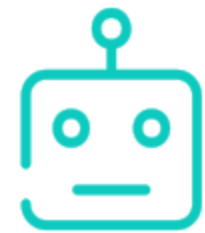


by code receipe



**BIBOT**



## 목차

01 기획배경

02 프로젝트 구성

03 프로젝트 진행

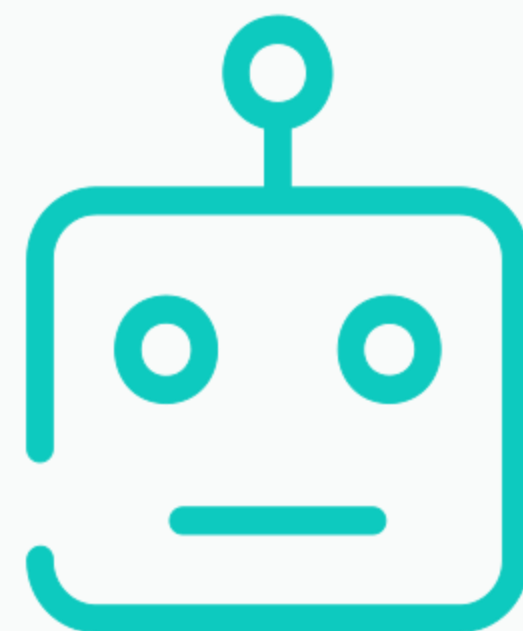
04 시연

05 후기

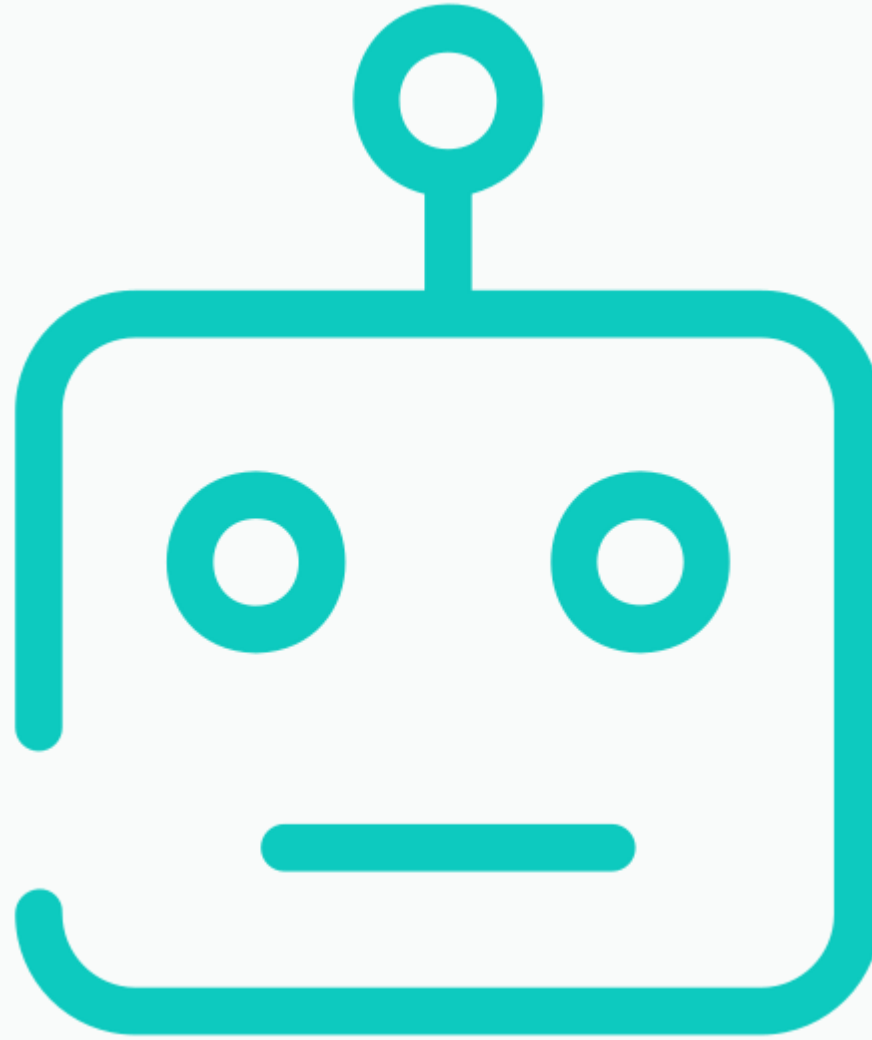
06 팀 소개

# 1. 기획 배경

---



## 서비스 소개

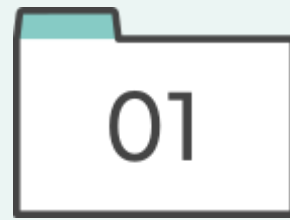


### 영수증 처리 자동화 서비스 BiBot

Bill와 Biz을 뜻하는 Bi와  
Robot을 뜻하는 Bot을 합친  
영수증 비즈니스 로봇 BiBot

## 서비스 기획 의도

인력의 효율적인 활용과 작업 시간 단축을 위한 자동 경비 처리  
OCR을 이용한 경비 처리 자동화 RPA 시스템



RPA

B2B RPA 서비스를  
기획 및 개발 함으로써  
실무 기획 능력 및  
비즈니스 역량 향상



Micro Service

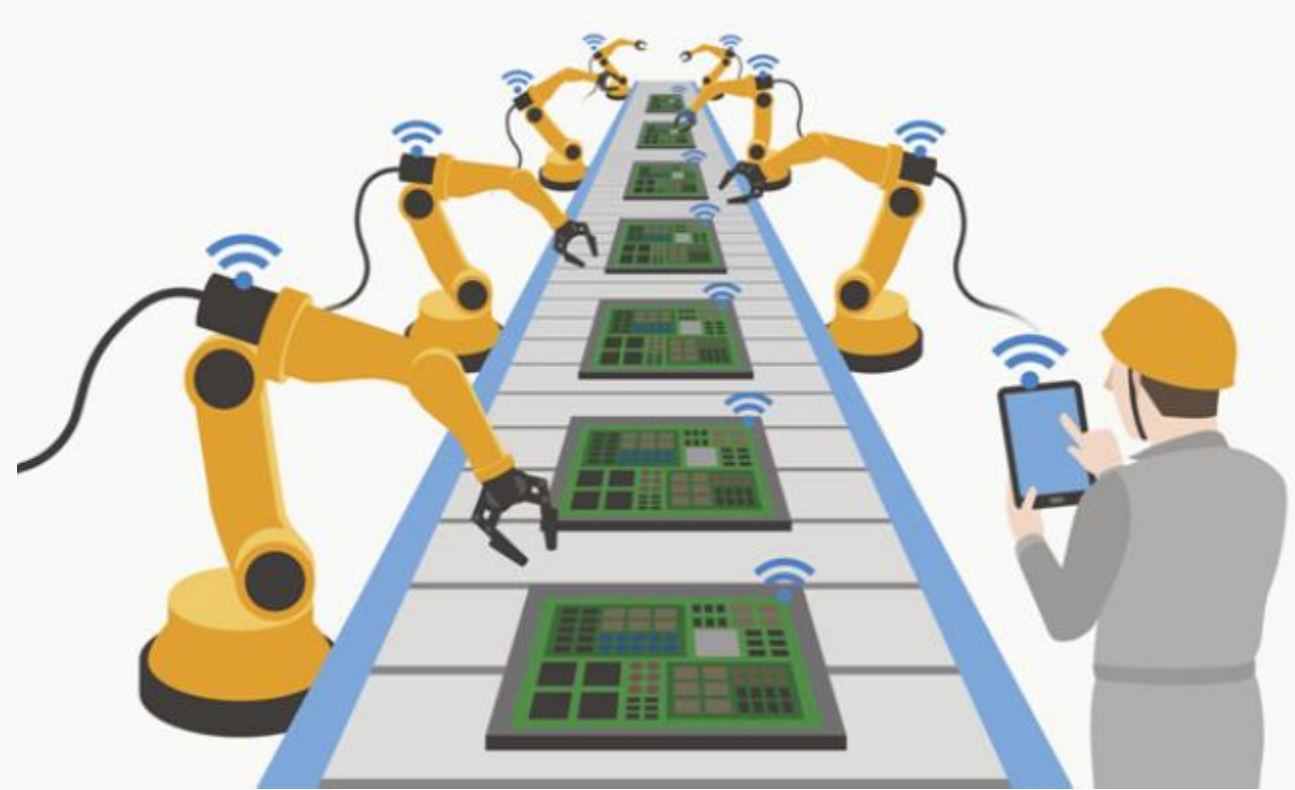
마이크로 서비스를  
설계 및 개발 함으로써  
개발 능력 및  
설계 능력 향상



DevOps

K8s 기반  
무중단 CI/CD 체계를  
직접 개발 함으로써  
개발 운영 역량 향상

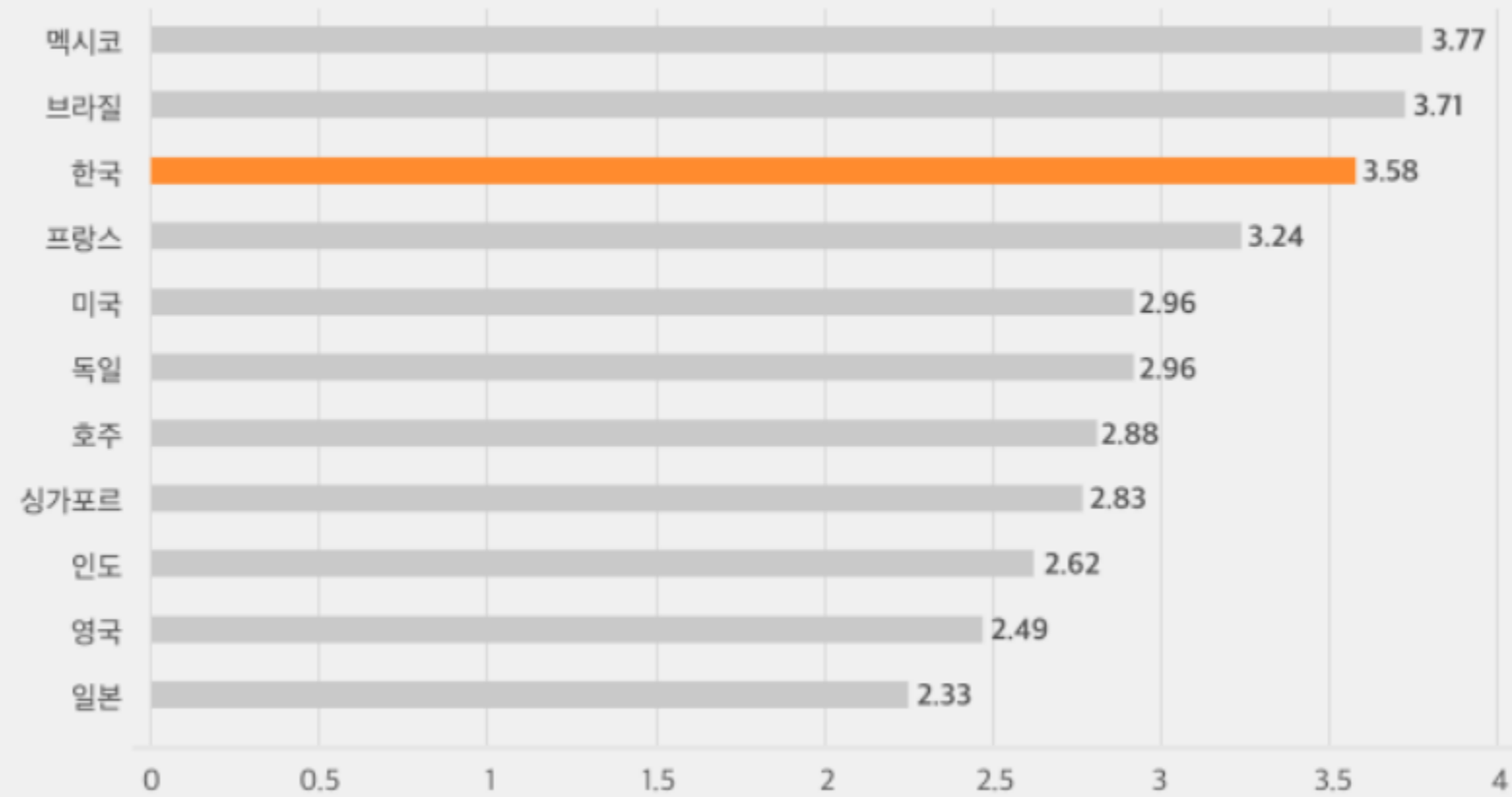
## RPA란? (Robotics Process Automation)



Robot이 공장의 실체적 기계였다면 RPA는 Software로 사람이 하는 단순 반복적인 일을 하는 Robot

# RPA 도입 배경

국가별 일 평균 부수적인 관리 업무 소요 시간



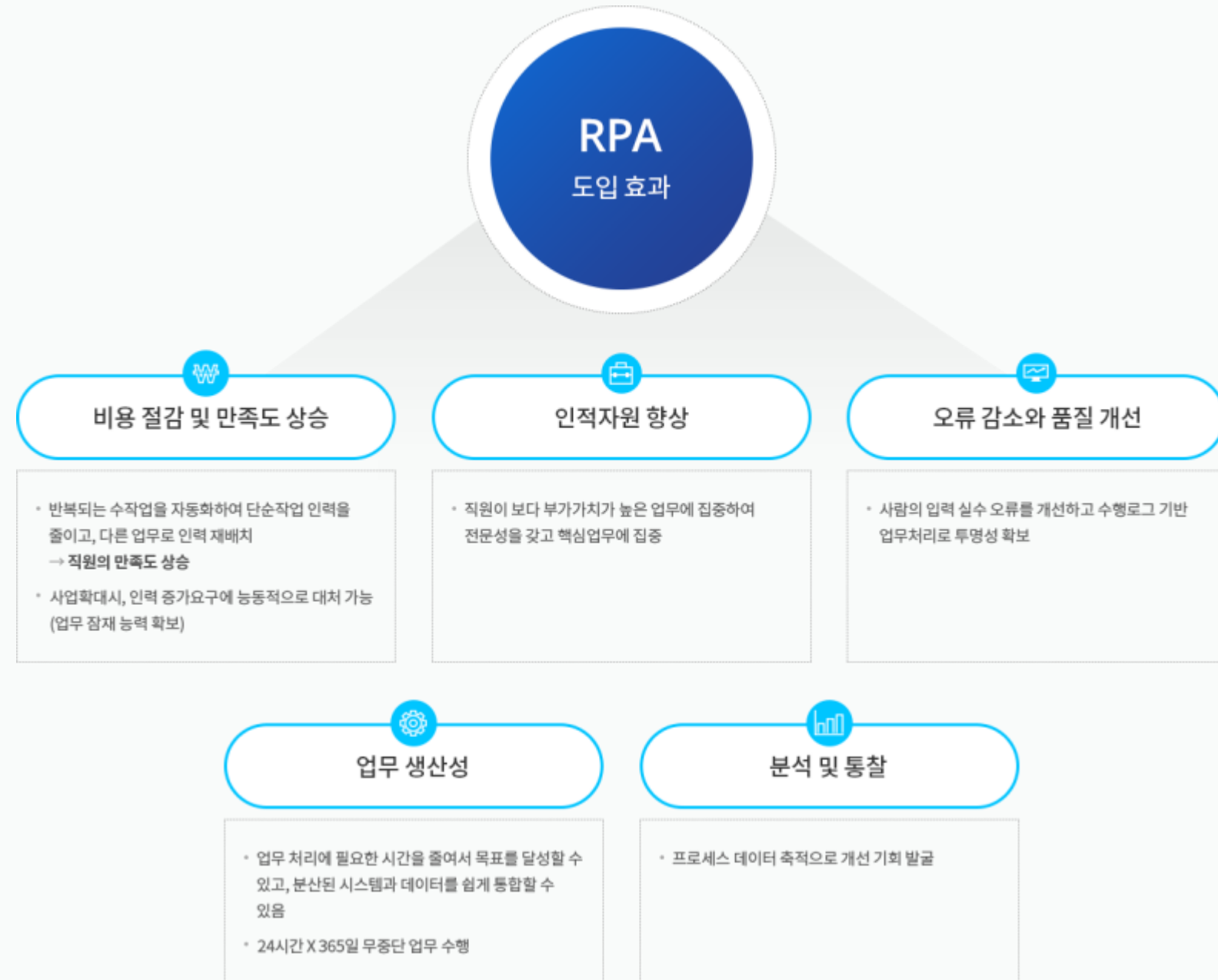
산업	적용 업무 예시	
보험	<ul style="list-style-type: none"> <li>고객 정보(프로파일) 갱신</li> <li>갱신 보험료 자동 생성</li> </ul>	<ul style="list-style-type: none"> <li>불편사항 처리</li> <li>정책 관리 및 서비스</li> </ul>
교통	<ul style="list-style-type: none"> <li>승강 처리</li> <li>서비스 및 수리 기록 모니터링</li> </ul>	<ul style="list-style-type: none"> <li>경로 변경에 대한 고지</li> </ul>
	<ul style="list-style-type: none"> <li>예약 처리</li> </ul>	<ul style="list-style-type: none"> <li>투자분석을 위한 정보 취합, 분석, 보고서 작성 및 발송</li> </ul>

