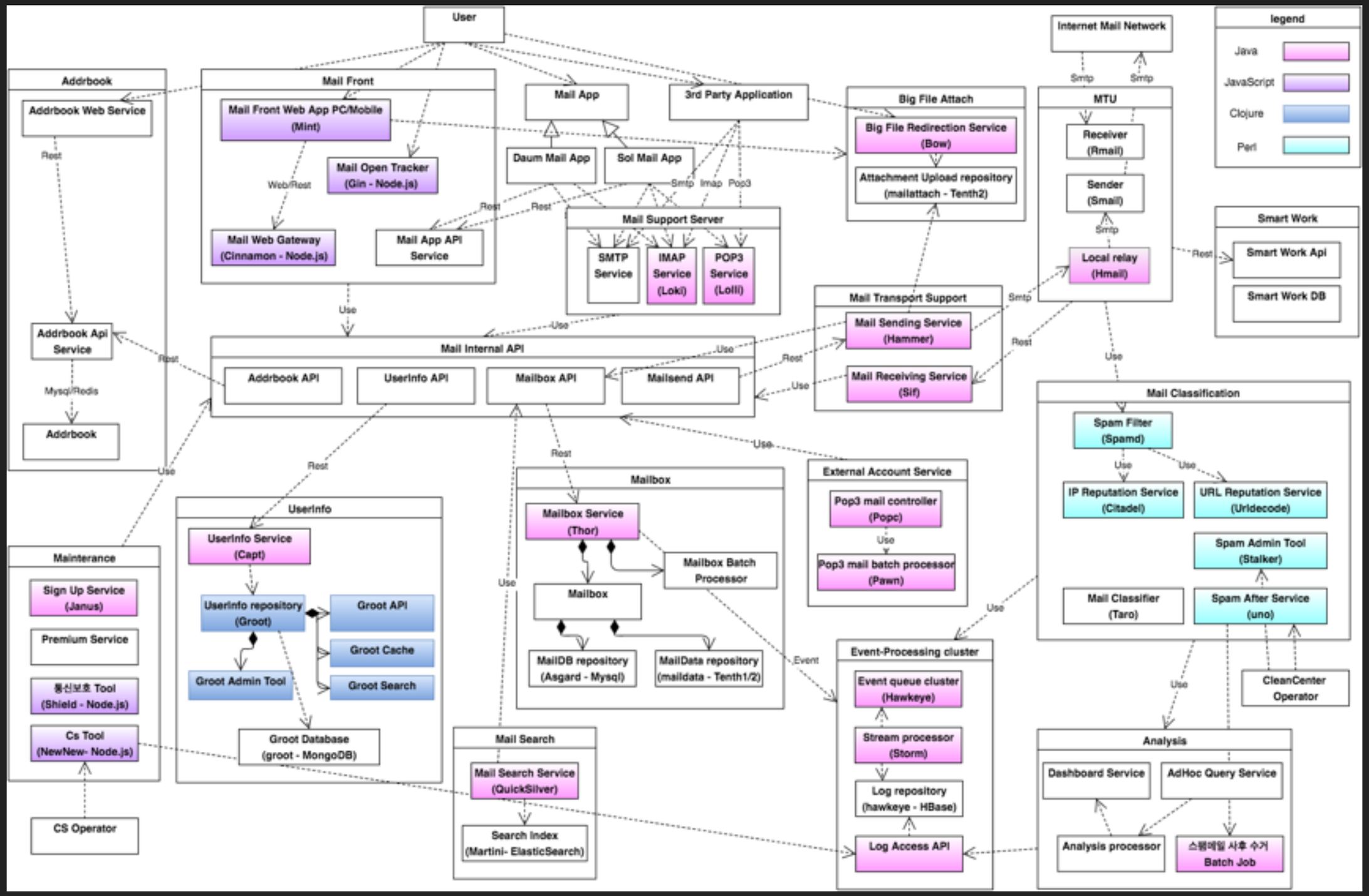
INTERFACE

APPLICATION

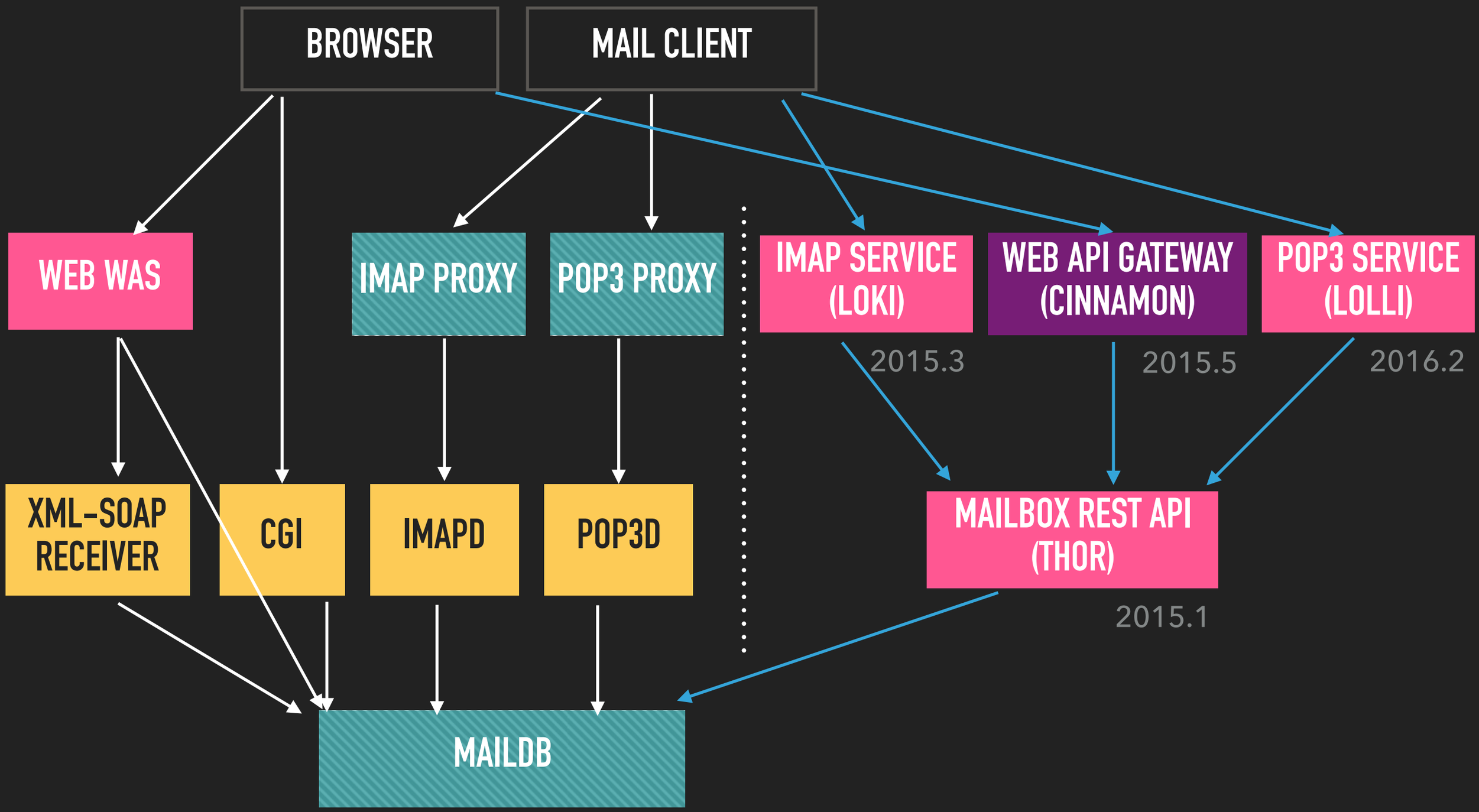
