BACK-END ENGINEER · WEB

만드는 데서 끝내지 않고, 배포와 시크릿 관리까지 직접 맡는 백엔드 개발자, 김지황

백엔드를 중심으로 설계·배포·운영까지 맡아 서비스를 안정적으로 이끌어갑니다.
B2B 플랫폼·B2C 이커머스·금융 도메인에서 개발과 운영을 경험했습니다.

6+ 년의 실무 경험 **3** 개 도메인 (제조·커머스·금융)

2022 — **LS ELECTRIC 재직 중** · 웹 플랫폼 서비스 개발

2018 중앙대학교 정보시스템학 학사 졸업

2016 정보처리기사 취득

2010 덕수고등학교 졸업 (디지털컨텐츠 전공)

2007 정보처리기능사 · 전산회계운용사 3급 · 워드프로세서 1급·2급 취득

개발자란, **소프트웨어를 통해 문제를 해결하고 소통하여 지속적인 비즈니스를 창출·확장해 나가는 사람**이라 믿습니다.

그래서 저는 안되는 이유 대신, 되게 만드는 기준과 절차를 만드는 개발자로 일합니다

- **운영까지 책임** — 만들고 끝이 아니라, 배포·모니터링·유지보수까지 이어갑니다.
- **풀스택 실무** — Java/Spring, Node.js 백엔드와 Vue/React 프론트엔드를 오가며 개발합니다.
- **개선 지향** — 레거시를 Java 11 → 21 → 25, MySQL 8.0 → 8.4, CRA → Vite로 단계적으로 현대화했습니다.
- **소통** — 고객사·운영자·개발자 사이에서 업무 맥락을 확인하고, 서로 이해할 수 있는 언어로 조율합니다.

01 — Career

경력

제조 B2B 플랫폼, 그룹 이커머스, 금융 인터페이스 — 세 도메인을 지나왔습니다.

LS ELECTRIC

전력·자동화 분야 글로벌 B2B 제조기업

2022.04 — 현재 · 자동화System)사업개발팀

Solution Square — 영업지원·대외 고객 웹 서비스

글로벌 제품·기술 포털의 BFF, SAML SSO, 데이터 연계, 검색·보안·배포 체계 현대화

Tech Square — B2B 스마트 제조 웹 플랫폼

수요·공급기업 매칭, 멘토링, 중고설비, 데이터 적재, 대시보드·알림을 연결하는 플랫폼 풀스택 개발·운영

Node.js · Vue 3 · Java · Spring Boot · React 18 · AWS · Azure · MSA/BFF · CI/CD

교원 크리에이티브

교원그룹의 IT 서비스 개발·운영 계열사

2020.01 — 2021.02 · IT서비스개발3팀

K멤버스 렌터카 서비스 — 그룹 통합 멤버십 쇼핑몰 개발·운영

K멤버스 쇼핑몰 개발·운영, 렌터카 상담 서비스와 백오피스, 계열사 매출 현황, SMS 인증 구현

Spring · MyBatis · Oracle · JSP · jQuery

웹케시씨앤에스

기업 자금관리 솔루션을 다루는 웹케시그룹 계열사

2019.01 — 2020.01 · KEB하나지사

기업 자금관리 CMS 기술지원 · ERP DB 연계

기업 고객의 자금관리 CMS 운영을 지원하고, MS-SQL 기반으로 CMS DB와 기업 ERP DB 사이의 금융 데이터 연계를 담당

MS-SQL · 금융 도메인

02 — Projects

프로젝트

회사에서 맡은 플랫폼부터 SI까지, 직접 개발하고 운영한 프로젝트를 소개합니다.

Solution Square (Global)

PROFESSIONAL PROJECT · 2025 – PRESENT

LS ELECTRIC의 글로벌 제품·엔지니어링 포털. 프론트엔드 요청을 받는 API 서버(BFF)를 갖춘 멀티모듈 MSA 환경에서 비즈니스 로직, 인증, 보안, 배포 파이프라인을 담당했습니다.

Tech Square

PROFESSIONAL PROJECT · 2022.04 – PRESENT

중소 제조기업의 스마트공장 전환을 지원하는 B2B 플랫폼. 풀스택 개발과 운영을 담당하며 매칭·멘토링·거래·데이터·알림 흐름을 연결했습니다.

