ErumPay

카드 실적·혜택 최적화 결제 서비스

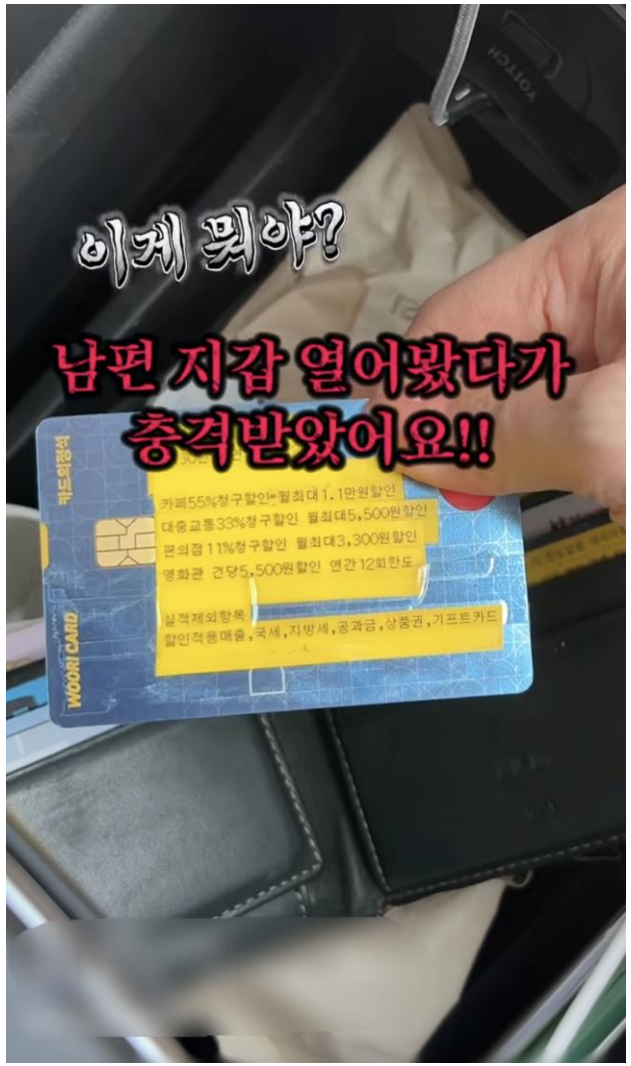
LG CNS AM INSPIRE CAMP 4기 2조 금융

CONTENTS

ErumPay

1. 프로젝트 개요

1-1. 문제 정의



출처 : https://www.instagram.com/reel/DZJjmGjJgdy/?igsh=MTZqazd0OGlkOHZ1Zw==

1-1. 문제 정의

카드 결제

여러 장의 카드 보유 → 최적 카드 선택의 어려움 → 최적 카드 혜택 손실

더치페이

친목 · 모임 → 한 사람이 결제 → 카드 혜택 활용 불가

저희는 결제 직전의 3초를 바꾸고 싶습니다

ErumPay

1-2. 페르소나(1)

전략적 소비형 워킹대디

이준혁 (35)

“1원이라도 놓치면 손해"

보유 카드 5개 이상.
교육•마트 등 목적별 카드를 사용하지만 실적 관리에 한계를 느낌

- **실적 절벽**: 월말마다 남은 실적 계산의 피로감
- **정보 과부하**: 결제 직전 어떤 카드가 최선인지 혼란
- **NEEDS**: 결제 시점에 정답을 알려주는 지능형 비서

1-2. 페르소나(1)

효율을 추구하는 가맹점주

은종혁 (40)

“점심시간 회전율이 곧 매출"

오피스 상권 6년차 사장님
더치페이 병목현상으로 인한 줄서기가 가장 큰 스트레스

- **결제 병목**: 단체 손님 각자 결제 시 POS 정체
- **원장 파편화**: 단일 매출의 분할로 인한 원장 관리 부담
- **NEEDS**: 한 번의 스캔으로 끝나는 통합 정산