DOMAIN

INFRASTRUCTURE

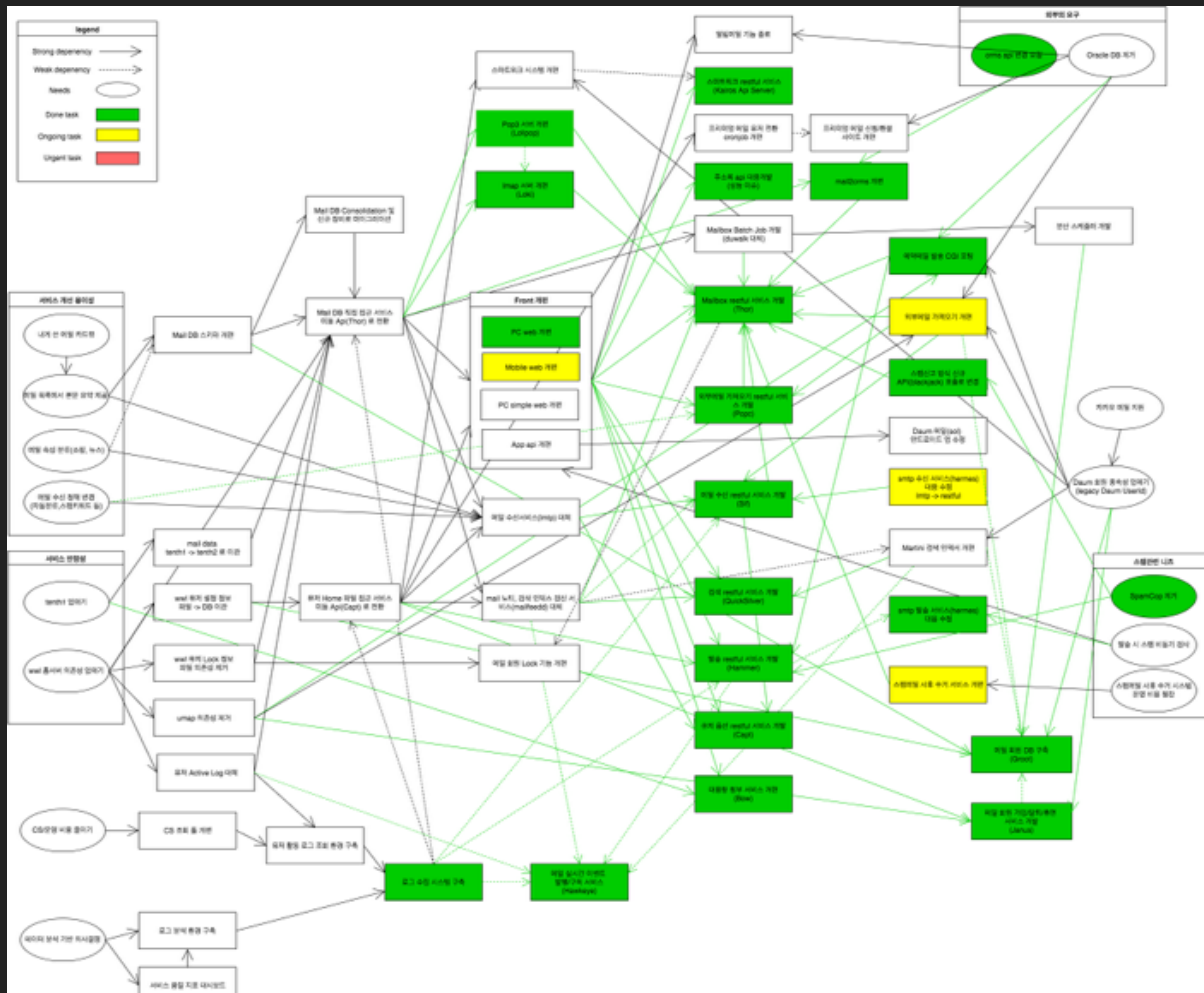
SYSTEM DOMAIN MODEL TO-BE (COMPONENT VIEW)