넷플레이스 고객사 관리 시스템

SI PROJECT · 2026

고객사 DB 관리 백오피스. Django 서버 렌더링과 Tailwind 빌드 파이프라인으로 구축.

넷플레이스 랜딩페이지

SI PROJECT · 2025

서비스 소개 정적 랜딩 페이지. 시맨틱 마크업과 sitemap/robots 구성으로 SEO 대응.

넷플레이스 마케팅 서비스 시스템

SI PROJECT · 2024

마케팅 서비스의 고객용 웹과 API 서버. 기획부터 배포·운영까지 풀스택으로 담당.

넷플레이스 마케팅 서비스 관리자 시스템

SI PROJECT · 2024

마케팅 서비스 운영을 위한 관리자 백오피스. 고객·서비스 현황 관리 기능 제공.

K멤버스

PROFESSIONAL PROJECT · 2020.01 – 2021.02

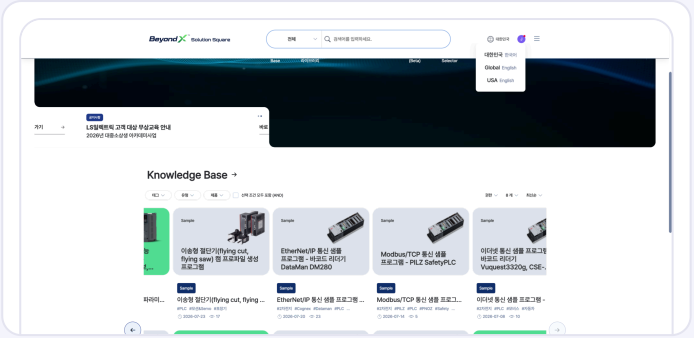
교원그룹 통합 멤버십 쇼핑몰. 렌터카 상품 탐색·상담 신청과 운영 백오피스를 개발하고 기존 서비스를 운영했습니다.

PROFESSIONAL PROJECT · 2026

Solution Square (Global)

2025 – Present

화면



목적

제품 정보, 기술 자료, 선택 도구와 엔지니어링 지식을 여러 지역과 언어로 제공하는 LS ELECTRIC의 글로벌 고객 포털입니다.

역할 · 기술스택

Java/Spring 기반 백엔드 서비스와 프론트엔드 요청을 받는 API 서버(BFF), React·TypeScript 프런트엔드에서 제품·대시보드·검색·AI 서비스 개발에 참여했습니다.

Java · Spring Boot · React · TypeScript · Azure · Elasticsearch · Jenkins

Java | Spring Boot | React | TypeScript | Azure | Elasticsearch | Jenkins

성과 01

인증 로직 공통화

배경·원인

글로벌 포털과 자산 관리 서비스는 같은 뿌리에서 출발한 제품이라, 프론트엔드의 요청을 받아 인증과 백엔드 연계를 담당하는 API 서버(BFF)도 처음에는 하나였습니다.
담당 조직이 분리되면서 자산 관리 서비스 쪽에 같은 역할의 API 서버가 별도로 생겼고, 인증 로직이 하드카피로 중복된 채 각자 수정되기 시작했습니다.
한쪽의 수정이 다른 쪽에 반영되지 않아 인증 동작이 미묘하게 달라졌고, 문제가 생겨도 원인 파악이 어려웠습니다.

고민·선택 이유

별도 인증 서버로 분리하는 안을 먼저 검토했지만 배제했습니다.
운영 대상이 하나 늘고 모든 요청에 네트워크 홉이 붙는 비용을, 두 서비스 규모에서 감수할 이유가 없었습니다.
대신 이미 공통 모듈을 두는 저장소 관례가 있어 그 관례를 따르면 구조 변화 없이 변경 폭이 가장 작았습니다.
추출은 세션 → 인증 토큰·SAML 유틸 → JWT 순으로 나눴는데, 한 번에 옮기면 문제가 생겼을 때 원인 범위가 넓어지므로 단계마다 검증하고 넘어가기 위해서였습니다.
어댑터를 둔 것도 같은 이유로, 두 애플리케이션의 코드를 한 번에 바꾸지 않고 연결점만 교체하기 위해서입니다.

