

PORTFOLIO

Daily24

초보 점주의 운영 관리를 돕는

데이터 분석 기반의 백오피스 시스템

정혜민 JEONG HYEMIN



porlife13@gmail.com



010-4311-5966



<https://github.com/Heim970>

목차

1. 자기소개	2p
2. 기술스택	3p
3. 주요 프로젝트 - Daily24	4~12p


ABOUT ME


자기소개




Java & React 개발자

정혜민 JEONG HYEMIN

 1997.09.09

 010-4311-5966

 porlife13@gmail.com

 github.com/Heim970

사용자의 행동 데이터를 분석해 의미를 찾아냅니다.
2년치 판매 데이터를 생성·가공·분석해 매출 통계와 인사이트를 도출했습니다.
기술을 넘어 사람을 이해하는 개발자를 지향합니다.

자격증

- **정보처리기사 (2024.12)**
 - CS 및 소프트웨어 공학 기초 학습
 - 평일 1시간, 주말 4시간씩 4개월 학습
 - 개발 방법론(애자일/스크럼)을 프로젝트 일정 및 기능 단위 계획 수립에 적용
- **SQLD(SQL개발자) (2025.04)**
 - 정규화/반정규화, 통계 테이블 설계, 중복 컬럼 처리 방법 학습
 - 프로젝트와 병행, 매일 1시간씩 1개월간 학습
 - 학습 내용을 프로젝트 DB 설계에 적용
- **TOEIC 830 (2025.02)**

경력

- **(주)문정 | 사원, 사무직 (2023.02 ~ 2024.08)**
 - 나라장터 제안서 작성 및 입찰 성공 경험
 - 사용자 관점에서 정보 설계 및 시각적 구성을 고려
- **부산보훈병원 | 청년인턴 방사선사 (2020.05 ~ 2020.11)**
 - 의사, 간호사 등 타 직군과의 협업 경험
 - 정보를 간결하고 정확하게 전달하는 법을 체득

교육

- **신세계아이앤씨 웹 풀스택 개발자 양성과정 (2024.10 ~ 2025.04, 920시간)**
Java, Spring Boot, MySQL, JPA, AWS, React
- **연세대학교 미래캠퍼스**
방사선학과 | 학사 (GPA 3.81/4.3)
2016.03 ~ 2019.08

기술스택

협업 툴



Git/GitHub
- 브랜치 전략, PR 경험



Notion
- 기획, 일정관리



Slack
- 자료공유, 커뮤니케이션



Google Docs/Sheets
- API, 요구사항 정리

Main Stack/BE



Spring,
Springboot

- RESTful API 구현, 프론트 연동
- 스케줄링을 적용한 주기적 DB 업데이트
- JPA 기반 CRUD 기능 개발



MySQL

- 수십만 건의 판매데이터를
통계 테이블 반정규화로 성능 향상
- AutoIncrement와 트리거로 데이터 무결성 관리

Main Stack/FE



React

- 외부 API로 실시간 날씨, 간편결제 구현
- RESTful API로 동적 페이지 구성
- Nivo 라이브러리로 판매 데이터 시각화

Soft Skills

문서화 및 커뮤니케이션 능력

- 사용자 매뉴얼, 제안서 등 문서 작성 경험
- 정보 정리 및 시각화를 통한 커뮤니케이션 능력

Sub Stack (사용경험)



Python

- 다중 파일 파싱 → SQL 쿼리 생성
- NumPy, Pandas로 데이터 전처리
(빅데이터분석기사 준비 중)



AWS

- Elastic Beanstalk 수동 배포
- RDS & S3 연동 → React + Spring + DB 통합 배포



Next.js

- 클론코딩 프로젝트에서
서버리스 렌더링, 페이지 라우팅 구조 경험



Typescript

- 클론코딩 프로젝트에서
타입 지정으로 코드 안정성 강화 경험



Tailwindcss

- 공식 문서 기반 UI 세부 조정
(여백, 색상, 간격 등)