WRAPPING UP WITH REST API



점진적인 전환 – TASK DEPENDENCY MAP



충실한 문서 – 목적

- 프로젝트 초기, 설계 단계에서는
  - 생각을 정리, 아이디어를 구체화
  - 다수에게 효율적으로 리뷰, 피드백 받기 위해
- 프로젝트 과정
  - 결정 사항을 공식화하고, 협업 시 명확한 커뮤니케이션 도구
  - 중도에 합류하는 멤버의 빠른 이해 도움
- 프로젝트 완료 후
  - 본인의 기억력은…
  - 회고, 유지보수, 새로운 이터레이션 시작의 기반
- 지속적으로 업데이트가 필요하고, 가능한 문서만 만들 것

문서 EX) LOKI- 프로젝트 참여원 및 역할

프로젝트 참여원 및 역할

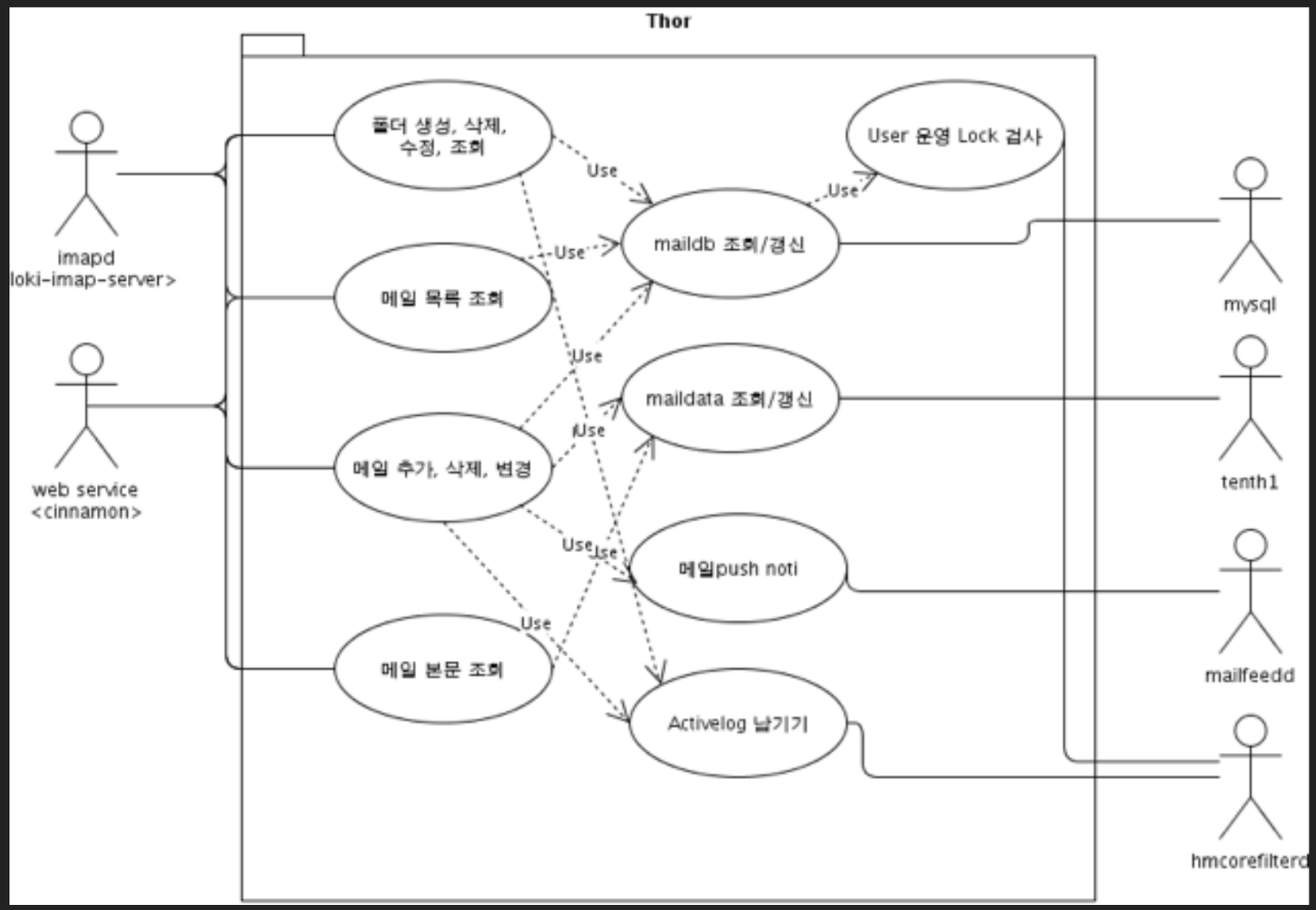
Name	Roles
Dex	Project Manager, Visionary, Developer, Tester
Riela	Developer, Tester, Ambassador User
Brandon	Observer
Amy	Developer, Tester
Irene	Tester, Ambassador User

- Agile DSDM Handbook - https://www.agilebusiness.org/resources/dsdm-handbooks
  - 프로젝트 구성원의 역할(Role)과 책임, 최소 필요 문서에 대한 풍부한 가이드

