사랑이 넘치는 ♥

최종발표 : 밋앤그릿 ♥

아형 성태 지원 민정 형린 아영 지민

목차

기획 의도 1 : 여성향 연애시뮬레이션 유저 특성 고려

20~30대 여성

그 외에도

- 2차 창작러

- 트위터, 커뮤니티 유저 등등

<수집욕> “repeatable” 유저의 반복적 플레이와 직결

= 반복적 매출 가능

엔딩이… 10개…? 다 가지고 만다… ㅎㅎ

그게 아무리 오래 걸리더라도…

아무리 고생스럽더라도.. ^-^!!

기획 의도 2 : 유저의 플레이 편의성 극대화

인증

1. 기존 (구글, 페이스 북) 계정과의 연동을 통해 유저의 게임 시작을 쉽고 빠르게!
2. 스토리게임의 특성, 수집욕 배려

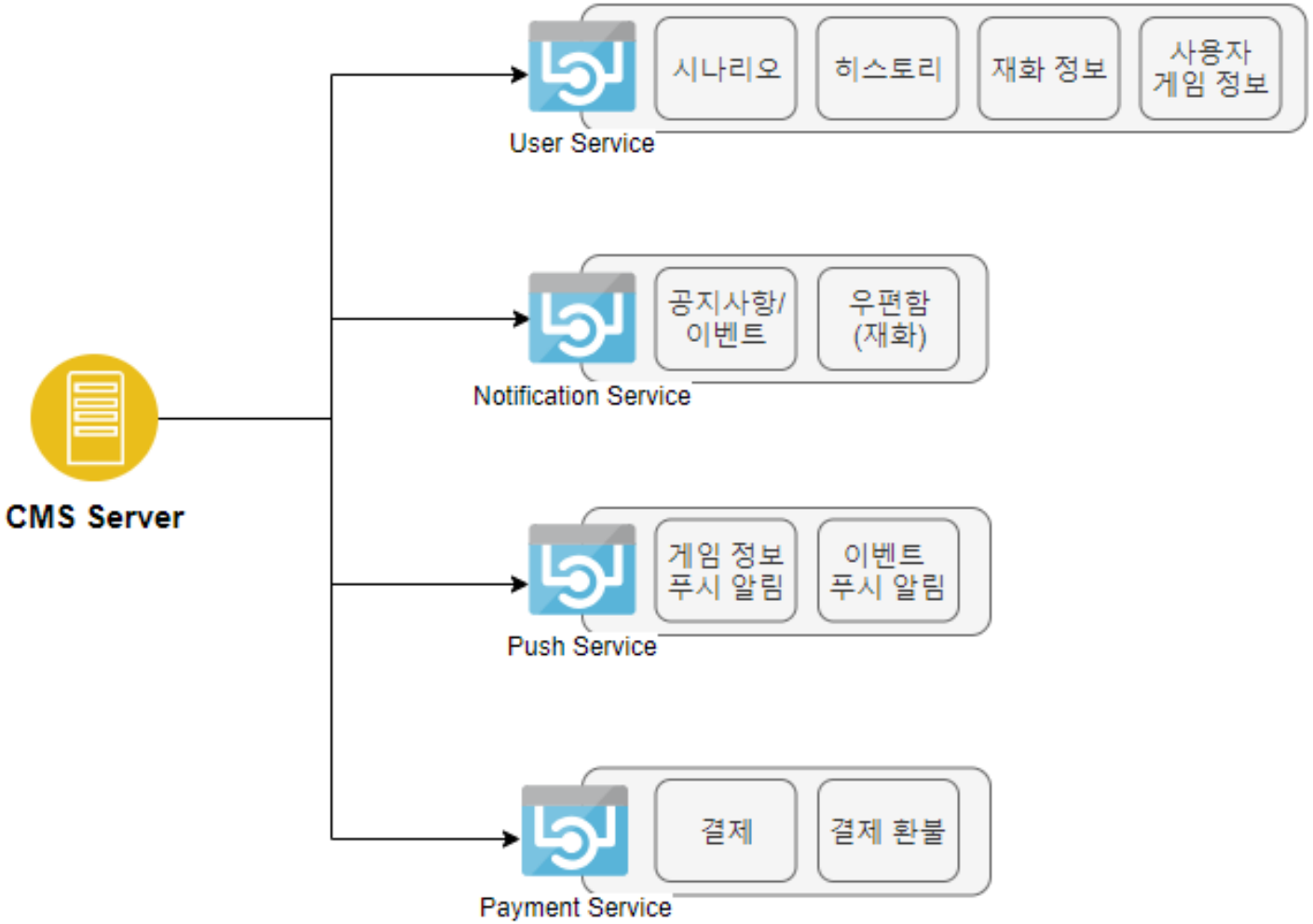
패치

1. 수정과 업데이트가 진행되어도 최소한의 시간만 소모되도록!
-> UX 향상
-> 유저의 이탈 방지

CMS

1. 푸쉬와 우편함 시스템을 통한 유저 잔존율(retention) 향상
2. 시나리오 저장을 통한 유저의 재플레이 유도

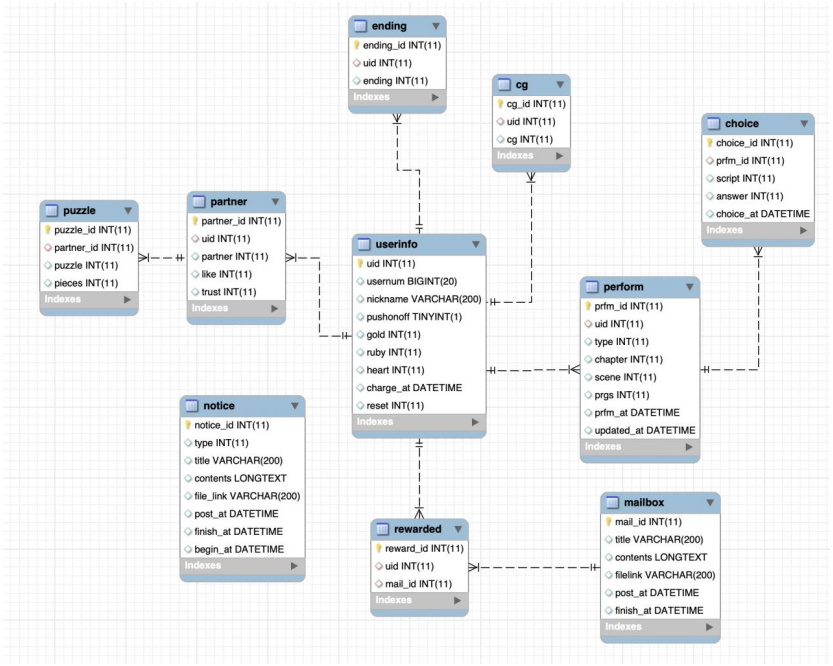
CMS Server 초기 구현 목표



1. CMS 서버

CMS Server 구현 준비 (1~2주)

번호	대메뉴	중메뉴	기능	설명
1	타이틀	로그인 화면	회원가입	1. 페이스북, 구글 인증을 통한 회원가입 허용
				2. 게스트 로그인
				3. 일반 회원가입
				속성 : 이메일, 비밀번호, 비밀번호 확인
2			로그인	로그인, 회원가입 버튼
				페이스북, 구글, 이메일(+비밀번호) 로그인
2-1			아이디 찾기	속성 : 실명, 이메일을 통한 인증
2-2			비밀번호 찾기	속성 : 실명, 아이디, 이메일을 통한 인증
3	타이틀		공지 팝업	ex) 점검 완료, 점검 진행 일정 등
touch to start				
4	메인UI	공지 팝업		ex) 운영자 보상 재화, 이벤트 알림, 출석체크
				속성 : 오늘 더 이상 보지 않기
		비주얼노벨	메인 스토리	(-) 스테미너
				(+) 게임 골드
			이벤트 스토리	(-) 게임 골드
				(+) 히든 조각
			히든 스토리	(-) 히든 조각 8개
				(+) 히든 스토리
			시나리오 저장	상황1 : 최초 플레이 시 비주얼 노벨 스테이지를 완료하지 않고 유저가 나가려고 할 때, 그 부분까지 시 상황2 : 유저가 클리어한 스테이지를 다시 플레이 할 때, 이전에 선택지 스토리대로 감상만 가능
		히스토리 클라우드		모든 플레이에서 획득한 CG이미지 누적 측적
		휴대폰	앨범	현재 플레이에서 획득한 CG이미지 누적 측적
			카톡	특정 스테이지 이후 오는 카톡 보상
			전화	특정 스테이지 이후 오는 전화 보상
		이벤트	전화	특정 시간에만 열리는 보상
			전화/ 카톡/ 이야기	특정 시간에만 열리는 보상
		설정	환경설정	푸시 기능 on/off
			리뷰 쓰기	구글/ 앱스토어로 연결
			공식 사이트 연결	공식 트위터/ 공식 블로그 / 공식 카페/ 홈페이지로 연결