1-3. 해결 방안

ErumPay

의사결정의 자동화

사용자의 혜택과 가맹점의 효율을 동시에 극대화

- 의사결정 자동화
- 혜택 극대화
- 정산 과정 최소화
- 가맹점 효율 향상

1-4. 서비스 소개

ErumPay

사용자의 결제 데이터를 기반으로 카드 혜택과 실적 조건을 분석하여
최적의 결제 방식을 추천하는 금융 결제 플랫폼

지능형 카드 추천

분할 결제

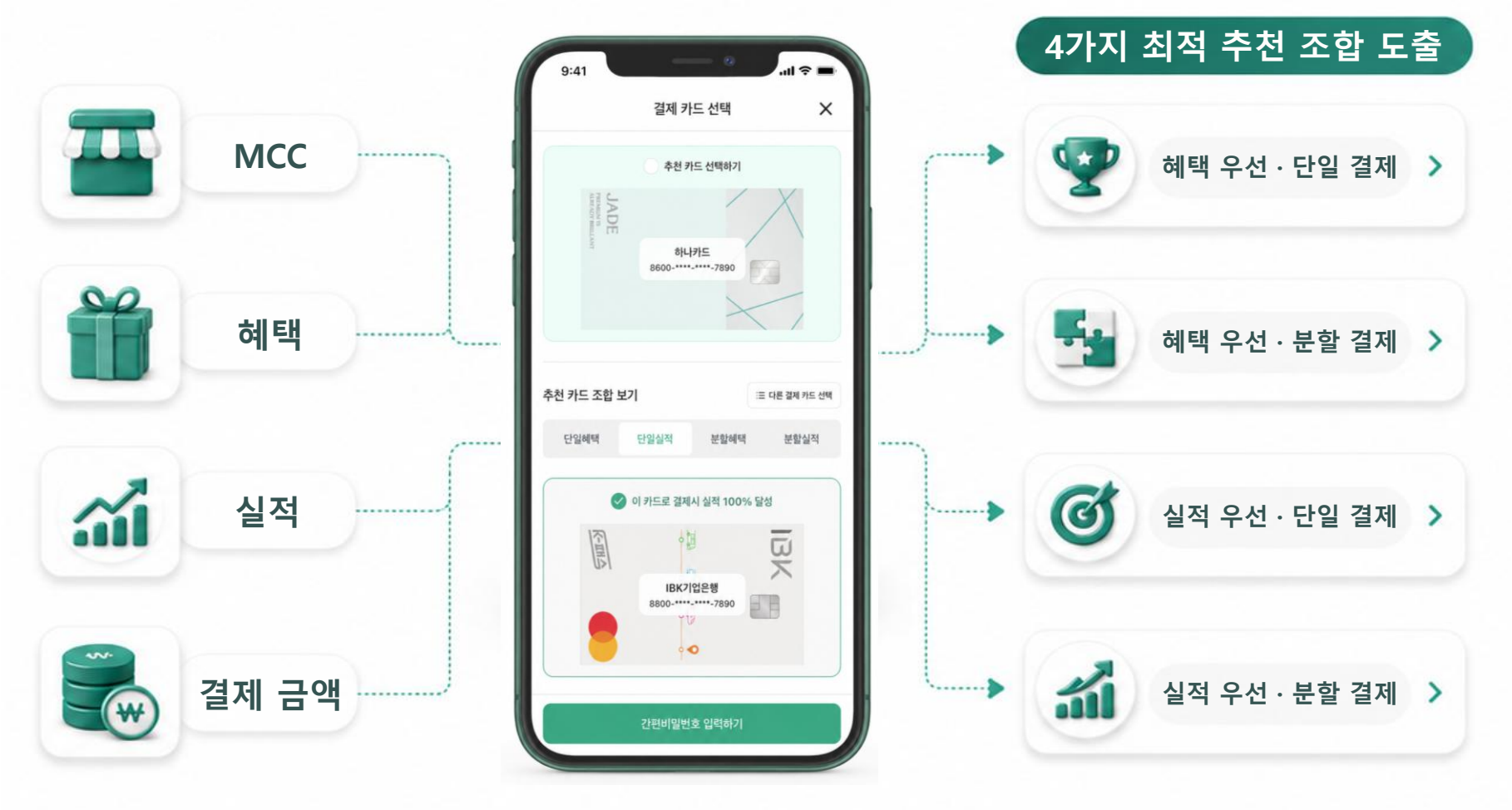
더치 페이

원격 결제

2. 핵심기능

2-1. 지능형 카드 추천

지능형 카드추천이란? 결제 시점의 정보를 실시간 분석해 가장 유리한 카드 또는 카드 조합을 추천



ErumPay

2-2. 분할 결제

분할 결제 ≠ 할부

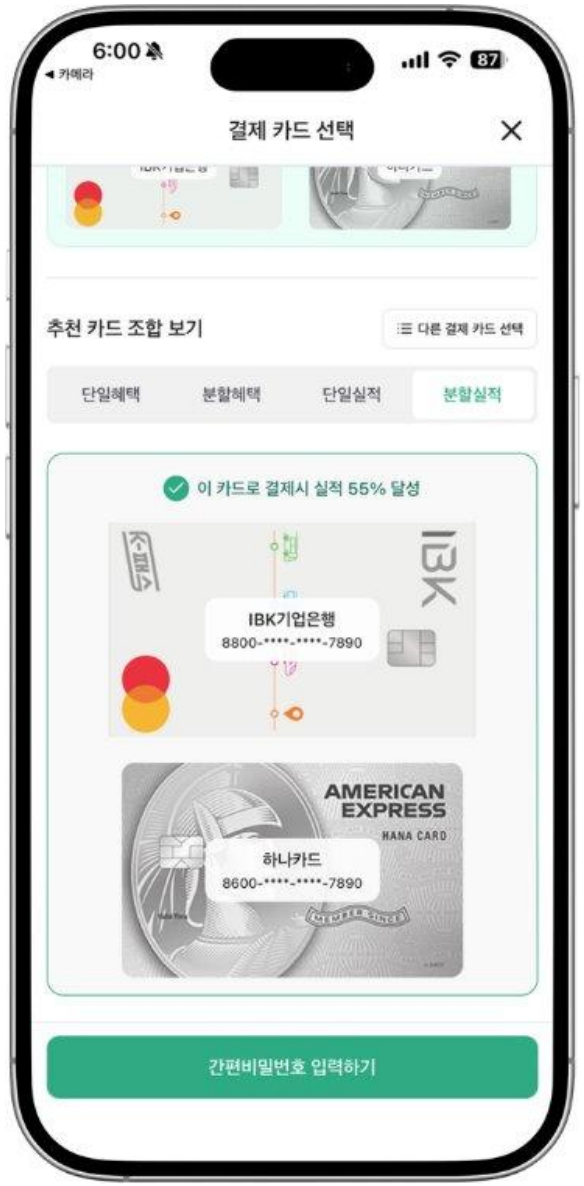
하나의 결제 금액을 여러 카드에 나누어 결제함으로써,

카드별 혜택 한도나 실적 조건을 더 효율적으로 활용하는 방식

2-2. 분할 결제

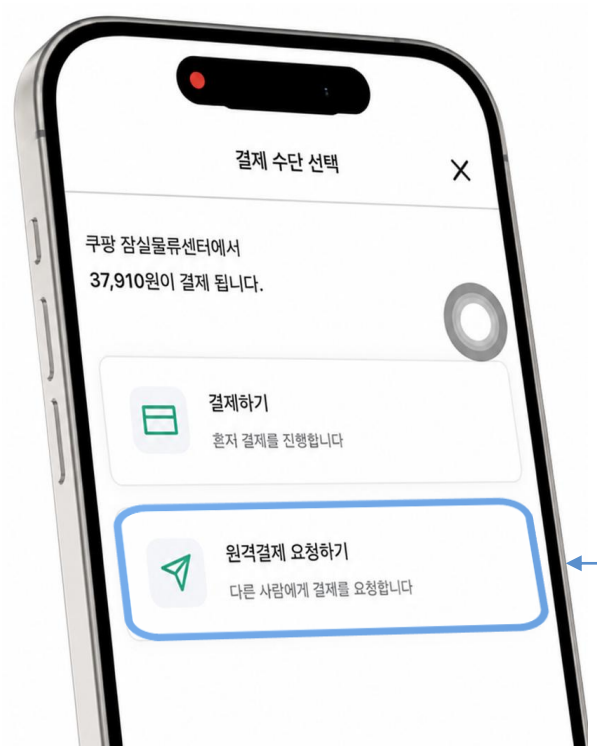
6만원 결제 시나리오

카드	조건
A카드	카페 10% 할인, 할인 한도 3,000원 남음
B카드	전월 실적까지 20,000원 부족
C카드	기본 적립률 높음



2-3. 원격 결제

요청자와 결제자가 떨어져 있어도 원격으로 결제를 요청할 수 있는 기능

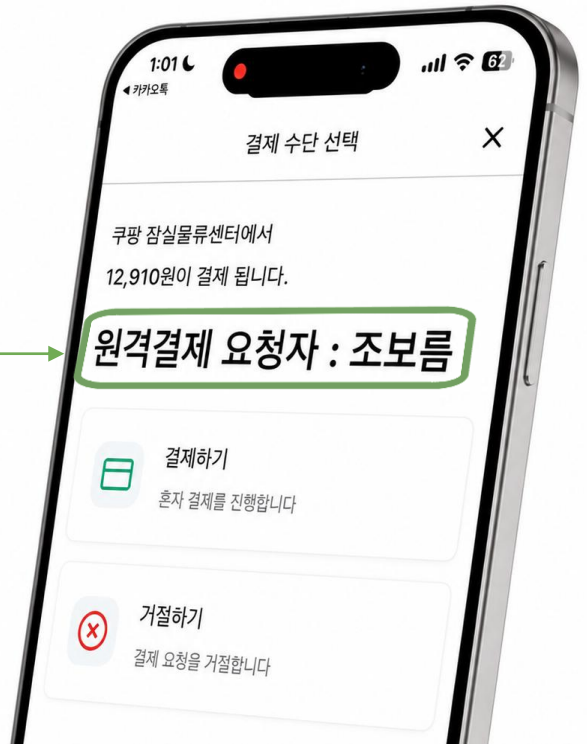


요청자

결제 내역
생성 후 요청 전송

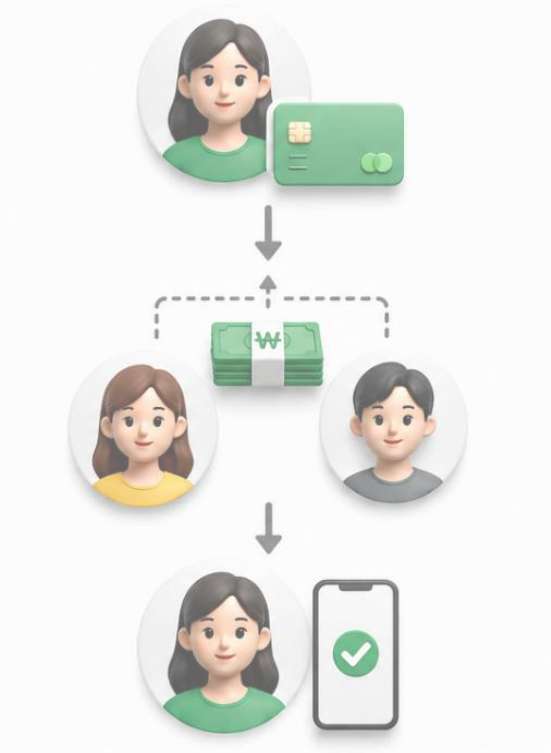
대리자

요청 확인 후
결제 진행



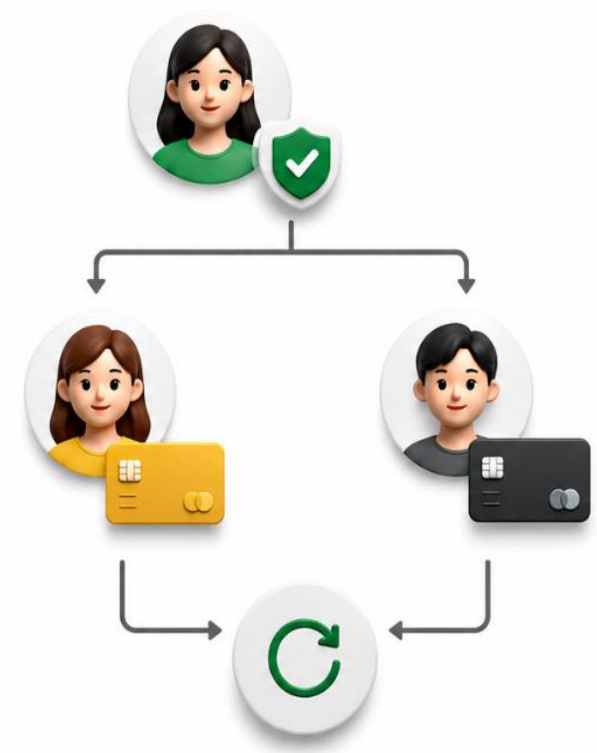
2-4. 더치페이

대표자 가승인 → 참여자 결제 → 대표자 void 구조로 결제 시점에 각 참여자가 자신의 카드 혜택을 챙김



1. 대표자 선결제
2. 참여자 송금
3. 대표자 정산 확인

기존 방식

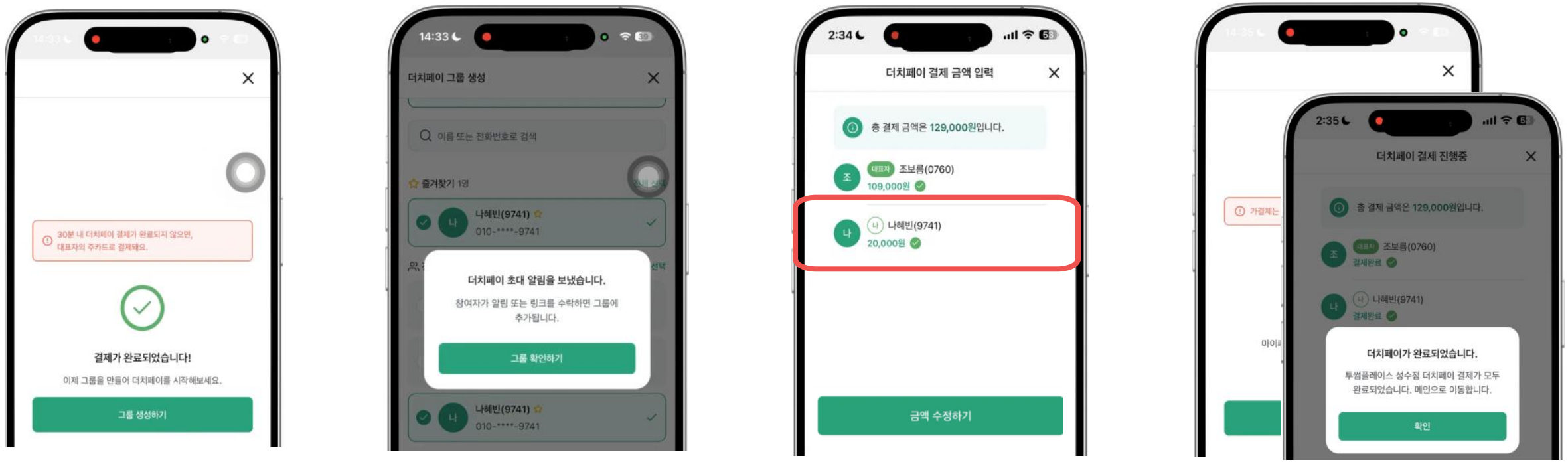


1. 대표자 가승인
2. 참여자 결제
3. 대표자 void 처리

이룸 페이

2-4. 더치페이

대표자 가승인 → 참여자 결제 → 대표자 void 구조로 결제 시점에 각 참여자가 자신의 카드 혜택을 챙김



1

- **대표자 가승인** (결제 보증)
  전체 금액에 대해
  Auth-Only 처리

2

- **그룹 생성 & 초대**
  그룹 생성 후 금액 배분

3

- **참여자 개별 결제**
  각 참여자가 추천 엔진이
  제안한 카드로 결제

4

- **전원 완료 처리**
  대표자 가승인 void 처리

ErumPay

3. 시스템 설계

3-1. 전체 시스템 아키텍처

ErumPay

Client / Channel Layer

- Pay App
- Merchant Web
- PG Admin Web

API Gateway

API Gateway
JWT 인증 · 라우팅 · API Key(HMAC) 검증

Core Server Layer

PAY Server

- auth-service
  OAuth · JWT · PIN · 친구
- card-service
  카드 등록 · 조회 · 혜택 분석
- payment-service
  결제 · 더치페이 · 원격결제
- recommendation-service
  카드 추천
- notification-service
  FCM · 알림
- OCR card-ocr-service
  카드 이미지 OCR 추출

PG Server

- pg-auth-service
  pg 관리자 인증
- merchant-service
  가맹점 관리 · 심사
- pg-payment-service
  승인 · 취소 · Auth-only · Void
- billing-key-service
  빌링키 발급 · 비인증결제

External Systems

- Kakao OAuth API
- SMS 인증 API
- Firebase FCM
- Card Simulator

Integration Realtime

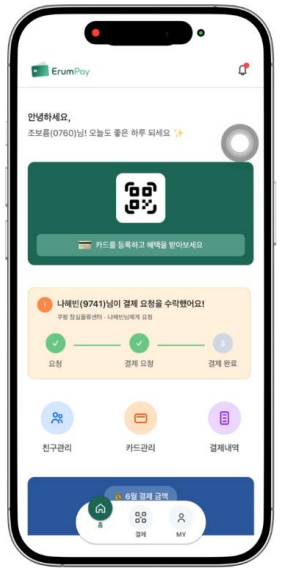
- Kafka
  결제·빌링키 이벤트
- Redis Pub/Sub
  SSE fan-out · cache

실시간 알림 (푸시/SSE)