▪ 비용 보고서, 급여, 법인카드, 출장비, 매입 세금계산서 처리  
 ▪ 매출 회계

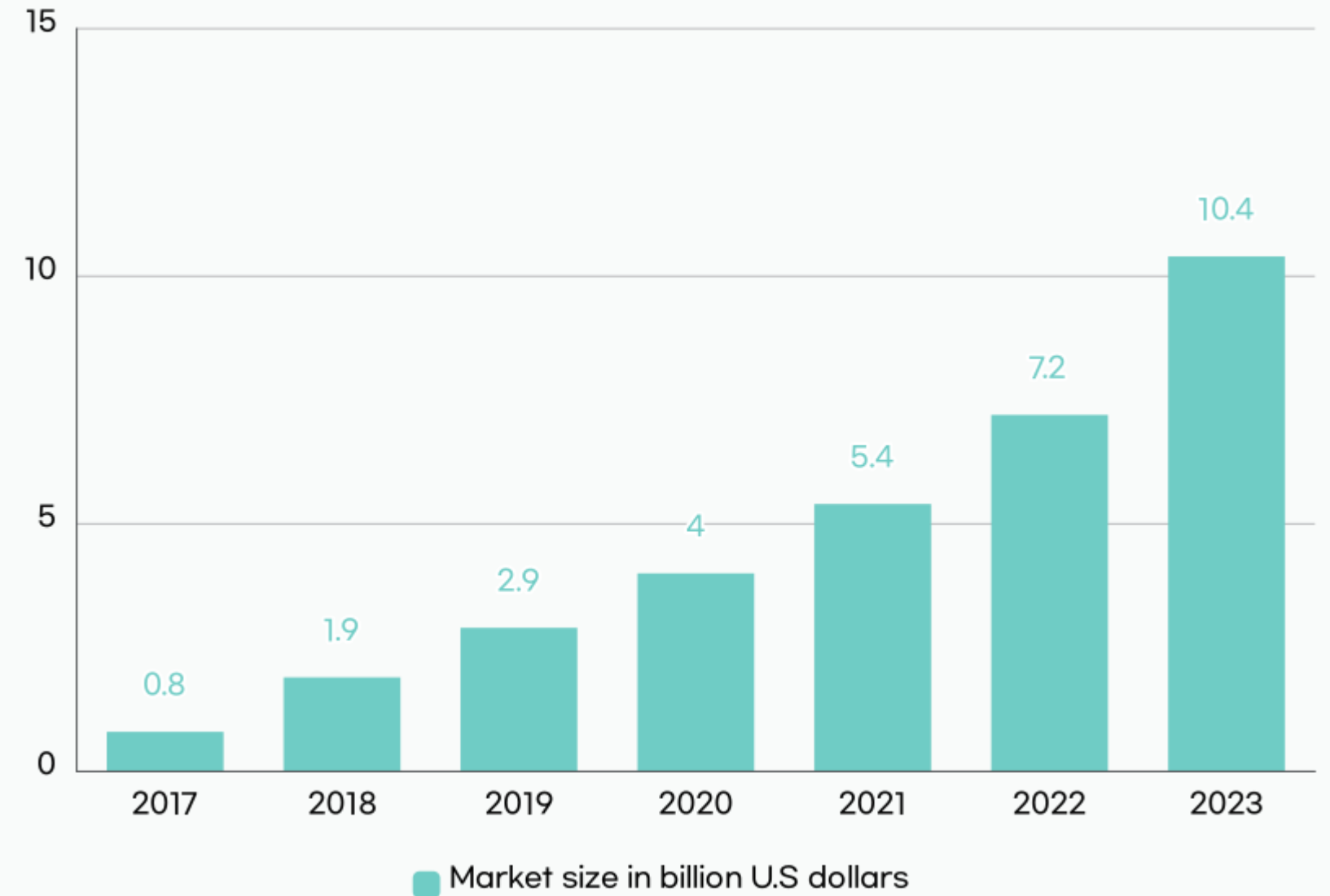
헬스케어	<ul style="list-style-type: none"> <li>공급자 정보 유효성 검사</li> <li>빌링 및 컴플라이언스 관리</li> <li>승강 발행</li> </ul>	<ul style="list-style-type: none"> <li>보험 등록 및 유효성 검사</li> <li>시스템 간 의료 기록 비교</li> <li>예약 정보 고지</li> </ul>
인사 등 경영 지원	<ul style="list-style-type: none"> <li>인사복지 관리</li> <li>컴플라이언스 리포팅</li> <li>원천징수 관리</li> </ul>	<ul style="list-style-type: none"> <li>주소 등록, 변경 처리</li> <li>데이터 클렌징</li> <li>주문 갱신</li> </ul>
물류	<ul style="list-style-type: none"> <li>온라인 주문, 생산 지시서 작성, 제품 발송 시 승강, 포장 목록 작성</li> <li>단순 정보 입력 및 이메일 발송</li> </ul>	<ul style="list-style-type: none"> <li>외부 계약자(물류창고, 물류회사 등)에게 출고, 배송 지시서, 해외주문 시 통관서류 작성</li> </ul>
소매업	<ul style="list-style-type: none"> <li>고객 불만사항 관리</li> <li>로열티 프로그램 고객 등록</li> </ul>	<ul style="list-style-type: none"> <li>배송 고지</li> <li>인벤토리 재정렬</li> </ul>
영업/회계	<ul style="list-style-type: none"> <li>신규 고객정보 등록 자동화</li> </ul>	<ul style="list-style-type: none"> <li>회계결산 업무 적용하여 오류개선 (수익결산/보고 관련 대량 데이터 정제, 가공)</li> </ul>

학벌주의로 인한 고급인력 양산과 저출산, 고령화로 인한 단순 노동 인력 감소

# RPA 시장



Statista의 글로벌 RPA 소프트웨어 시장 전망 2022

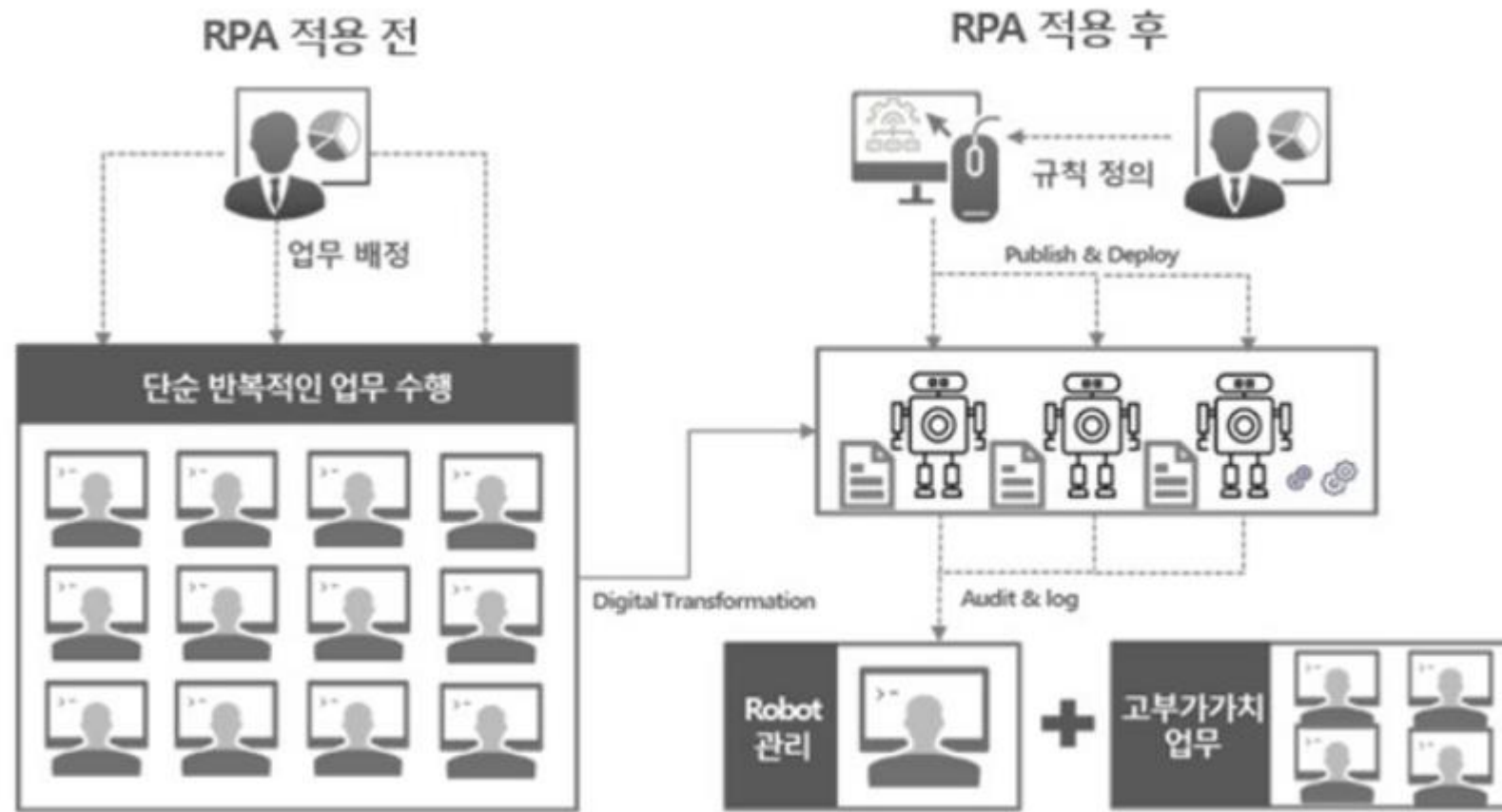


이에 대한 해결방안으로 세계적으로 **RPA 도입의 확산**

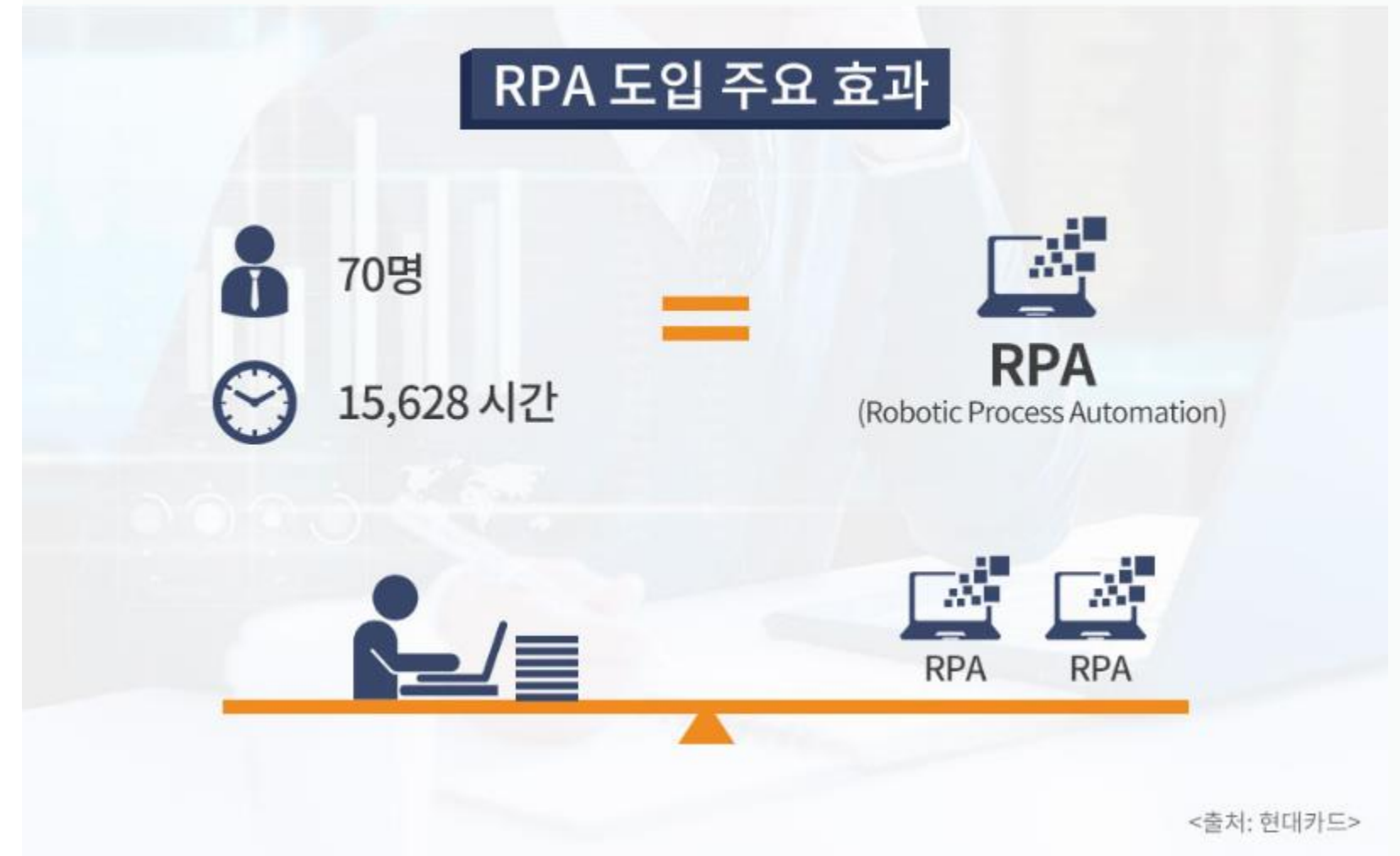


# RPA의 장점

**그림2** RPA 적용 전/후 비교



※ 출처: KT DS 공식 홈페이지

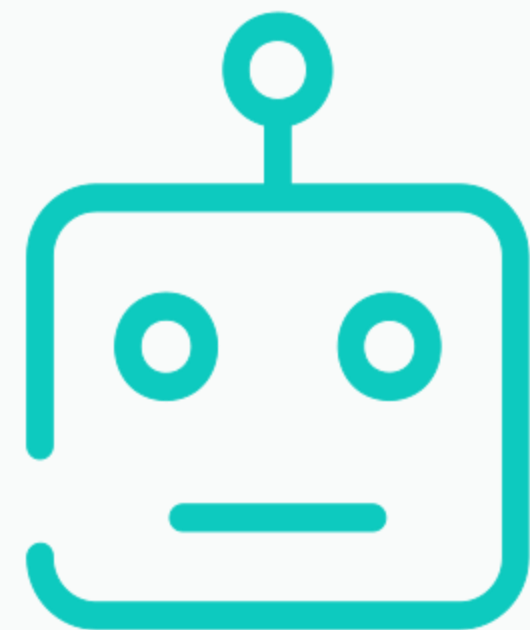


<출처: 현대카드>

단순 노동을 로봇으로 대체함으로써 임직원은 **고부가** 업무에 집중 -> **업무 생산성 향상**

## 2. 프로젝트 구성

---



# 프로젝트 구성 - 기술 스택

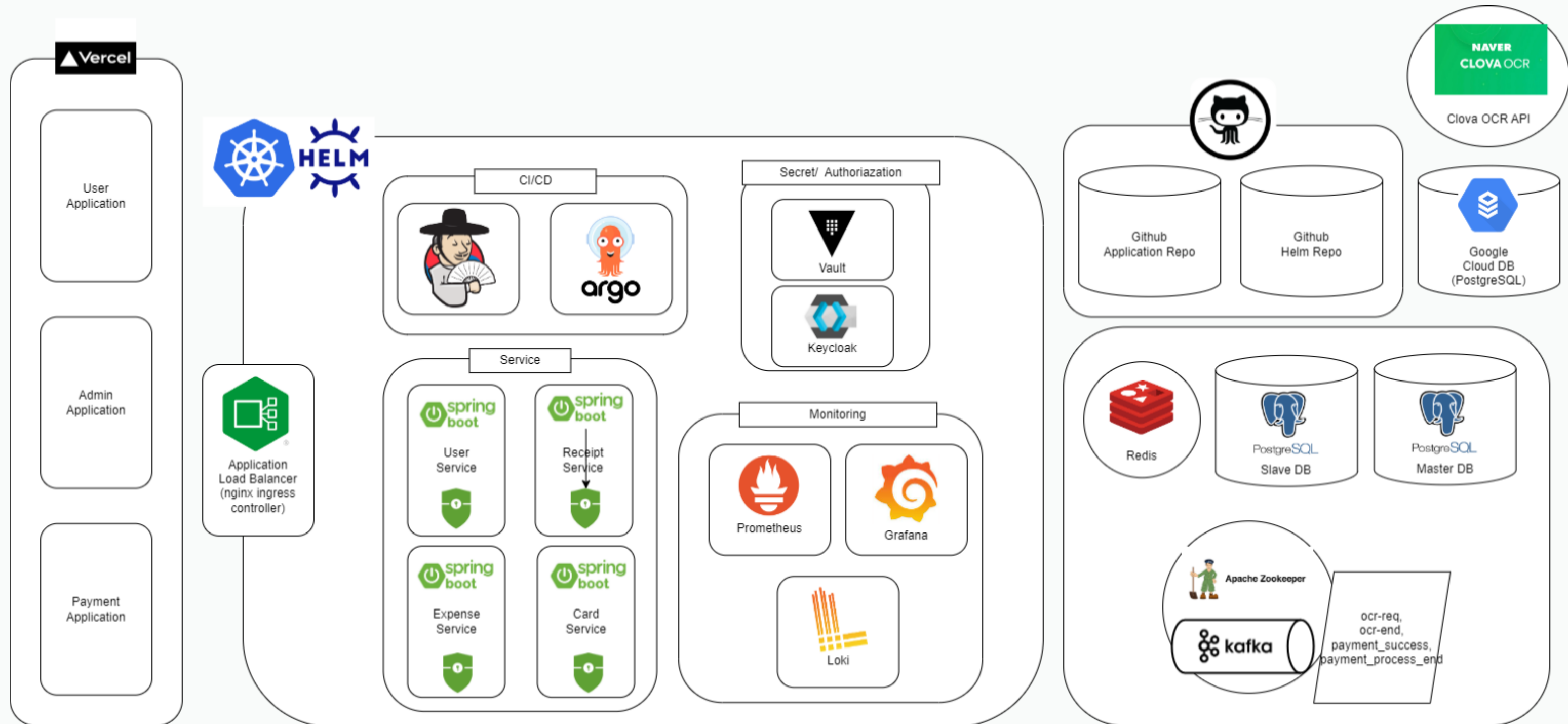
Front End

The Front End stack includes: HTML, CSS, JS, ESLint, TS, PWA, NextAuth.js, React Query, and Axios. A person icon is also present.

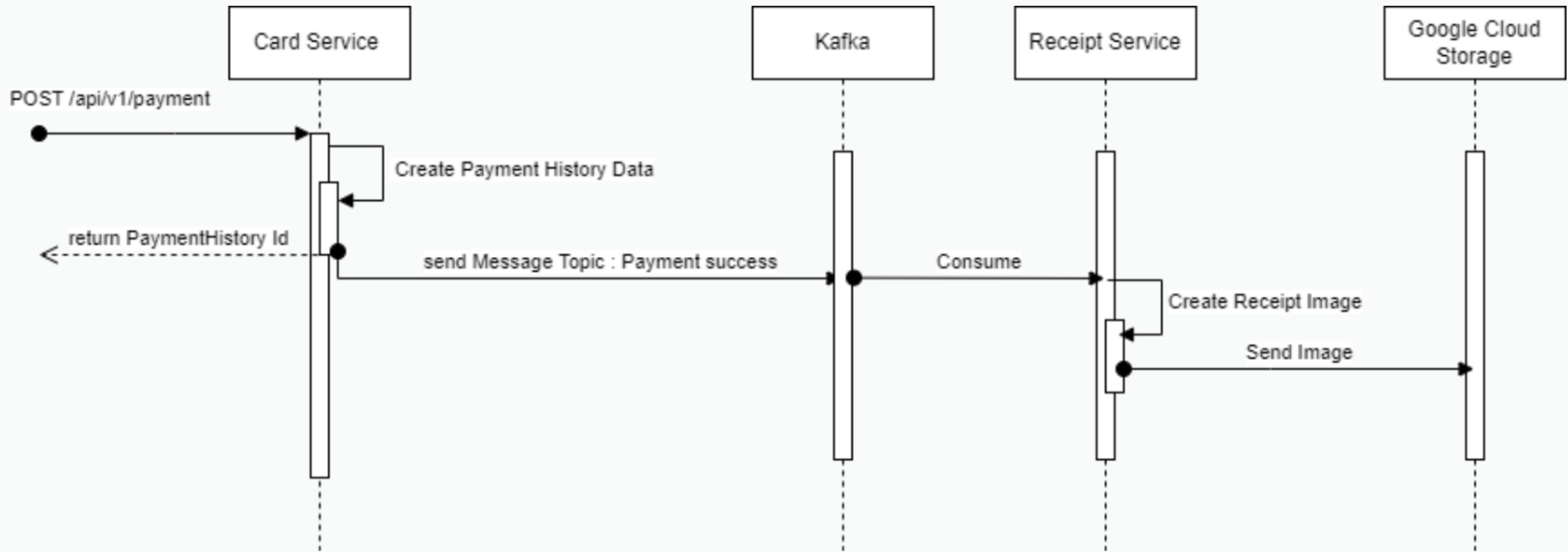
Back End

The Back End stack includes: Kafka, Redis, Naver Clova, Argo, Helm, and Google Cloud. A cartoon character icon is also present.