1. 기획 파악

2. CMS 기능 정리

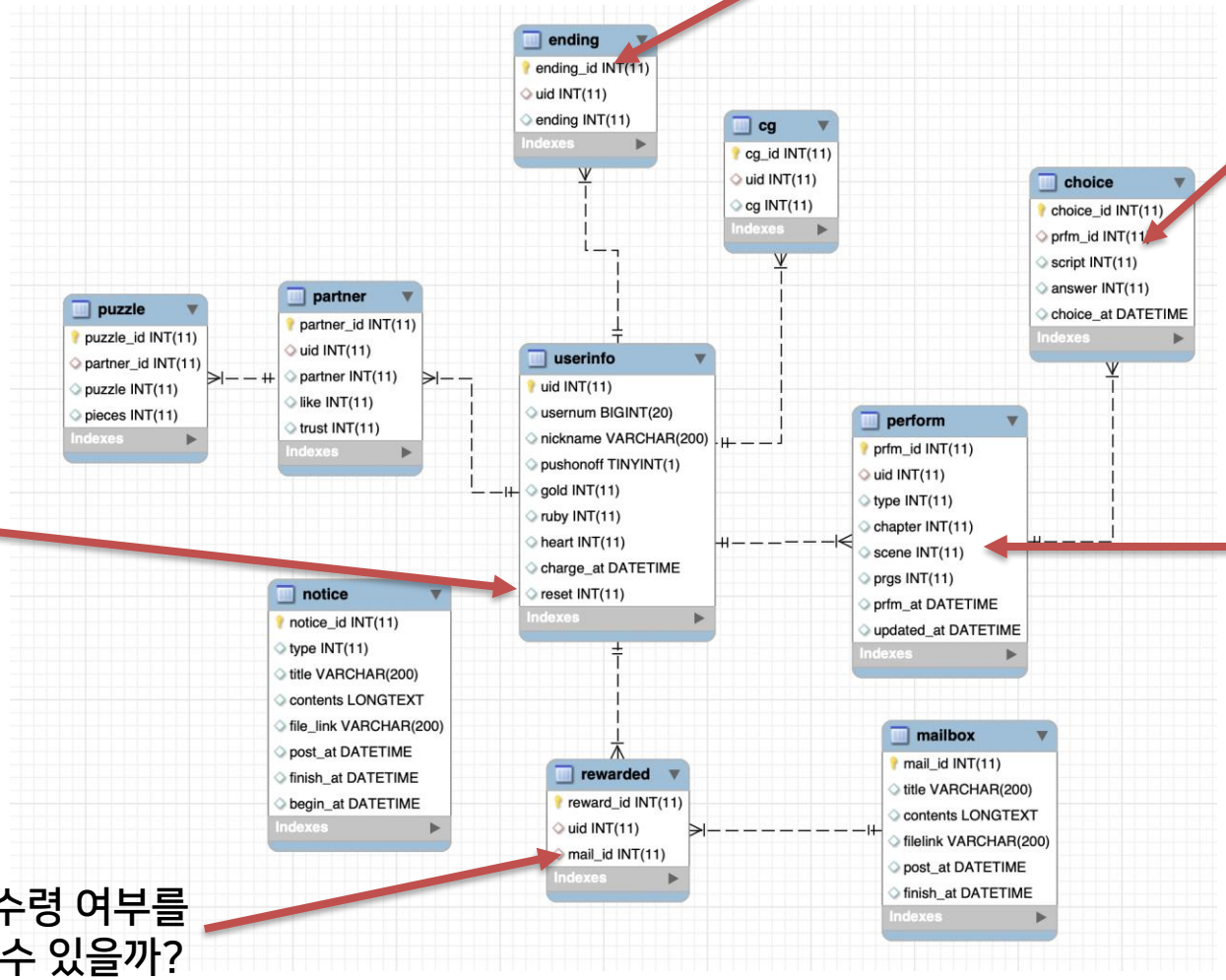
3. DB 설계

CMS Server 구현 준비 (1~2주) + 3주

DB 재설계
+ 재구현

엔딩 정보는
어디까지 저장해야?

게임 진행 정보는
어디까지 저장해야하지?
진행 상황? 선택지?
시간까지?



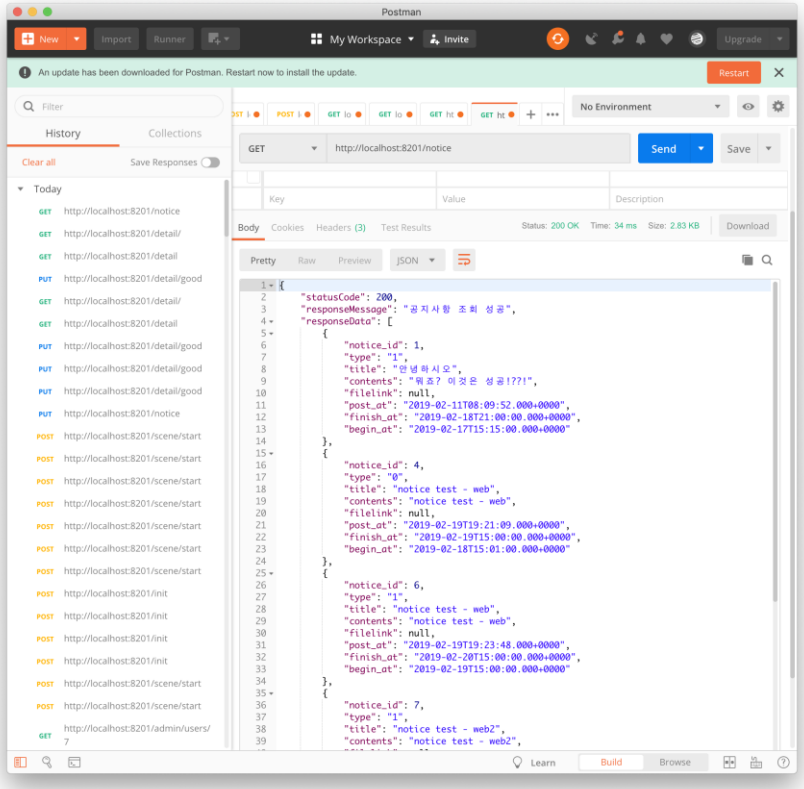
리셋 정보에 따라
usenum과 uid를..?

시나리오 정보는
마지막만 저장해도 되나?
시간이 필요할까?

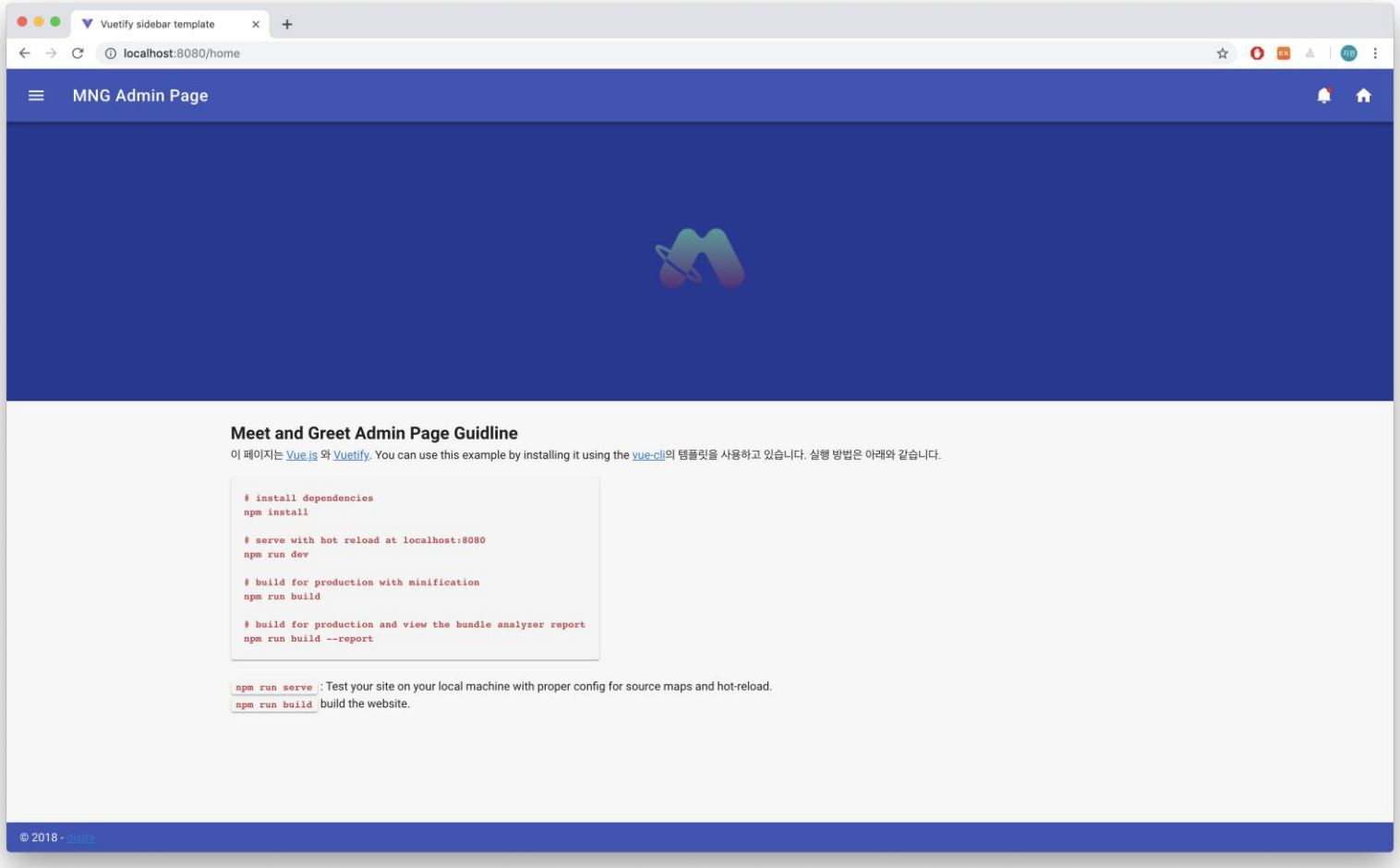
우편함의 수령 여부를
어떻게 알 수 있을까?