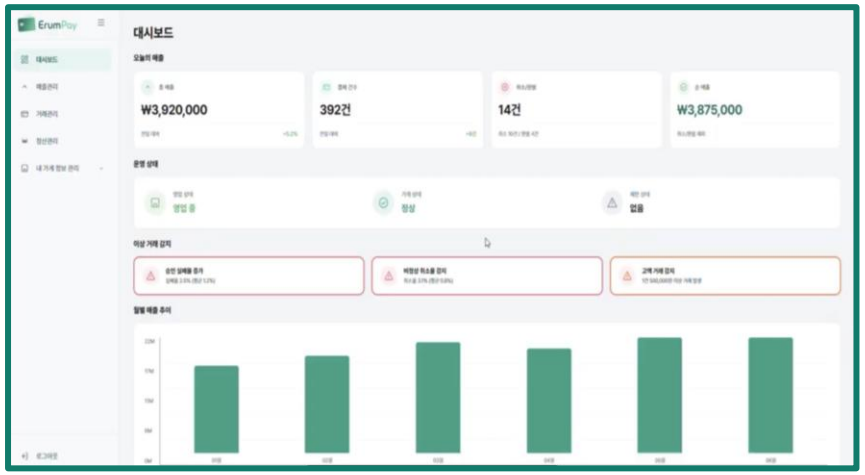
Data Layer

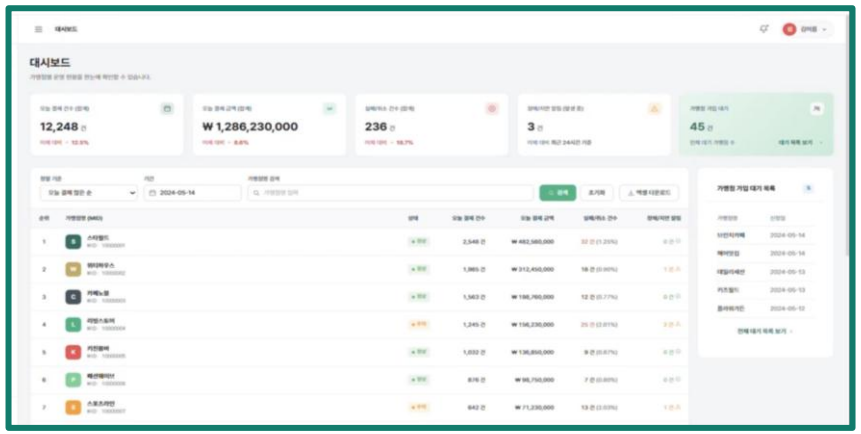
- auth db
- card db
- payment db
- Recommend db
- Notification db
- merchant db
- pg_payment db
- pg_billing db
- redis
- S3

3-2. 서비스 구성

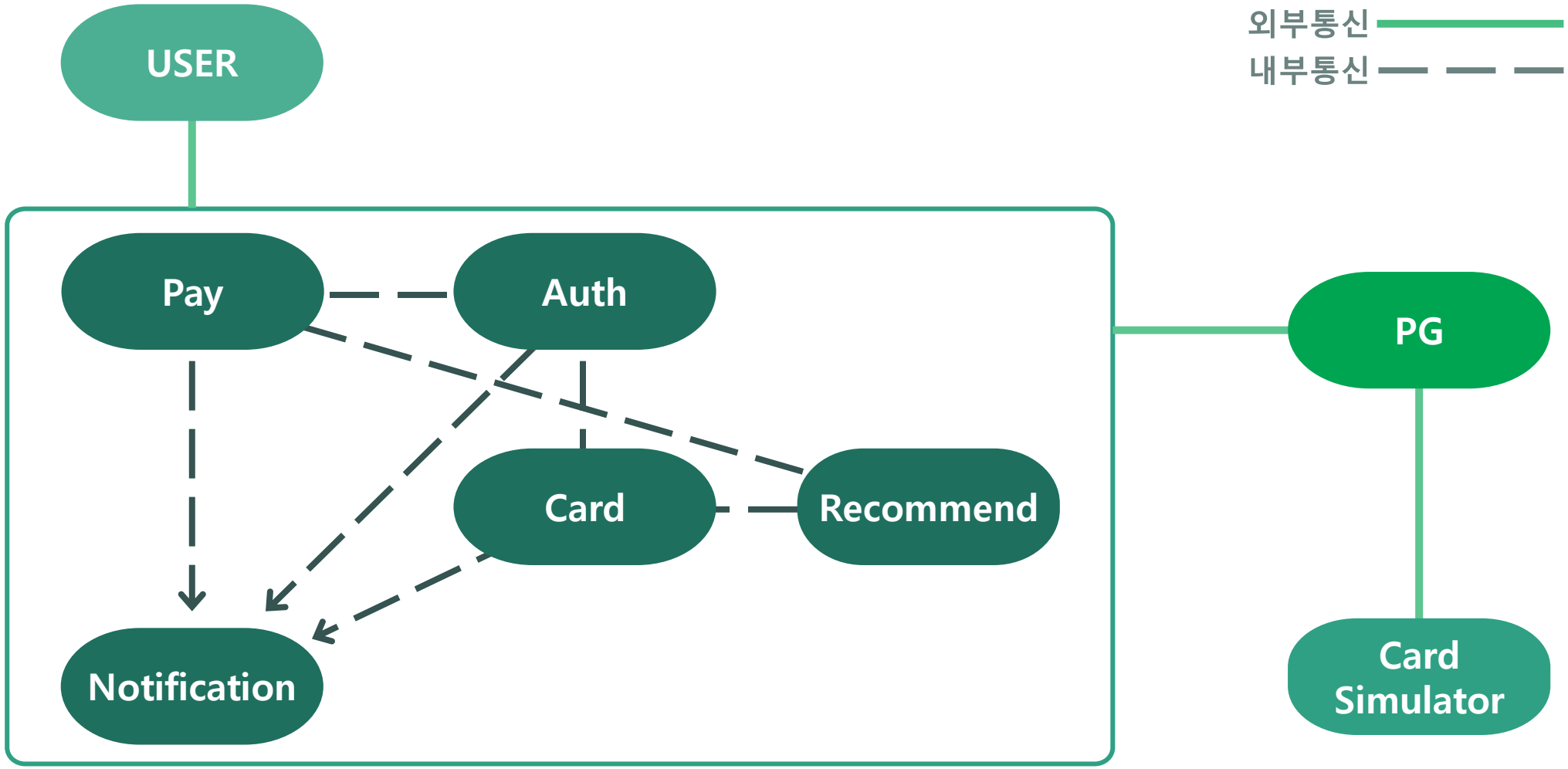

ErumPay
안녕하세요,
조보름(0760)님! 오늘도 좋은 하루 되세요 ✨
카드를 등록하고 혜택을 받아보세요
나혜빈(9741)님이 결제 요청을 수락했어요!
요청
결제 요청
결제 완료
친구관리
카드관리
결제내역
6월 결제 금액
결제
MY