## 프로젝트 구성 - 서비스 아키텍처 구성

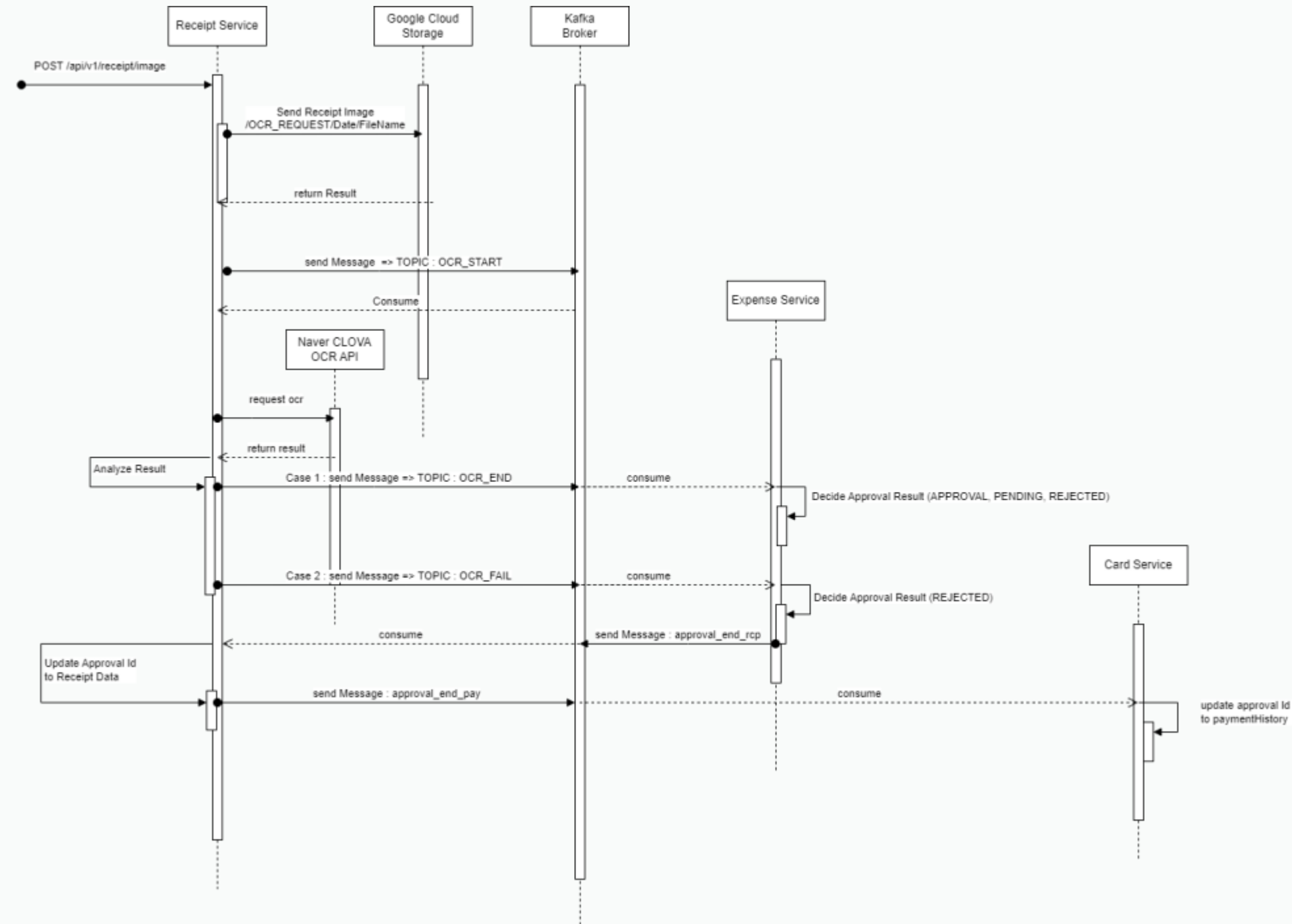


## 프로젝트 구성 - Sequence Diagram



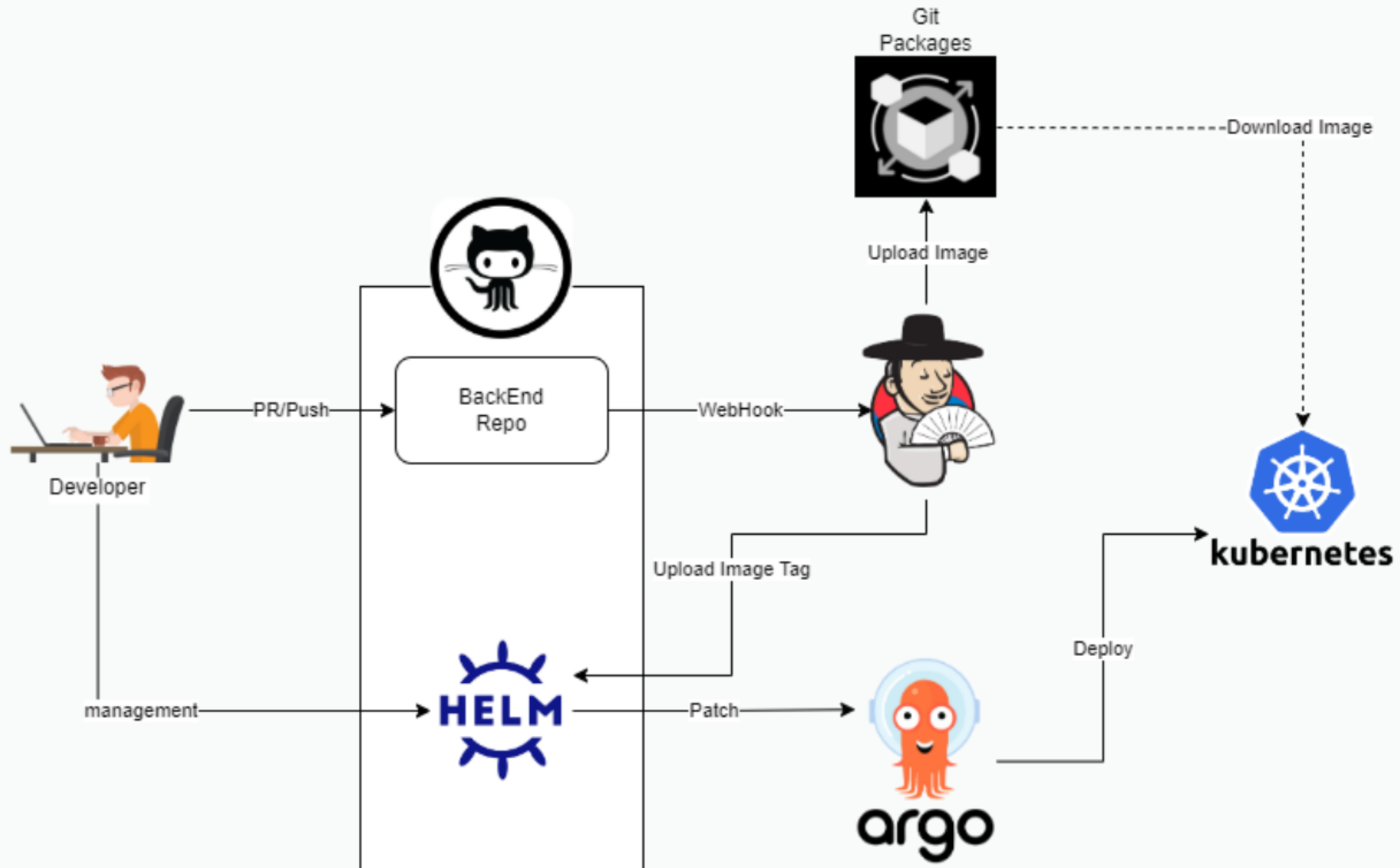
영수증 이미지 생성 로직

# 프로젝트 구성 - Sequence Diagram



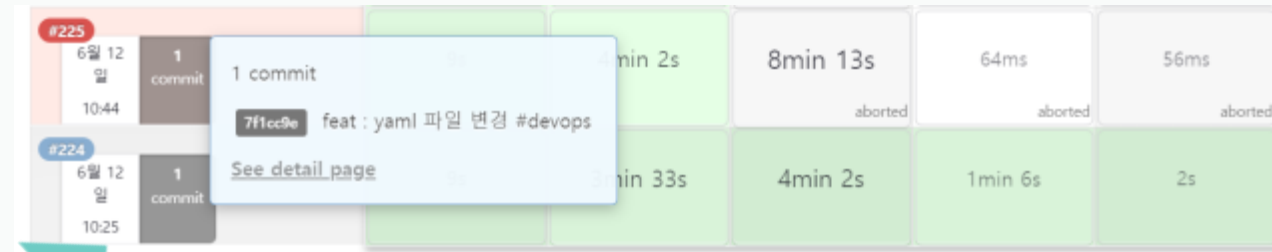
경비처리 로직

## 프로젝트 구성 - CI/CD 구성도





# 프로젝트 구성 - CI/CD 구성도



Commits

develop

Commits on Jun 12, 2023

- Merge pull request #54 from BiBot-org/S2P-153-Back-feature
- feat: global cache duration 수정 및 receipt service cache 제거 #52P-153
- Merge pull request #53 from BiBot-org/S2P-153-Back-feature
- feature: global cache set #52P-153
- feat: yml 파일 변경 #devops
- feat: yml 파일 변경 #devops
- feat: sh 파일 변경 #devops
- chore: 공용모듈에만 vault dependency 적
- chore: 공용모듈 vault config 활성화
- feat: 경비 처리 예러처리 및 메시지 파이프라인 통합 수정 #52P-43

- #215 (2023. 6. 11. 오후 3:12:23)
  - chore : application.yml 파일 분리 및 수정 #52P-43 (commit: f2d0495) — swj1265 / githubweb
  - feat : 비밀번호 변경 구현 & 캐싱 및 볼트 속성 비활성화 #52P-43 (commit: 468b657) — swj1265 / githubweb
  - feat : 경비 처리 예러처리 #52P-43 (commit: ce86aea) — swj1265 / githubweb
  - feat : 경비 처리 예러처리 및 메시지 파이프라인 통합 수 #52P-43 (commit: f892a0c) — swj1265 / githubweb
- #214 (2023. 6. 11. 오후 2:47:18)
  - feat : 배포파일 수정 #devops (commit: 0e81298) — 102888442+Junseokie / githubweb
- #213 (2023. 6. 11. 오후 1:32:43)
  - feat : 로그인 로직 변경 #devops (commit: e3862bf) — 102888442+Junseokie / githubweb
- #212 (2023. 6. 11. 오전 10:03:47)
  - backup : kafka 연동 백업 #52P-145 (commit: 154a534) — shghdr1234 / githubweb
- #210 (2023. 6. 10. 오후 6:28:39)
  - fix : sh file host 수정 (commit: 8d2ae95) — 102888442+Junseokie / githubweb
- #207 (2023. 6. 10. 오후 4:31:54)
  - fix : vault host 수정 #devops (commit: b0aa207) — 102888442+Junseokie / githubweb
- #206 (2023. 6. 9. 오전 11:58:44)
  - 긴급 패치 (commit: c0e9905) — shghdr1234 / githubweb
- #204 (2023. 6. 9. 오전 1:47:17)
  - backup : user profile CUD API 백업 #52P-140 (commit: 0f48629) — shghdr1234 / githubweb
  - refactor : user profile CUD 로직 refactoring #52P-140 (commit: ef35ff2) — shghdr1234 / githubweb
  - feat : yml file secret 설정 #devops (commit: 45b155e) — weq156 / githubweb
  - fix : yml file #devops (commit: 2399454) — weq156 / githubweb
  - feat : kubernetes dockerfile sh #devops (commit: 9980555) — weq156 / githubweb
  - feat : kubernetes dockerfile 수정 #devops (commit: 2f1dcea) — weq156 / githubweb
  - backup : global cache 백업 #52P-141 (commit: 5da7afb) — shghdr1234 / githubweb
  - feat : 수동 경비처리 구현 #52P-43 (commit: cdef26f) — swj1265 / githubweb
  - feat : 연수증 경비처리 연 #52P-43 (commit: 4c7ae68) — swj1265 / githubweb
  - fix : 시크릿 데이터 삭제 (commit: a04ba11) — swj1265 / githubweb
  - fix : profiles 환경 변경 # devops (commit: a9a1121) — weq156 / githubweb

개발자가 Git Repo에 Push -> Jenkin에서 Build



## 프로젝트 구성 - CI/CD 구성도

BiBot-org

Overview Repositories 8 Projects Packages Teams People 7 Settings

user-service

Install from the command line [Learn more about packages](#)

```
$ docker pull ghcr.io/bibot-org/user-service:231
```

Last published **3 hours ago**

Total downloads **52**

Package settings

Recent tagged image versions

231	0
Published about 3 hours ago · Digest	
230	3
Published about 13 hours ago · Digest	
229	3
Published about 13 hours ago · Digest	
228	0
Published about 13 hours ago · Digest	
227	4
Published about 14 hours ago · Digest	

main 1 branch 0 tags

Go to file Add file <> Code Use this template

Junseokee change tag : 230 6b4a902 12 hours ago 102 commits

idea	service account 수정 deploy 수정	3 days ago
templates	sh 명령 삭제	yesterday
.helmignore	add helm chart	3 months ago
Chart.yaml	Update Chart.yaml	last month
README.md	권한확인	last month
values.yaml	change tag : 230	12 hours ago

```
spec:  
  containers:  
  - image: 'ghcr.io/bibot-org/user-service:230'  
    imagePullPolicy: IfNotPresent  
    name: bibot
```

Build가 완료되면 Image Push -> HelmChart Tag 변경 -> ArgoCD에서 Patch

# 프로젝트 구성 - CI/CD 구성도

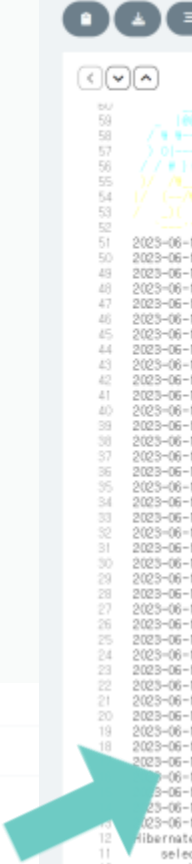
```

argoCD / templates / deployment.yaml
Junseooke sh 명령 삭제

Code Blame 66 lines (65 loc) · 2 KB

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: {{ include "bibot.fullname" . }}
5    labels:
6      {{- include "bibot.labels" . | nindent 4 }}
7
8  spec:
9    {{- if not .Values.autoscaling.enabled }}
10   replicas: {{ .Values.replicaCount }}
11   {{- end }}
12   selector:
13     matchLabels:
14       {{- include "bibot.selectorLabels" . | nindent 6 }}
15   template:
16     metadata:
17       {{- with .Values.podAnnotations }}
18       annotations:
19         {{- toYaml . | nindent 8 }}
20       vault.hashicorp.com/agent-inject: "true"
21       vault.hashicorp.com/role: "secret-read"
22       vault.hashicorp.com/address: "http://34.22.82.97:8200"
23       vault.hashicorp.com/token: $VAULT_TOKEN
24     {{- end }}
25     labels:
26       {{- include "bibot.selectorLabels" . | nindent 6 }}
27   spec:
28     {{- with .Values.imagePullSecrets }}
29     imagePullSecrets:
30       {{- toYaml . | nindent 8 }}
31     {{- end }}

```

```

user-bibot-5b5b68575-9pbct
SUMMARY EVENTS LOGS
Page 1 (Lines 1 to 62)

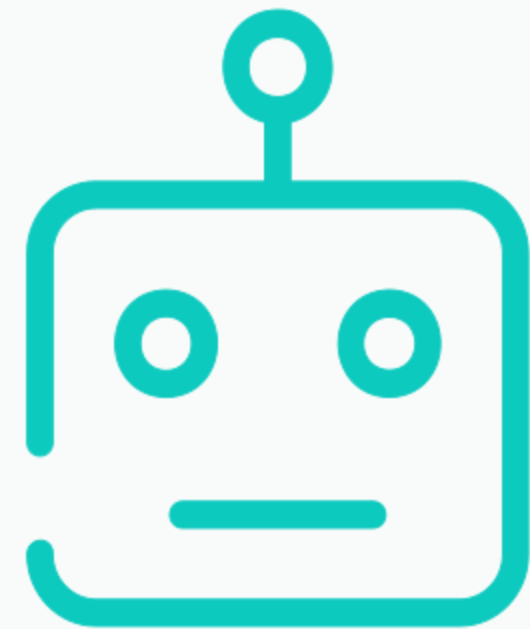
51 2023-06-12T03:27:15.602Z INFO 11 --- [
52 2023-06-12T03:27:15.643Z INFO 11 --- [
53 2023-06-12T03:27:16.325Z INFO 11 --- [
54 2023-06-12T03:27:16.335Z INFO 11 --- [
55 2023-06-12T03:27:16.336Z INFO 11 --- [
56 2023-06-12T03:27:16.339Z INFO 11 --- [
57 2023-06-12T03:27:16.340Z INFO 11 --- [
58 2023-06-12T03:27:16.352Z INFO 11 --- [
59 2023-06-12T03:27:28.145Z INFO 11 --- [
60 2023-06-12T03:27:28.327Z INFO 11 --- [
61 2023-06-12T03:27:34.570Z INFO 11 --- [
62 2023-06-12T03:27:34.586Z INFO 11 --- [
63 2023-06-12T03:27:36.726Z INFO 11 --- [
64 2023-06-12T03:27:36.821Z INFO 11 --- [
65 2023-06-12T03:27:36.822Z INFO 11 --- [
66 2023-06-12T03:27:41.256Z INFO 11 --- [
67 2023-06-12T03:27:48.209Z INFO 11 --- [
68 2023-06-12T03:27:48.325Z INFO 11 --- [
69 2023-06-12T03:27:48.326Z INFO 11 --- [
70 2023-06-12T03:27:48.198Z INFO 11 --- [
71 2023-06-12T03:27:48.239Z INFO 11 --- [
72 2023-06-12T03:27:57.027Z INFO 11 --- [
73 2023-06-12T03:27:57.484Z INFO 11 --- [
74 2023-06-12T03:28:01.017Z INFO 11 --- [
75 2023-06-12T03:28:02.838Z INFO 11 --- [
76 2023-06-12T03:28:02.874Z INFO 11 --- [
77 2023-06-12T03:28:03.292Z INFO 11 --- [
78 2023-06-12T03:28:20.034Z INFO 11 --- [
79 2023-06-12T03:28:20.145Z INFO 11 --- [
80 2023-06-12T03:28:26.652Z INFO 11 --- [
81 2023-06-12T03:28:26.897Z INFO 11 --- [
82 2023-06-12T03:28:27.000Z INFO 11 --- [
83 2023-06-12T03:28:27.000Z INFO 11 --- [
84 2023-06-12T03:28:27.000Z INFO 11 --- [
85 2023-06-12T03:28:27.000Z INFO 11 --- [
86 2023-06-12T03:28:27.000Z INFO 11 --- [
87 2023-06-12T03:28:27.000Z INFO 11 --- [
88 2023-06-12T03:28:27.000Z INFO 11 --- [
89 2023-06-12T03:28:27.000Z INFO 11 --- [
90 2023-06-12T03:28:27.000Z INFO 11 --- [
91 2023-06-12T03:28:27.000Z INFO 11 --- [
92 2023-06-12T03:28:27.000Z INFO 11 --- [
93 2023-06-12T03:28:27.000Z INFO 11 --- [
94 2023-06-12T03:28:27.000Z INFO 11 --- [
95 2023-06-12T03:28:27.000Z INFO 11 --- [
96 2023-06-12T03:28:27.000Z INFO 11 --- [
97 2023-06-12T03:28:27.000Z INFO 11 --- [
98 2023-06-12T03:28:27.000Z INFO 11 --- [
99 2023-06-12T03:28:27.000Z INFO 11 --- [
100 2023-06-12T03:28:27.000Z INFO 11 --- [