CMS Server 중간 결과

서비스는 구현하였으나 결과적으로는 보여줄 수 있는 클라이언트가 없다!
클라이언트가 없는 CMS는 무용지물이 된다.



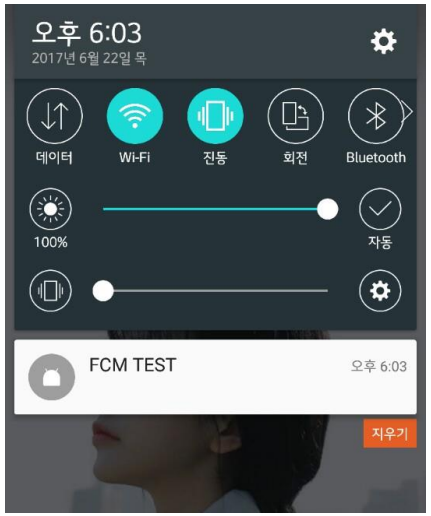
CMS Server 보수 (4~5주)



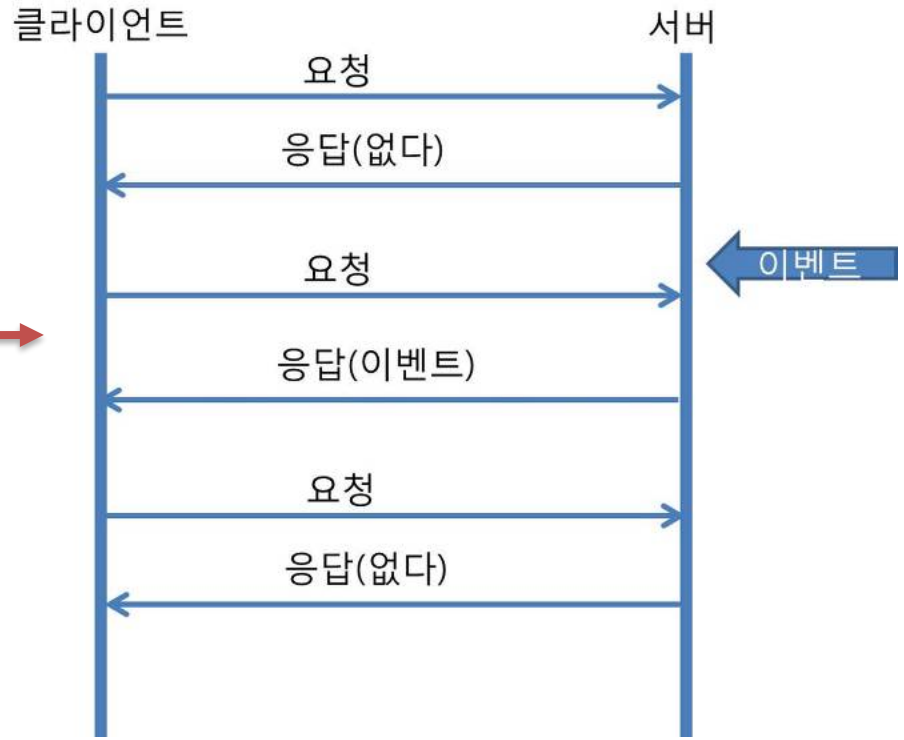
관리자 페이지 생성으로 실제 클라이언트와의 통신 테스트

CMS Server 보수 (4~5주)

Firebase

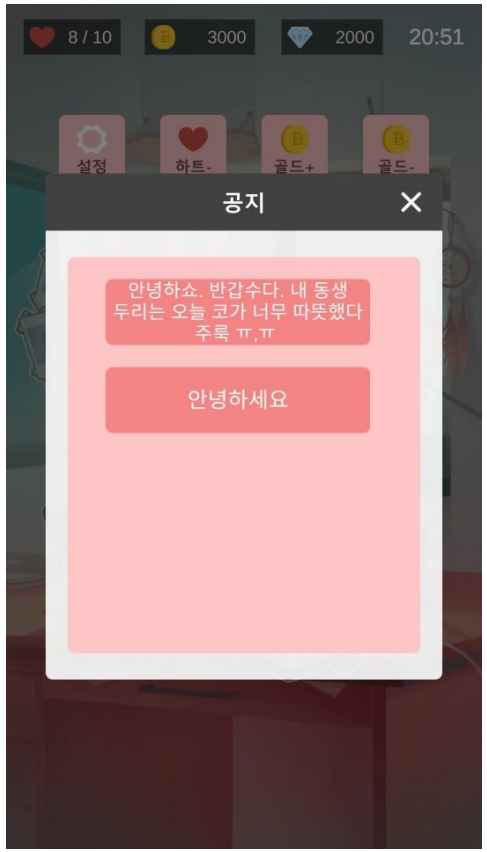


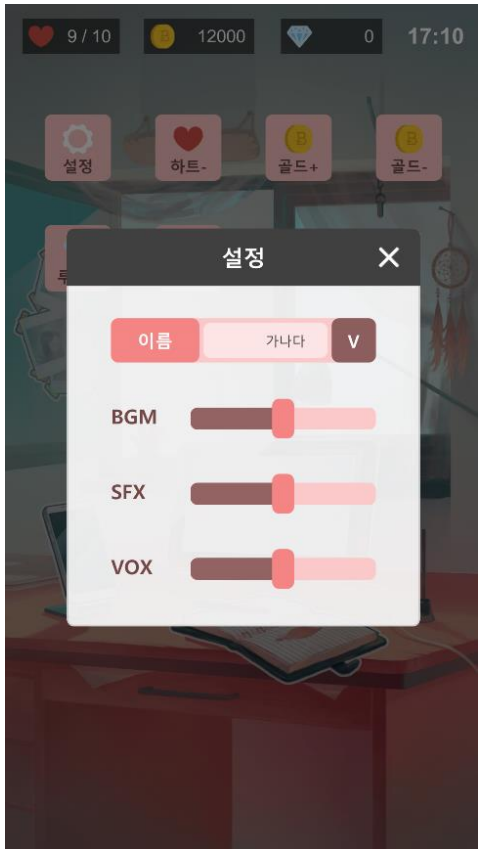
대체

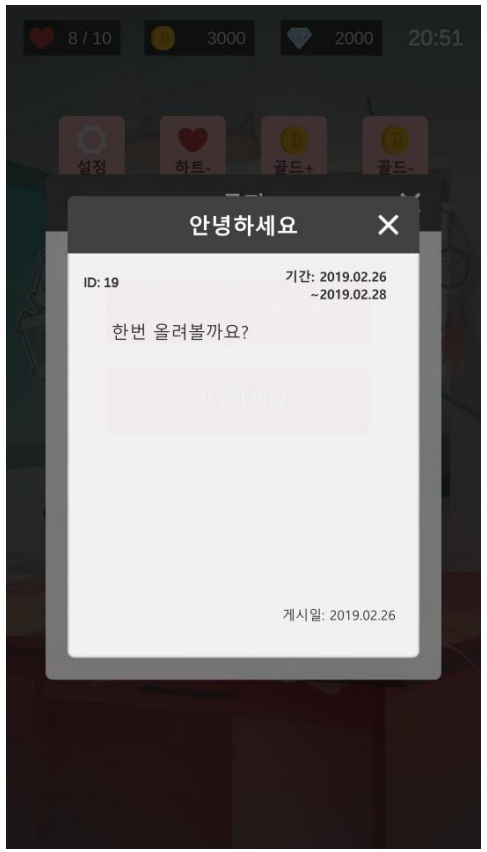


앱에서 웹으로 클라이언트가 바뀌면서 푸시알림 방식을 바꿈
FCM -> Polling 으로 웹에 푸시알림이 가는 것처럼!