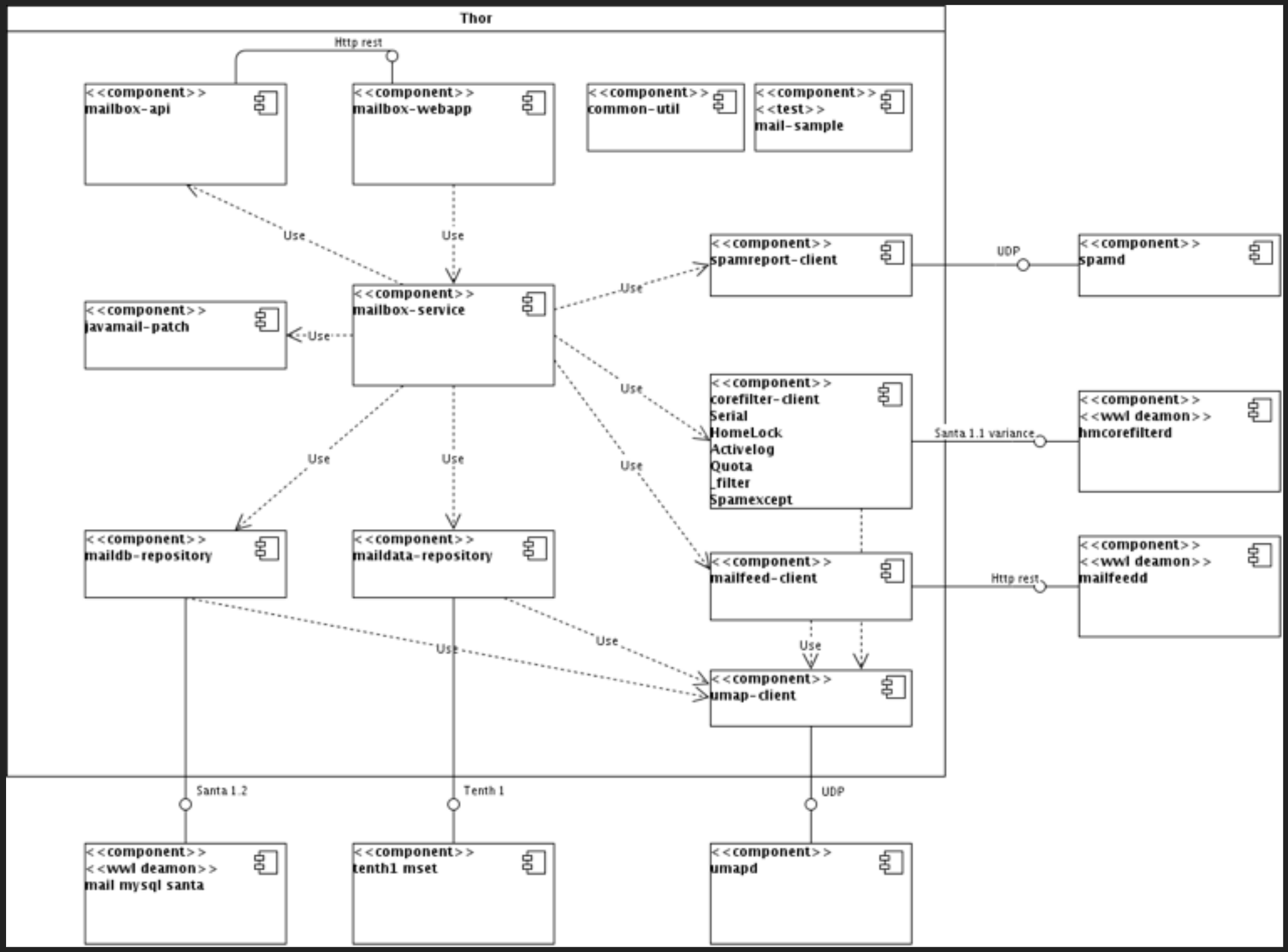
문서 EX) THOR- 이터레이션 진행 상황

단계	설명	Start Date	End Date
Setup	pre-project(project설계), 개발환경셋팅	2014-12-17	2014-12-29
FMI #1	Mail Data(tenth 1) CRUD 를 위한 api 1차 스펙 구현	2015-1-5	2015-2-13
FMI #2	Mail DB(mysql) 조회, 관리를 위한 api 2차 스팩 구현	2015-2-23	2015-3-13
DBI #1	test서버 셋팅 및 기능 테스트	2015-3-16	2015-3-27
FMI #3	Imap 지원(해더 읽기, 메일 저장, 라벨 변경 지원)을 위한 api 3차 스팩 구현	2015-3-30	2015-4-24
FMI #4	wwl 레거시 서비스와 연동을 위한 기능 개발(mailfeed, serial, lock, activelog)	2015-4-27	2015-5-22
DBI #2	stage 서버 셋팅 및 배포 환경 준비(tenth1 전체 mail map 접근 테스트)	2015-5-26	2015-6-19
FMI #5	웹 프론트 개편을 위한 api 4차 스팩 구현	2015-6-22	2015-7-22
FMI #6	환경설정의 메일함 관리(초기화,지난메일 일괄삭제) 기능 구현	2015-11-19	2015-12-1