```

Helm 템플릿 바탕으로 Pod 배포 -> Kubernetes에서 Application 실행

### 3. 프로젝트 진행

---



# 프로젝트 진행 - 협업툴

프로젝트 / spharos\_2nd\_project

## 로드맵

상태 범주 | 에픽 | 사용자 지정 필터

	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2
릴리스															
S2P-2 유즈케이스 분석 및 UID 문서 작성															
S2P-6 기능명세 및 ERD 작성 [BACKEND] ...															
S2P-8 [Back] 백엔드 초기 셋업															
S2P-11 [Back] 멀티모듈 설계															
S2P-16 [Front] 프론트엔드 초기 셋업															
S2P-28 [Back] OCR 로직 설계 및 테스트															
S2P-31 [Back] 백엔드 1차 개발															
S2P-37 [Front] 프론트 1차 개발															
S2P-86 [Back] 백엔드 2차 개발															
S2P-87 [Front] 프론트 2차 개발															

프로젝트 / spharos\_2nd\_project

## rpa\_code\_recipe

에픽 | 사용자 지정 필터

[BACKEND] 완료 23 이슈

- [Back]feature : 경비 내역 조회 및 관리 기능 구현 [BACK] 백엔드 2차 개발 S2P-55
- [Back]feature : 스케줄러 기반 환경설정 구현 S2P-124
- [Back]feature : 카드 결제 데이터 생성 및 영수증 이미지 생성 프로그램 개발 [BACK] OCR 로직 설계 및 테스트 S2P-29
- [Back]feature : vault setup [BACK] 백엔드 2차 개발 S2P-111
- [Back]feature : 영수증 결제 기능 구현 [BACK] 백엔드 2차 개발 S2P-60
- [Back]feature : 결제 내역 조회 구현 [BACK] 백엔드 2차 개발 S2P-43
- [Back]feature : OCR API 연동 [BACK] 백엔드 2차 개발 S2P-96
- [Back]feature : 유저 애플리케이션 대응

[FRONTEND] 진행 중 1 이슈

- [Front]feature : 마무리 작업 [FRONT] 프론트 2차 개발 S2P-154

S2P-87 / S2P-139

### [Front]feature : 마이페이지 기능 수정

첨부 | 하위 이슈 추가 | 이슈 연결

기반: 없음

설명: 설명 편집

활동: 표시: 모두 | 댓글 | 기록 | 최신 항목 먼저

댓글 추가...

프로젝트 | 댓글 | 댓글 추가

[FRONTEND] 완료 | 완료 | 작업

세부 정보

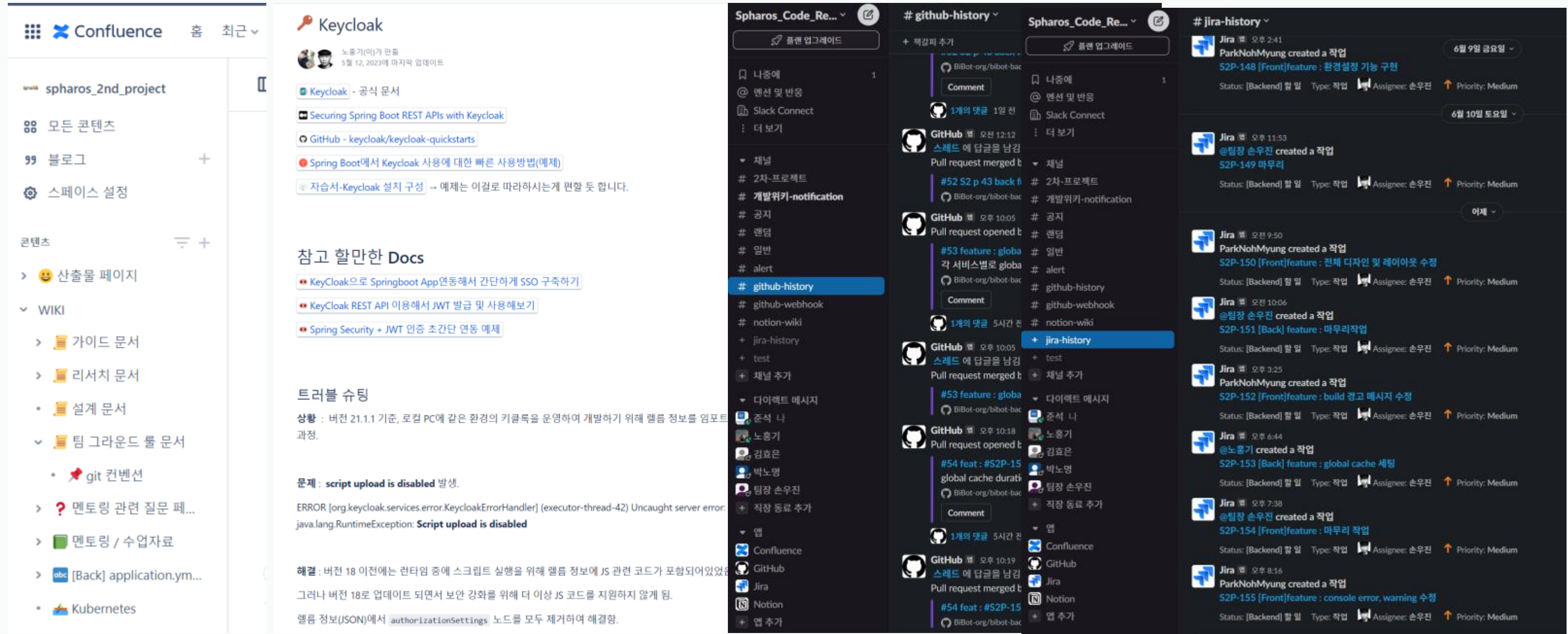
- 담당자: ParkNohMyung (나에게 할당)
- 레이블: 없음
- 수익 버전: 없음
- 개발: 브랜치 만들기 (7개 커밋, 1개 풀리퀘스트, 5일 전, MERGED)
- 모고자: ParkNohMyung
- Automation: Rule executions

만용기 2023년 6월 2일 오후 3:18 | 업데이트됨 지난주 | 해설됨 지난주 | 구성

Jira를 활용하여 개발 일정 관리와 자동화를 활용한 이슈, 커밋 관리

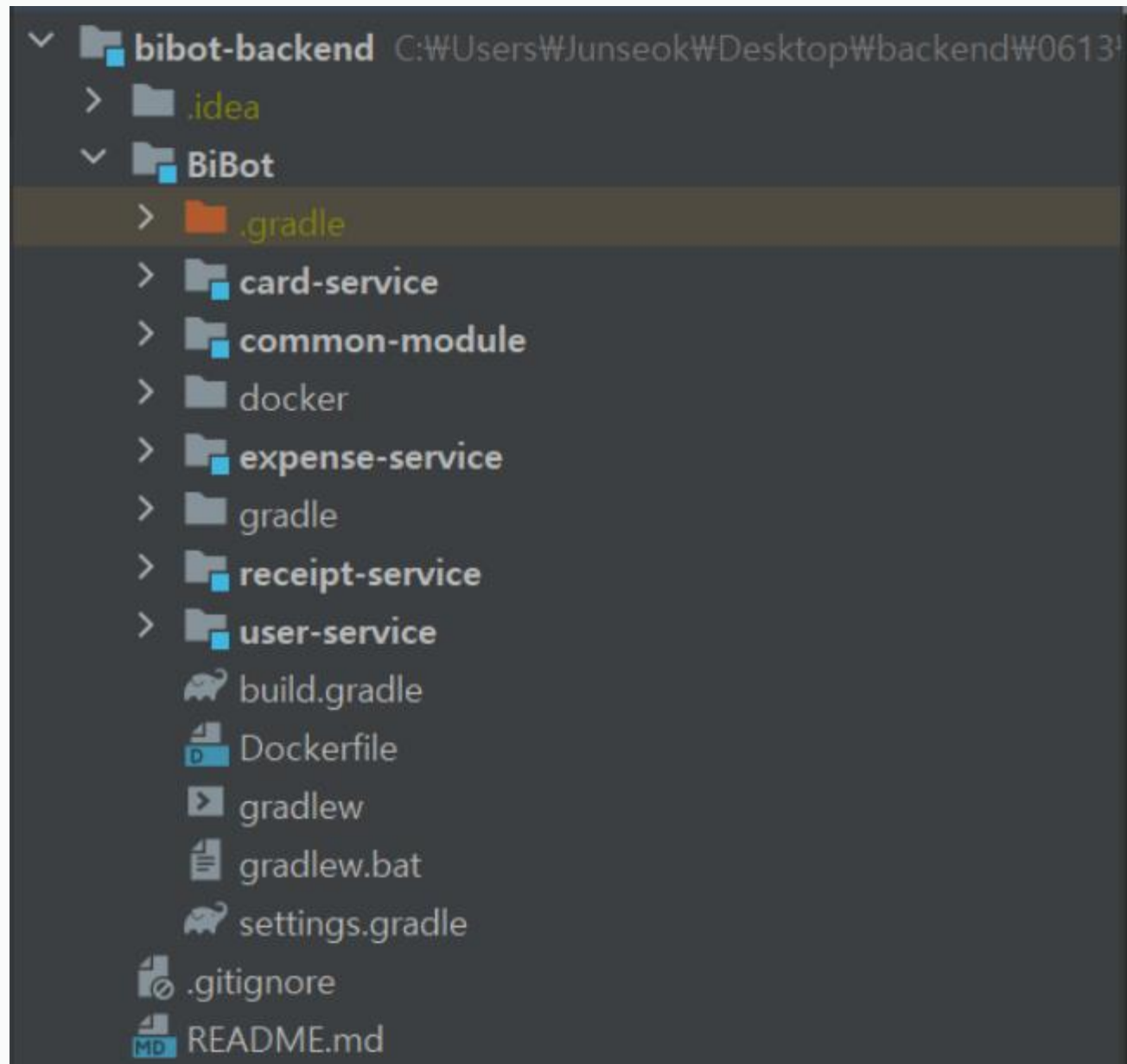


# 프로젝트 진행 - 협업툴



Confluence를 활용하여 산출물과 Wiki 관리, Slack을 협업툴과 연동하여 프로젝트 관리

## 프로젝트 진행 - 멀티 모듈 구조

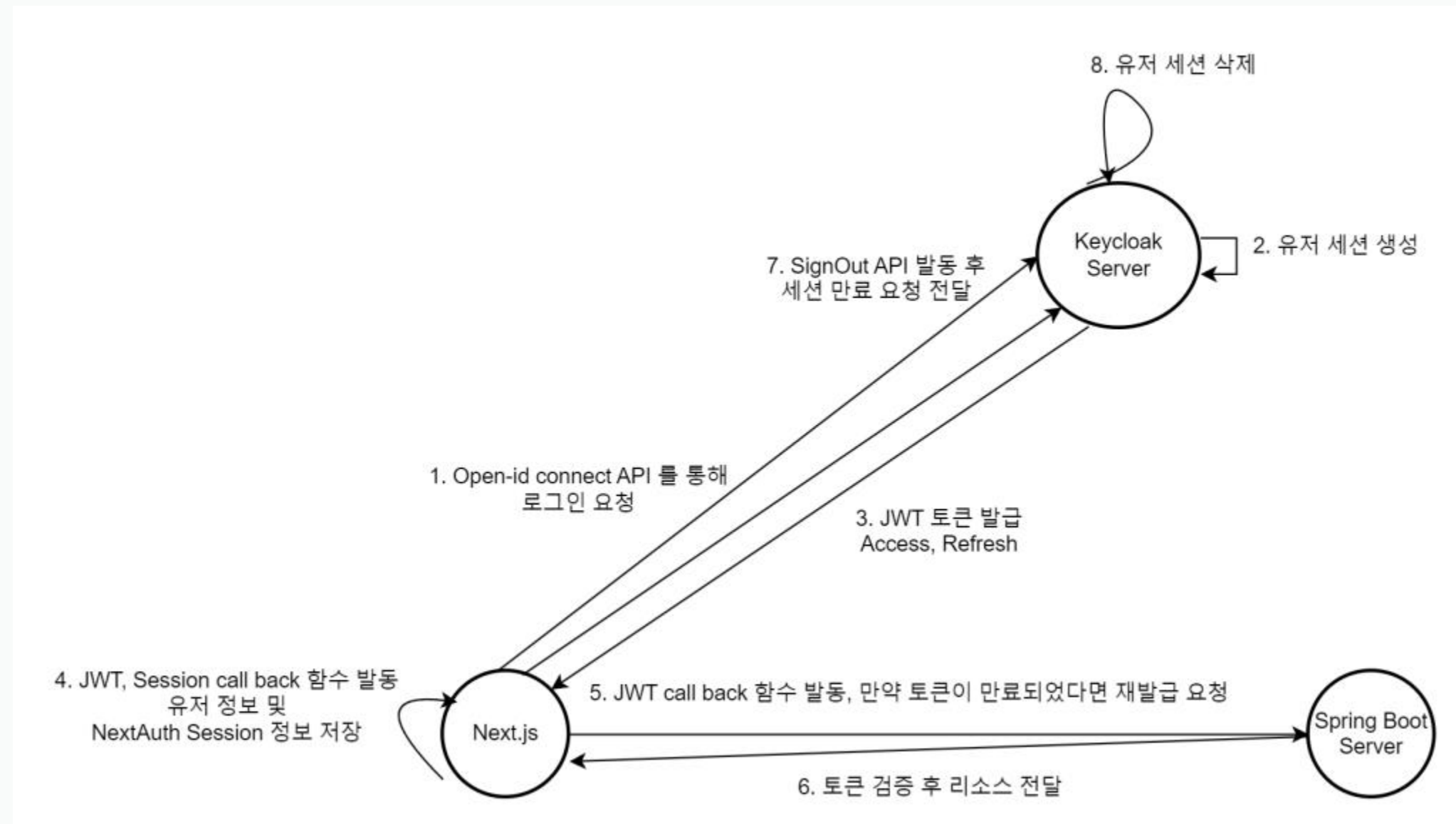


```
def commonProject : ArrayList<Project> =
    [project(':common-module')]
configure(commonProject) {
    bootJar {
        enabled = false
    }
    jar {
        enabled = true
    }
    bootBuildImage {
        enabled = false
    }
}

def domainProjects : ArrayList<Project> = [project(':card-service'), project(':expense-service')
    , project(':user-service'), project(':receipt-service')]
configure(domainProjects) {
    dependencies {
        implementation project(':common-module')
        // implementation 'org.springframework.cloud:spring-cloud-starter-vault-config'
        implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
        implementation 'org.springframework.boot:spring-boot-starter-cache'
        implementation 'org.springframework.boot:spring-boot-starter-web'
        implementation 'org.springframework.boot:spring-boot-starter-validation'
        runtimeOnly 'org.postgresql:postgresql'
    }
}
```