CMS Server 결과 (6주)

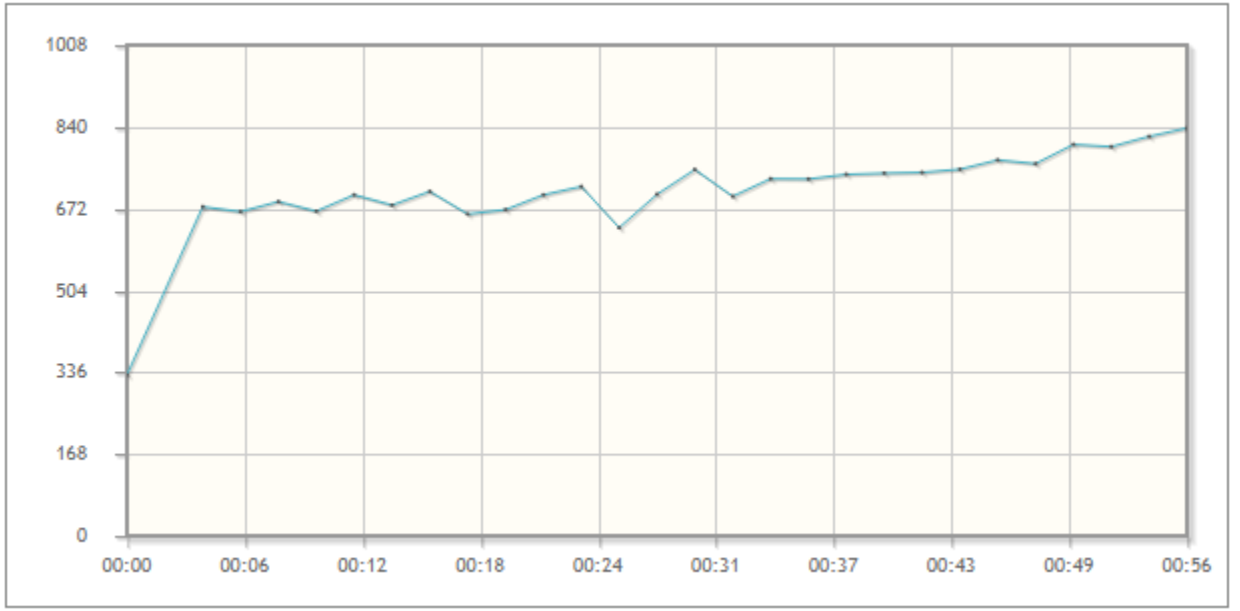






유니티 클라이언트와의 통신도 성공하였다.

CMS Server 결과 (6주)



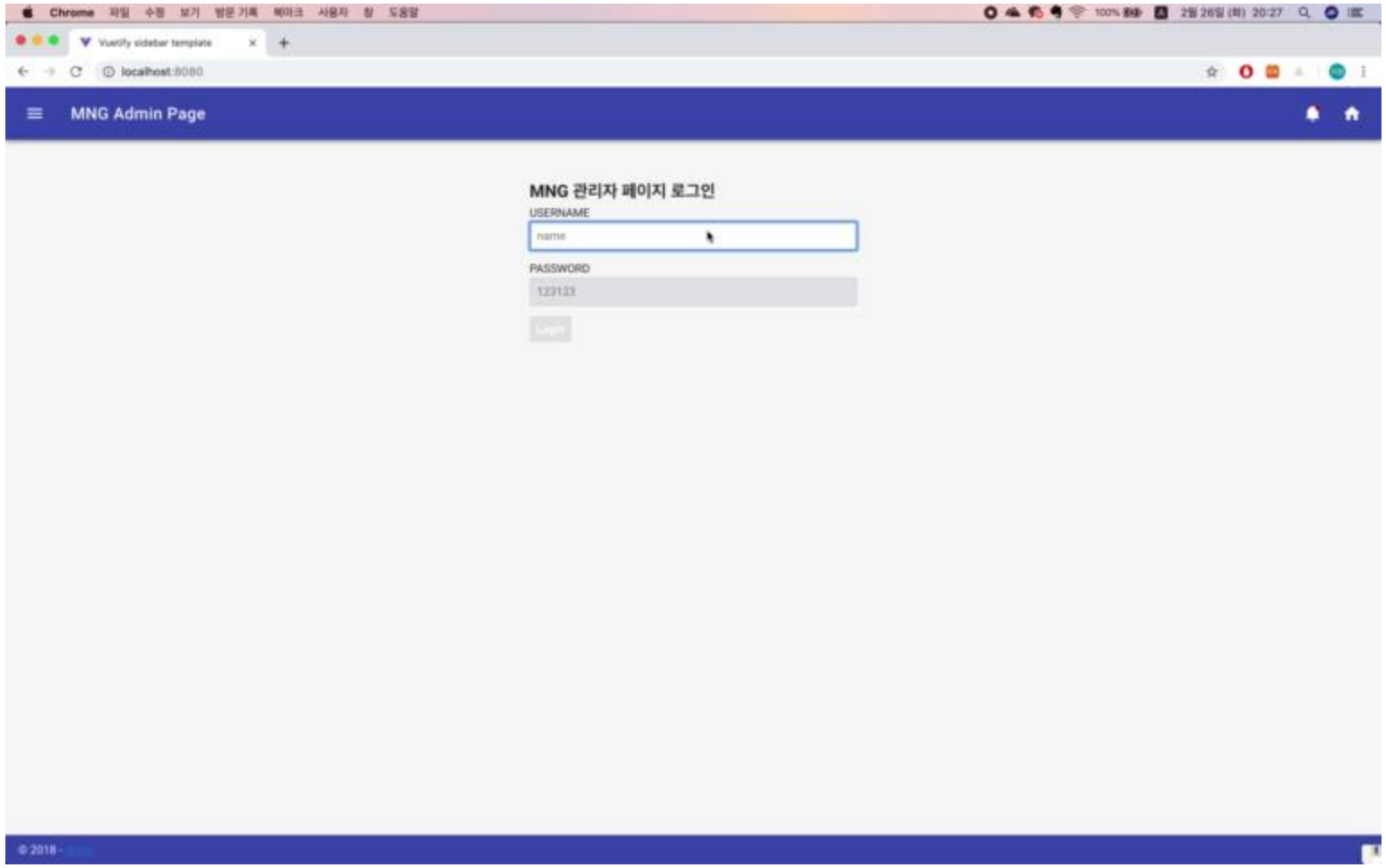
CMS Server의 TPS 그래프!

….이하 생략….

1. CMS 서버

시연 영상

시연 영상


MNG Admin Page
MNG 관리자 페이지 로그인
USERNAME
name
PASSWORD
123123
Login
© 2018 -

1. 시작 1~2주차, 인증 서버…? 그거 그냥 하면 되는 거 아니야?!

1. 시작 1~2주차, 인증 서버…? 그거 그냥 하면 되는 거 아니야?!

오잉…?

2. Membership Server

2. 시작 3~6 주차 개발 시작!!