프로젝트

팀 구성 | 기여도

- 총 2명(팀장) | 50%

프로젝트 기간

- 2025.02.28 ~
2025.04.24

프로젝트 코드



BE

github.com/KDT7team1/backend
백엔드 서버 로직(Spring)



FE

github.com/KDT7team1/frontend
React 및 챗봇 Python 서버

프로젝트 개요

무인 매장의 초보 점주들이 **운영 데이터**를 쉽게 관리하고,
실시간 대응까지 가능하도록 돕는 **백오피스 시스템**

기획 의도

운영 핵심 요소(매출, 재고, 발주, 폐기)를 **데이터 기반으로 연결**하고,
분석과 시각화를 통해 **점주의 의사결정을 지원**

맡은 업무 (Feature 단위로 기능 전체 개발)

관리자 페이지

매출 시각화 구현

- Nivo 라이브러리 활용
- 시간 단위 매출량과 매출 추이 데이터 시각화

통계 테이블 설계

- MySQL로 설계, JPA로 관리
- 매출 데이터 반정규화로 조회 성능을 향상

분석 레포트 자동화

- Spring Scheduler를 활용
- 매 시간 매출 분석 자동 실행
- Nivo 라이브러리의 바 차트, 트리맵으로 시각화

사용자 페이지

결제 API 연동

- Toss Pay API를 활용한 결제 프로세스 구현
- React에서 결제 요청을 처리하고, Spring Boot에서 결제 이후 로직 구현

장바구니 세션 처리

- 사용자의 장바구니 정보를 **세션으로 관리**
- 비회원 사용자의 장바구니 데이터도 유지되도록 처리 로직 구성

공통

기획 및 설계