멀티 모듈을 활용한 Repo 간소화와 Dependency 버전 통일과 확장성 확보

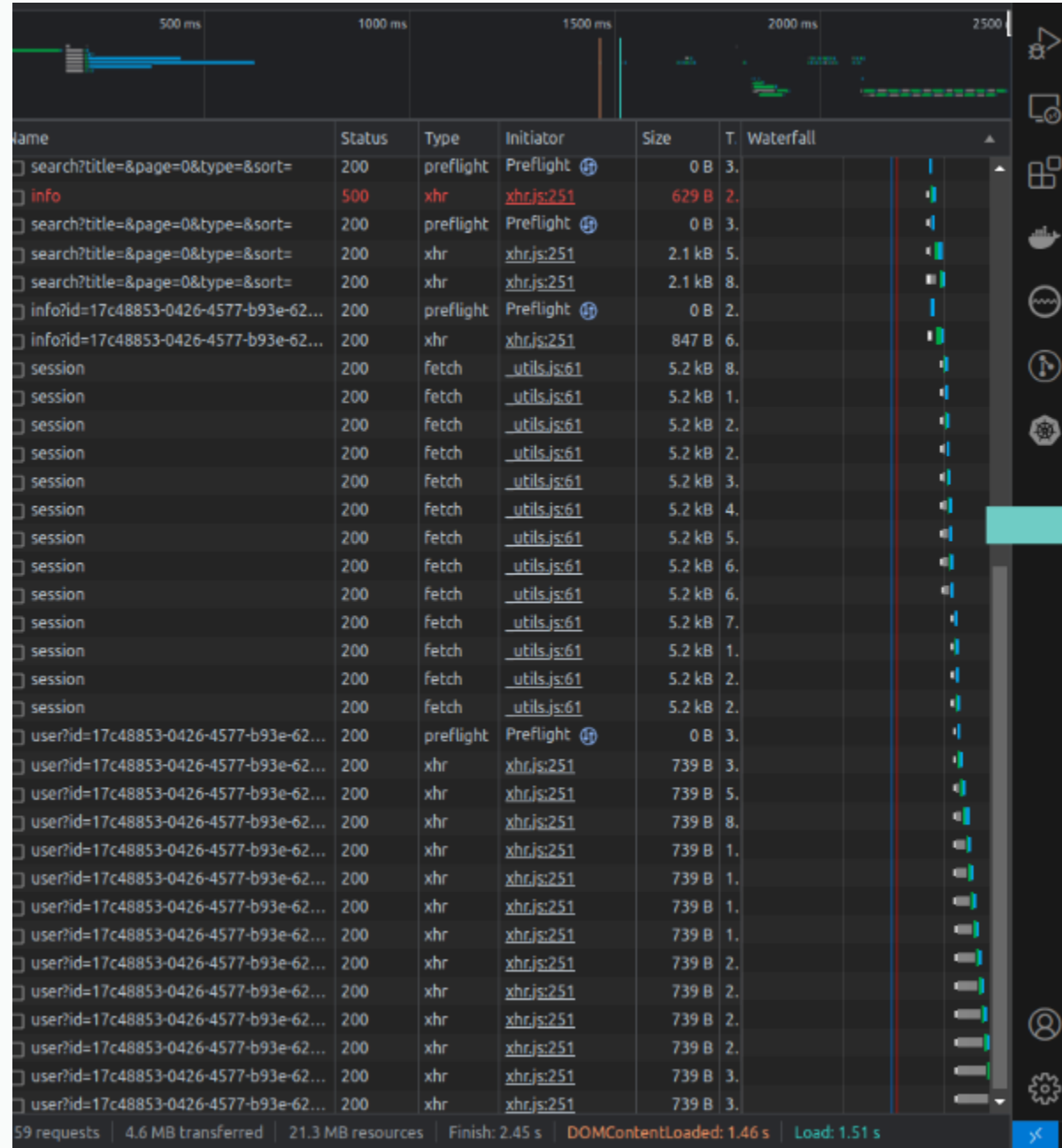
## 프로젝트 진행 - Keycloak, NextAuth



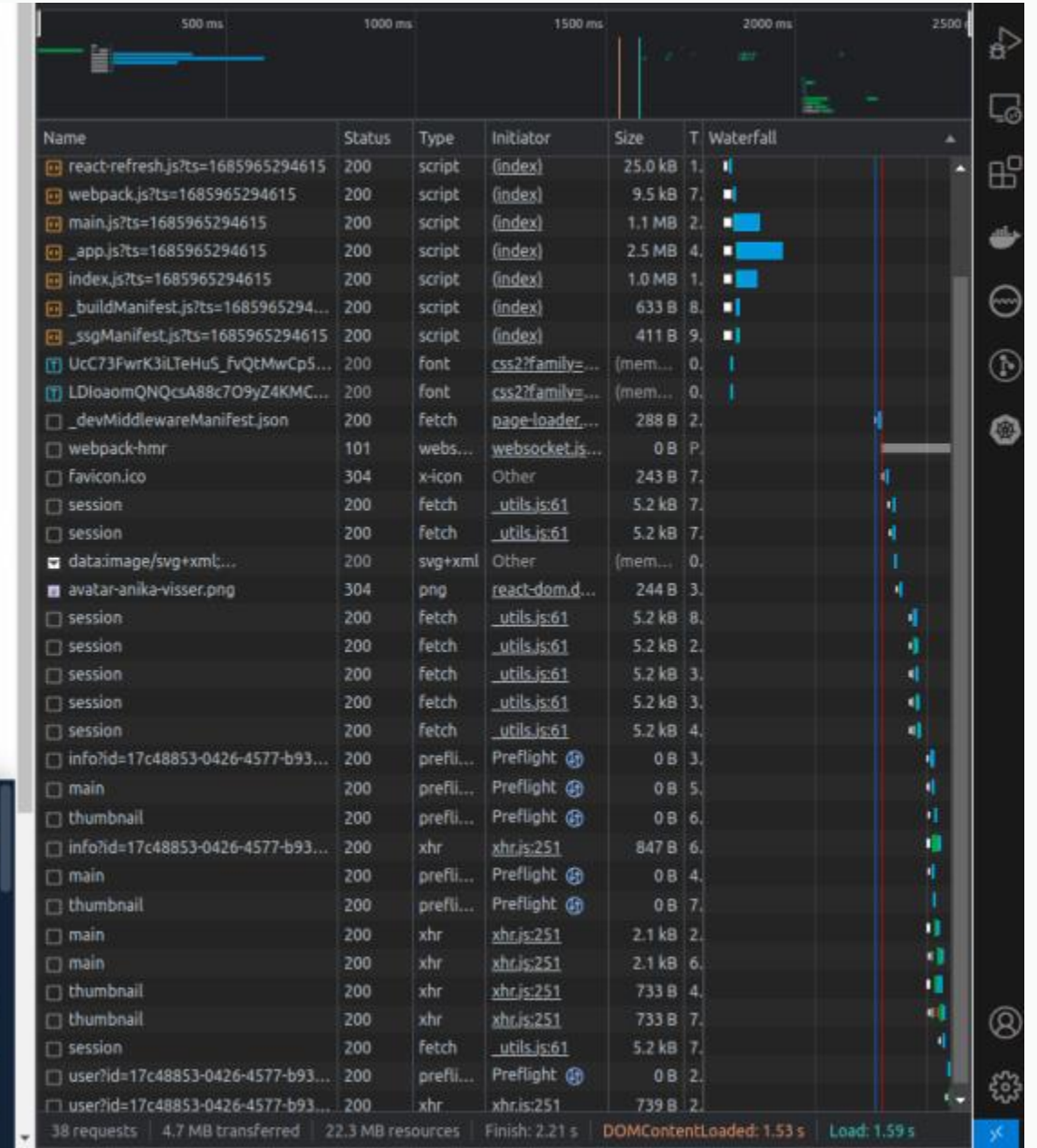
Keycloak을 통한 SSO 구현과 NextAuth 연동을 통해 확장성과 사용성 확보



## 프로젝트 진행 - React Query



The screenshot shows a web application interface. The main content area displays a list of items with columns for '이름' (Name), '부서 / 팀' (Department / Team), '결제 항목' (Payment Item), 'DATE', and '승인 여부' (Approval Status). One item is highlighted: '사원이름' (Employee Name), '개발 1센터 / 스파로스 개발팀' (Development 1 Center / Sparos Development Team), '식비' (Meal), '12/04/2019', and 'pending'. A modal window is open, showing 'Query Details' for a query: `"getUser : 17c48853-0426-4577-b93e-625abe639e59"`. The query is marked as 'stale' and has 4 observers. The last updated time is 8:41:36 PM. The modal also includes 'Actions' like 'Refetch', 'Invalidate', 'Reset', and 'Remove'.



API 호출이 많은 MSA에서 React Query를 이용한 캐싱처리를 통한 API 호출 감소와 렌더링 성능 향상



## 프로젝트 진행 - Global cache server

Case 1 : 100명의 유저가 초당 1회씩 1000번 요청

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput ↑	Received KB/sec	Sent KB/sec	Avg. Bytes
before Caching	100000	46	1	1075	74.04	0.00%	2049.9/sec	1214.54	264.24	606.7
after Caching	100000	23	0	608	48.58	0.00%	3842.2/sec	2276.44	495.28	606.7
TOTAL	200000	34	0	1075	63.67	0.00%	1811.9/sec	1073.55	233.57	606.7

user 100, period 1, Loop 1000

Case 2 : 1339명의 유저가 초당 1회씩 1회 요청

Label	# Samples	Average ↑	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
after caching	1339	12	2	103	13.05	0.00%	948.3/sec	562.13	122.24	607.0
before caching	1339	1180	258	2713	384.84	0.00%	471.1/sec	279.28	60.73	607.0
TOTAL	2678	596	2	2713	644.14	0.00%	95.7/sec	56.75	12.34	607.0

user 1339, period 1, Loop 1

Case 3 : 1339명의 유저가 초당 1회씩 2회 요청

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput ↓	Received KB/sec	Sent KB/sec	Avg. Bytes
after caching	2678	547	5	1102	225.66	0.00%	1150.3/sec	681.89	148.29	607.0
before caching	2678	829	5	2394	426.34	0.00%	811.0/sec	480.75	104.55	607.0
TOTAL	5356	688	5	2394	369.03	0.00%	125.4/sec	74.33	16.16	607.0

user 1339, period 1, Loop 2

Case 4 : 1339명의 유저가 초당 1회씩 3회 요청

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput ↓	Received KB/sec	Sent KB/sec	Avg. Bytes
after caching	4017	627	2	2989	436.73	0.00%	1324.9/sec	785.35	170.78	607.0
before caching	4017	1263	4	4050	587.49	0.00%	697.6/sec	413.54	89.93	607.0
TOTAL	8034	945	2	4050	607.45	0.00%	254.9/sec	151.12	32.86	607.0

user 1339, period 1, Loop 3

Case 5 : 1339명의 유저가 초당 1회씩 4회 요청

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput ↓	Received KB/sec	Sent KB/sec	Avg. Bytes
after caching	5356	530	2	3184	355.87	0.00%	1674.8/sec	992.78	215.89	607.0
before caching	5356	846	4	2818	472.95	0.00%	1064.6/sec	631.07	137.23	607.0
TOTAL	10712	688	2	3184	447.25	0.00%	247.9/sec	146.97	31.96	607.0

user 1339, period 1, Loop 4

Redis와 spring-boot-cache를 활용한 **Global cache server**를 구축  
**JMeter**를 이용한 Metric Data 분석으로 **리소스 관리**

## 프로젝트 진행 - Global cache server

- 22:40 ~ 22:41 구간은 캐싱 처리를 하지 않았습니다.
- 22:43 ~ 22:44 구간은 캐싱 처리를 진행하였습니다.



CPU Usage

### CPU 사용량과의 관계.

- 캐시는 자주 사용되는 데이터를 메모리에 저장하여 CPU가 데이터를 메모리에서 직접 읽습니다.
- CPU가 하드 디스크에서 정보를 가지고 오지 않아도 되기 때문에 CPU 사용량이 줄어듭니다.



Direct Buffers

### Memory와의 관계.

- 캐싱을 사용하면 메모리 할당 및 프로모션 작업이 증가합니다.



Classes Loaded

### Classes Loaded와의 관계.

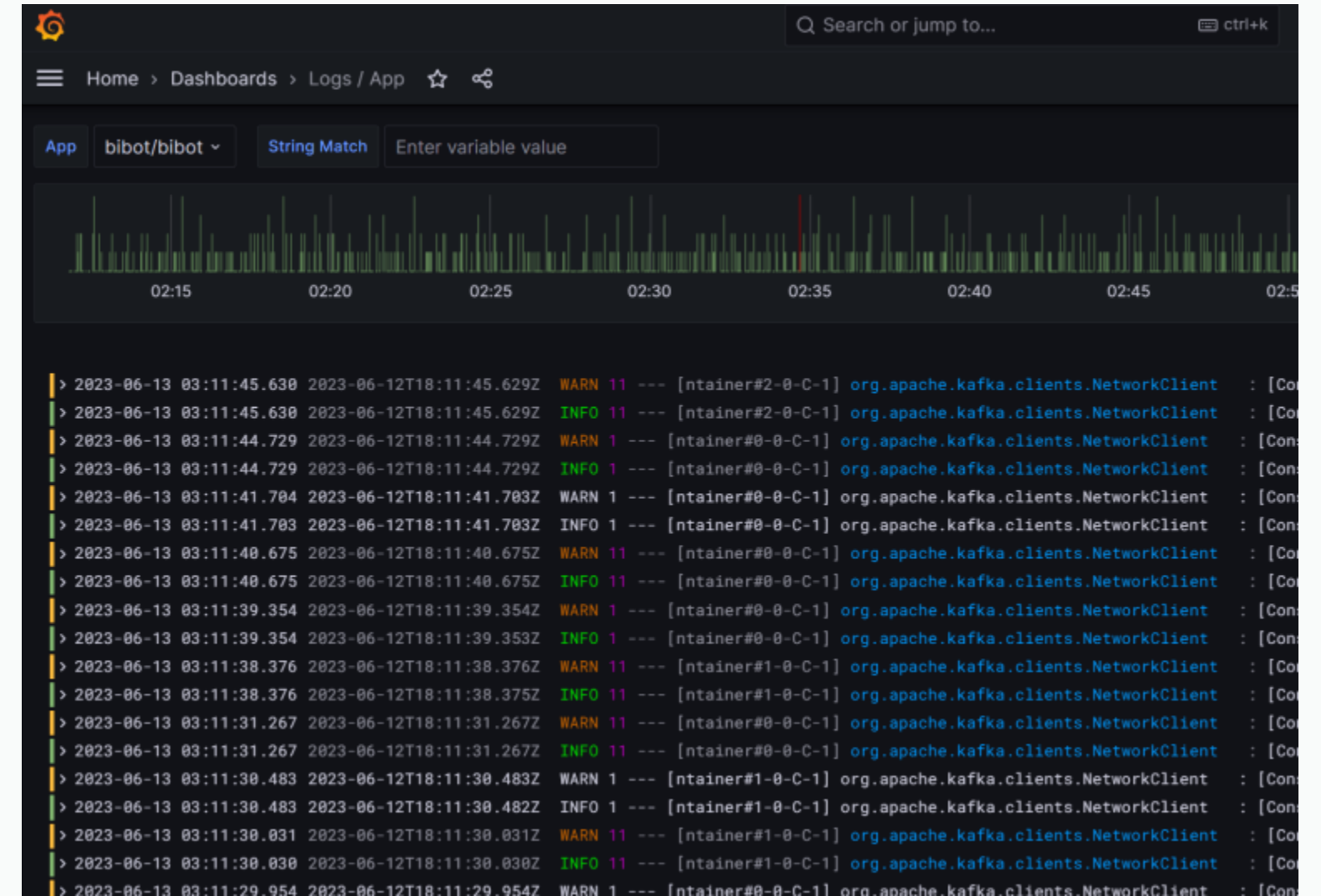
- 캐싱 처리를 적용할 때 클래스 로드의 증가가 발생할 수도 있습니다.
- 캐싱을 구현하기 위해 사용하는 라이브러리, 프레임 워크에서 자체적인 캐싱 기능을 사용한다면 관련된 클래스들이 추가로 로드될 수 있으며, 캐시 키/값의 직렬화, 역직렬화를 수행하는 관련 클래스들이 로드될 수 있습니다.



Hikari Connections

Grafana를 이용한 Metric Data 분석으로 테스트 시 리소스 추적

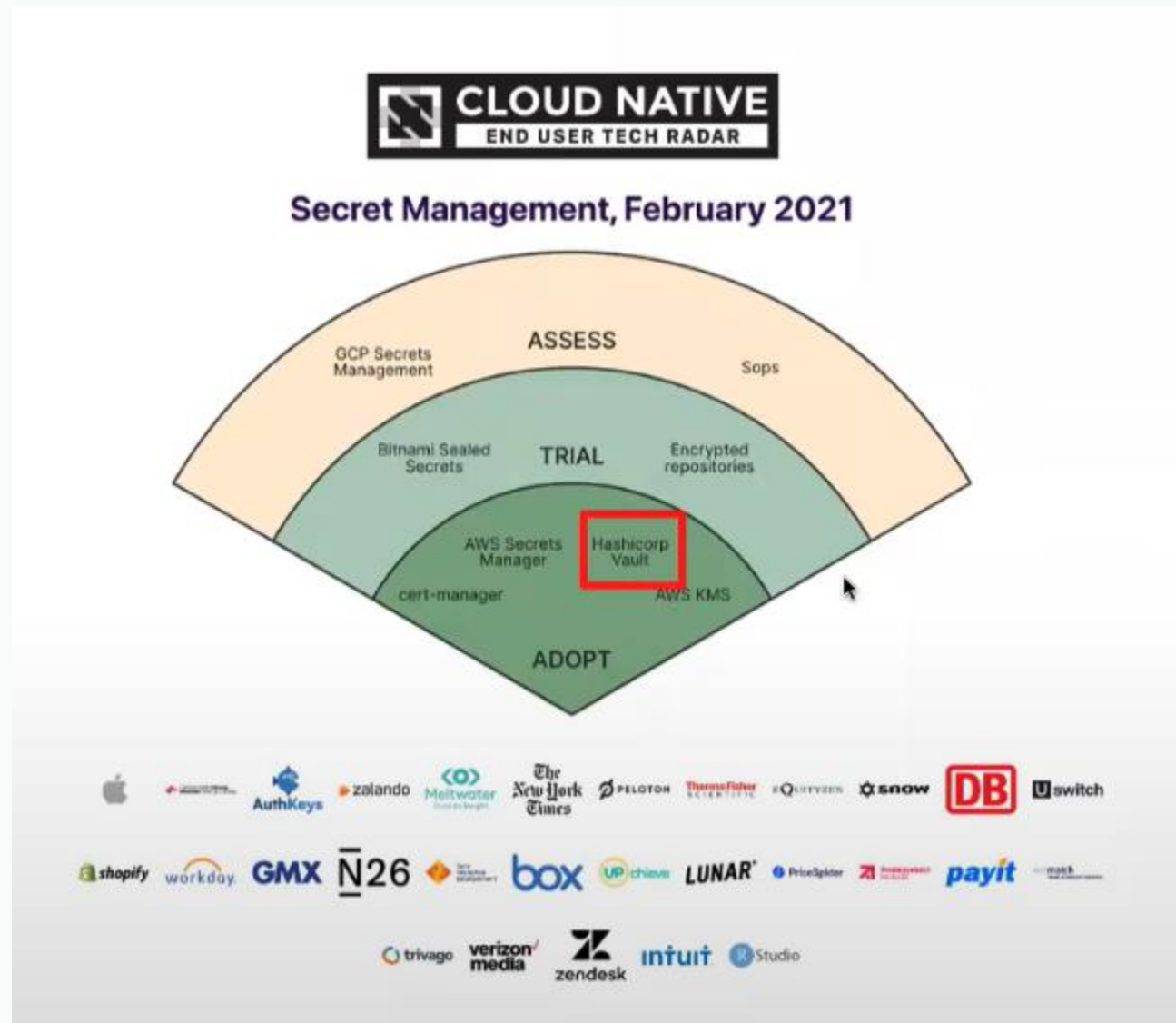
# 프로젝트 진행 - PLG Stack



Prometheus, Loki, Grafana(PLG)를 활용한 Kubernetes 리소스, 로그 관리 목적 **모니터링 시스템 구축**



# 프로젝트 진행 - Vault



Secrets Access Policies Tools

< secrets < secret < bibot

secret Version 2

Secrets Configuration

Q bibot/ Tab to autocomplete

cloak

db

jwt

mail

ocr

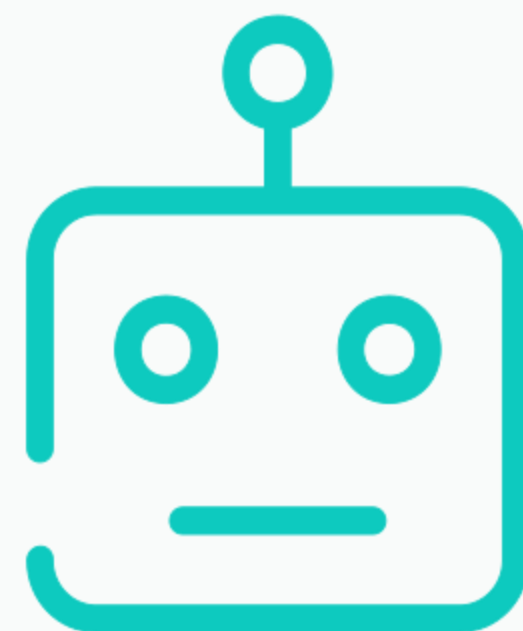
```
datasource:
  driver-class-name: org.postgresql.Driver
  url: jdbc:postgresql://${host_name}/users
  username: ${db_name}
  password: ${db_pwd}

data:
  redis:
    host: ${redis_host}
    port: ${redis_port}
  mail:
    host: smtp.gmail.com
    port: 587
    username: ${mail_id}
    password: ${mail_pwd}
    properties:
      mail.smtp.auth: true
      mail.smtp.starttls.enable: true
  kafka:
    bootstrap-servers: ${kafka_url}
  output:
    ansi:
      enabled: always
  security:
    oauth2:
      resourceserver:
        jwt:
          jwk-set-uri: ${jwk_uri}
          issuer-uri: ${iss_uri}
```