JWT : 토큰 자체 데이터 저장이 가능하여 Rest API에 적합

Client

Server

ETC

Spring boot

Spring security

인증, 인가 기능을 Spring Security에 위임

JPA
Java Persistence API
HIBERNATE

JWT Token 값 저장

서버가 토큰 상태를 가지고 있지 않아, 강제로 만료 시킬 방법이 없어 접근 속도가 빠른 redis에 저장

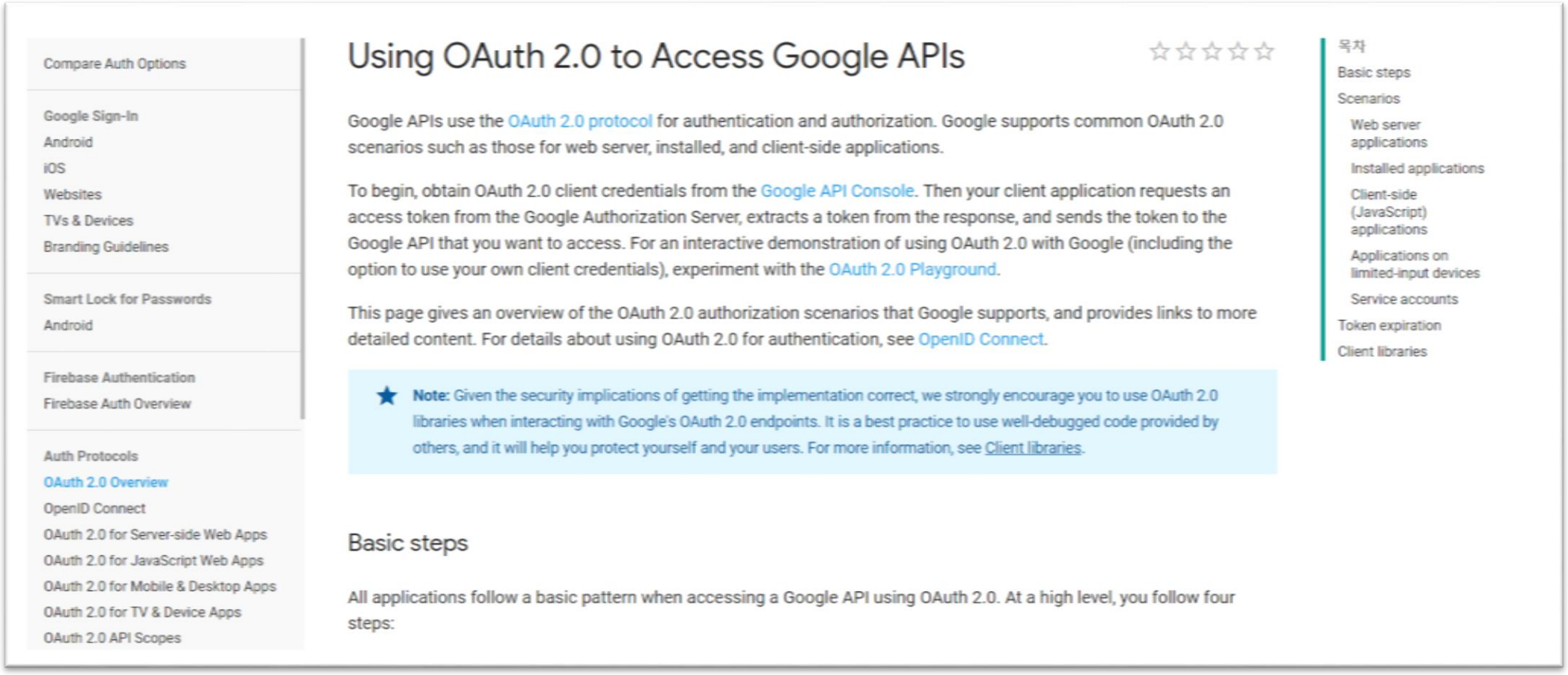
User 계정 정보 저장

User 계정 정보 저장

3. 위기 잘못된 소셜 로그인 방식

RESTful 하지 못한 소셜 로그인 방식으로 개발 진행

4. 해소 공식 문서 정독.. 정독..



Compare Auth Options

Google Sign-In
Android
iOS
Websites
TVs & Devices
Branding Guidelines

Smart Lock for Passwords
Android

Firebase Authentication
Firebase Auth Overview

Auth Protocols
OAuth 2.0 Overview
OpenID Connect
OAuth 2.0 for Server-side Web Apps
OAuth 2.0 for JavaScript Web Apps
OAuth 2.0 for Mobile & Desktop Apps
OAuth 2.0 for TV & Device Apps
OAuth 2.0 API Scopes

Using OAuth 2.0 to Access Google APIs

☆☆☆☆☆

Google APIs use the OAuth 2.0 protocol for authentication and authorization. Google supports common OAuth 2.0 scenarios such as those for web server, installed, and client-side applications.

To begin, obtain OAuth 2.0 client credentials from the Google API Console. Then your client application requests an access token from the Google Authorization Server, extracts a token from the response, and sends the token to the Google API that you want to access. For an interactive demonstration of using OAuth 2.0 with Google (including the option to use your own client credentials), experiment with the OAuth 2.0 Playground.

This page gives an overview of the OAuth 2.0 authorization scenarios that Google supports, and provides links to more detailed content. For details about using OAuth 2.0 for authentication, see OpenID Connect.

★ **Note:** Given the security implications of getting the implementation correct, we strongly encourage you to use OAuth 2.0 libraries when interacting with Google's OAuth 2.0 endpoints. It is a best practice to use well-debugged code provided by others, and it will help you protect yourself and your users. For more information, see Client libraries.

Basic steps

All applications follow a basic pattern when accessing a Google API using OAuth 2.0. At a high level, you follow four steps:

목차
Basic steps
Scenarios
Web server applications
Installed applications
Client-side (JavaScript) applications
Applications on limited-input devices
Service accounts
Token expiration
Client libraries

문제 해결!

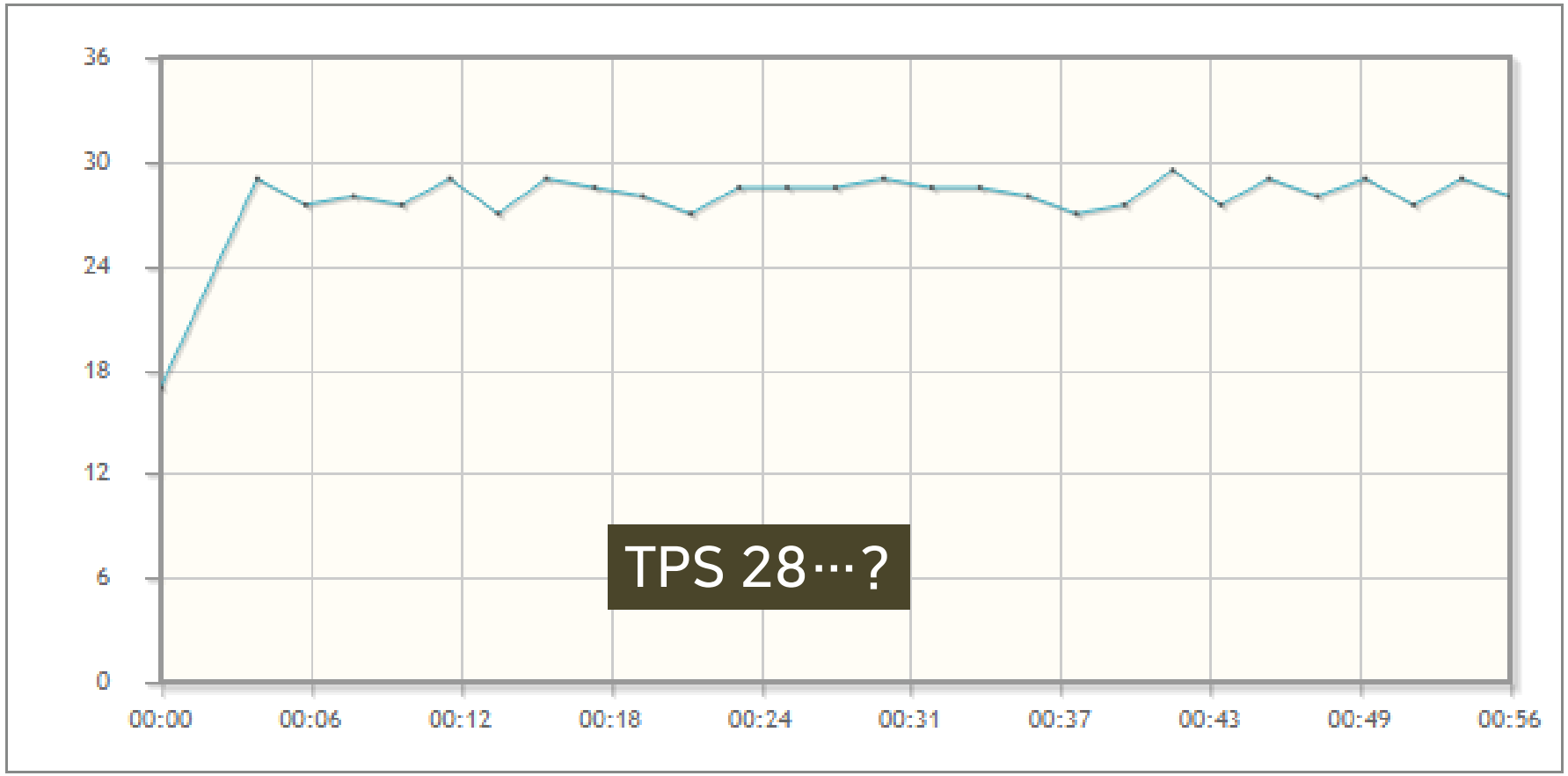
2. Membership Server

끝!

2. Membership Server

인줄만 알았는데..