결과

인증 로직 변경이 한 곳의 수정으로 끝나게 되었고 두 애플리케이션의 인증 동작이 같아졌습니다.
이관하며 남은 dead code와 중복 의존성을 정리해 JWT 라이브러리도 단일 소유로 통일했습니다.

회고

레거시 개선은 한 번에 갈아엎는 것보다, 기존 관례를 따라 변경 폭을 줄이고 검증 가능한 단위로 나눠 옮기는 쪽이 안전하다는 기준을 얻었습니다.

성과 02

알림 템플릿·수신자 관리 시스템 구축

배경·원인

이메일 알림 템플릿이 언어별 HTML 파일 54개로 하드코딩돼 있었습니다.
템플릿마다 요청 클래스와 엔드포인트가 따로 있어 알림 종류가 늘수록 코드가 선형으로 늘었습니다.
문구나 수신자를 바꾸는 데도 배포가 필요해 단일 관리 지점이 없었습니다.

고민·선택 이유

템플릿을 코드가 아니라 데이터로 옮기기로 한 것은, 문구·수신자 변경을 배포에서 분리해야 관리자가 직접 다룰 수 있기 때문입니다.
템플릿의 식별을 채널·이벤트 타입·서비스 타입·언어 네 개 축의 조합으로 모델링한 것은, 새 알림이 코드 추가가 아니라 행 추가로 끝나게 하기 위해서입니다.
수신자를 템플릿에 묶지 않고 별도 레지스트리로 분리한 것은, 조직이나 담당자가 바뀔 때 수신자만 바꾸면 되게 하려는 것입니다.
개별 REST API 위에 관리 화면용 통합 API를 더한 것은 화면의 API 왕복을 줄이기 위해서입니다.
발송 구현체는 공통 템플릿 발송 하나로 수렴시켜 템플릿마다 발송 코드가 생기던 구조를 끊었습니다.

결과

관리자가 배포 없이 템플릿과 수신자를 동적으로 관리할 수 있게 됐고, 이후 신규 기능의 알림이 코드 추가 없이 기존 발송 로직을 재활용해 연결됐습니다.
수신자·발송 이력은 개인정보 암호화 대상으로 함께 정리했습니다.

회고

같은 모양의 코드가 반복 증식하는 것은 데이터 모델 부재의 신호이며, 식별 축을 데이터로 옮기면 새 요구가 코드 추가가 아니라 행 추가로 끝난다는 기준을 얻었습니다.

성과 03

개인정보 암호화와 검색의 충돌

배경·원인

개인정보 보호 기준을 높이기 위해 이름·연락처 같은 필드를 암호화해야 했는데, 암호화하면 그 값을 조건으로 하던 기존 동등 검색이 깨집니다.
보호와 검색 가능성이 정면으로 충돌하는 문제였습니다.

고민·선택 이유

암호화를 먼저 하고 검색을 나중에 푸는 순서도 가능했습니다.
다만 그 사이 평문 검색 경로가 남으면 영구히 유지될 위험이 있어, 봉투 암호화와 블라인드 인덱스를 처음부터 함께 설계했습니다.
봉투 방식을 고른 것은 암호문에 키 버전을 함께 저장해 나중에 키를 교체해도 재암호화 경로가 남게 하기 위해서입니다.
운영 중인 데이터가 있어 한 번의 배포 대신 쓰기 → 검색 → 백필 순으로 단계를 나눴습니다.
전환 기간에는 평문과 암호문을 모두 읽게 두었습니다 — 단계 중간에 문제가 생겨도 서비스가 멈추지 않아야 하기 때문입니다.

결과

암호화된 값으로도 동등 검색이 유지되는 상태로 운영 데이터에 단계적으로 적용했습니다.
백필은 중단 후 재실행해도 안전하도록 행 상태 기준으로 재개하게 만들었습니다.

회고

보안 요구는 기능을 깨지 않는 전환 경로까지 설계해야 운영 중인 시스템에 적용할 수 있습니다.
암호화 전환은 각 단계가 재진입 가능한 상태 전이여야 한다는 기준을 얻었습니다.