- 요구사항 분석
- API 명세서 작성
- DB 테이블 구조 설계

배포 및 인프라 구성

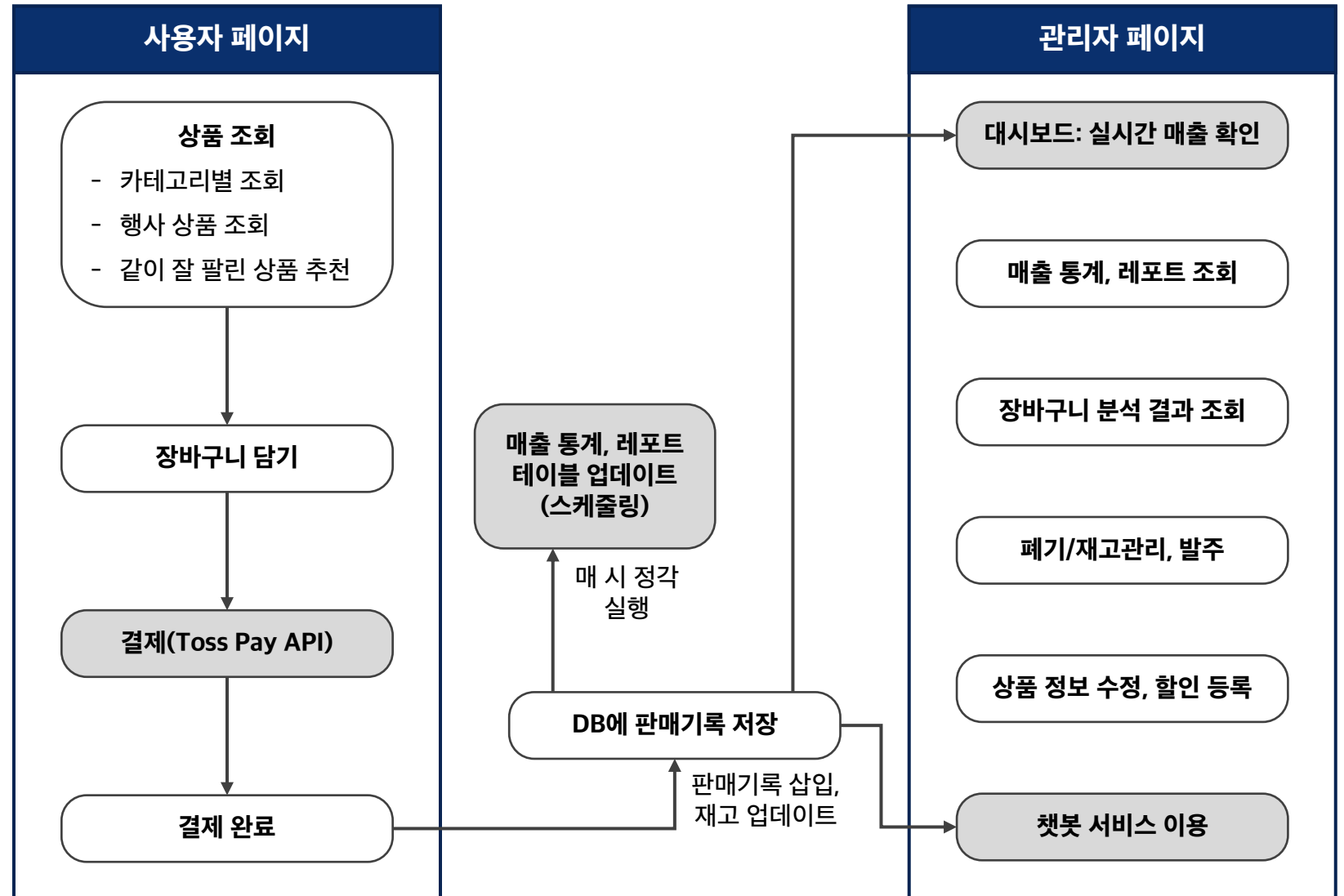
- AWS RDS, S3, Elastic Beanstalk를 이용

문서 및 발표 자료 제작

- 프로젝트 소개 및 발표용 PPT 작성
- 최종 제출 문서 정리

프로젝트

서비스 흐름도



프로젝트

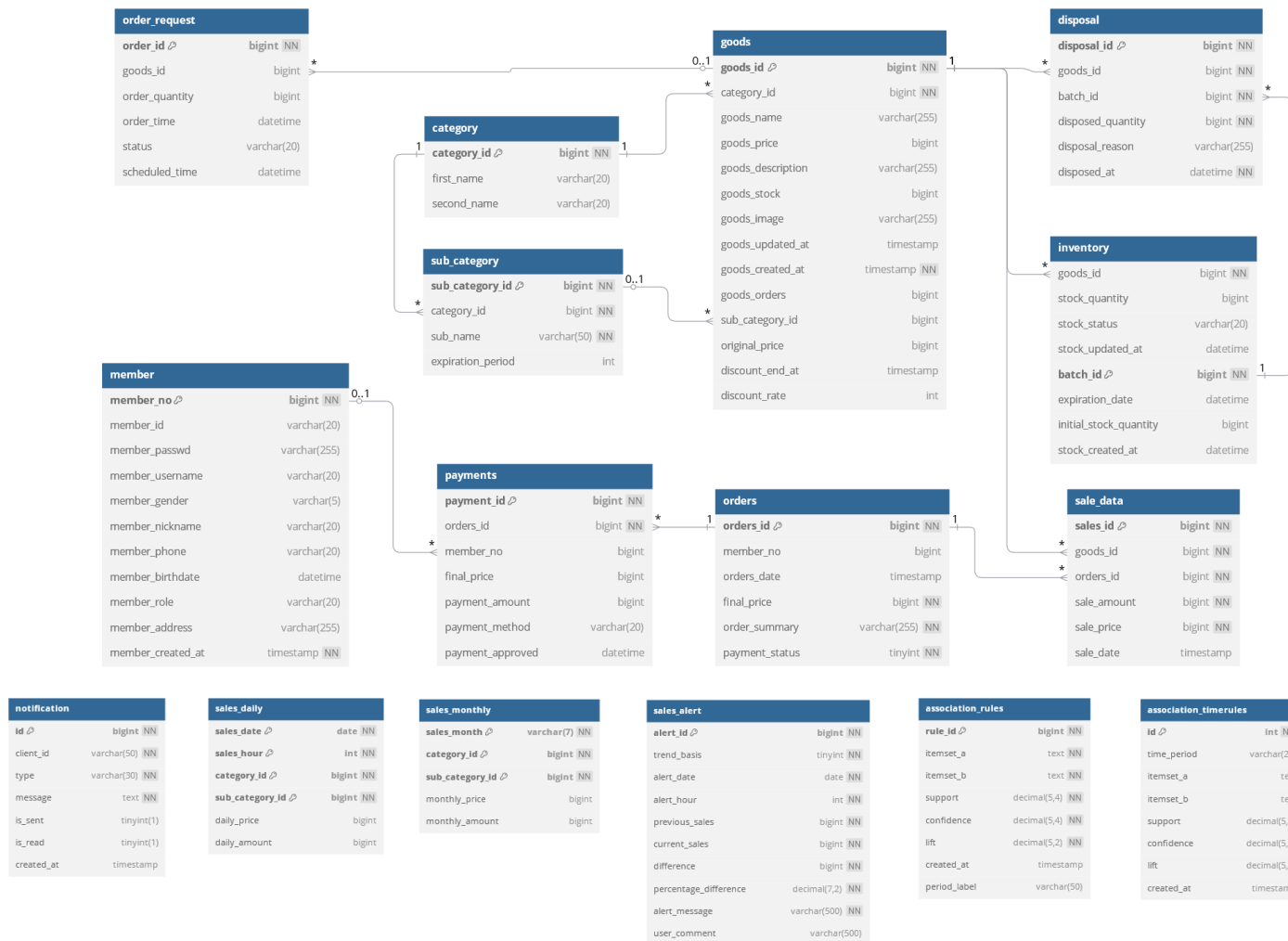
주요 테이블

- 상품 | goods
- 재고 | inventory
- 폐기 | disposal
- 판매기록 | sale_data
- 반정규화된 집계 테이블
: 일간 | sales_daily
: 월간 | sales_monthly

DB 설계 및 관계 구조(ERD)