5. 또다른 위기 봉착 마지막 테스팅…

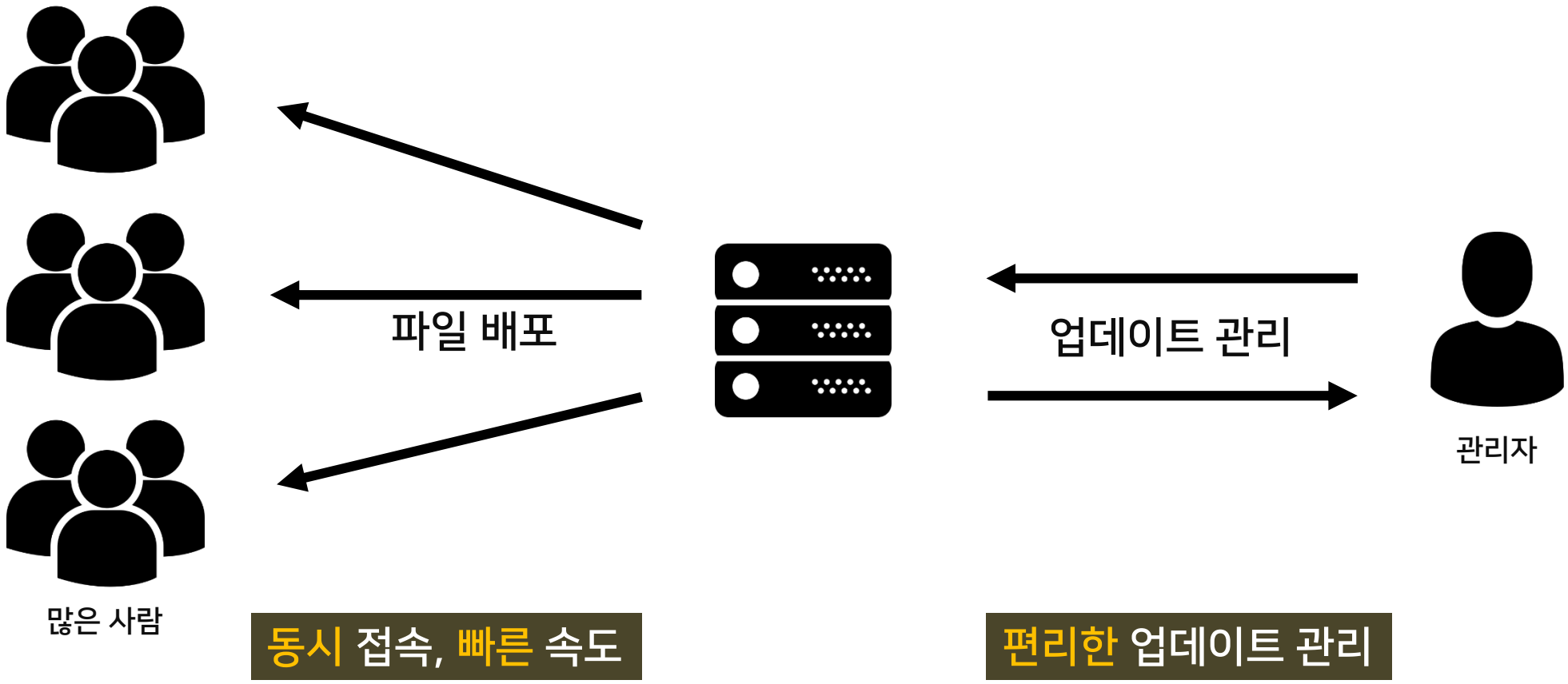


To Be Continue..?