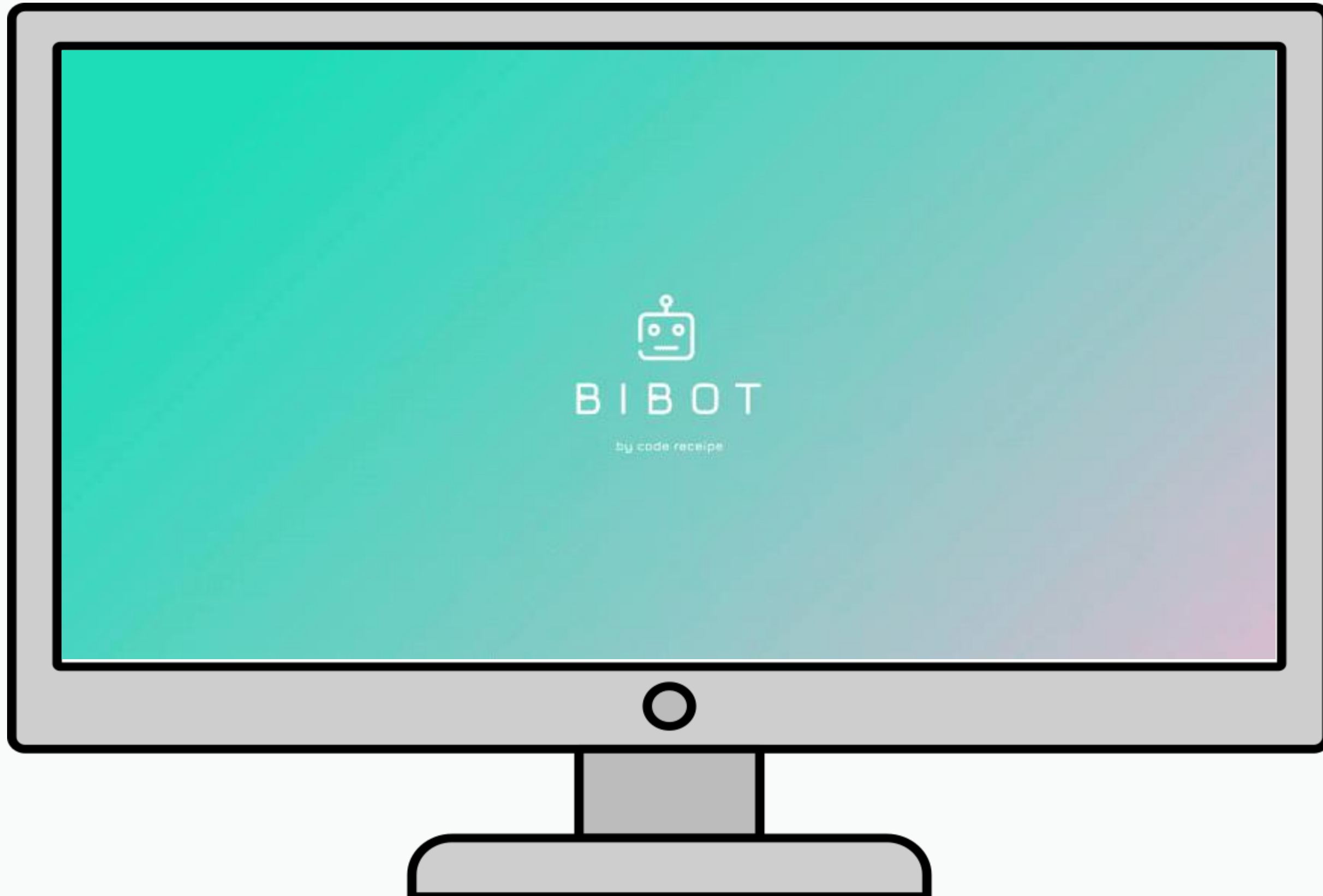
Kubernetes에서 Secret 관리를 위한 Vault 사용으로 보안 강화와 관리 간소화

## 4. 시연

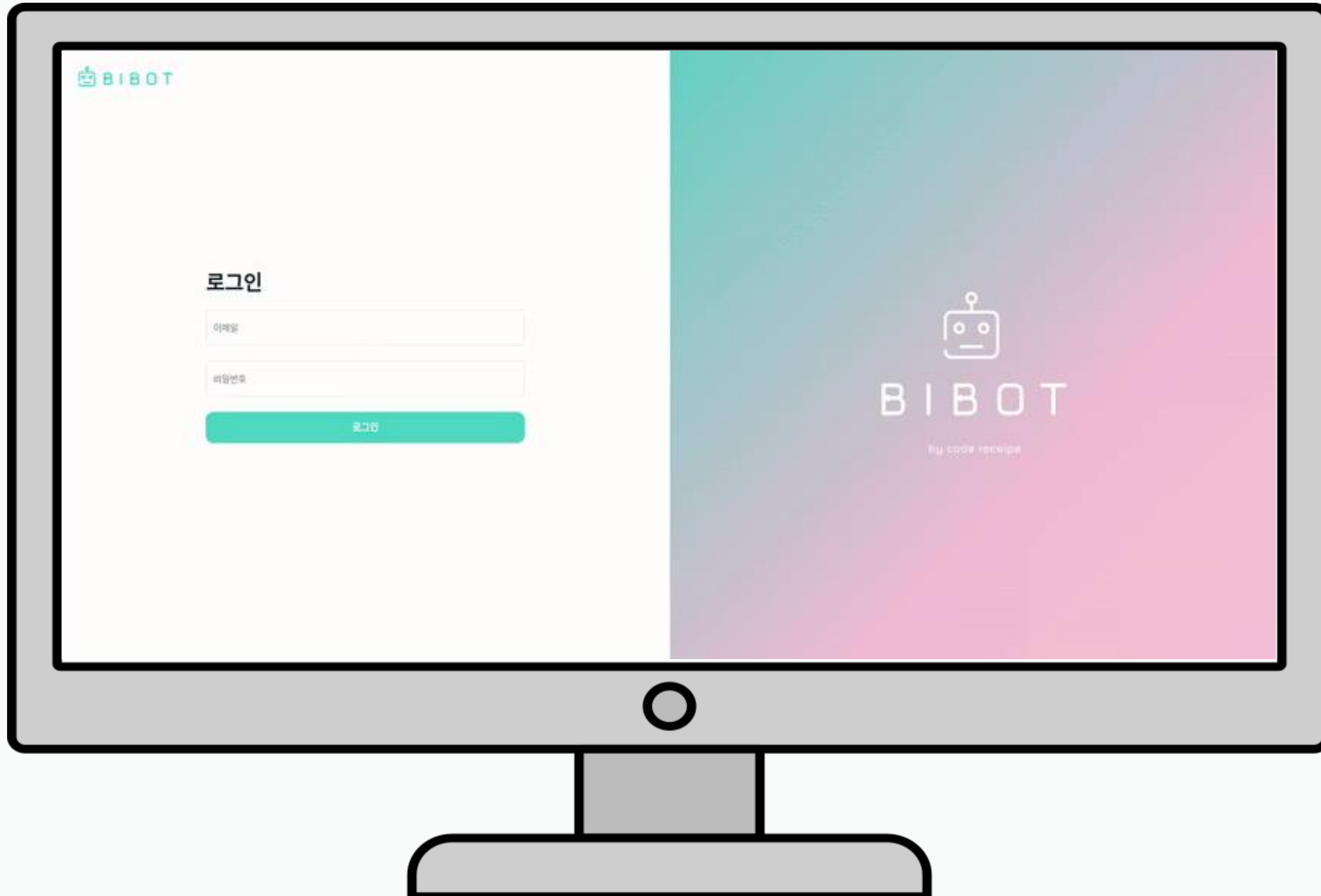
---



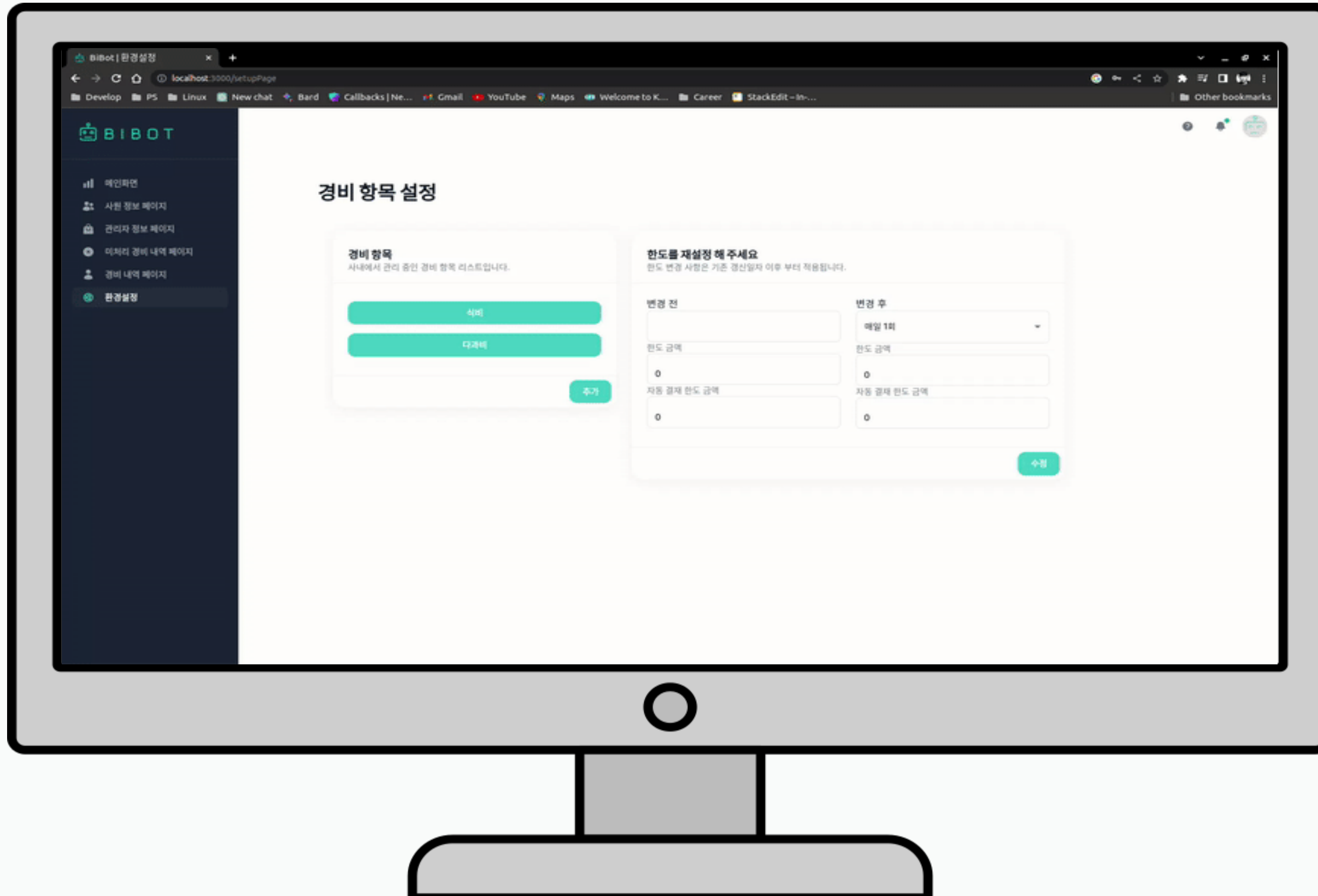
## 시연 - 관리자 메인 화면



## 시연 - 관리자 로그인 화면

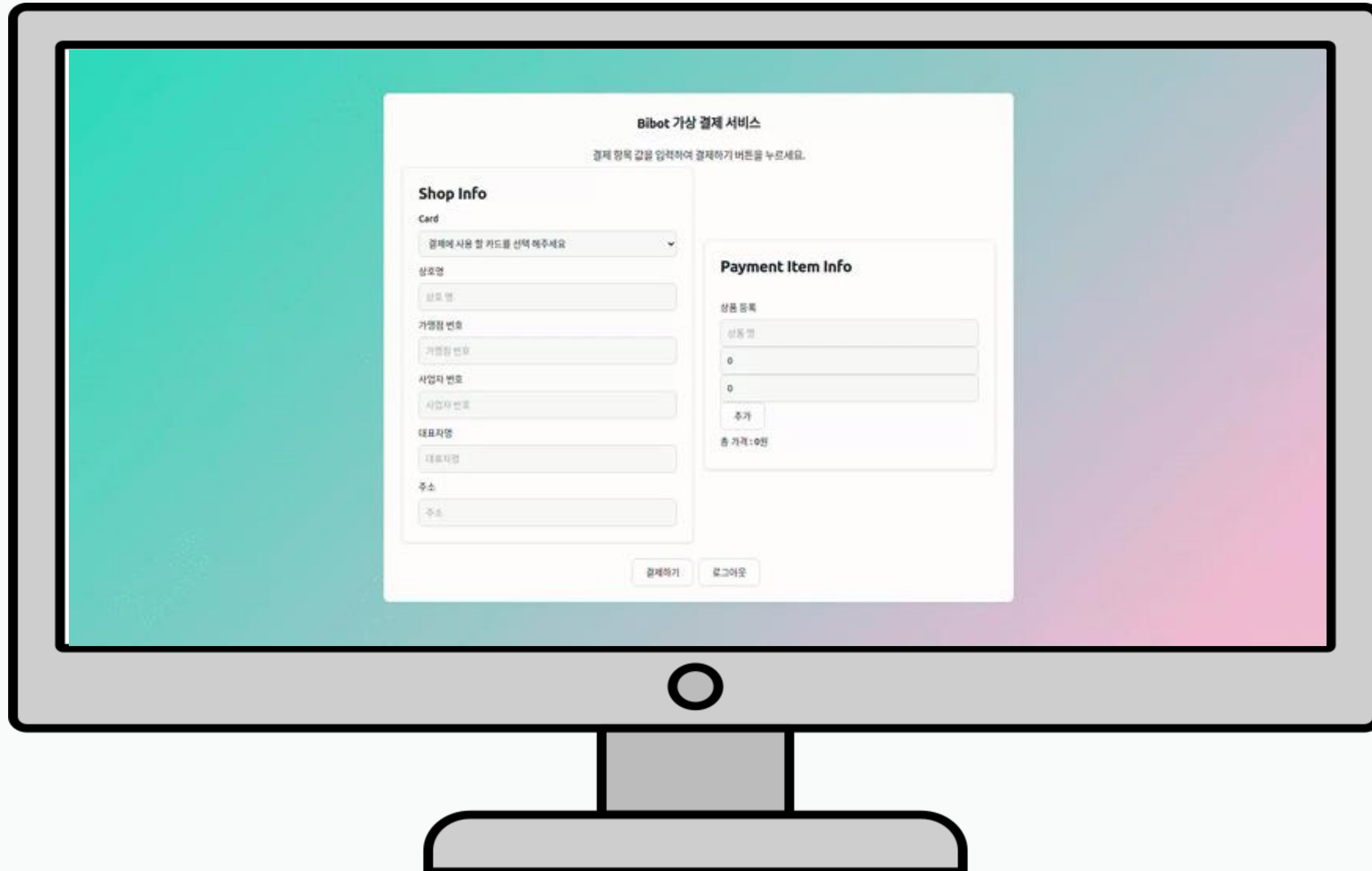


## 시연 - 관리자 경비 환경 설정





## 시연 - 영수증 이미지 생성



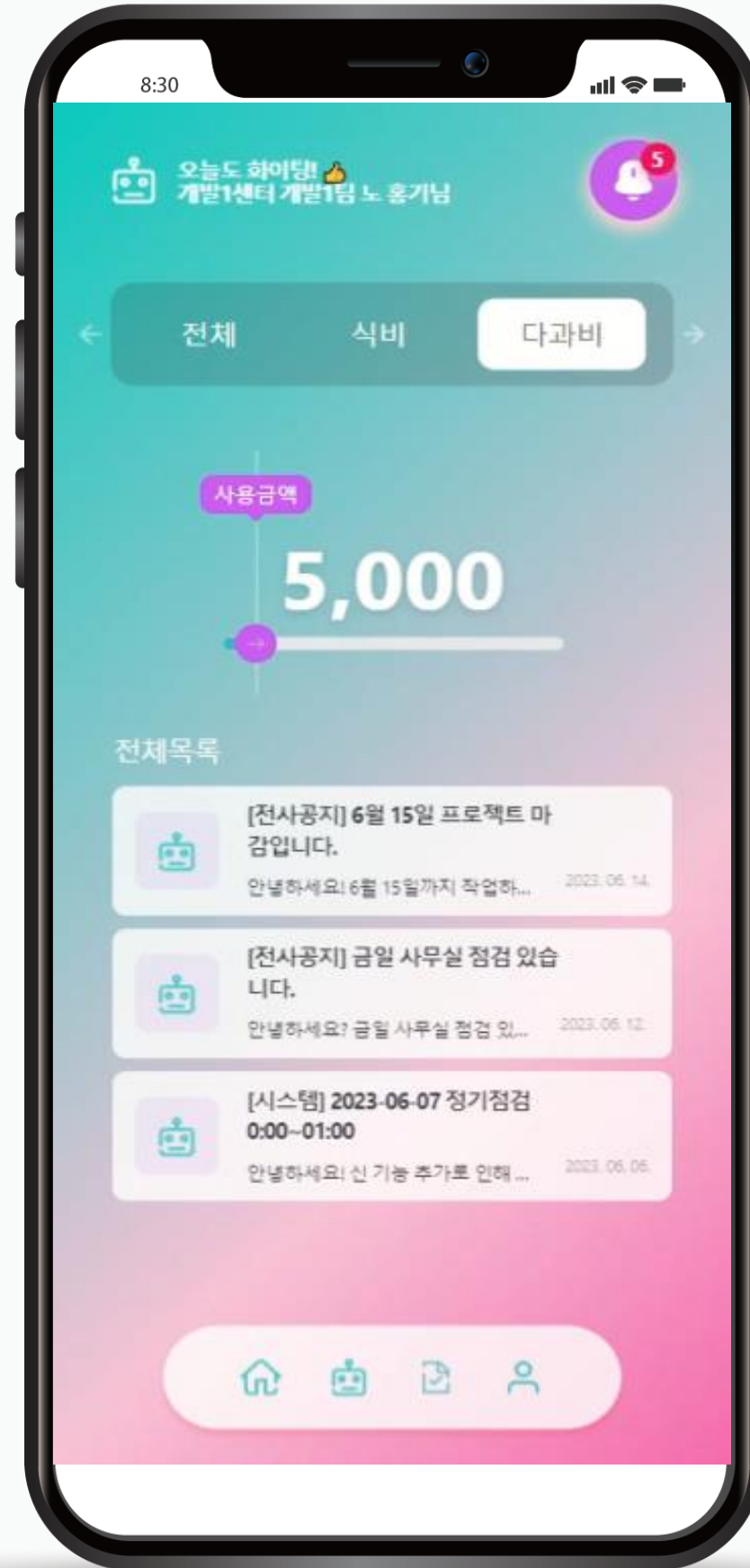
## 시연 - 메인 화면 및 공지 기능



## 시연 - 경비처리 성공 케이스

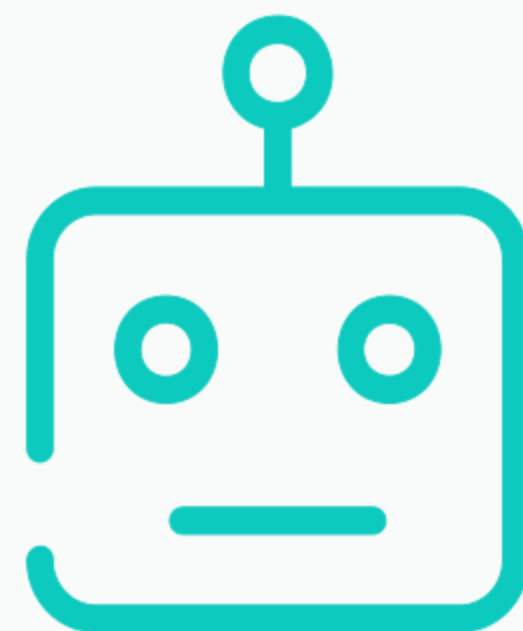


## 시연 - 경비 처리 실패, 재전송



## 5. 후기

---



# 후기

README.md

## Bibot-org / Code recipe

반갑습니다! Bibot 자동 경비처리 프로젝트 Code Recipe 팀 리포지토리입니다.