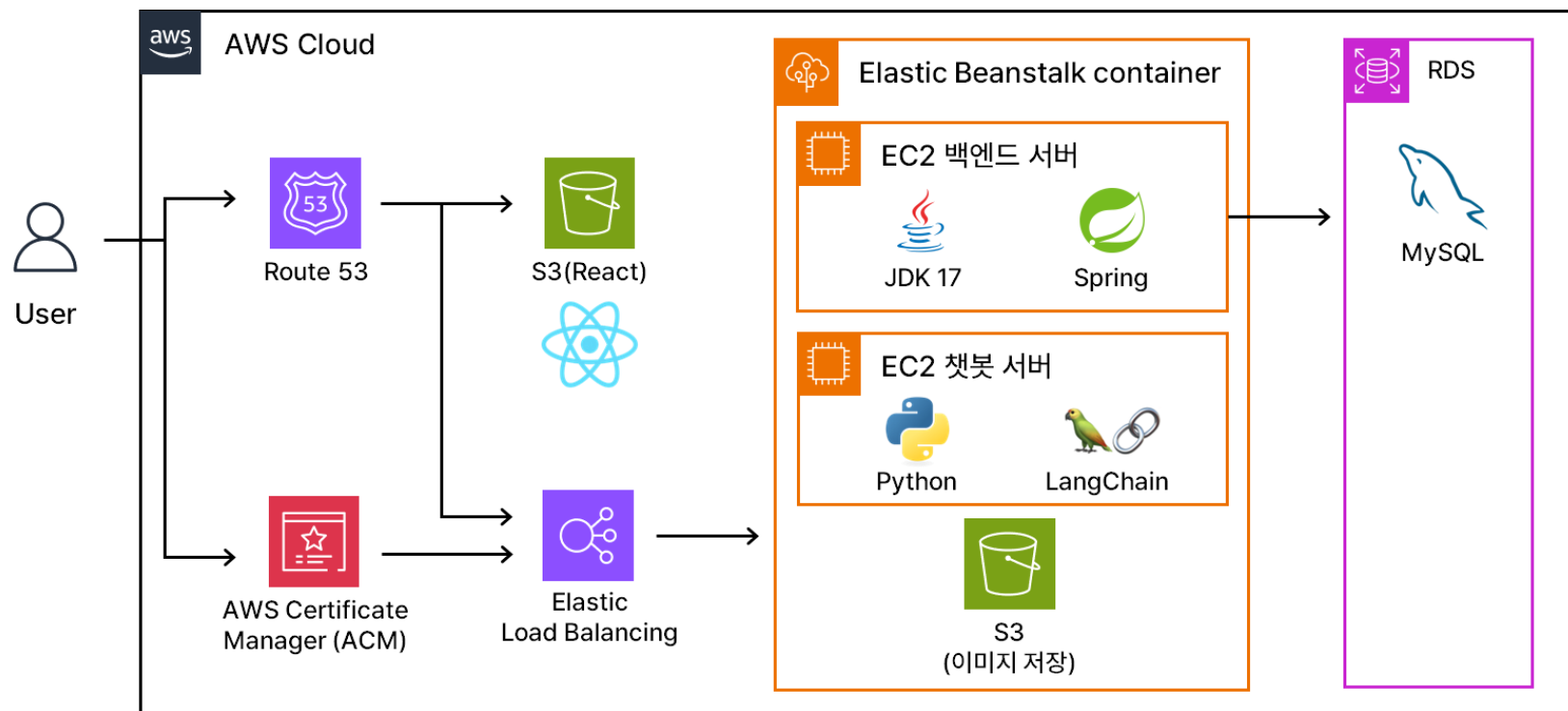
실시간 분석과 통계 기능을 고려하여, 반정규화 테이블 추가 설계

외래키 관계를 통해 연관된 데이터 흐름을 명확히 구성



프로젝트

시스템 아키텍처



EC2 챗봇 서버

- Python + LangChain
- DB의 판매 데이터를 기반으로 한 답변 제공

시스템 플로우

- 도메인 요청은 Route53 → ACM 인증(HTTPS)
- 로드밸런싱 → S3(React) 또는 EC2(서버)로 요청 전달
- EC2 내 백엔드(Spring)와 챗봇 서버(Python)는 S3/RDS와 연결