2. Membership Server

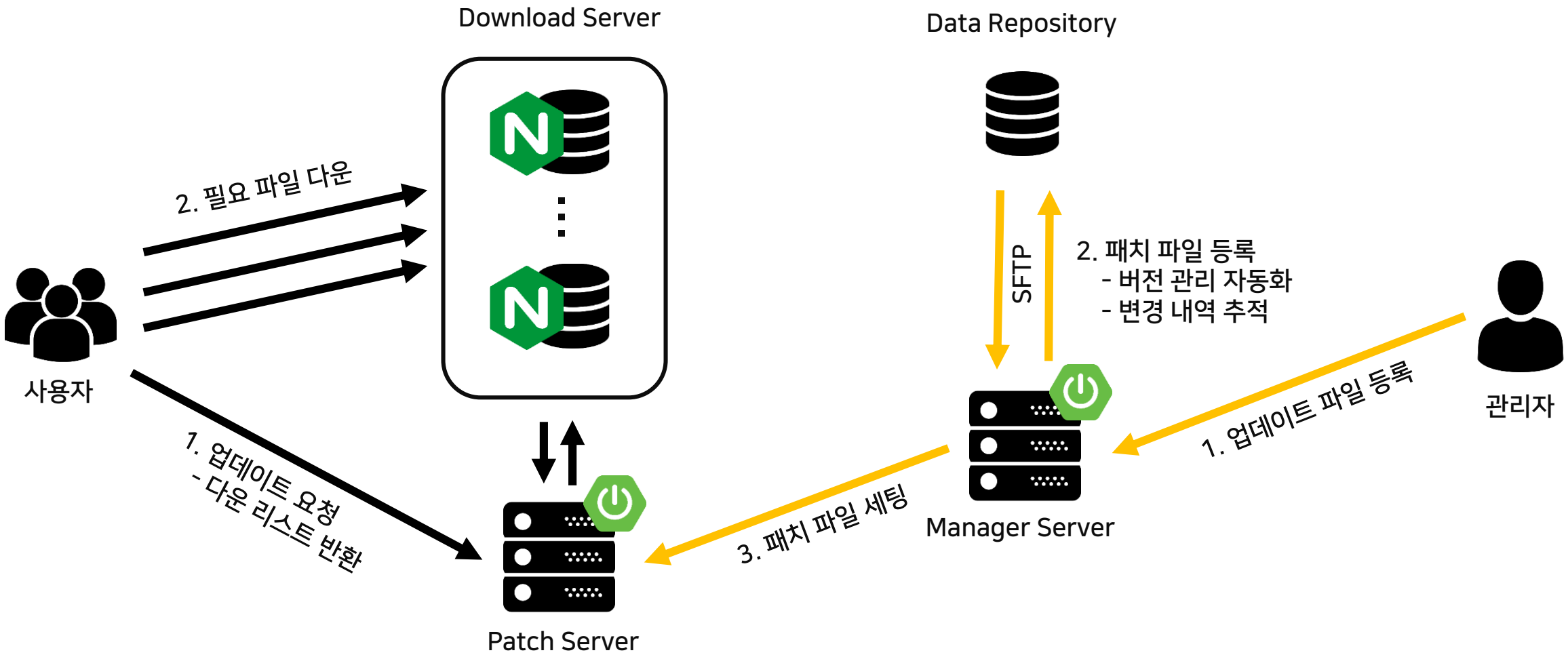
시연 영상

1. 시작 1~5주차, 우리도 실제 패치 업로드 서비스 만들 수 있어요



1. 시작 어떻게 하면 사용자들이 빠르게 업데이트를 다운 할까요

쪼개야 사는 마이크로 서비스



1. 시작 어떻게 하면 사용자들이 빠르게 업데이트를 다운 할까요

원하는 것만 받는 부분 패치 시스템

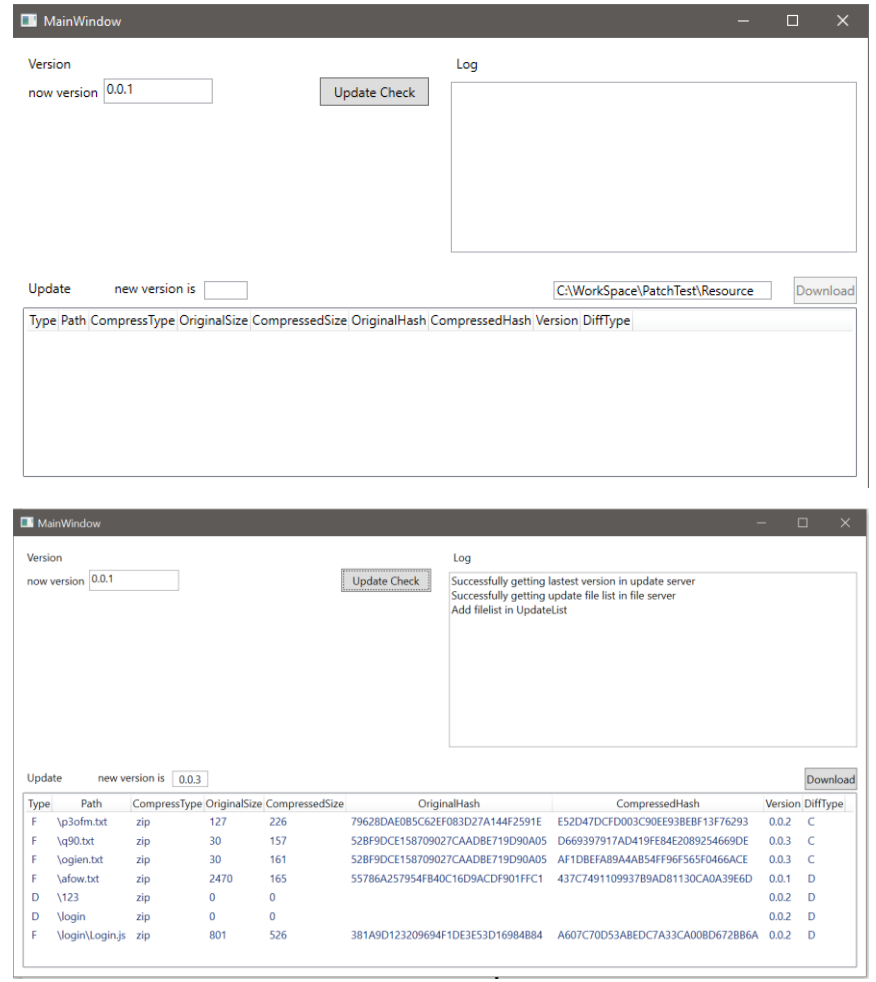
사용자가 자신의 버전 서버에 전달

Version : v1

사용자 별 다운 패치 목록 생성

v1_v3_Patch.json

사용자는 필요 자원만 다운로드



<Client 패치 파일 다운 테스트 프로그램(C# WPF)>

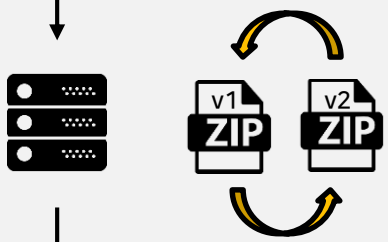
3. Patch 서버

1. 시작 어떻게 하면 관리자들이 편리하게 업데이트를 관리 할까요

Git보다는 못하지만 버전 시스템

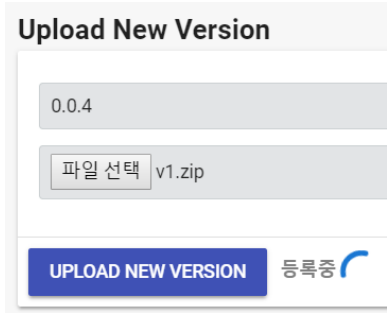
관리자가 최신 파일과 버전명 업로드

이전 버전 파일과 최신 버전 파일 비교



이전 버전과 변경 내역 출력

Patch_Ver_v2.json



Version List

version	full	patch
0.0.1	log/full/Full_Ver_0.0.1.json	log/patch/Patch_Ver_0.0.1.json
0.0.2	log/full/Full_Ver_0.0.2.json	log/patch/Patch_Ver_0.0.2.json
0.0.3	log/full/Full_Ver_0.0.3.json	log/patch/Patch_Ver_0.0.3.json

Rows per page: 5 1-3 of 3

```
[
  "F | \\afow.txt | zip | 2470 | 165 | 55786A257954FB40C16D9ACDF901FFC1 | 437C7491109937B9AD81130CA0A39E6D | v1 | D",
  "D | \\i4on | zip | v2 | C",
  "F | \\p3ofm.txt | zip | 127 | 96 | 79628DAE0B5C62EF083D27A144F2591E | E52D47DCFD003C90EE93BEBF13F76293 | v2 | C"
]
```

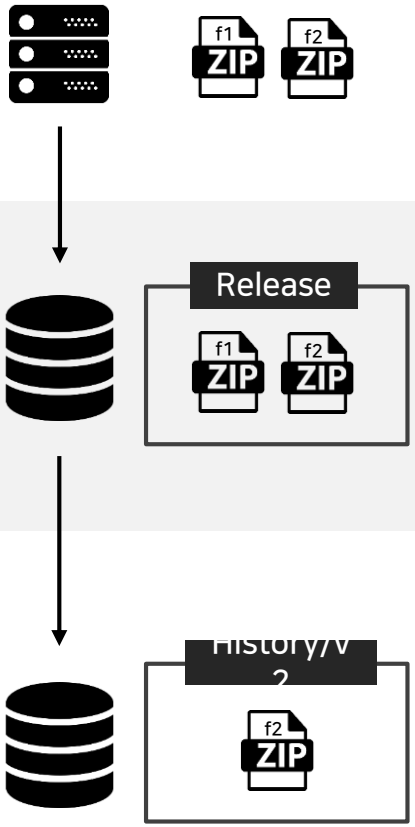
1. 시작 어떻게 하면 관리자들이 편리하게 업데이트를 관리 할까요

다 해주는 배포 시스템

용량이 작게 파일 압축

Release 버전 업로드

버전별 Patch 파일 업로드




```
#ftp directory
 └ masta
    ├ file
    |  ├ backup
    |  |  └ [full files]
    |  ├ history
    |  |  ├ 0.0.1
    |  |  |  └ [patch files]
    |  |  ├ 0.0.2
    |  |  |  └ [patch files]
    |  |  └ 0.0.3
    |  |     └ [patch files]
    |  └ release
    |     └ [full files]
    └ log
       ├ full
       |  └ [json files : FULL_VER_0.0.1.json]
       └ patch
          └ [json files : PATCH_VER_0.0.1.json]
```

변경 파일(.zip) 저장

버전별 변경 내역저장

<패치 파일 서버 디렉토리>

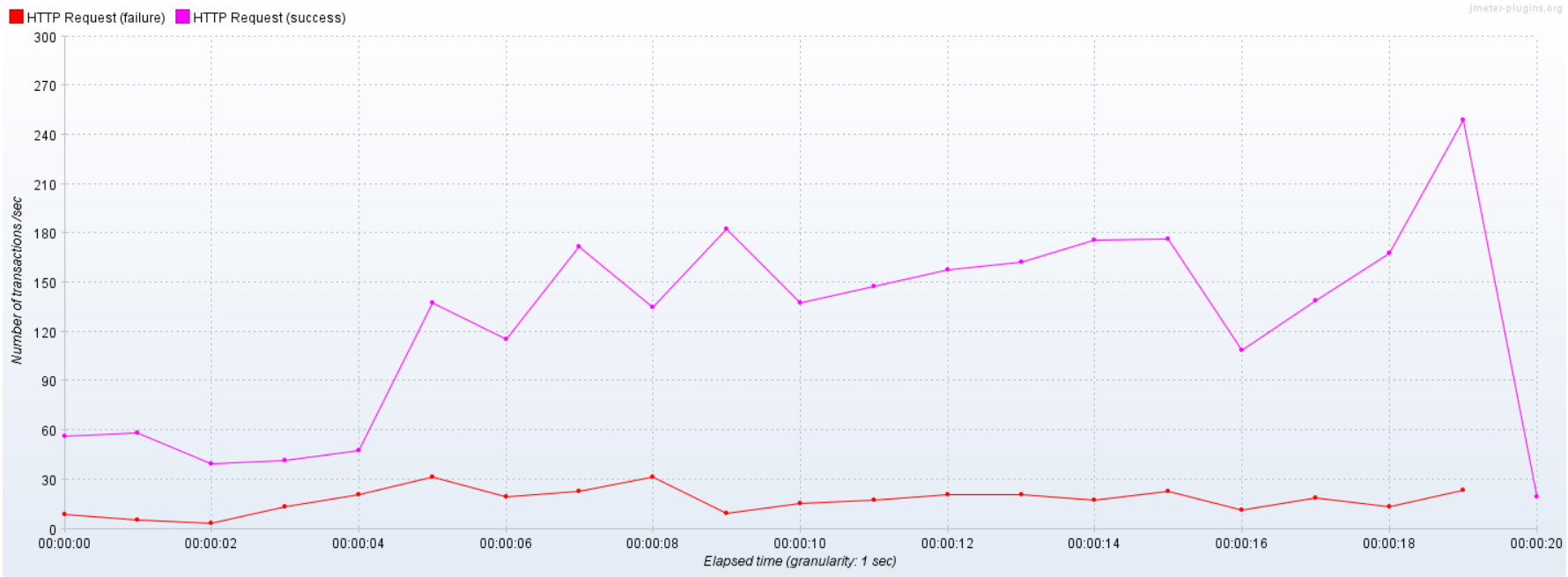
2. 점검 6주차, 우리 서버는 완벽해 터지지 않아요

로직별 예외 처리

```
try {
    if (!checkFileType(newVersionFile,  fileType: "zip")) {
        return DefaultRes.res(StatusCode.NOT_FORMAT, ResponseMessage.NOT_ZIP_FILE);
    }

    String checkRightVersionResult = versionService.checkRightVersion(newVersion);
    if (checkRightVersionResult != ResponseMessage.SUCCESS_TO_GET_LATEST_VERSION) {
        return DefaultRes.res(StatusCode.NOT_FORMAT, checkRightVersionResult);
    }

     String makePatchResult = preparePatchService.makePatch(newVersionFile, newVersion);
    if (makePatchResult != ResponseMessage.SUCCESS_TO_SAVE_JSON_IN_LOCAL) {
        return DefaultRes.res(StatusCode.INTERNAL_SERVER_ERROR, checkRightVersionResult);
    }

    String uploadPatchResult = uploadPatchService.uploadPatch(newVersion);
    if (uploadPatchResult != ResponseMessage.SUCCESS_TO_UPLOAD_PATCH) {
        return DefaultRes.res(StatusCode.INTERNAL_SERVER_ERROR, uploadPatchResult);
    }

    return DefaultRes.res(StatusCode.OK, ResponseMessage.SUCCESS_TO_NEW_VERSION);
} catch (Exception e) {
    TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();
    log.error(e.getMessage());
    uploadPatchService.initSftpServer(newVersion);
    return DefaultRes.res(StatusCode.INTERNAL_SERVER_ERROR, ResponseMessage.DB_ERROR);
}
```

처음부터 소스코드 정리

3. 결과 7주차, 인생은 실전이야.