문서 EX) THOR - USE CASE



문서 EX) THOR - COMPONENT DIAGRAM



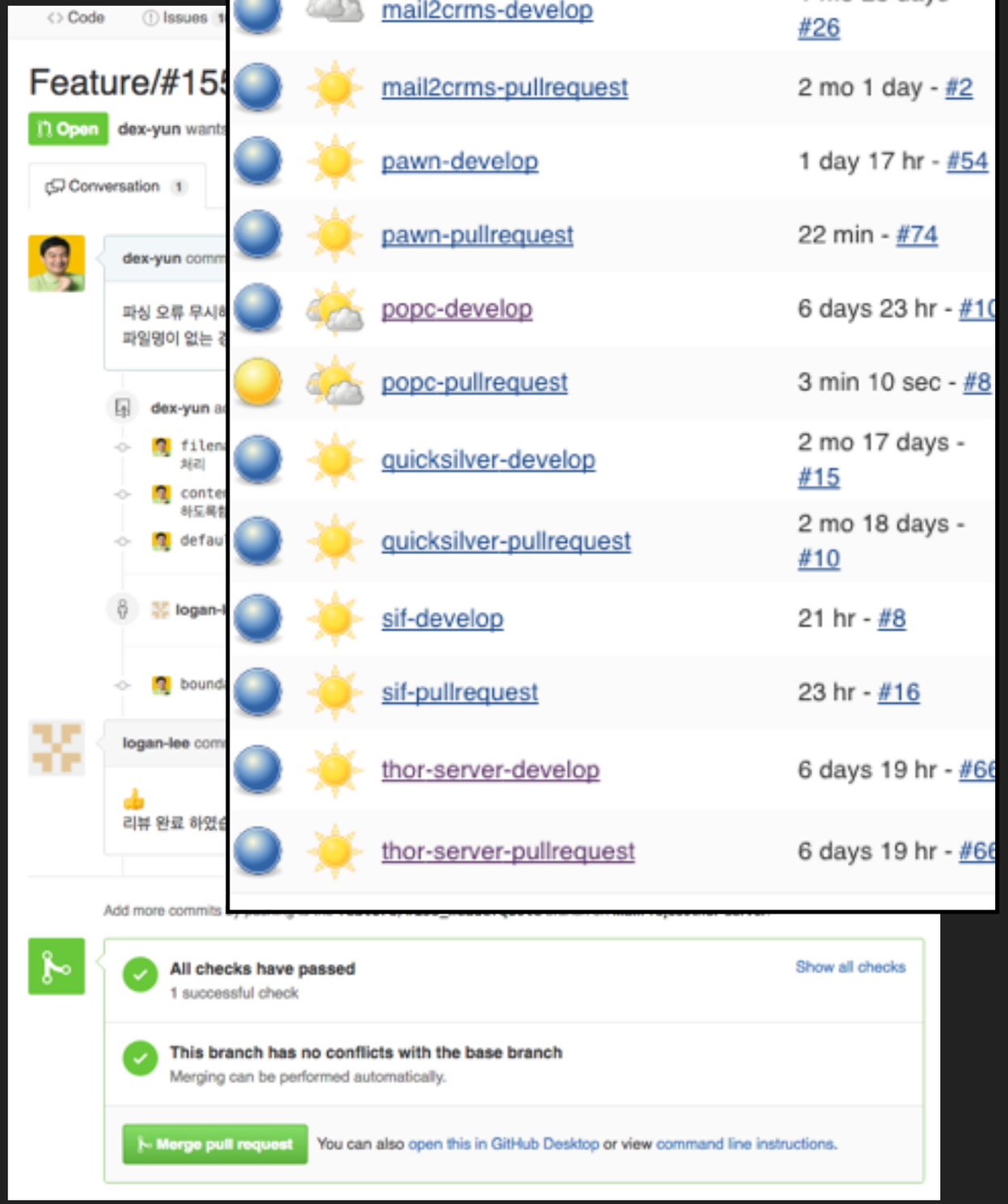
TEST DRIVEN DEVELOPMENT

프로젝트명	Lines	#Test Case	Coverage	Duplications
Thor	23k	602	52.8%	1.5%
Hammer	5.5k	106	67.5%	0.8%
Sif	4k	97	72.0%	6.9%
Capt	4.8k	84	57.6%	2.0%
Loki	27k	133	46.7%	3.2%
Groot-api	2k	84	93.48%	-
Hermes	21k	132	18.0%	7.1%

- Hermes: 2007년에 개발된 Java SMTP 서버. 최근 JDK 1.5->1.6->1.8, Spring 1.2 -> 4.3 로 변경 함

CONTIUOUS INTEGRATION

- Gitflow workflow
- Forking workflow
- Jenkins + Sonar
  - develop branch 5분 간격 polling
  - Github pull request 시 webhook
- 마이크로서비스 별 독립적인 배포 사이클
- 참고문서
  - http://blog.appkr.kr/learn-n-think/comparing-workflows/
  - http://wiki.daumkakao.com/pages/viewpage.action?pageId=376675865