성과 04

배포 알림·연속 배포 자동화

배경·원인

여러 모듈을 배포하는 환경에서 누가 언제 배포했는지 팀에 공유되지 않았고, 커밋 단위 코드리뷰에서 놓치는 부분이 생길 수 있었습니다.
빌드 서버의 메모리 제약으로 모듈을 병렬 빌드할 수 없어 한 개씩 연속으로 빌드해야 했습니다.
앞 빌드가 끝나기를 사람이 기다렸다가 수동으로 다음 배포를 이어가고 있었습니다.

고민·선택 이유

알림 채널은 팀이 일상적으로 보는 팀즈를 기본으로 두고, 전송 실패 시 이메일로 폴백되게 이중화했습니다.
한 채널의 장애로 알림이 끊기는 순간 자동화 자체를 신뢰할 수 없게 되기 때문입니다.
AI 코드리뷰를 MR 단위와 Push 단위로 분리한 것은, MR만 보면 커밋 단위에서 놓치는 부분이 생길 수 있어서입니다.
배포는 빌드 서버 메모리를 늘리는 대신, 파이프라인이 앞 빌드 완료를 감지해 다음 빌드·배포를 자동으로 잇게 했습니다.
제약은 그대로 두고 사람의 대기만 없애는 쪽이 비용 없이 같은 효과를 내기 때문입니다.

결과

배포 사실과 리뷰 결과가 자동으로 공유되고, 연속 배포에서 사람의 대기가 사라졌습니다.

회고

알림은 성공 경로만이 아니라 전송 실패의 폴백까지 설계해야 신뢰할 수 있고, 자동화의 대상은 작업 자체보다 작업 사이의 대기라는 기준을 얻었습니다.

성과 05

Java 11 → 21 → 25 런타임 현대화

배경·원인

모바일 클라이언트 인증과 고객사 사이트 단위 파일 관리를 담당할 신규 애플리케이션을 올려야 했습니다.
그런데 Java 11 기반의 기존 방식으로는 구현에 제약이 있었고, 오래된 런타임을 그대로 두는 것 자체도 장기 유지보수 위험이었습니다.

고민·선택 이유

현대화는 신규 애플리케이션 구축과 맞물려 시작했습니다.
21을 먼저 병합해 새 기능 개발과 런타임 전환을 같은 구간에서 진행했고, 이후 25로 올리며 Spring Boot 4 BOM으로 버전 관리를 통합했습니다.
Jackson 3 마이그레이션과 JWT 라이브러리 단일 소유 정리, 모듈별 빌드 JDK 전환까지 함께 처리해 모듈마다 의존성이 갈리지 않게 했습니다.

결과

업그레이드가 깨뜨린 것들을 함께 복구했습니다.
HTTPS 판정이 뒤집힌 SAML 헤더 처리와 entity-id 스킴, 비표준 포트 문제가 드러났습니다.
스테이징의 403 차단과 프록시 뒤 클라이언트 IP 유실도 진단 필터까지 넣어 되돌렸습니다.
같은 흐름에서 프론트엔드 빌드를 CRA에서 Vite로 옮기고 검색 클라이언트도 9.x로 올려, 백엔드와 프론트엔드의 낡은 의존성을 함께 걷어냈습니다.
이 전환 덕분에 이후 SAML 표준화도 프레임워크가 지원하는 방식 위에서 진행할 수 있었습니다.

회고

런타임 업그레이드에서 실제로 손이 가는 곳은 버전을 올리는 데가 아닙니다.
환경 차이 때문에 조용히 깨진 동작을 찾아 되돌리는 일이 본체입니다.

성과 06

SAML 표준화와 SSO 확장 기반

배경·원인

새 파트너 IdP와의 SSO 연동 요구가 들어왔는데, 레거시 SAML 구현은 프레임워크 특성상 유연하고 확장 가능한 구조를 만들기 어려웠습니다.
특히 Spring Boot 4에서는 기존 방식의 IdP 구현이 제한되는 상황이라, 당장의 연동을 넘어 향후 확장 개발을 위해 표준화를 사전에 해두는 것이 좋겠다고 판단했습니다.