😊 산출물 Wiki 바로가기 😊

### 프론트엔드 Wiki

- 요구사항 정의서
- 화면 정의서
- 기능 명세서

### 어드민 Wiki

- 화면 정의서
- 기능 명세서

### 백엔드 Wiki

- API 문서
- 아키텍처 문서
- ERD 문서

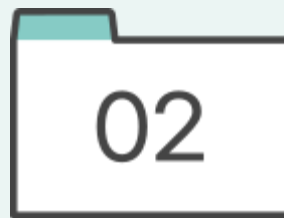
코드레시피 깃허브 주소  
-> <https://github.com/BiBot-org>

# 후기



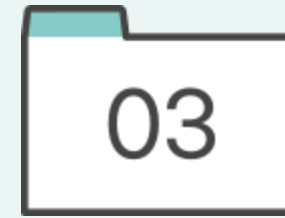
## 협업툴을 이용한 개발

협업툴과 git 정책에 따라 개발을 진행하여 프로젝트 진행시 협업 및 관리가 잘 이루어짐



## MSA 환경 개발

MSA 서비스를 구현하면서 Next.js, TypeScript 이벤트기반 아키텍처, Kubernetes 등 여러 기술 스택을 사용



## 기업 연계 프로젝트

기업 연계 프로젝트를 수행하면서 B2B 서비스에 대한 이해와 실제 기업에서 사용하는 경비처리 로직을 개선함



# 후기 - 기획 산출물

## 요구사항 정의서

## 화면 정의서

## API 명세서

### 사용자

식별코드	요구명	설명	비고
RQ-USR-01	초기 셋업	<ul style="list-style-type: none"> <li>사용자는 마켓 (웹스토어 및 플레이스토어) 에서 애플리케이션을 설치할 수 있어야 한다.</li> <li>사용자는 회사 별 워크스페이스 URL을 입력해서 자신의 회사 워크스페이스에 접속할 수 있어야 한다.                             <ul style="list-style-type: none"> <li>해당 설정이 적용되지 않는다면 사용자는 서비스를 사용할 수 없다.</li> </ul> </li> </ul>	
RQ-USR-02	로그인 / 로그아웃	<ul style="list-style-type: none"> <li>사용자는 발급 된 사원 전용 계정을 통해 해당 서비스에 로그인 할 수 있어야 한다.</li> <li>사용자는 회원가입을 직접 진행할 수 없다.                             <ul style="list-style-type: none"> <li>사내 시스템과 별개로 서비스는 Third Party로써 제공된다.</li> </ul> </li> </ul>	
RQ-USR-03	비밀번호 변경	<ul style="list-style-type: none"> <li>사용자는 자신의 비밀번호를 변경할 수 있어야 한다.</li> </ul>	
RQ-USR-04	유저 정보 확인 (소속 부서, 직급)	<ul style="list-style-type: none"> <li>사용자는 자신의 정보를 해당 앱에서도 확인할 수 있어야 한다.</li> <li>확인할 수 있는 정보는 아래와 같다.                             <ul style="list-style-type: none"> <li>소속 부서</li> <li>경비 처리 한도                                     <ul style="list-style-type: none"> <li>지금까지 사용한 경비</li> <li>앞으로 사용할 수 있는 경비</li> </ul> </li> </ul> </li> </ul>	
RQ-USR-05	영수증 경비 처리 요청	<ul style="list-style-type: none"> <li>사용자는 업무 시 사용한 경비 처리를 해당 시스템을 통해 할 수 있다.</li> <li>사용자는 영수증의 사진을 찍음으로써, 영수증 데이터 취합 서비스에 데이터 분석 요청을 보낼 수 있다.                             <ul style="list-style-type: none"> <li>하나의 결제 건에서 1개의 영수증이 등록된다.</li> <li>사용자는 영수증 분석 서비스의 분석결과를 직접 확인 할 수 있다.                                     <ul style="list-style-type: none"> <li>45000원이 총 합인데, 47000원이 입력 되었 던 경우</li> </ul> </li> </ul> </li> <li>사용자는 경비 처리 항목을 선택 할 수 있다. 처리할 수 있는 경비 항목은 아래와 같다. 경비 처리 항목은 사내 정책에 따라 달라질 수 있다.                             <ul style="list-style-type: none"> <li>식비</li> <li>유류비</li> </ul> </li> <li>사용자는 사내 정책에 따라 자동 결제 유무를 확인할 수 있다.                             <ul style="list-style-type: none"> <li>해당 요구사항은 사내 정책에 따라 달라 질 수 있다.                                     <ul style="list-style-type: none"> <li>한 번의 결제에서 자동 결제 처리되는 식비의 범위는?</li> </ul> </li> </ul> </li> <li>사용자는 카드 결제내역에서 경비 처리요청을 진행할 수 있다.</li> </ul>	
RQ-USR-06	영수증 경비 처리 내역 확인	<ul style="list-style-type: none"> <li>사용자는 처리하였던 경비들에 대한 내역을 조회할 수 있다.</li> <li>사용자는 기간 별로 경비 처리 내역을 조회할 수 있다.</li> <li>사용자는 처리 항목 별로 경비 처리 내역을 조회할 수 있다.</li> <li>사용자는 요청의 승인 여부에 따라 경비 처리 내역을 조회 할 수 있다.</li> </ul>	

### 애플리케이션(사용자)

화면 명	비고
메인 로고 화면	<ul style="list-style-type: none"> <li>앱 실행 및 로고화면</li> </ul>
Url 초기 셋업 화면	<ul style="list-style-type: none"> <li>최초 앱 접속 시 회사 URL 입력 화면</li> </ul>
로그인 화면	<ul style="list-style-type: none"> <li>로그인(이메일, 비밀번호)</li> <li>비밀번호 찾기</li> </ul>
비밀번호 찾기	<ul style="list-style-type: none"> <li>등록된 이메일로 임시 비밀번호 발급</li> </ul>
메인 화면	<ul style="list-style-type: none"> <li>유저 정보                             <ul style="list-style-type: none"> <li>유저 이름 / 팀 =&gt; 환영합니다</li> <li>경비 항목 별 경비 처리 현황                                     <ul style="list-style-type: none"> <li>ex) 식비 / 경비 처리 금액 / 앞으로 청구할 수 있는 금액</li> </ul> </li> </ul> </li> <li>알림 아이콘 (알림 개수)</li> <li>공지사항 ( ex) 가장 최근 3개-5개)</li> <li>하단 메뉴바                             <ul style="list-style-type: none"> <li>홈</li> <li>카드 사용내역</li> <li>결제 내역</li> <li>마이페이지</li> </ul> </li> </ul>
알림 내역	<ul style="list-style-type: none"> <li>알림 내역 (읽음 여부 표시)</li> </ul>
공지 사항 목록 화면	<ul style="list-style-type: none"> <li>공지사항 페이지 조회</li> <li>공지사항 조건 별 정렬 (미정)                             <ul style="list-style-type: none"> <li>일반 공지</li> <li>시스템 공지</li> </ul> </li> </ul>
공지 사항 상세 화면	<ul style="list-style-type: none"> <li>해당 공지사항 상세 내역</li> <li>이전글 / 다음글</li> </ul>
카드 목록 및 카드 결제 내역 조회 화면	<ul style="list-style-type: none"> <li>카드 등록 / 삭제</li> <li>사용자에게 등록 된 카드 목록</li> <li>카드 별 결제 내역 조회 (결제 별 결제 요청 여부)</li> </ul>
카드 사용내역 결제 요청 화면	<ul style="list-style-type: none"> <li>영수증 촬영 버튼 및 영수증 이미지 등록</li> <li>경비 항목 선택 버튼</li> <li>결제 요청</li> </ul>
결제 내역 목록 화면	<ul style="list-style-type: none"> <li>전체 사용 경비와 남은 경비</li> <li>결제 내역 목록</li> <li>검색조건(최신순, 기간순)</li> </ul>

### Receipt Service

#### Receipt

API-RECEIPT-001	영수증 조회	/api/v1/receipt	GET	id : number	<pre> 1 { 2   status : HttpStatus, 3   data : { 4     id : String, 5     cardId : number, 6     paymentDestination : String, 7     amount : number, 8     approvalId : String, 9     isRequested : boolean 10  } 11  message : "SUCCESS" 12 } </pre>
API-RECEIPT-002	영수증 사진 업로드	/api/v1/receipt/image	POST	<pre> 1 { 2   file : MultipartFile, 3   cardId : number, 4   categoryId : number, 5   paymentId : String, 6   userId : String 7 } </pre>	<pre> 1 { 2   status : HttpStatus, 3   data : {stragglURL : String}, 4   message : "SUCCESS" 5 } </pre>
API-RECEIPT-003		/api/v1/receipt/test	POST		

#### Department

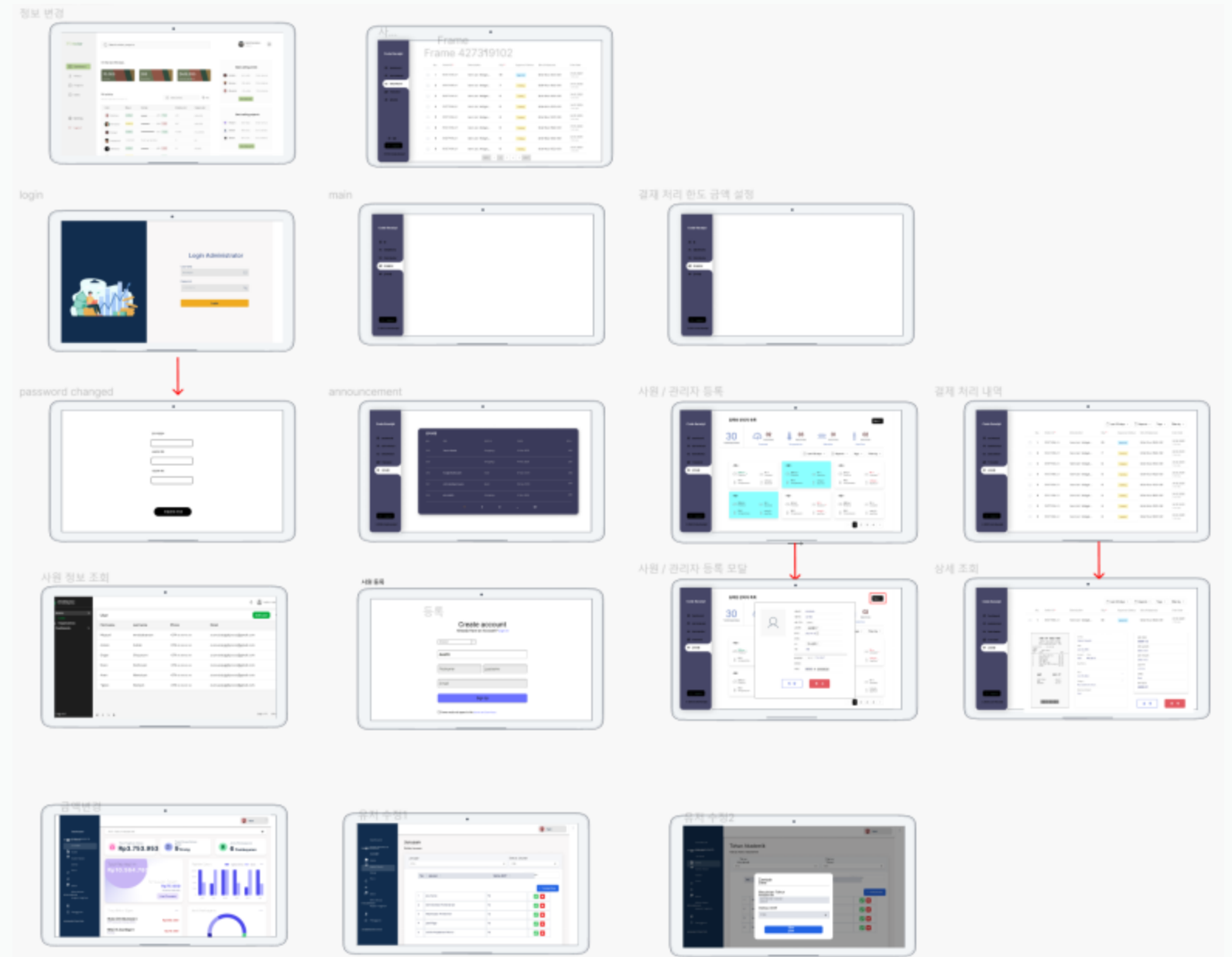
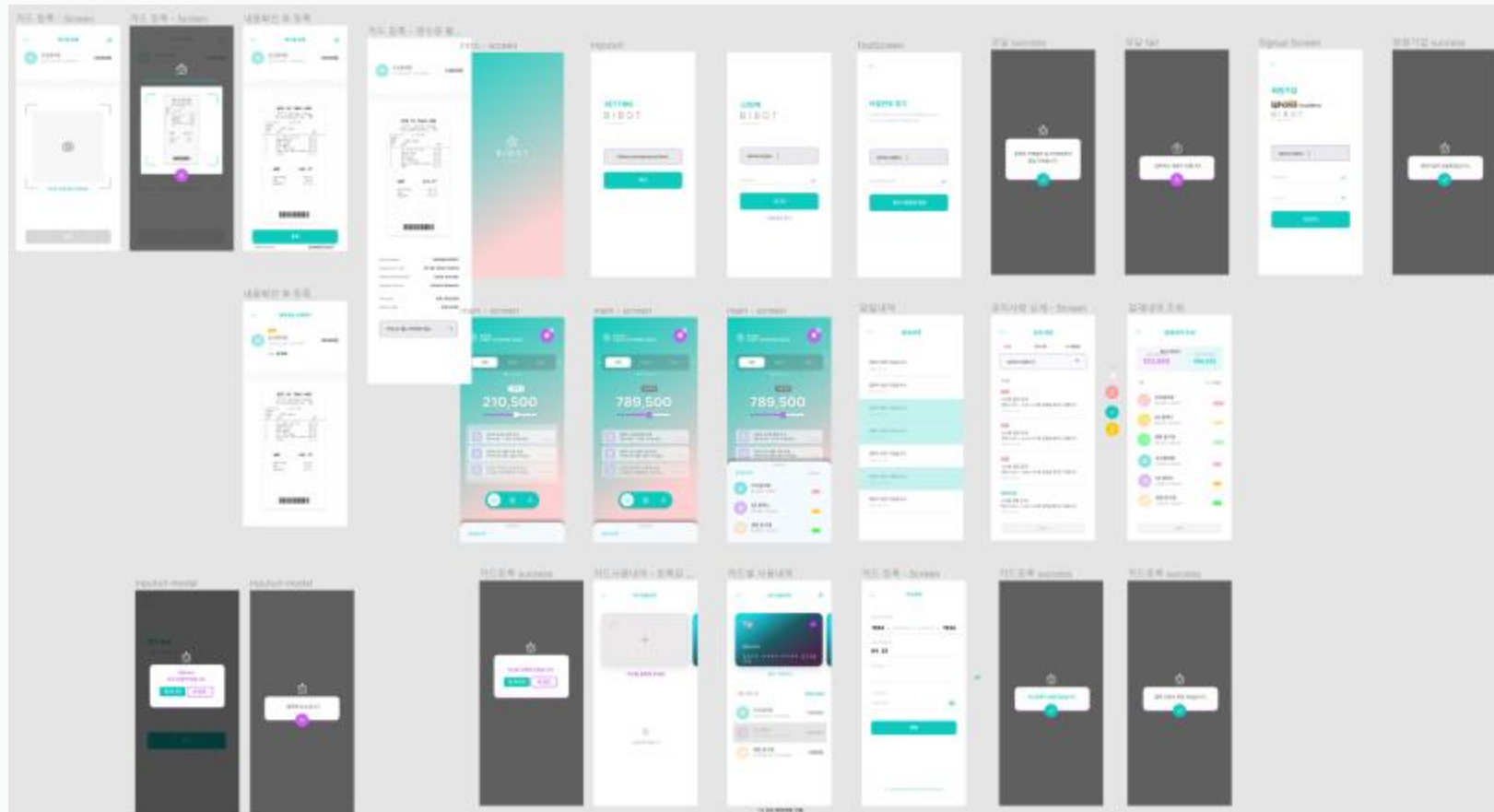
식별 코드	이름	URI	Method	Parameter	Return
API-USER-ADM-009	부서 추가	/api/admin/v1/department	POST	<pre> 1 { 2   name : string; 3 } </pre>	<pre> 1 { 2   status : HttpStatus, 3   data : {departmentId : number}, 4   message : "SUCCESS" 5 } </pre>
API-USER-ADM-010	모든 부서 정보 조회	/api/admin/v1/department/all	GET		<pre> 1 { 2   status : HttpStatus, 3   data : { 4     id : number, 5     name : string; 6   }, 7   message : "SUCCESS" 8 } </pre>
API-USER-ADM-011	모든 부서 상세 정보 조회	/api/admin/v1/department/info/all	GET		<pre> 1 { 2   status : HttpStatus, 3   data : { 4     department : { 5       id : number, 6       name : string 7     }, 8     team : { 9       id : number, 10      name : string 11     }, 12   }, 13   message : "SUCCESS" 14 } </pre>