PAIR PROGRAMMING

Thor

Loki

Hammer

Bow

Quicksilver

Capt

Utopia

Janus

Mail2crms

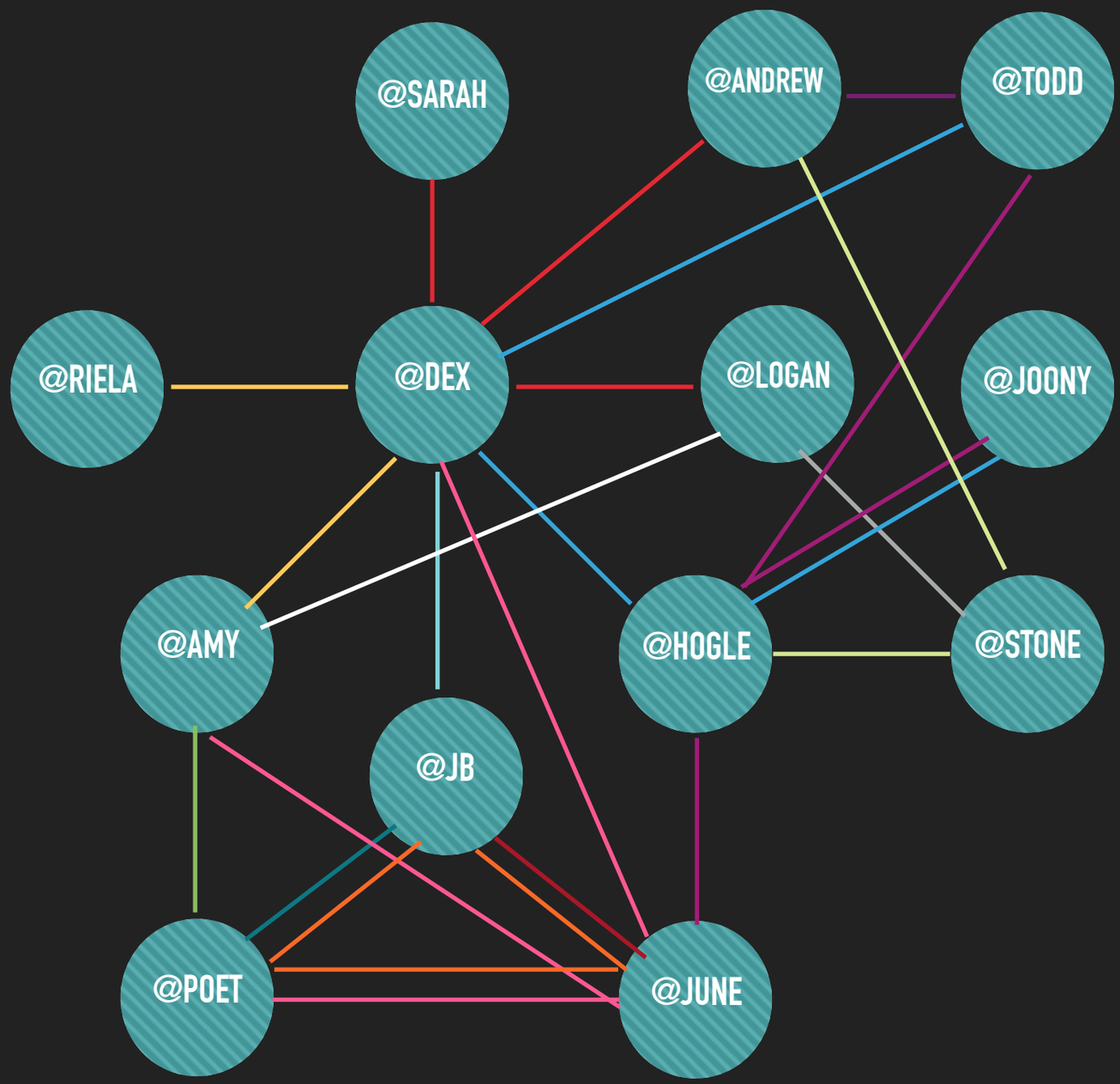
Hawkeye

Hermes

Lolli

Sif

Pawn



AGILE COMMUNICATION

- 2~3 : Pair programming
- ~10 : Daily stand-up meeting
- 10~
  - Weekly focus review - https://kakao.agit.in/g/300008057/wall
  - 프로젝트 별 pair, stand-up meeting, scrum, kanban

데이터 파이프 라인

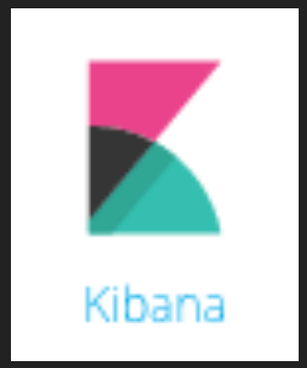
- 실시간 서비스 목적

(HAWKEYE)

- Maintenance 목적

(DKOS)

(KEMI)