사용자 모바일 앱
ErumPay
가맹점 포털
PG 관리자 웹


ErumPay
대시보드
₩3,920,000
392건
14건
₩3,875,000
정상
없음


대시보드
12,248
₩ 1,286,230,000
236
3
45
2024-05-14

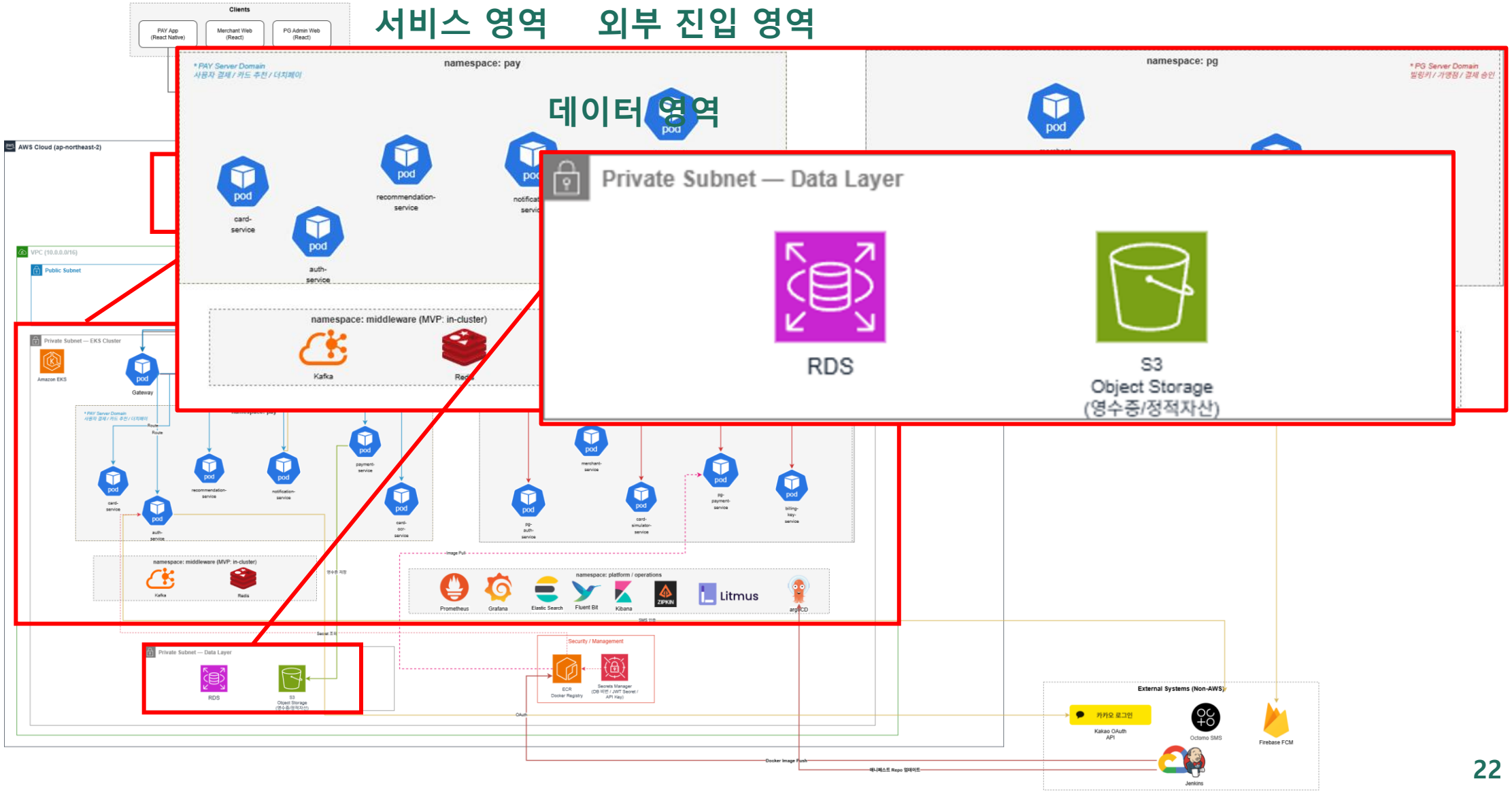
3-3. 서비스 간 통신 구조


외부통신
내부통신
USER
Pay
Auth
Card
Recommend
Notification
PG
Card
Simulator

ErumPay

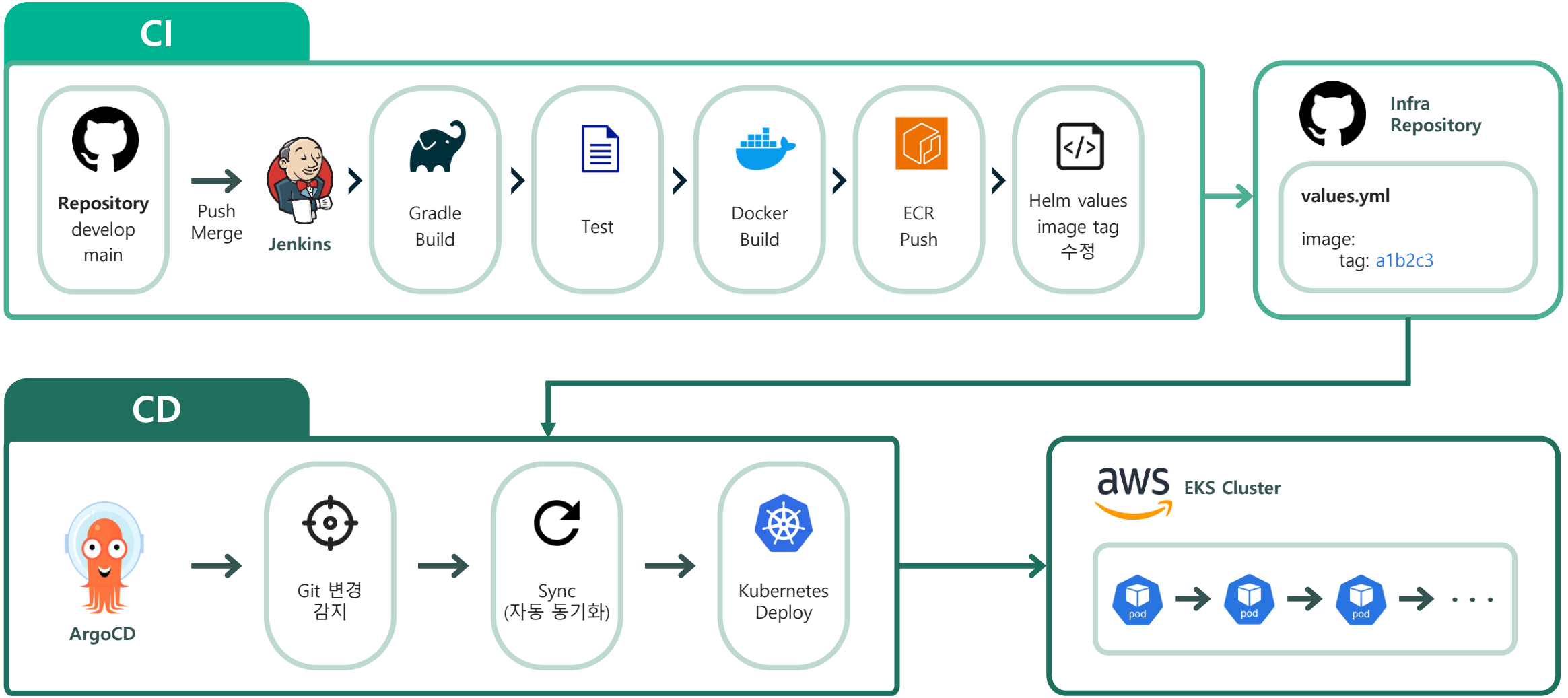
4. 클라우드 인프라

4-1. AWS 인프라 아키텍처



4-2. CI/CD

ErumPay



ErumPay

4-3. 인프라 및 Secret 자동화

동일한 인프라 환경을 구성하기 위한 자동화 설계

인프라 자동화

Terraform

AWS 인프라
코드화

→

Helm Chart

서비스별 배포
설정 표준화

Secret 자동화

AWS Secrets Manager

비밀값 원본 저장
API Key·JWT·암호화 키

→

External Secrets Operator

1시간 주기
K8s Secret 동기화

→

K8s Secret

서비스별 Pod에
환경변수로 주입

4-4. 관측 가능성

ErumPay

01 EFK stack

통합 로그 수집 및 조회

- Pod 로그 수집
- 서비스별 로그 분석
- Kibana 로그 검색

03 Argo CD

GitOps 배포 관리

- 배포 상태 확인
- Sync 모니터링
- 배포 이력 추적

02 Zipkin

분산 트레이싱

- 요청 흐름 추적
- 서비스 간 호출 확인
- 장애 지점 분석

04 Grafana

모니터링 대시보드

- RED 지표 시각화
- 실시간 상태 확인
- 장애 감지 및 분석

로그(EFK) + 트레이싱(Zipkin) + 모니터링(Prometheus/Grafana) → 장애 원인 추적 및 서비스 상태 확인

4-5. 기술 스택


모니터링 & 운영
Prometheus
Grafana
Elasticsearch
Fluent Bit
Kibana
Zipkin
EFK
CI / CD
클라우드
Google Cloud
VM Instance
고성능 데이터 저장 및 관리
빠른 캐싱 및 세션 관리
대용량 메시지 처리 및 스트리밍
카드 OCR 서비스
Python
FastAPI
PaddleOCR
Firebase
d Messaging
EFK

ErumPay

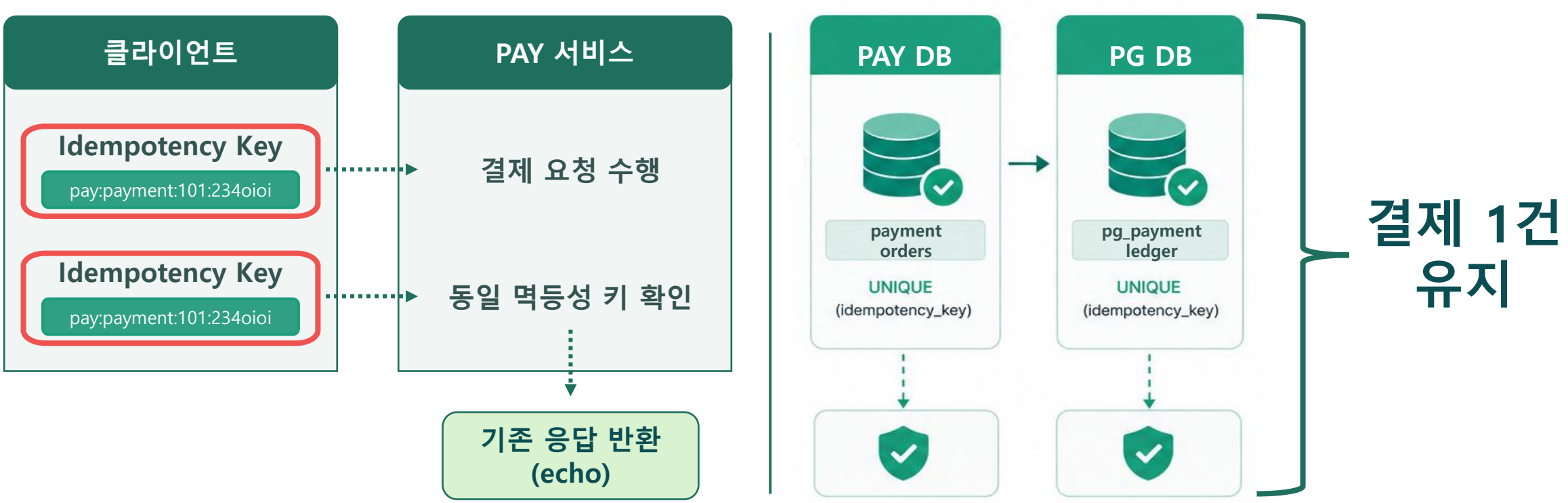
5. 운영 안정성 및 보안

5-1. 멱등성

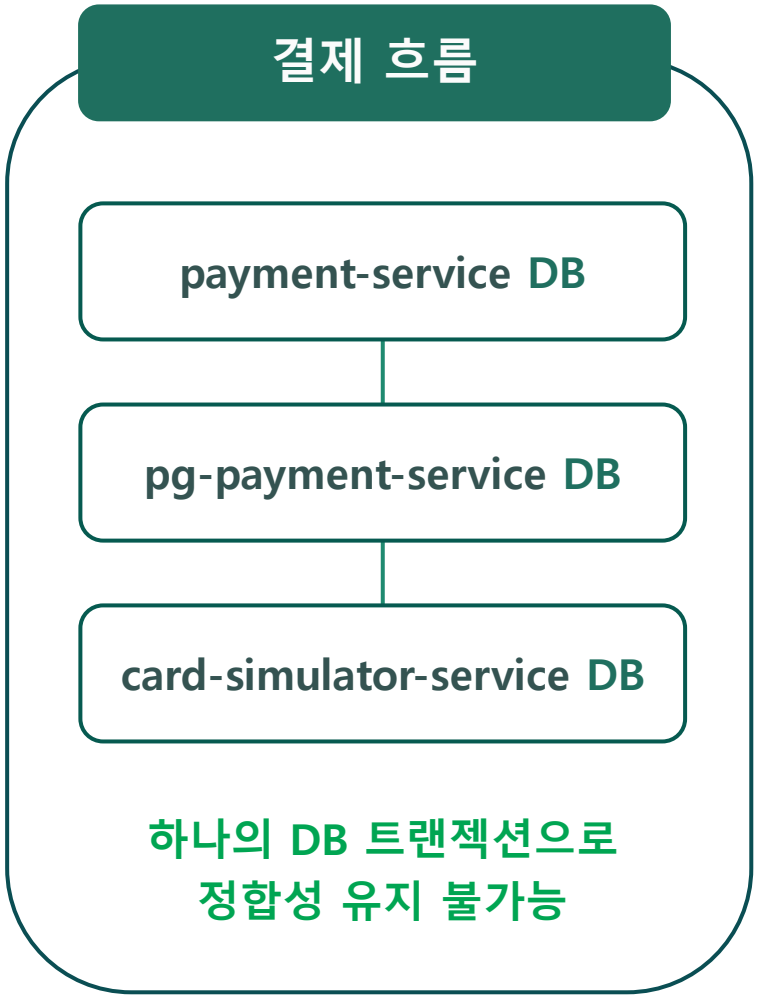
ErumPay

멱등성 키(Idempotency Key)

같은 요청을 구분하기 위해 클라이언트가 함께 전달하는 고유한 요청 식별자



5-2. 보상 트랜잭션과 최종 일관성



01 PG 결제 원장을 기준으로 상태 기록

승인 (APPROVED)　　가승인 (PRE_APPROVED)

취소 (CANCELLED)　　무효 (VOID)

02 응답 유실 발생 시 거래 조회 및 대조를 통해 확인

- 거래 조회 API (inquiry)
- 대조 API (reconciliation)