결과값 정확성 57%

서버 에러 12%

4. 수습 8주차, 기존 프로젝트에서 배운 상식이 통하지 않는 백앤드 서버

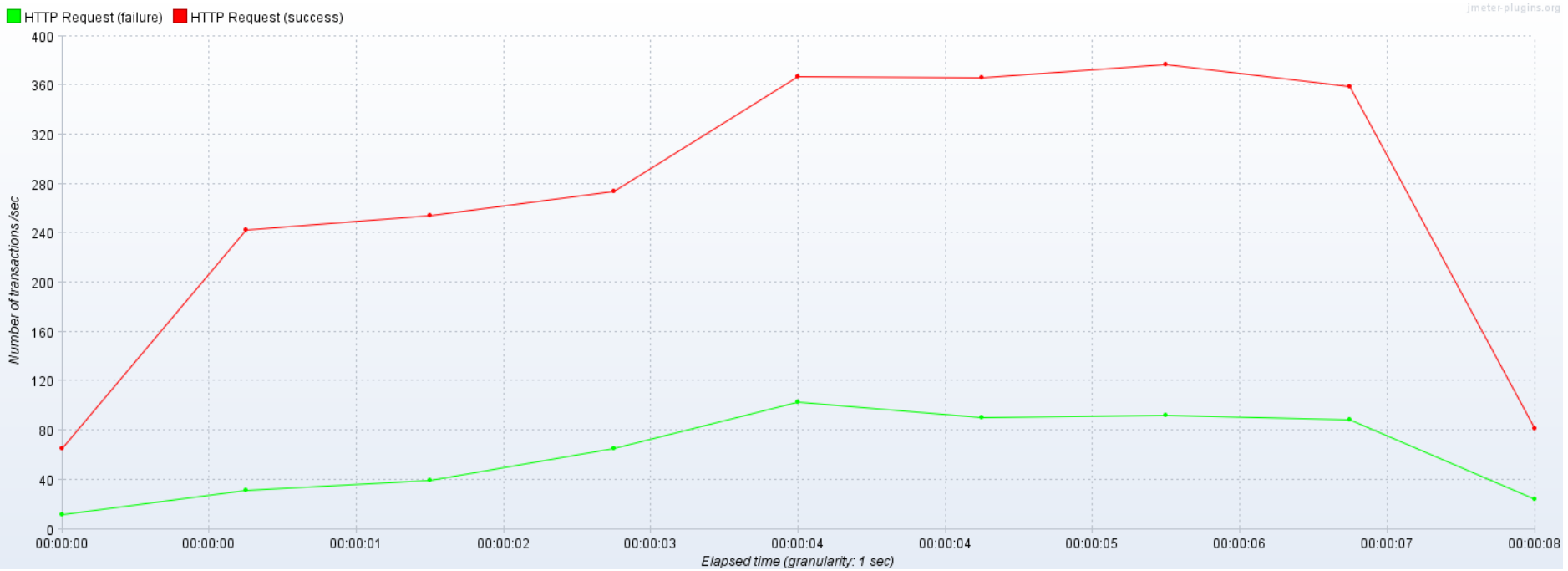
메모리 낭비가 아닌 **메모리 상주**

```
in = new BufferedReader(new InputStreamReader

String inputLine;

while ((inputLine = in.readLine()) != null) {
    content.append(inputLine);
}

in.close();
```

오래 놔두면 안되는 **커넥션 유지**

```
sftpServer.init();

File fullJsonFile = new File( pathname: localPath +
File patchJsonFile = new File( pathname: localPath

sftpServer.upload(fullJsonFile, dir: "log/full");
sftpServer.upload(patchJsonFile, dir: "log/patch");

remoteFullJsonPath = sftpServer.checkFile(newVersi
remotePatchJsonPath = sftpServer.checkFile(newVers

sftpServer.disconnect();
```

4. 수습 8주차, 발상의 전환 후에 결과



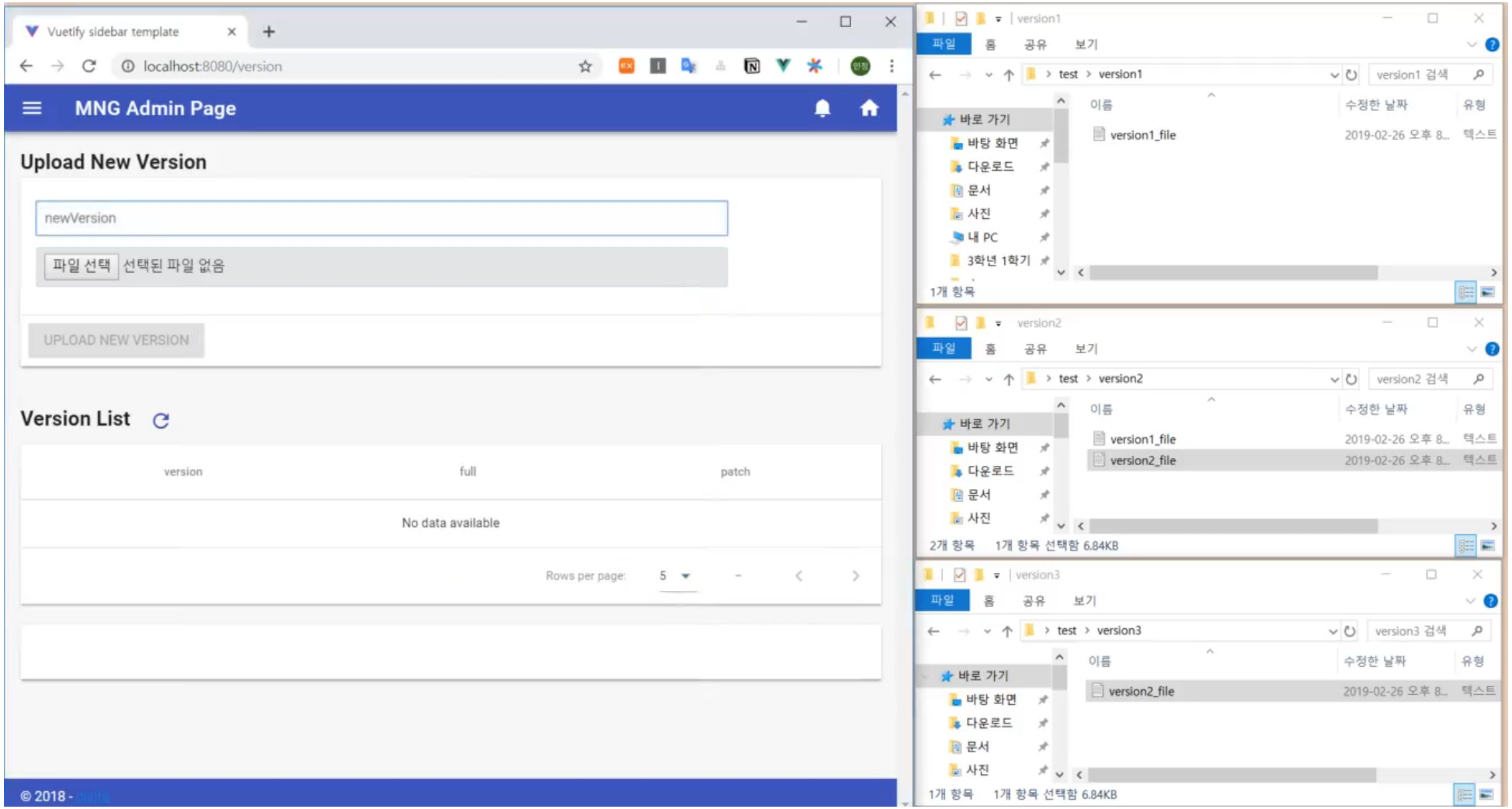
결과값 정확성 100%

서버 에러 8%

테스트를 조금 더 빨리 했으면 좋았을 뻔했다..

3. Patch 서버

시연 영상



4. 후기

개발 PM이 있었다면…

팀마다 담당 멘토가 있었다면…

테스트를 빨리 해 볼걸!

여기서 밤 새고 싶었는데…

실제 서비스 개발 및 운영 체험

다양한 직군과의 협업

맛있는 밥 ♥ 맛있는 간식 ♥

감사합니다 ❤️

(이어지는 두근두근 피드백)