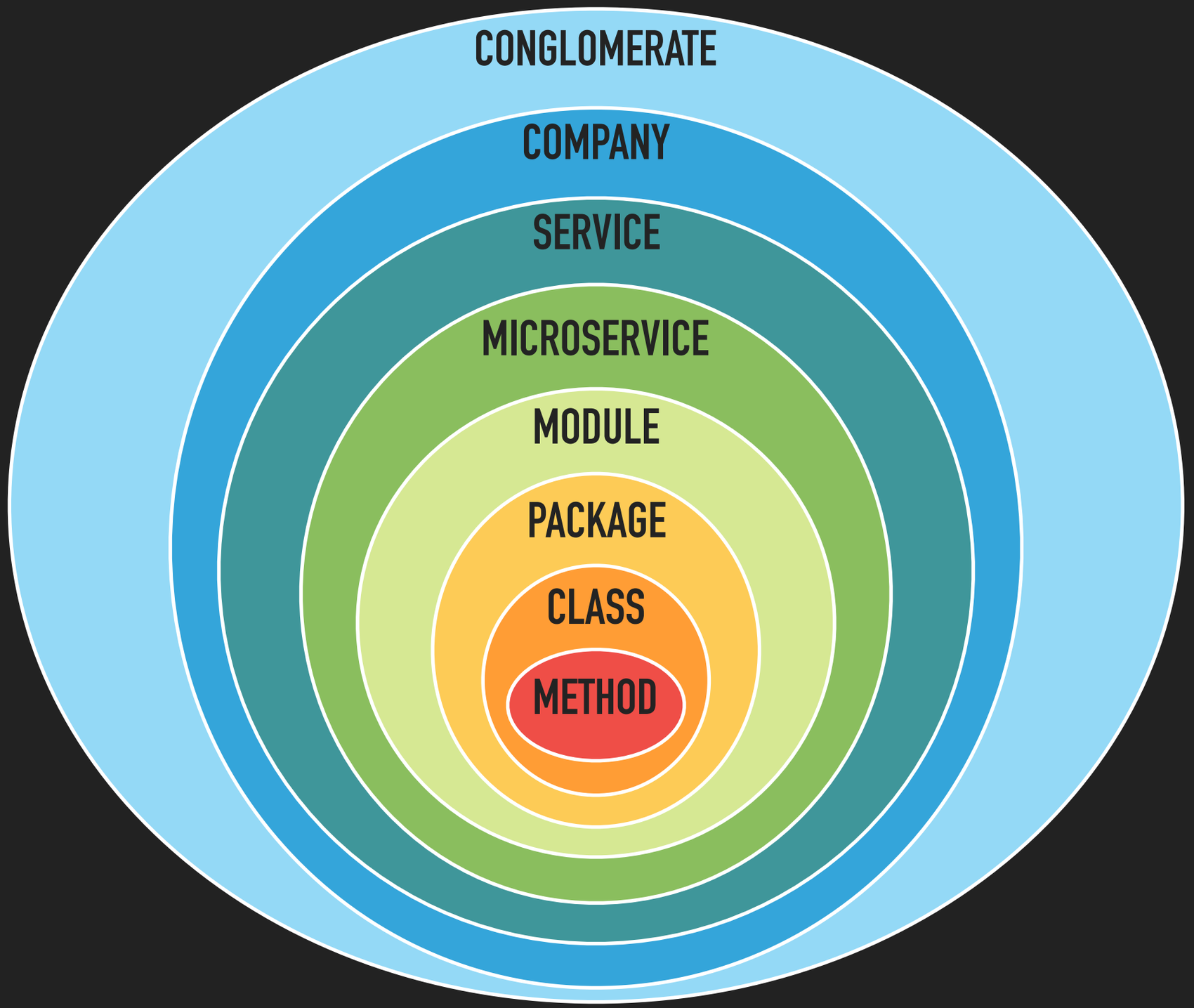


마이크로서비스 IN PRACTICE

MICRO SERVICE ARCHITECTURE

- Context boudary 를 잘 잡아야 (Domain-Driven Design)
  - 서비스 간 트랜잭션 롤백은 곤란
  - 물리적으로 분리된 서비스 간 통신 오버해드 최소화
- 서비스 간 순환 Depenency 는 피할 것
  - 독립적인 배포 사이클을 방해

높은 응집도(COHESION), 낮은 결합도(COUPLING)



TIP – 프로젝트 이름 정하기

- 고유한 프로젝트명과 직관적인 서비스명이 함께 필요

프로젝트명	서비스명
Thor	Mailbox service
Hammer	Mail sending service
Sif	Mail receiving service
Capt	UserInfo service
Loki	IMAP service
Mint	PC web app
Cinnamon	Web api gateway