고민·선택 이유

자체 방식을 고쳐 쓰는 대신 표준 방식으로 전환했습니다.
프레임워크가 지원을 보장하는 표준 위에 있어야 버전 업에서 살아남고, 확장도 프레임워크의 방식대로 풀리기 때문입니다.
제공자별 분기 코드 대신 SSO 제공자 레지스트리를 도입한 것은 새 파트너 연동이 코드 수정이 아니라 등록으로 끝나게 하기 위해서입니다.
IdP 엔드포인트를 전용 보안 체인으로 분리해, IdP에만 필요한 보안 규칙이 나머지 서비스 경로에 영향을 주지 않게 했습니다.
범용 Mock SP 테스트 애플리케이션은 파트너마다 실제 연동 환경을 기다리지 않고 표준 준수를 먼저 검증하려고 만들었습니다.

결과

IdP-initiated 자동 로그인이 동작하는 표준 기반을 갖췄습니다.
연동 검증용 범용 SAML 2.0 Mock SP 테스트 애플리케이션도 마련해 새 제공자 연동을 반복 가능하게 만들었습니다.

회고

프레임워크가 다음 버전에서 제한하는 방식을 먼저 확인하고 표준을 따라두면, 당장의 요구를 넘어 다음 확장의 비용이 줄어든다는 기준을 얻었습니다.

성과 07

룰 기반 제품 구성기 (Drive Panel Selector)

배경·원인

고객이 사양에 맞는 드라이브·패널 구성을 찾으려면 매번 기술 문의를 거쳐야 했습니다.
영업에서는 시스템 안에서 상담과 동시에 견적 문의까지 이어지는 도구를 요구했고, 이를 다단계 구성기로 풀기로 했습니다.

고민·선택 이유

추천을 학습 모델이 아니라 명시적인 룰로 설계한 것은, 제품 구성이 옵션 간 제약이 뚜렷한 공학 문제이기 때문입니다.
잘못된 구성이 견적과 수주로 이어지는 도메인에서는 예측보다 누구나 검증하고 설명할 수 있는 규칙이 맞습니다.
기동방식 같은 선택지는 앞 단계에서 고른 드라이브 용량에 따라 동적으로 좁혀지게 해, 고객이 성립하지 않는 조합에 도달할 수 없게 했습니다.
마지막 단계의 모델 이미지 렌더링, 고객 직접 입력 항목, 관리자 문의 내역 연동과 중복 문의 방지도 함께 설계했습니다.
구성기를 화면에서 끝내지 않고 상담·견적 흐름까지 잇기 위해서입니다.

결과

고객이 기술 문의 없이 스스로 사양에 맞는 구성을 좁혀 결과 모델에 도달하고, 그 자리에서 문의·견적 요청으로 이어지는 흐름이 생겼습니다.

회고

추천처럼 보이는 요구도 옵션 간 의존성을 명시적인 룰로 정리하면 예측 가능하게 풀립니다.
결과 화면이 아니라 상담·견적까지 잇는 것이 도구의 완성이라는 기준을 얻었습니다.

바로가기 — https://ssq.ls-electric.com

PROFESSIONAL PROJECT · 2026

Tech Square

2022.04 – Present

화면

목적

수요기업과 공급기업이 스마트공장 구축 파트너를 찾고 멘토링, 중고설비 거래, 운영 정보를 이용할 수 있는 B2B 제조 플랫폼입니다.

역할 · 기술스택

Node.js · Express · TypeORM · MySQL · Vue 3 · AWS · GitLab CI/CD

Node.js | Express | TypeORM | MySQL | Vue 3 | AWS | GitLab CI/CD

성과 01

Java 시스템에서 Node.js 플랫폼으로 단계적 전환

배경·원인

기존 Java 8 시스템은 품질과 유지보수에 한계가 있었고, Vue 3 고객·관리자 채널과 기술 구성을 맞춘 신규 플랫폼이 필요했습니다.

고민·선택 이유

전환 결정에는 참여하지 않았지만 Node.js·Express·TypeORM 기반 시스템 공동 구축과 데이터 이관을 맡았습니다.
프론트엔드와 언어 생태계를 통일해 협업 경계를 줄이고, 구·신 시스템을 병행해 기능과 데이터를 검증한 뒤 운영을 인수했습니다.

결과

운영 중단 없이 신규 플랫폼으로 전환했으며, 이후 API와 고객·관리자 채널의 운영·고도화를 단독으로 이어갔습니다.