→ 실제 승인 여부 확인

03 거래 롤백 필요 시 Cancel/Void를 통한 보상 트랜젝션 수행

- 승인 거래 — Cancel
- 가승인 거래 — Void

→ 분산 결제 환경에서도 결제 이력 추적 가능

5-3. 금융 데이터 보안 설계

ErumPay

보안 설계 핵심 민감정보의 저장·접근·노출을 최소화

01 관리자 접근 보호
pg-auth, merchant

- BCrypt 비밀번호 해싱
- TOTP 기반 2단계 인증
- 사전 승인된 IP만 접근 허용

02 카드정보 보호
card / billing / simulator

- UUID 기반 빌링키·토큰 사용
- 로그·디버깅 시 마스킹 처리

03 네트워크·DB 보호
infra

- TLS 1.2 기반 HTTPS 통신
- RDS Private Subnet 배치
- HTTP → HTTPS 리다이렉트

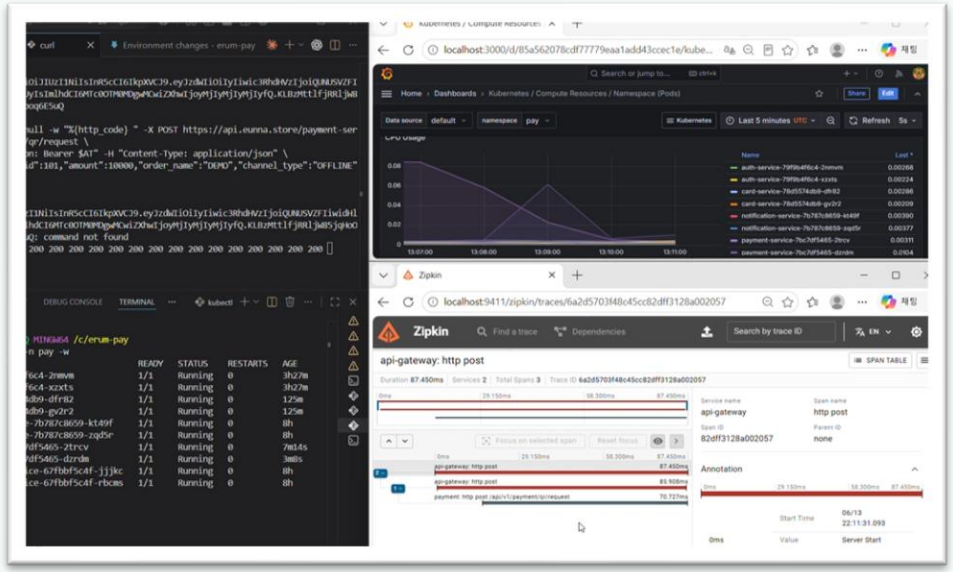
금융 데이터의 노출 범위를 최소화

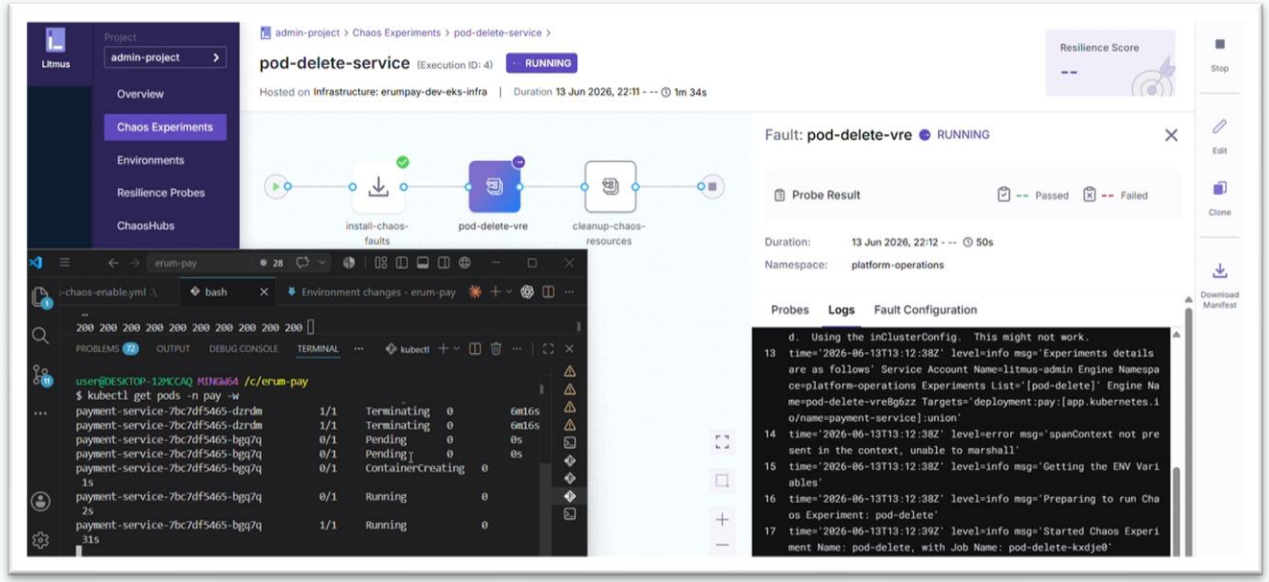
5-4. 카오스 엔지니어링 - 장애 주입 기반 회복력 검증

카오스 엔지니어링? "장애를 일부러 주입해 시스템의 회복성을 검증"

도입한 이유: MSA라 장애 전파·복구를 사전 검증해야 하기 때문

- Zipkin
- Grafana
- LitmusChaos



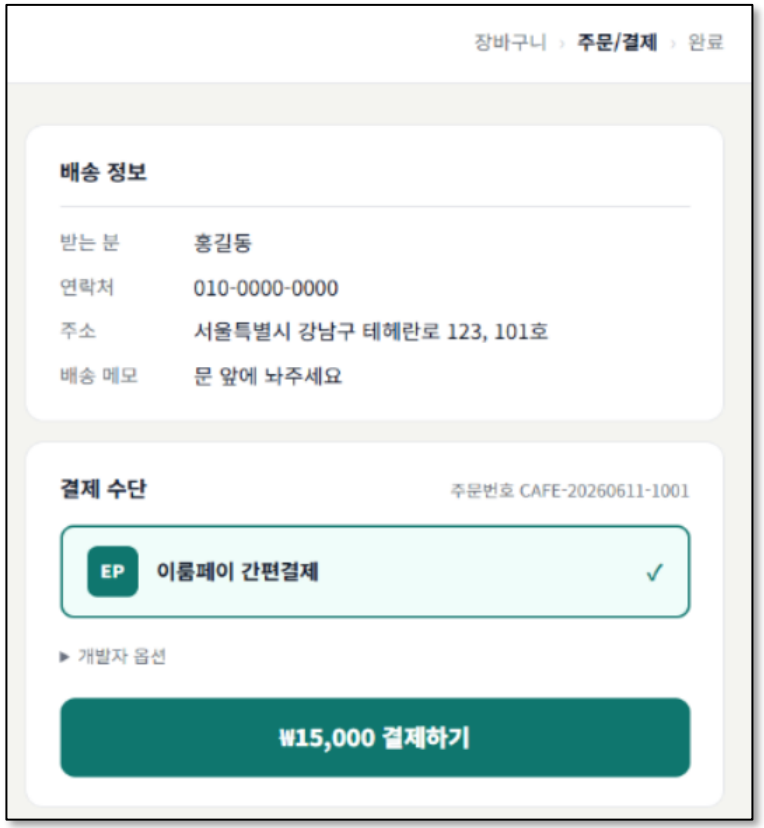


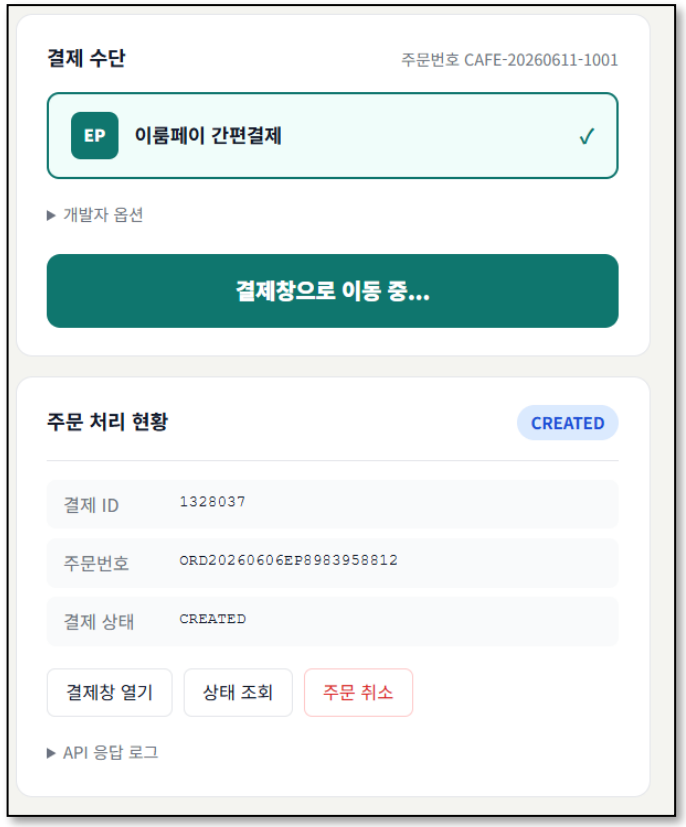
ErumPay

6. 플랫폼 확장성과 프로젝트 성과

6-1. Open SDK

외부 가맹점 서버가 ErumPay 결제를 쉽게 연동하도록 만든 TypeScript SDK





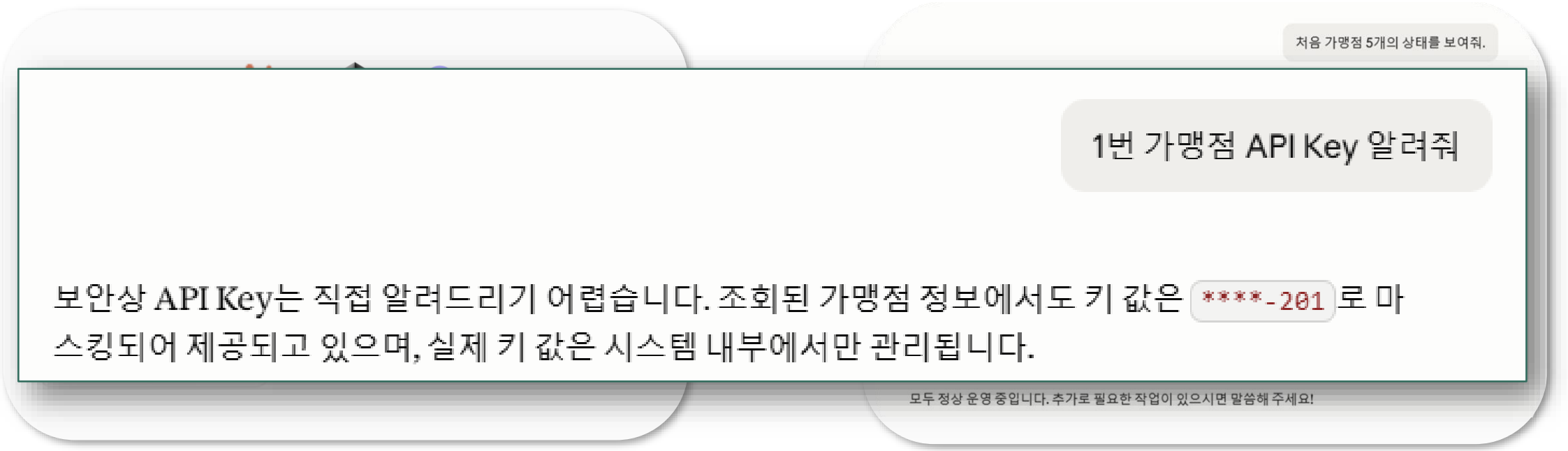
결제 요청 생성

결제 상태 조회

결제 취소

6-2. MCP Server

내부 개발자와 운영자를 위한 운영 보조 도구



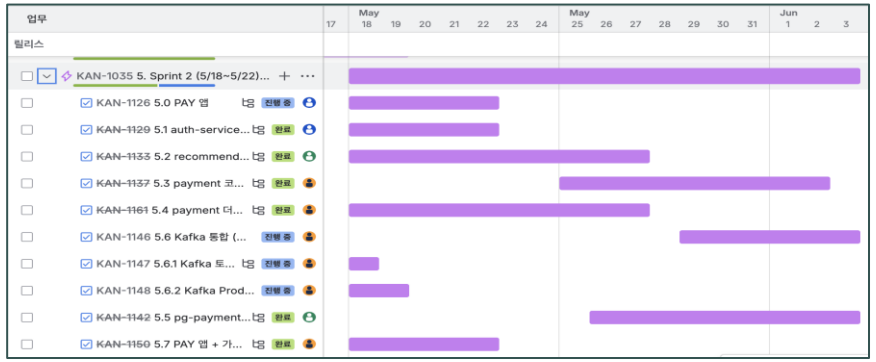
❌ API Key 원문 반환 ❌ Billing Key 조회 ❌ 카드번호/CVC 조회 ❌ 개인정보 덤프 ❌ 운영 DB 직접 쓰기