프로젝트

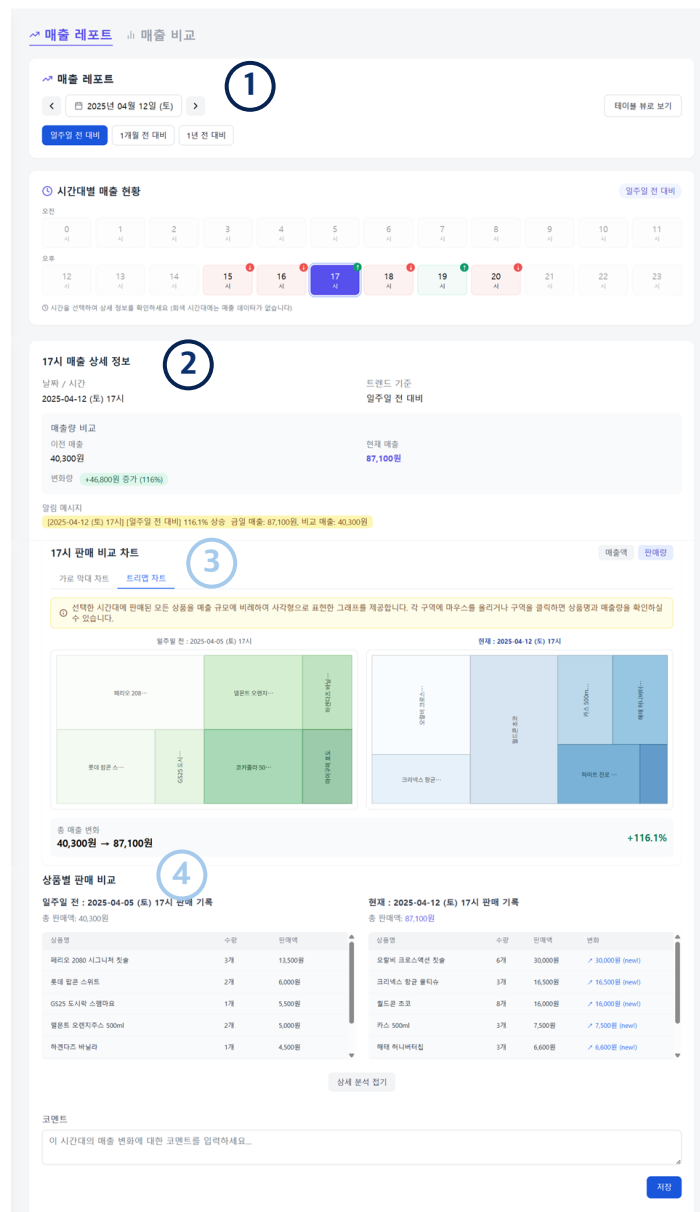
기능 요약

- 시간 단위로 매출 데이터를 분석/비교한 레포트 제공
- 분석 기준: 저번 주 / 한 달 전 / 1년 전 동일 시간대

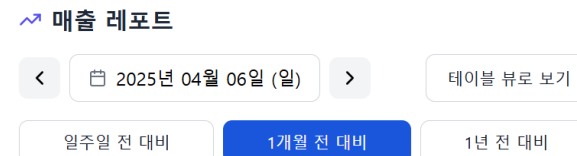
기술 구현

- Spring Scheduler로 매 시 정각 분석 스케줄 실행
- MySQL 기반 반정규화 테이블로 빠른 조회
- 분석 결과는 프론트에서 차트로 시각화