회고

운영 시스템 전환은 새 기술보다 데이터 이관, 병행 운영과 인수 종료 기준을 함께 설계해야 안정적으로 끝낼 수 있습니다.

성과 02

공통 트랜잭션 계층으로 데이터 정합성 기준 통일

배경·원인

업무 모듈이 늘면서 SQL 실행과 트랜잭션 경계가 서비스마다 흩어졌고, 동시 요청이나 부분 실패에서 데이터 정합성을 일관되게 다루기 어려웠습니다.

고민·선택 이유

각 서비스가 TypeORM을 직접 제어하게 두는 대신 공통 SQL·트랜잭션 계층을 만들었습니다.
격리 수준과 커밋·롤백 책임을 한곳에 모아 업무 단위의 성공·실패 기준을 통일하고, 중복 요청은 업무 조건과 트랜잭션으로 함께 차단했습니다.

결과

여러 업무 모듈이 같은 실행·복구 기준을 사용하게 되었고, 동시 요청에서도 중복 저장과 부분 반영을 제어할 기반을 마련했습니다.

회고

트랜잭션 공통화의 목적은 코드 재사용보다 업무 단위의 성공과 실패 경계를 일관되게 만드는 데 있습니다.

성과 03

반복 조회 구조를 쿼리 재구성으로 개선

배경·원인

회원·기업·멘토링 데이터가 늘면서 목록 한 건마다 연관 정보를 다시 조회하는 구조가 생겼고, 검색 조건이 추가될수록 쿼리 조합도 복잡해졌습니다.

고민·선택 이유

제한 시간을 늘리거나 인덱스를 무작정 추가하기보다 반복 조회를 조인·일괄 조회로 전환했습니다.
동적 검색 조건은 파라미터 바인딩 기반 쿼리 빌더로 정리해 실행 구조와 입력 안전성을 함께 개선했습니다.

결과

반복 조회를 줄이고 검색 조건을 안전하게 조합할 수 있게 했으며, 목록과 엑셀 다운로드가 같은 조회 기준을 공유하도록 정리했습니다.

회고

느린 조회는 결과 집합과 반복 횟수를 먼저 확인하고 쿼리 구조를 바로잡은 뒤 필요한 인덱스를 선택해야 합니다.

성과 04

로컬 파일을 S3 기반 공통 파일 흐름으로 전환

배경·원인

업로드 파일이 서버 로컬 경로에 묶여 있어 환경이 바뀌면 탐색·다운로드 처리가 달라졌고, PDF·Excel·이미지마다 별도 예외가 쌓였습니다.

고민·선택 이유

서버 인스턴스와 파일 생명주기를 분리하기 위해 S3로 전환하되 기존 파일을 한 번에 폐기하지 않고 두 경로를 함께 읽는 기간을 뒀습니다.
업로드·조회·삭제·원본 파일명 복원을 공통 유틸리티로 모아 저장 방식 변경이 업무 코드로 퍼지지 않게 했습니다.

결과

공지·보고서·견적·이미지 등 서로 다른 파일 흐름이 공통 저장 방식을 사용하게 되었고, 환경에 종속된 파일 경로와 중복 처리를 줄였습니다.

회고

파일 저장소 전환은 객체 복사뿐 아니라 기존 키, 원본 이름과 조회 경로의 호환성을 함께 보존해야 하는 데이터 이관입니다.

성과 05

인증과 입력 경계를 단계적으로 보안 강화

배경·원인

브라우저에서 접근 가능한 인증 토큰, 동적 SQL, HTML 입력과 다양한 업로드 경로가 각각 다른 보안 위험을 만들고 있었습니다.

고민·선택 이유

JWT는 HttpOnly 쿠키로 옮겨 브라우저 스크립트의 직접 접근을 막고, 동시 401 요청은 한 번의 토큰 갱신으로 수렴시켰습니다.
SQL은 파라미터화하고 HTML과 파일은 허용 목록으로 검증해 인증·데이터·입력·파일 경계마다 적합한 방어를 적용했습니다.

결과

고객·관리자 채널의 인증 흐름을 통일하고, 스크립트 입력·SQL 삽입·위험 파일 렌더링으로 이어질 수 있는 노출 경로를 줄였습니다.