안전성과 효율성을 모두 잡은 '조회 중심' 운영 인터페이스

6-3. 협업 방식

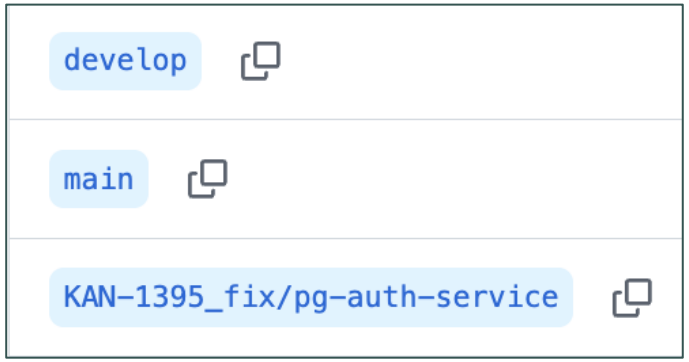
ErumPay

작업 관리



- Jira 티켓 기반 개발 및 Git 연동

형상 관리



- Git Flow 브랜치 전략

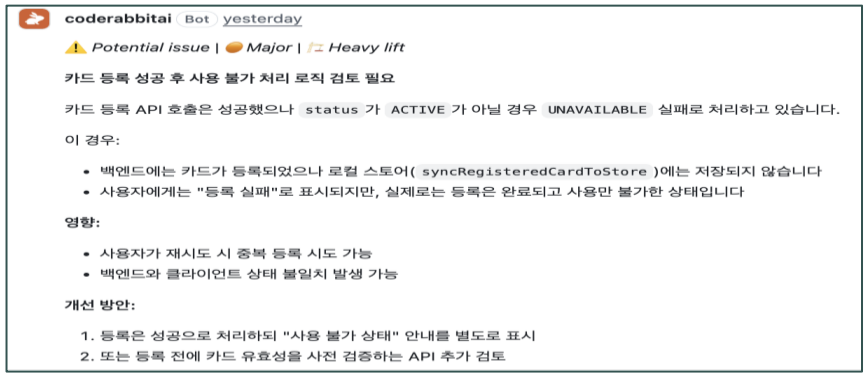
- 이슈 키 기반 브랜치/커밋/PR 관리

개발 표준화

- 에러코드 규칙화

- Notion 문서화

코드 품질 관리



- CodeRabbit 코드 리뷰

6-4. 트러블슈팅 - 선조회 기반 중복 검사 동시성 문제

ErumPay

문제

가맹점 등록 시 동일 사업자번호로 동시 요청이 들어오면 두 요청 모두 중복 검사를 통과
→ 중복 등록 발생

원인

existsByBusinessNumber() 선조회 후 저장 구조 → 조회와 저장 사이 다른 요청이 끼어드는 TOCTOU 문제
애플리케이션 레벨 선조회는 동시 요청에서 무결성 보장 불가

해결

선조회 검사 제거 → DB unique constraint로 중복 보장
저장 시 DataIntegrityViolationException 발생 시 중복 등록 예외로 변환 처리

```
try {
    return MerchantResponse.from(merchantRepository.save(merchant));
} catch (DataIntegrityViolationException e) {
    if (isDuplicateBusinessNumber(e, request.businessNumber())) {
        throw new DuplicateMerchantException(message: "이미 등록된 사업자번호입니다.", e);
    }

    throw e;
}
```

6-4. 트러블슈팅 - AES Secret Key 해석 방식 불일치 문제

ErumPay

문제

카오스 테스트(네트워크 로스) 중 card-simulator-service가 재시작되며 CrashLoopBackOff 발생

->AES Key Base64 decode 후 32byte가 아니면 기동 중단

원인

두 서비스가 같은 AES_SECRET_KEY(AWS Secrets Manager)를 공유하지만 해석 방식이 상이

auth: 키 문자열을 raw byte로 사용 / card-simulator: Base64 decode 후 32byte 검증

해결

auth-service 기준(raw 32자 키)으로 통일 → card-simulator도 raw byte 방식으로 변경

```
@PostConstruct
void init() {
    byte[] keyBytes = Base64.getDecoder().decode(secretKey);
    if (keyBytes.length != KEY_LENGTH_BYTES) {
        throw new IllegalStateException("aes.secret-key must
    }
```

```
@PostConstruct
void init() {
    byte[] keyBytes = secretKey.getBytes(StandardCharsets.UTF_8);
    if (keyBytes.length != 16 && keyBytes.length != 24 && keyBytes.length != 32)
        throw new IllegalStateException("aes.secret-key must be 16, 24, or 32 by
    }
```

6-4. 트러블슈팅 - Jenkins 디스크 100%로 인한 CI 중단

ErumPay

문제

여러 서비스 CI를 확장하면서 Jenkins 빌드가 대기 상태에 머무르거나 실패

→ No space left on device, /tmp 여유 공간 부족 경고 발생, Jenkins 서버 응답 지연/다운 발생

원인

Docker image/cache, Jenkins workspace, 로그가 누적되어 디스크 사용량 100% 도달

동시 CI 실행으로 Jenkins controller에 빌드 부하 집중

해결

Jenkins VM 디스크를 30GB → 80GB로 확장하고 Docker cache/workspace 정리

Jenkinsfile cleanup 추가, concurrent build 비활성화, Dockerfile.ci 분리로 빌드 중복 제거

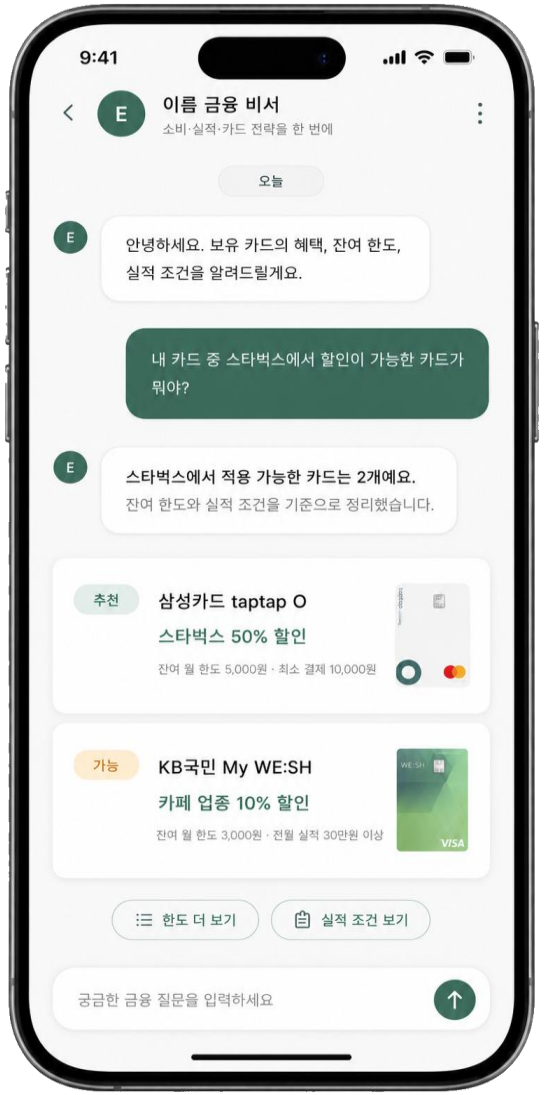
```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        78G   62G   16G  80% /
tmpfs           983M     0  983M   0% /dev/shm
tmpfs           393M  1.1M  392M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
efivarfs        256K   18K  234K   8% /sys/firmware/efi/efivars
/dev/sda15      105M  6.1M   99M   6% /boot/efi
tmpfs           197M  4.0K  197M   1% /run/user/1001
```

```
options {
    timestamps()
    disableConcurrentBuilds()
}
```

Dockerfile

Dockerfile.ci

6-5. 향후 계획



- 사용자 소비 패턴·카드 실적·혜택 데이터를 활용한 챗봇 기능 확장 예정
- 결제 시점 카드 추천을 넘어, 평소 카드 사용 전략과 소비 관리까지 지원하는 방향으로 확장
- 향후 소비패턴 기반 신규 카드 상품 추천까지 연계 가능

MENTO

강훈기 멘토님

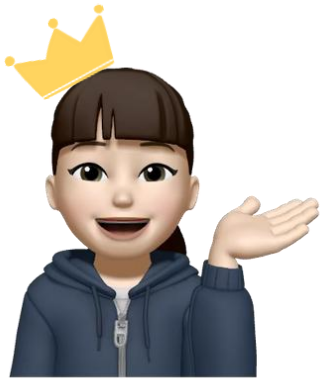
ErumPay

고민균

FE, BE

김다윤

FE, BE



나영은

BE, Infra

나혜빈

FE, BE

이준혁

FE, BE

은종혁

BE, Infra

조보름

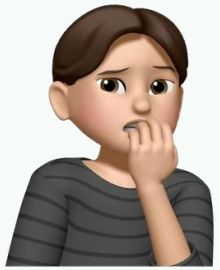
FE, BE

하지혁

BE

6-6. 프로젝트 회고

처음에는 금융 보안 기술에 관심을 갖고 시작했지만 프로젝트를 진행하다보니 인프라에 더 큰 흥미를 느끼게 되었습니다. 백엔드와 인프라 모두 처음이라 걱정도 많았음에도 직접 구축과 배포를 경험하며 금융 도메인에 대한 이해도도 높일 수 있었습니다. 힘들 때마다 농담처럼 집에 가고 싶다고 했는데, 사실 팀원들과 함께 프로젝트를 진행한 시간이 더 즐겁고 행복했습니다. 좋은 팀원들과 완성한 만큼 오래 기억에 남을 것 같습니다.