주요 기능 - 매출 레포트



- 날짜와 분석 기준을 선택하면 24시간의 매출 변화가 간단하게 나타남



- 시간대를 클릭하면 매출 상세 정보를 표시함



프로젝트

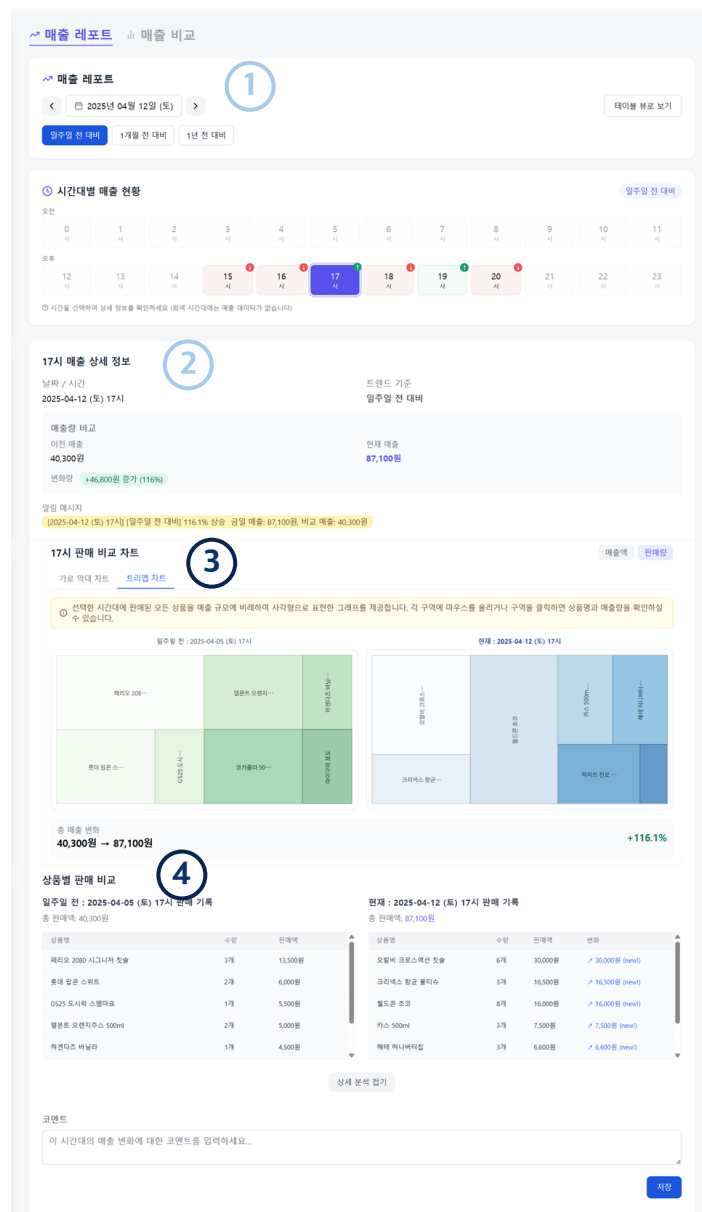
기능 요약

- 시간 단위로 매출 데이터를 분석/비교한 레포트 제공
- 분석 기준: 저번 주 / 한 달 전 / 1년 전 동일 시간대

기술 구현

- Spring Scheduler로 매 시 정각 분석 스케줄 실행
- MySQL 기반 반정규화 테이블로 빠른 조회
- 분석 결과는 프론트에서 차트로 시각화

주요 기능 - 매출 레포트



③ 판매 데이터는 매출액/판매량 기준으로 바 차트와 트리맵 차트로 시각화

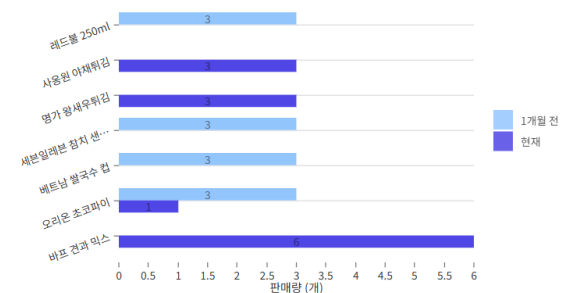
23시 판매 비교 차트

매출액 판매량

가로 막대 차트

트리맵 차트

① 선택한 시간대에서 매출이 가장 높은 7개 상품을 골라, 매출과 판매 수량을 비교한 그래프를 제공합니다.



④ 자세한 비교를 위한 테이블과 데이터로 나타나지 않는 특이사항을 기록하는 코멘트 공간

상품별 판매 비교

일주일 전 : 2025-03-30 (일) 19시 판매 기록

총 판매액: 226,000원

현재 : 2025-04-06 (일) 19시 판매 기록

총 판매액: 87,000원

상품명	수량	판매액	변화
BHC 닭강정	7개	55,300원	
래 한방 삼투	6개	45,000원	
나베와 배드워시	3개	21,000원	
아사히 맥주 500ml	6개	21,000원	
고온 허니콤보 닭강정	4개	32,800원	↗ 32,800원 (new!)
아사히 맥주 500ml	5개	17,500원	↓ -3,500원 (-16.7%)
크리넥스 합금 롤티슈	3개	16,500원	↑ +11,000원 (+200.0%)
하이트 진로 소주	3개	6,000원	↗ 6,000원 (new!)

코멘트

프로모션 행사 진행함

저장

프로젝트

기술 포인트 요약

- **Scheduler 트리거:** 정각
실행 (cron = "0 0 * * *")
- **분석 대상:**
sales_data, sales_daily
- **분석 기준**(1주 전/전월/전년)
+ 동일 시간대 비교
- **결과 저장:** sales_alert
- **예외 처리:** 이미 처리된
시간대는 중복 실행 방지