회고

보안은 하나의 필터로 해결되지 않으며 인증, 입력, 데이터와 파일 경계를 각각 확인해 단계적으로 좁혀야 합니다.

성과 06

로그 단편을 요청 흐름으로 연결한 관측성 구축

배경·원인

서비스별 `console` 로그만으로는 한 요청이 API와 데이터 처리 과정에서 어디까지 진행됐는지 연결해 보기 어려웠습니다.

고민·선택 이유

로그를 더 추가하는 대신 전 소스를 구조화 로거로 통일하고 요청별 상관관계 ID를 부여했습니다.
OpenTelemetry traceId까지 연결해 로그 형식을 유지하면서도 요청 흐름을 서비스 경계 너머로 추적할 수 있게 했습니다.

결과

개별 로그를 따로 검색하던 방식에서 벗어나 요청 단위로 문제 발생 지점을 좁힐 수 있는 운영 기반을 마련했습니다.

회고

관측성은 로그의 양보다 서로 다른 기록을 같은 요청 흐름으로 연결할 수 있는지가 중요합니다.

성과 07

시크릿과 로컬 개발환경을 재현 가능한 구성으로 전환

배경·원인

환경별 설정과 시크릿 관리 방식이 분산되어 있었고, 로컬 데이터베이스 터널이 끊기면 애플리케이션 연결도 함께 불안정해졌습니다.

고민·선택 이유

소규모 혼합 인력 팀의 운영 부담을 고려해 Vault보다 진입 장벽이 낮은 Infisical을 선택했습니다.
Docker Compose로 실행 환경을 통일하고 DB 터널은 별도 프로세스로 분리해 끊김 감지, 지수 백오프 재연결과 중복 실행 방지를 적용했습니다.

결과

시크릿을 코드와 배포 파일에서 분리하고 개발자가 같은 실행 절차를 재현할 수 있게 했습니다.
터널 장애도 애플리케이션 전체 재시작에 의존하지 않고 복구할 수 있게 했습니다.

회고

개발환경 표준화는 도구를 맞추는 것보다 설정·시크릿·외부 연결의 실패와 복구 방식을 함께 정의하는 일입니다.

성과 08

반복 업무를 배치·알림 파이프라인으로 자동화

배경·원인

외부 데이터 적재, 상태 확인과 담당자 안내를 사람이 반복하면 누락과 처리 지연이 발생할 수 있었습니다.

고민·선택 이유

대량 데이터는 건별 요청 대신 일괄 적재해 데이터베이스 왕복을 줄이고, 업무 조건별 cron 작업과 알림 흐름을 구성했습니다.
중복 실행·빈 결과·환경별 발송 여부를 분리해 자동화가 예외 상황에서 불필요한 작업을 만들지 않게 했습니다.

결과

기업 데이터 적재와 운영 알림이 정해진 조건과 일정에 따라 실행되어 담당자의 반복 확인 업무를 줄였습니다.

회고

배치 자동화는 주기적 실행보다 중복 처리, 빈 결과와 부분 실패를 안전하게 다루는 것이 핵심입니다.

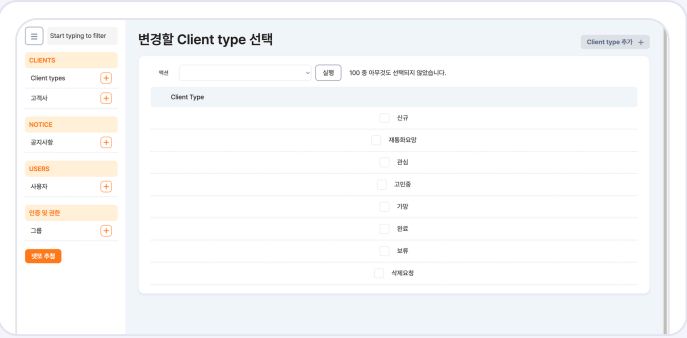
바로가기 — https://tech-square.co.kr

SI PROJECT · 2026

넷플레이스 고객사 관리 시스템

2026

화면



목적

고객사 DB 관리 업무를 시스템화하기 위해 진행했습니다.