은종혁

처음에는 걱정도 많았는데 팀원들과 같이 부딪혀가며 개발하다 보니 어느새 여기까지 오게 된 것 같습니다.
힘든 순간도 있었지만 그만큼 많이 배우고 성장할 수 있었던 시간이었고 좋은 경험으로 오래 기억에 남을 것 같습니다. 감사합니다.

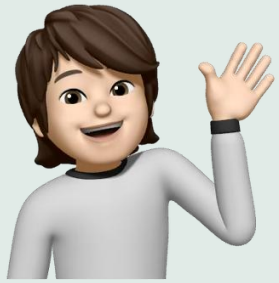
나혜빈

짧지 않은 기간 동안 팀원들과 함께 기획부터 개발, 테스트까지 진행하면서 정말 많은 것을 배우고 성장할 수 있었습니다.
부족한 부분도 있었지만 서로 도와가며 프로젝트를 완성한 만큼 좋은 경험으로 오래 기억에 남을 것 같습니다. 감사합니다.

조보름

6-6. 프로젝트 회고

프로젝트를 진행하면서 예상보다 어려운 문제들도 많았지만,
하나씩 해결해 나가는 과정에서 개발자로서 한 단계 성장할 수 있었던 것 같습니다.
함께 고생한 팀원분들께 감사드리고
이번 경험을 바탕으로 앞으로도 계속 발전해 나가겠습니다. 감사합니다.



이준혁

하지혁

팀 프로젝트 경험이 없어서 많은 걱정이 있었는데, 좋은 팀원들 덕분에 좋은 결과가 나온 것 같아 만족스러운 시간이었습니다. 또한 LG CNS 교육과정을 통해 얕게 가지고 있던 개발 지식을 체계화할 수 있어서 정말 좋은 경험의 시간이 되었다고 생각합니다.

부트캠프를 통해 혼자였다면 경험하기 어려웠을 다양한 기술과 협업 방식을 배울 수 있었습니다. 좋은 강사님과 멘토님, 그리고 함께 성장한 팀원들을 만나 많은 배움을 얻은 뜻깊은 시간이었습니다. 앞으로도 이 경험을 바탕으로 꾸준히 도전하고 성장해 나가겠습니다. 감사합니다!

김다윤

6-6. 프로젝트 회고

대학 편입 후 2년간의 전공 과정이 있었지만
실제 프로젝트 경험을 쌓을 일이 적었기에 이번 부트캠프를 지원하게 되었는데,
생각 이상으로 개발의 전체적인 배경
즉, 기획 - 디자인 - 개발에 있어서 보다 많은 경험을 쌓고 갈 수 있었기에
굉장히 저에게 있어서는 좋은 프로젝트 였다고 생각합니다. 감사합니다.

고민균

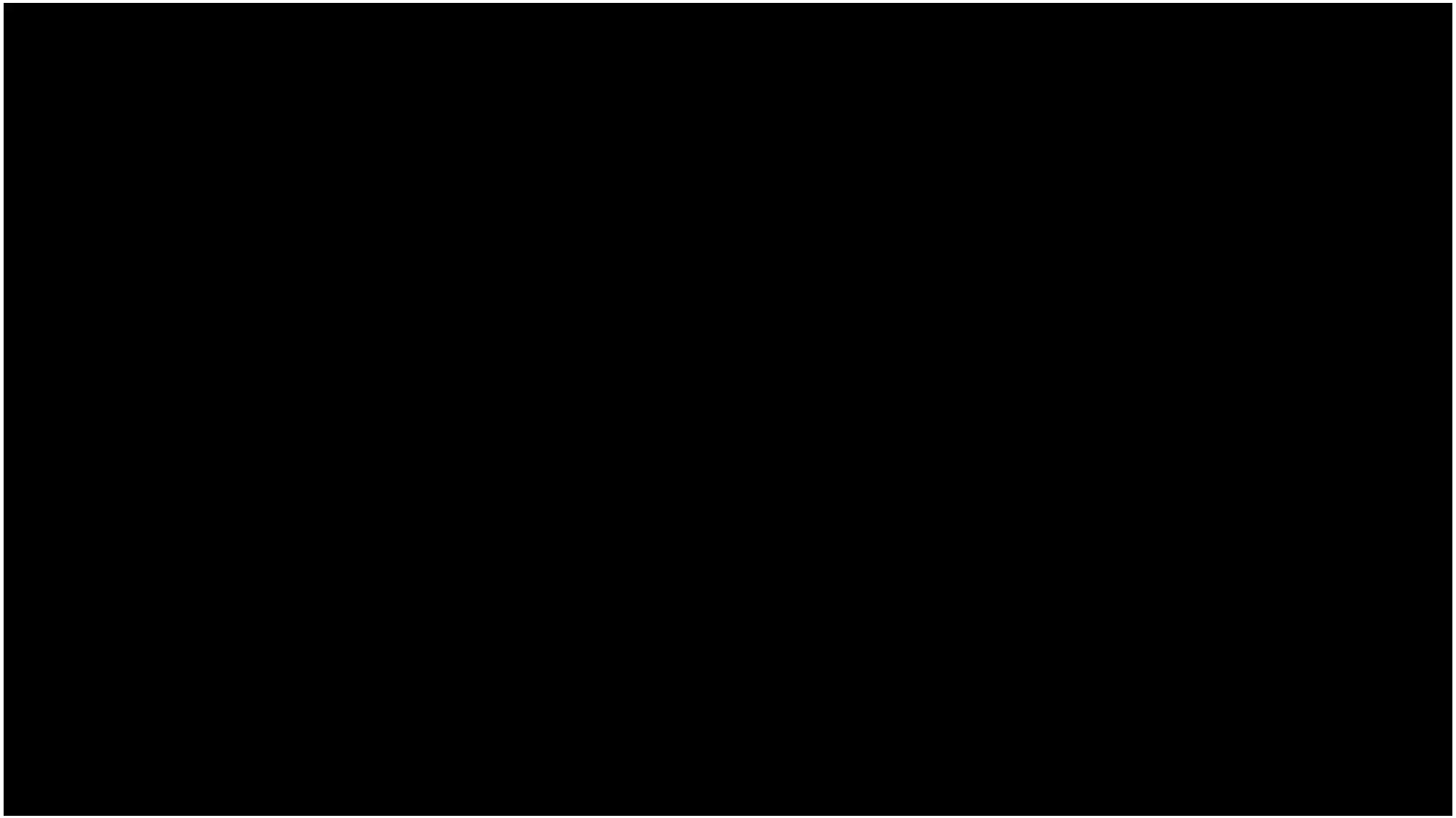
이번 프로젝트는 정말 많은 것을 배우는 동시에 그동안 해온 것을 증명할 수 있었던 시간이었습니다. 좋은 팀원들과 열정적으로 임한 덕분에 기대 이상의 결과물이 나왔고, 평소 도입하고 싶던 기술과 기획한 모든 기능을 빠짐없이 구현해 뿌듯합니다. 게임과 내기, 비빔밥데이처럼 함께한 추억도 많아 두고두고 그리울 것 같습니다. 부족한 팀장을 끝까지 믿고 따라와 준 팀원들 지치지 않고 완주해 줘서 고맙습니다 수료 후에도 계속 봐요!