기술 설계 의도

- **실시간 트렌드 파악**을 위한
정기 분석
- **점주의 의사결정**을 돕기 위한
인사이트 제공

주요 기능 - 매출 레포트

```
@Scheduled(cron = "0 0 * * *") // 매 정각(00분 00초)에 실행 1 usage - Hyemin
public void runSalesAnalysis() {

    LocalDateTime now = LocalDateTime.now();
    log.info("LOGGER: [매출기록 분석기] 자동 실행 - 시스템 시간: {}", now);

    LocalDateTime target = now.minusHours(1);
    LocalDate targetDate = target.toLocalDate();
    int targetHour = target.getHour();

    // 오늘 매출 데이터 가져오기
    List<SalesDailyDTO> todaySalesList = salesAnalysisService.getSalesDailyByDateAndHour(targetDate, targetHour);
    long todaySales = todaySalesList.stream().mapToLong(SalesDailyDTO::getDailyPrice).sum();

    // 데이터가 이미 처리되었는지 확인
    boolean isDataProcessed = isDataProcessed(targetDate, targetHour);

    if (!isDataProcessed) {
        log.info("LOGGER: [매출기록 분석기] 매출 기록 분석을 시작합니다.");
        salesAnalysisService.detectSalesAnomaly(targetDate, targetHour);
    } else {
        log.info("LOGGER: [매출기록 분석기] 이미 처리된 데이터입니다. 스케줄러를 실행하지 않습니다.");
    }
}
```

처리 흐름

- `@Scheduled` 어노테이션을 사용한 cron 스케줄 설정
- `LocalDateTime.now()`로 현재 시간 기준 분석 대상 결정
- 과거 매출 데이터 조회 → `sum()`으로 매출 총합 계산
- **이상치 탐지 알고리즘 호출**(서비스 레이어 내부에 정의)
- 기존에 처리된 데이터가 있을 경우 **로그만 출력하고 종료**

프로젝트

철저한 사전 설계로
치명적인 리스크가 없는
시스템을 구현했습니다.

트러블 슈팅 - 리스크 사전 대응

서비스 설계 및 구현 과정에서 발생 가능한 주요 이슈들을
프로젝트 초기 단계부터 사전에 예측하고, 이를 고려한 구조 및 로직 설계를 통해
운영 중 성능 저하 및 장애 발생 없이 안정적으로 기능을 수행할 수 있었습니다.

💡 사전 대응한 리스크

1. 📊 통계 테이블 분리 설계

- 대용량 데이터 집계로 인한 응답 지연 방지
- 통계 테이블(sales_daily/monthly) 별도 구성

2. ⚙️ JPA 성능 이슈 예방

- N+1 문제 방지를 위해 연관된 데이터는 필요 시점에만 접근
- 응답 시 불필요한 필드만 활용해 string으로 출력

3. 🕒 스케줄링 로직 성능 고려

- 정각 집계 기능 구현 시
- DB 부하를 최소화하여 평균 응답시간 1초 이내 유지

프로젝트

AS-IS 🔍

- 부모님이 운영하시는 편의점 관리 시스템을 자주 확인한 경험이 있음
- 시간대별 매출 차트 하나만 보이고, 나머지는 수치를 직접 분석하거나 계산해야 하는 시스템
- 초보 점주라면 📉 데이터를 읽는 것 자체가 어려울 수 있을 것이라 생각함

TO-BE 🎯

- 📊 시각화 중심으로 매출 데이터를 보여주는 관리자 페이지 구축
- 🕒 매 시간 정각 자동 분석으로 실시간 매출 흐름 파악
- 팀원 두 명의 중도 이탈 후, 핵심 기능(매출 분석, 통계 시각화)에 집중하여 완성

성과 🏆

- 👥 2인 팀으로 개발 일정을 조율하고, 서비스가 실행되는 MVP 완성
- 반정규화 테이블 설계로 ⚡ 빠른 데이터 조회 성능 확보. 매출 데이터 조회에 1초 미만 소요
- JPA N+1 문제 없이 필요한 데이터만 🔒 안전하게 조회
- 숫자가 아닌 흐름을 읽게 해 주는 📈 시각화 + 분석 시스템 구축