# 후기 - 화면설계서

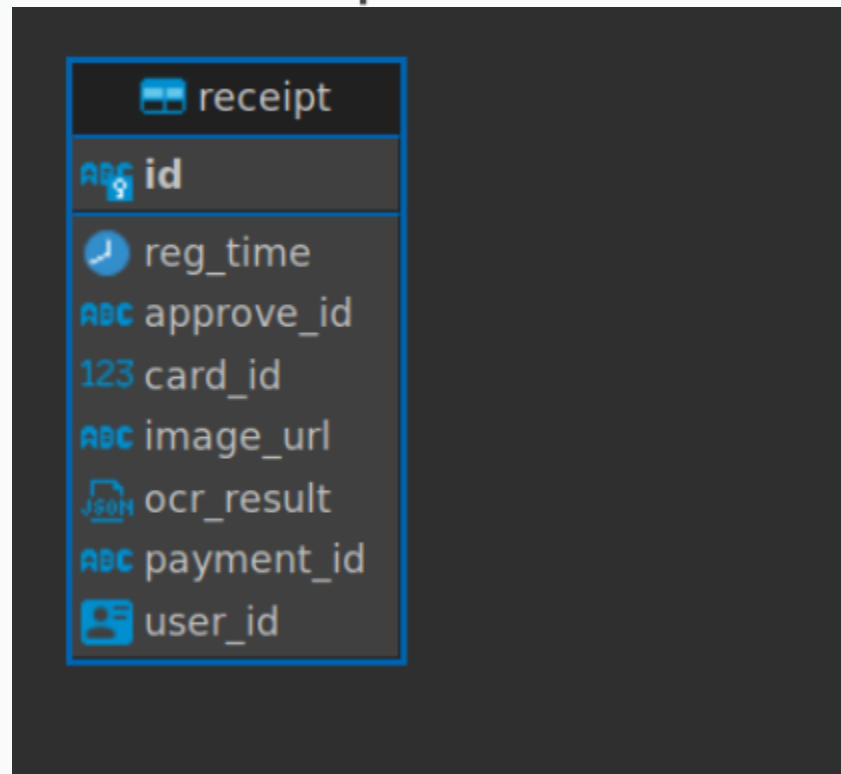
User

Admin

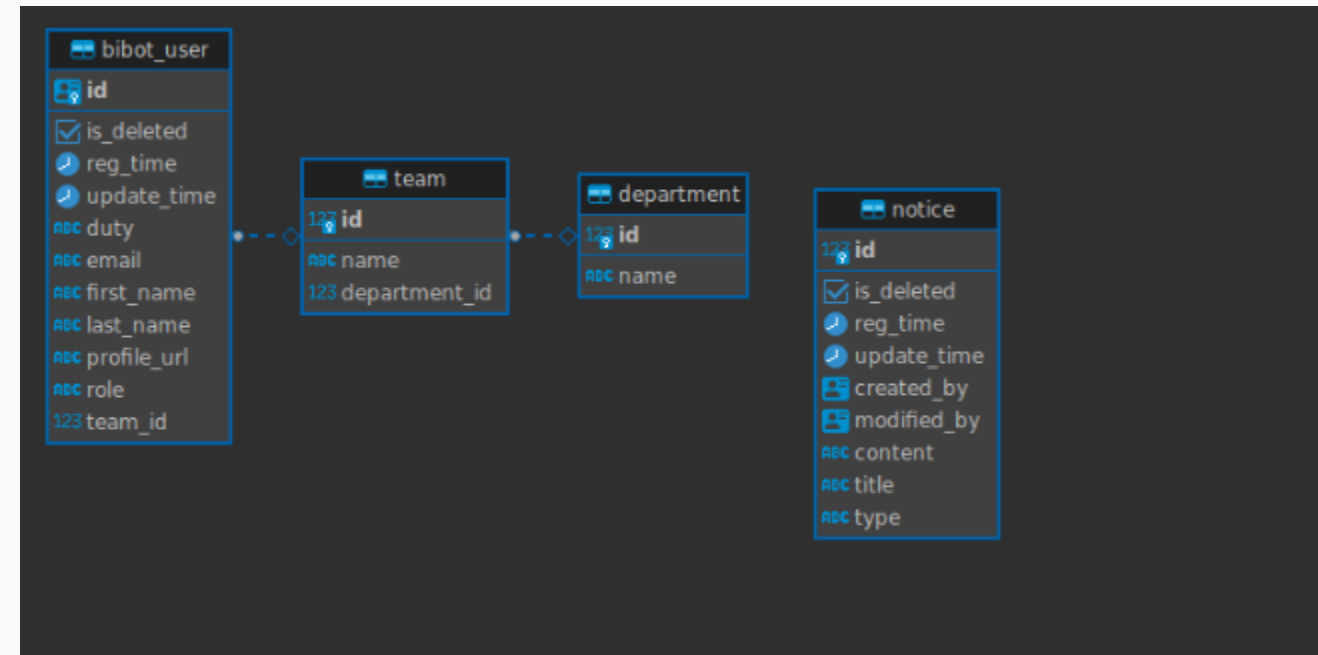


# 후기 - ERD Diagram

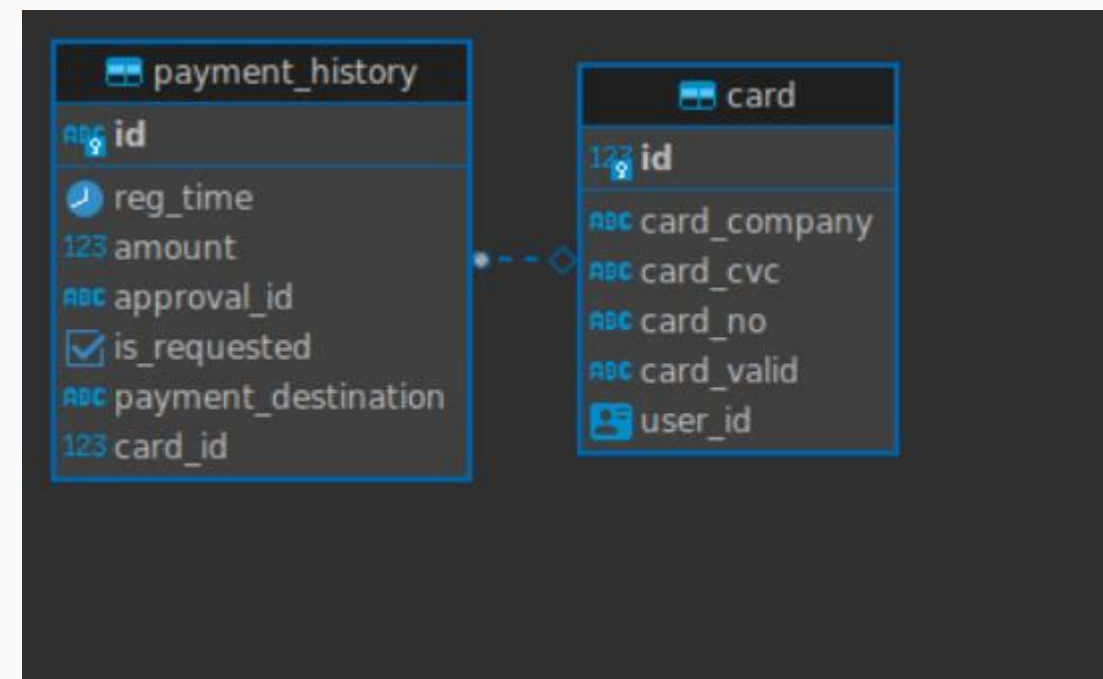
## Receipt Service



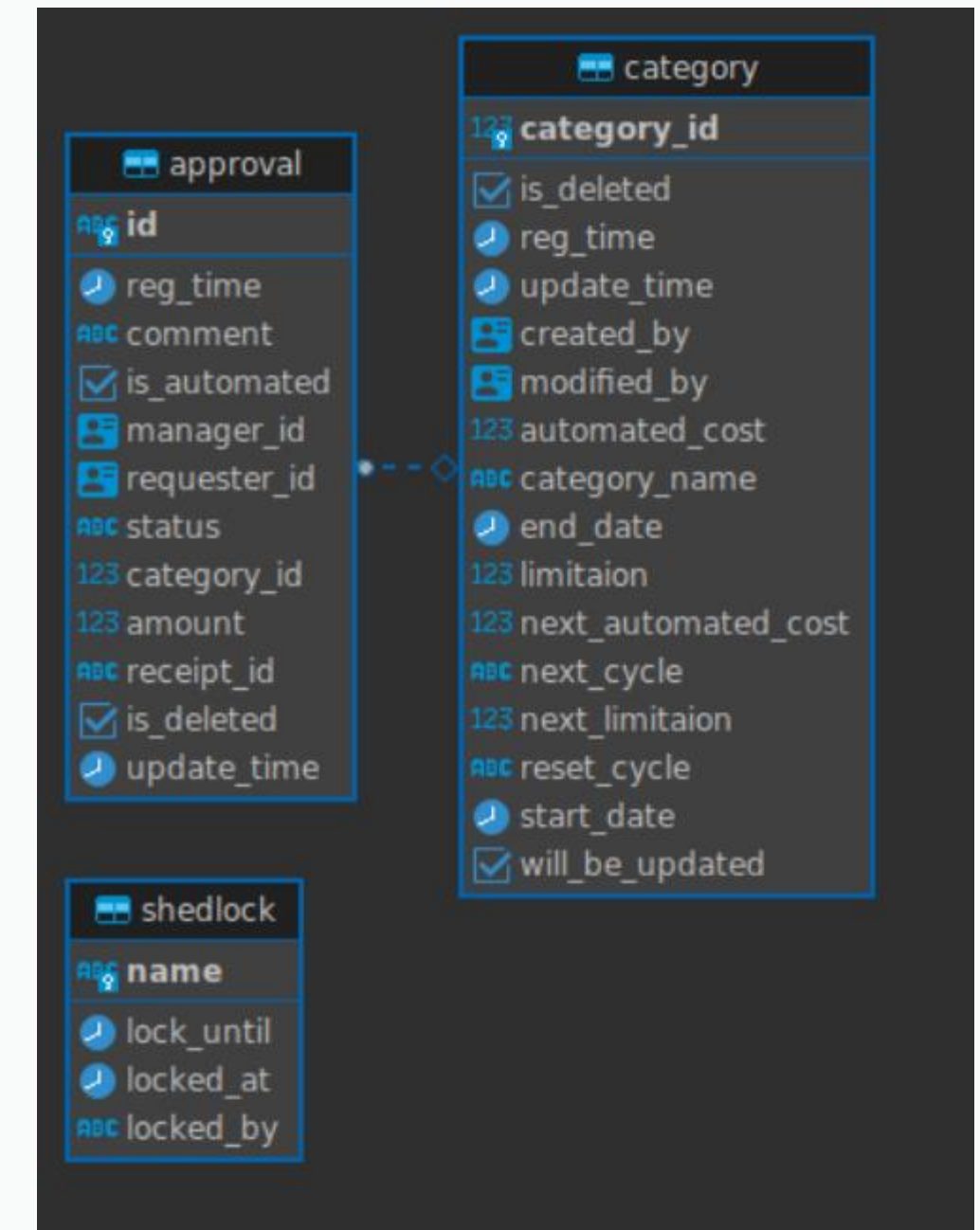
## User Service



## Card Service

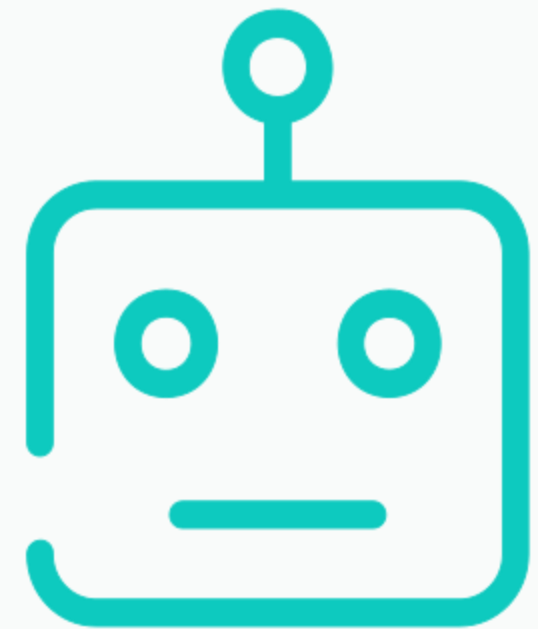


## Expense Service



## 6. 팀 소개

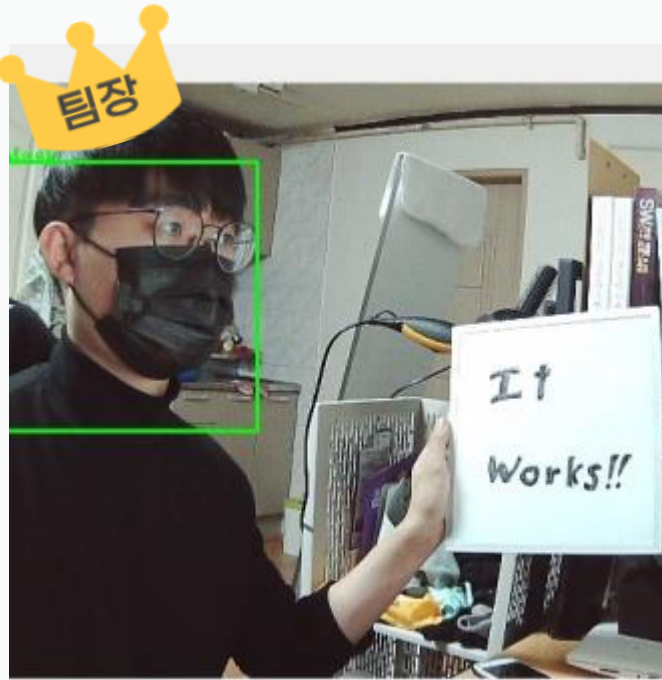
---







코드를 더 맛있게! 우리는 코드레시피!



### 손우진(FE&BE)

- 애플리케이션 아키텍트
- Kafka 비동기 개발
- SSO 구축(Keycloak)
- 앱인증 로직 개발 (NextAuth)
- 관리자 애플리케이션 개발



### 박노명(FE)

- 유저 애플리케이션 화면 및 기능 구성
- 유저 애플리케이션 퍼블리싱
- 서버 api(유저, 공지사항, 카드내역, 결제내역) 연동
- 이미지 업로드 기능



### 김효은(FE)

- 유저 애플리케이션 화면 및 기능 구성
- 유저 애플리케이션 퍼블리싱
- NextUI(components) 적용
- PWA



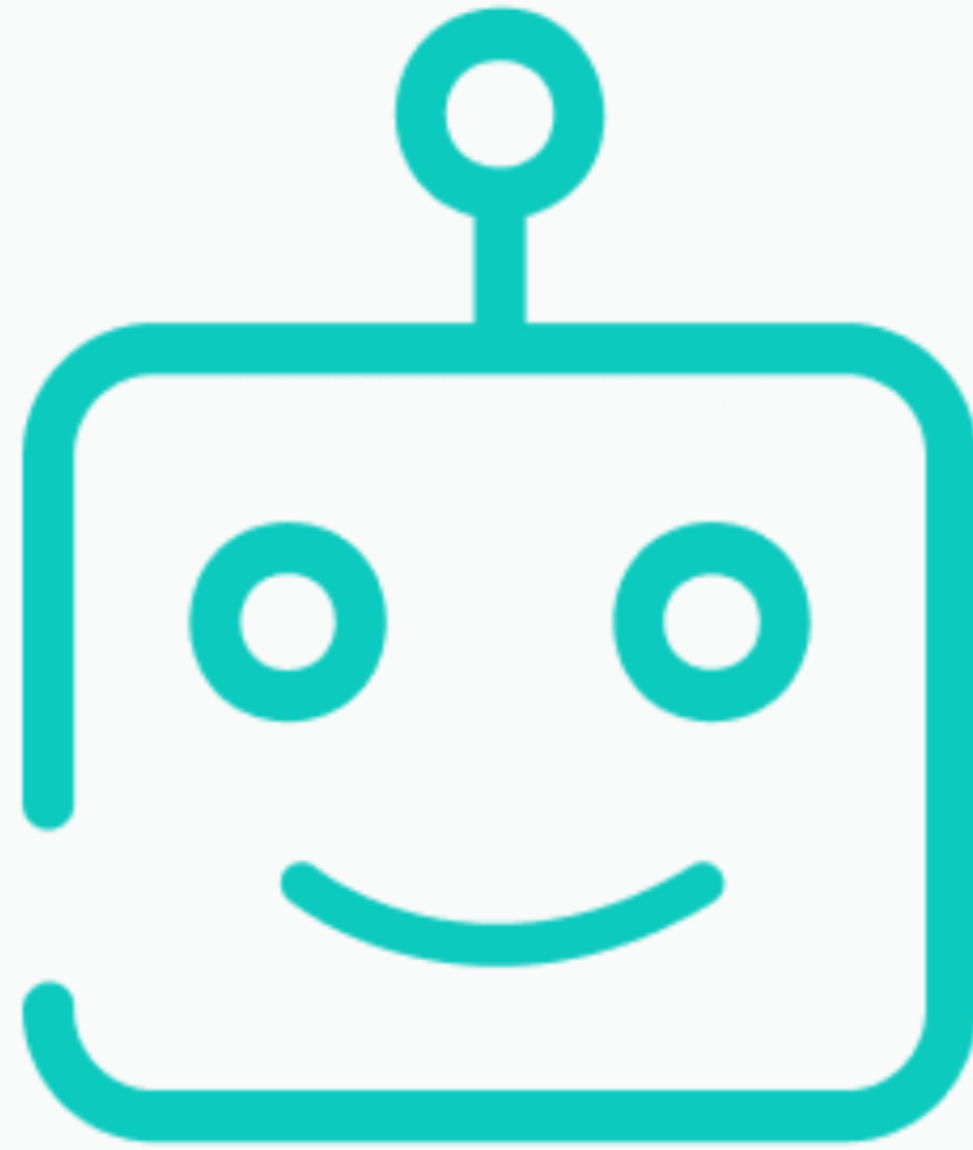
### 노홍기(BE)

- OCR API 연동 및 로직 개발
- 스케줄러 기반 경비처리 주기 업데이트 로직 개발
- Redis 기반 API 캐싱 환경 구축
- 테스트 엔지니어



### 박준석(DevOps)

- Google Cloud Platform k8s 서버 운영
- GitOps CI/CD 구축
- Vault 기반 시크릿 서버 구축
- Prometheus & Grafana & Loki 기반 로그 수집 및 모니터링 구축
- DB 서버 운영 및 연결



**Thanks!**