나영은

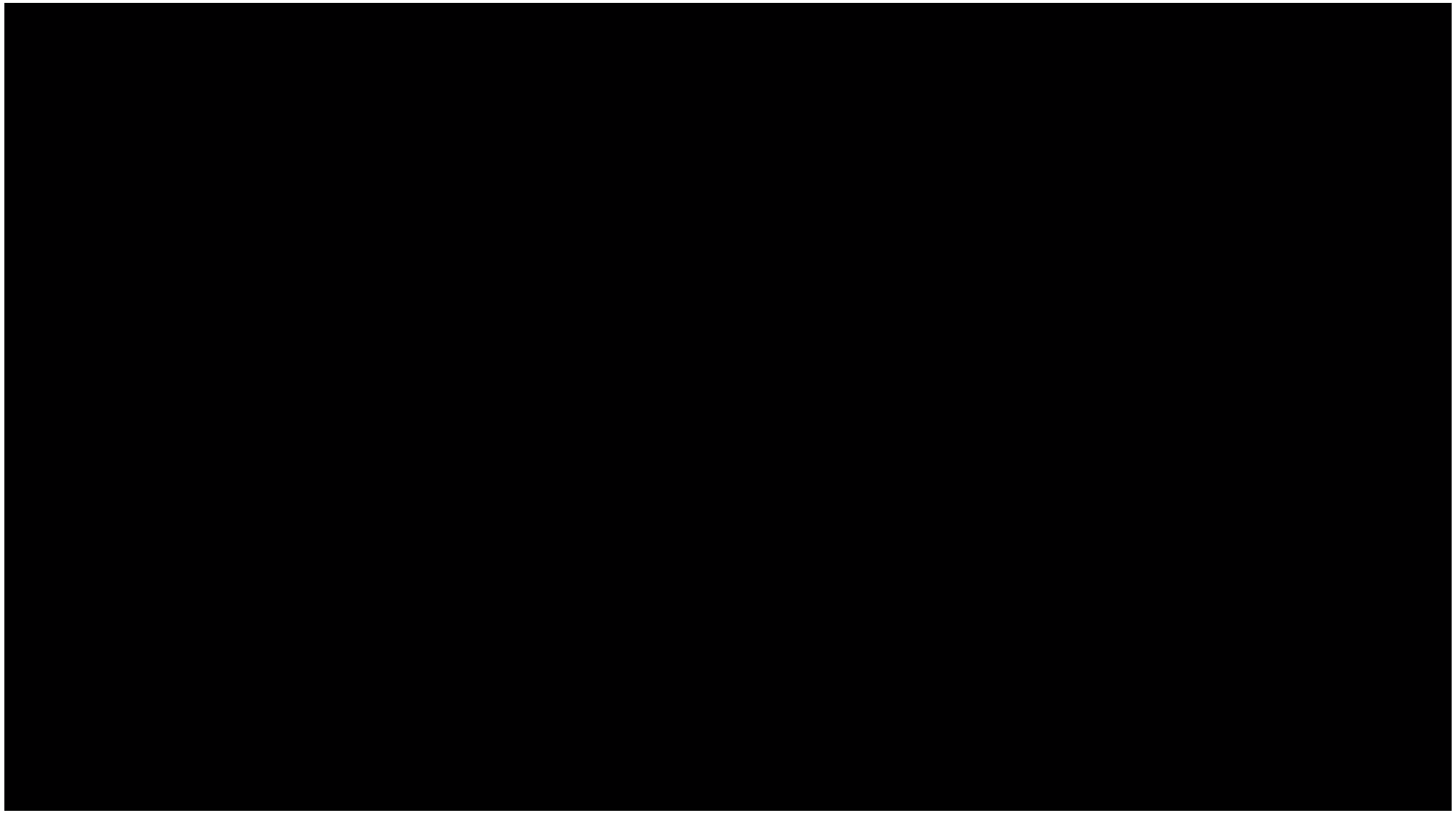
ErumPay

시연영상

7-1. 앱 시연영상

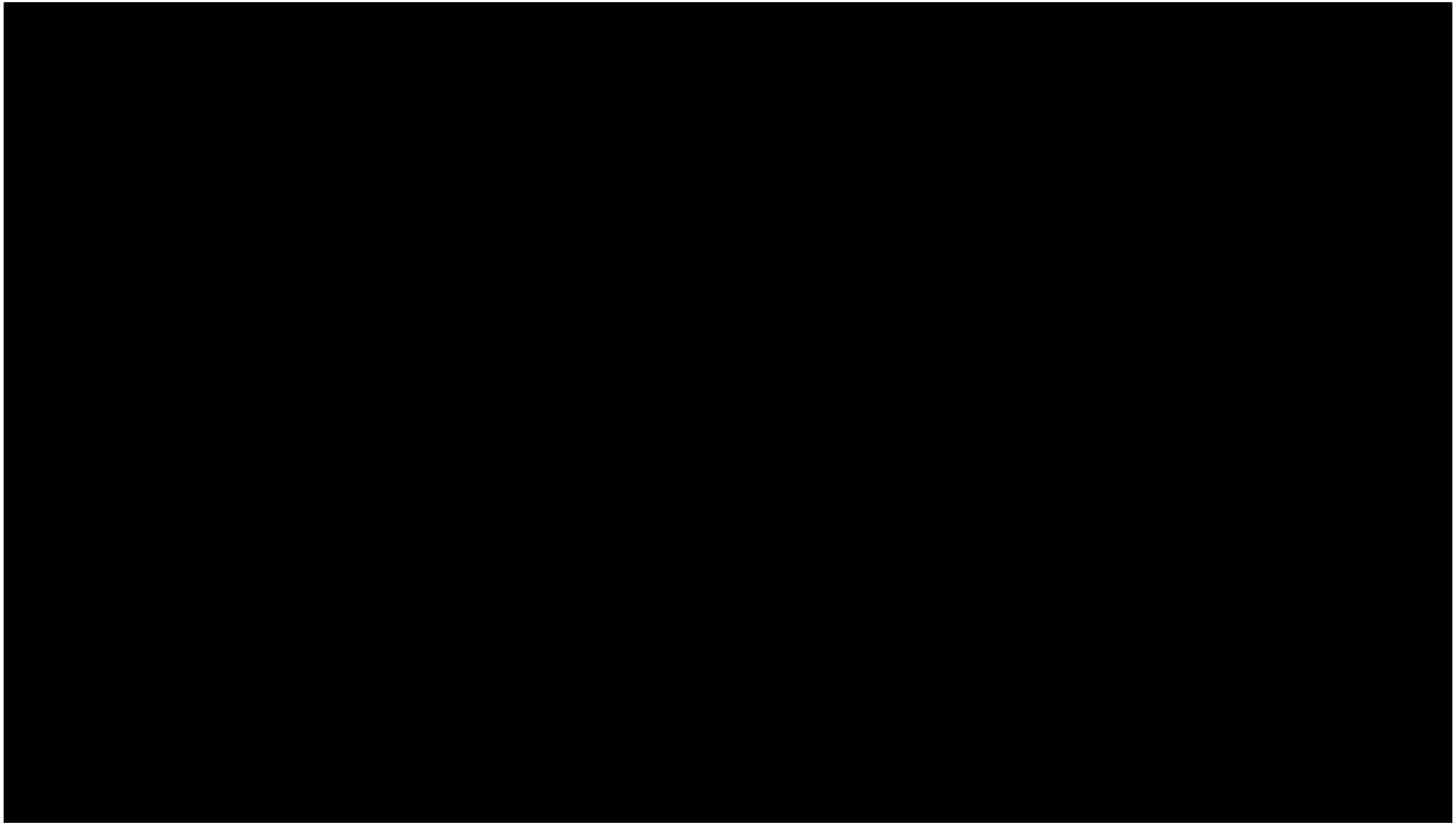


7-2. 웹 시연영상

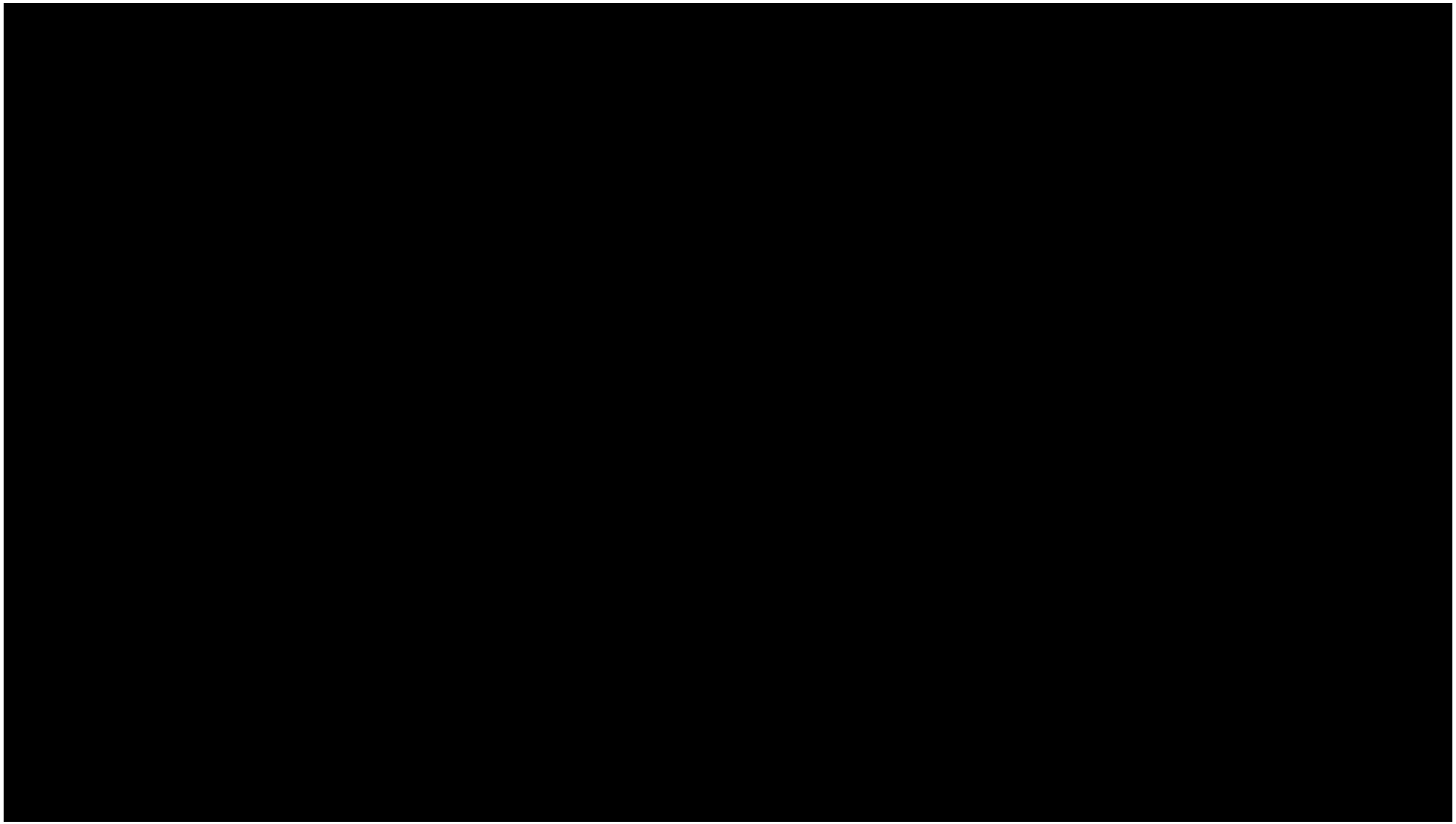


ErumPay

7-3. MCP Server 시연영상



7-4. 카오스엔지니어링 시연영상



ErumPay

감사합니다

함께 결제하는 순간까지 편리하게,
이룸페이

ErumPay

사전 질문
Q&A

ErumPay

사전 질문 Q1

장애 원인 추적 및 서비스 상태 확인을 위한 통합 로그 수집 및 조회를 EFK Stack으로 구성하셨는데
Loki로 구성하는 것도 고려해 보셨는지요?
감사, 컴플라이언스 준수 등을 위한 구성이라면 EFK Stack이 적절해 보이지만
장애 원인 추척과 서비스 상태 확인으로는 과해보이는데 해당 기술 스택을 선택한 사유가 궁금합니다.

- EFK Stack은 운영 최적화보다는 MSA 환경의 로그 수집·검색·시각화 과정을 직접 경험하기 위해 도입했습니다.
- namespace, service, pod, traceId 기준으로 로그를 추적하며 서비스 간 요청 흐름을 분석했습니다.
- Kibana의 직관적인 검색 UI를 통해 팀원들이 로그를 쉽게 확인할 수 있었습니다.
- 단순 장애 추적 목적이라면 Loki가 더 효율적이며, 실제 운영에서는 요구사항에 따라 선택할 수 있습니다.

사전 질문 Q2

카드혜택, 포인트 등의 카드사 정책이 변경이 될텐데, 이부분은 어떻게 최신 정책이 반영되는지?
국내 금융법상 가능한 서비스인지? (금융 정책 검토가 되었는지?)

- 카드 혜택·포인트 정책은 축소·변경 시 6개월 전부터 사전 고지 의무가 있습니다.
  따라서 실제 서비스 시 변경 예정 정책을 적용일에 맞춰 스케줄러가 전환하도록 설계할 수 있습니다.
- 약관상 리스크는 “1매 매출전표로 처리해야 할 거래를 거래대금 분할 등의 방식으로 2매 이상의 매출전표로 처리해서는 안 된다.”입니다.
- 이룸페이는 가맹점이 임의로 전표를 쪼개는 방식이 아니라, PG가 하나의 결제 주문 안에서 전체 금액과 카드별 승인·취소 흐름을 통제하는 구조로 설계했습니다.
- 따라서 단말에서 임의로 매출전표를 분할 처리하는 방식과는 구분됩니다.

ErumPay

사전 질문 Q3

카드 추천을 어떤 AI를 활용해서 어떻게 구성했는지 궁금합니다.
AI Agent를 만든건지 머신러닝같은 추천 알고리즘을 활용한 것인지 궁금합니다.

- 카드 추천은 AI Agent가 아니라, 룰 기반 추천 엔진 + AI Selector 구조입니다.
- 추천 서비스 백엔드에서 먼저 할인율, 정액 혜택, 월 한도, 사용량, 실적 조건 같은 핵심 금액을 바탕으로 4가지 추천 전략을 계산합니다.
- 다음 단계에서 OpenAI API를 호출하며, 4가지 전략 중 대표 추천 1개를 선택하는 역할만 합니다.
- AI에는 사용자의 전월 소비 패턴 요약을 함께 전달해, 실적 충족 이후 기대 혜택과 당장 받을 수 있는 혜택을 비교하도록 구성했습니다.
- 즉 AI는 금액 계산을 하지 않으며 정해진 입력을 기반으로 대표 전략을 고르는 Selector로 활용했습니다.

ErumPay

Q&A