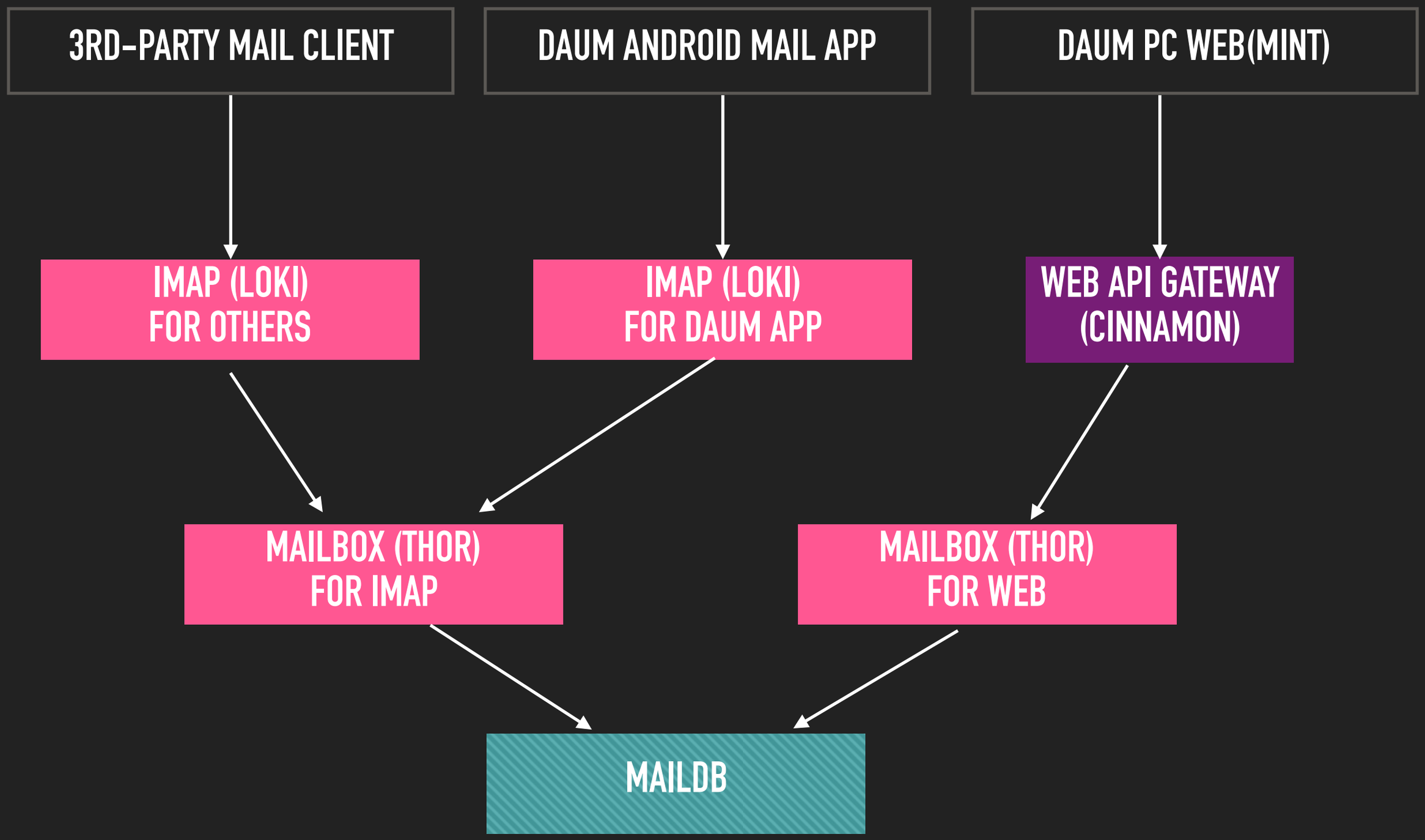
TIP - REST(JSON) VS THRIFT/PROTOBUF

- REST(Json) 채택
  - 손쉬운 디버깅, 배경이 다양한 개발자들의 학습 비용 낮음
    - XML-SOAP 의 안좋은 기억들...
  - 성능이 실제로 부족한 경우에 다른 방안을 고려하기로
    - 큰 바이너리 데이터는 POST multipart/form-data, GET raw data 활용
- Performance - Web gateway(Cinnamon) api response 기준
  - 120 request/sec * 50ea = 6000 request/sec
  - average processing time: 60ms
    - IDC 배치는 중요함
    - Cinnamon(목동IDC) - Thor-md1(목동IDC) - 가산IDC - Mail DB(일산/논현IDC) : 160ms
    - Cinnamon(목동IDC) - Thor-gs2(가산IDC) - Mail DB(일산/논현IDC) : 60ms

TIP - REST URL SCHEME

- http://beta-thor.daumkakao.io:8080/mailbox/v1/users/…
  - alpha-, beta-, inhouse- 등의 domain prefix 사용
  - 내부 api 임으로 80포트 일 필요는 없음
  - context-path 로 서비스명 사용
  - 하위 호환이 되지 않는 api 개선 시에는 버전(v1) 변경

TIP - 트래픽 격리


3RD-PARTY MAIL CLIENT
DAUM ANDROID MAIL APP
DAUM PC WEB(MINT)
IMAP (LOKI)
FOR OTHERS
IMAP (LOKI)
FOR DAUM APP
WEB API GATEWAY
(CINNAMON)
MAILBOX (THOR)
FOR IMAP
MAILBOX (THOR)
FOR WEB
MAILDB

폴리그랏 IN PRACTICE

POLYGLOT PROGRAMMING

- Java -> C# -> Scala?
- 다양한 언어를 문제에 따라 골라서 사용하는?
  - 오래된 Specialist vs Generalist 논쟁? -> T자형 인재(흔한 결론)
    - Full-stack developer?
- 한 가지의 언어로는 충족하기 어려운, 다양한 언어들의 기능성과 효율성을 모두 얻기 위함

FULL-STACK DEVELOPER

- Full-stack 개발자 유행의 한계
  - LAMP(Linux, Apache, Mysql, PHP) stack?
  - MEAN(MongoDB, Express.js, Angular.js, Node.js) stack?
- 최신의 경쟁력있는 서비스 개발을 위한 필요 기술들
  - AWS, Cloud
  - iOS, Android
  - NoSQL DB, Hadoop
  - RDBMS, ElasticSearch
  - Data analysis, Machine Learning
- 현재 시장은 각 분야의 Expert 를 원함

폴리그랏 시대의 개발자

- 다양한 언어에 능숙한 한 명의 개발자 vs 각 언어별 Expert 가 여러 명인 개발팀
- 개인
  - 자신의 주력 언어 하나를 마스터 하자
  - 다양한 언어를 장점을 학습하여 나의 주력 언어에 적용 연습
  - 주력 언어를 변경해야 되는 시장 상황이 되었을 때 주력 언어 갈아타기
- 팀
  - 해당 문제의 도메인에 효율적인 언어를 선택
  - 해당 언어의 Expert 를 영입 or 키우기
  - 폴리그랏 협업에 적합한 아키텍쳐를 설계하고 각 분야 Expert에게 오너쉽 부여
  - 문제에 더 적합한 언어 vs Expert 가 있는 언어

@DEX 는?

- 현재 주력 언어 : Java
- 과거(~2004) 주력 언어 : C
- 실무에 활용해 본 언어
  - Php, Shell, Perl, Ruby, Flash ActionScript, JavaScript, Python, R
- 기초 공부를 해본 언어
  - Basic, C++, C#, ASP, JSP, Object-C, Scala, Clojure, Lisp
- 미래에 관심 있는 언어
  - Go, Kotlin, Swift

메일 시스템의 폴리그랏

- 다양한 레거시 인프라에 접근, 엄격한 모델 관리
  - Java 같은 static type 언어
- Json 모델 변환
  - JavaScript (Node.js)
- 보다 강력한 추상화와 부작용 최소화
  - 함수형 프로그래밍 (최후의 언어 Clojure)
- 스팸 검사와 같은 다양한 텍스트 필터 필요
  - Unmatched power for text processing and scripting (Perl)

Q & A