역할 · 기술스택

Django 5 · MySQL · Tailwind CSS · Gunicorn

Django 5 | MySQL | Tailwind CSS | Gunicorn

성과 01

재구축 없이 환경을 분리하고 SQLite를 MySQL로 전환

배경·원인

고객사 정보가 엑셀·문서·담당자별 자료에 분산되어 최신 정보와 중복 데이터를 구분하기 어려웠고, 검색·분류와 인수인계에도 제약이 있었습니다.
기존 외주사가 구축한 Python 시스템은 로컬·개발·운영 환경이 분리되지 않은 채 SQLite를 사용하고 있었습니다.

고민·선택 이유

전면 재구축은 비용이 크므로 기존 Python 시스템을 유지하고 위험이 큰 실행환경과 데이터베이스 경계부터 개선했습니다.
환경별 설정과 비밀정보를 분리해 변경 영향을 격리했습니다.
파일 단위 SQLite 대신 서버형 MySQL을 선택해 환경별 데이터베이스와 다중 사용자 연결을 독립적으로 운영하게 했습니다.
데이터는 Django `dumpdata/loaddata`로 이전해 기존 모델과 관계를 최대한 보존했습니다.

결과

개발 변경이 운영 데이터에 직접 영향을 주지 않게 되었고 환경별 테스트와 독립 배포가 가능해졌습니다.
이전 전후 테이블 건수와 핵심 CRUD 화면을 대조해 데이터 누락 없이 기존 기능이 유지되는 것도 확인했습니다.

회고

기존 시스템을 무조건 다시 만들기보다 쓸 수 있는 자산은 보존하고, 데이터와 배포처럼 위험이 큰 경계부터 단계적으로 분리하는 편이 비용과 전환 위험을 함께 낮춥니다.

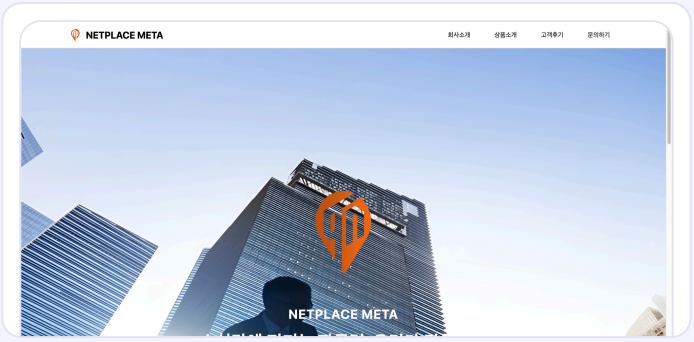
바로가기 — https://dbcs.net-place.co.kr

SI PROJECT · 2025

넷플레이스 랜딩페이지

2025

화면



목적

서비스 소개와 검색 유입(SEO) 확보를 위해 진행했습니다.

역할 · 기술스택

HTML5 · CSS3 · SEO

HTML5 CSS3 SEO

성과 01

검색 유입과 문의 전환을 연결한 랜딩페이지 개편

배경·원인

기존 페이지는 검색 노출과 모바일 대응이 부족했고 유지보수 비용과 기술 부채가 누적되어 있었습니다.
디자인도 고객사가 원하는 방향과 달라 검색·광고 또는 영업 안내로 방문한 잠재 고객에게 서비스 가치를 명확히 전달하기 어려웠습니다.

고민·선택 이유

콘텐츠 중심 페이지에는 별도 프레임워크보다 시맨틱 정적 구성이 적합하다고 판단했습니다.
브라우저 실행 코드와 런타임 의존성을 줄여 초기 로딩과 검색엔진 해석에 유리하고, 서버 애플리케이션 없이 단순하게 배포할 수 있기 때문입니다.
디자인 담당자와 협업해 정보 구조·반응형·CSS 애니메이션을 개선하고, 기존 Marketing API와 AWS SES를 재사용해 메일 자격증명과 발송 로직의 중복을 피했습니다.

결과

고객사가 새 디자인과 구성을 승인해 운영에 반영했고 모바일 사용성과 검색엔진 색인·노출을 개선했습니다.
문의 폼에서 기존 API를 거쳐 상담 이메일이 발송되는 흐름도 완성했으며 정적 배포로 유지보수 절차를 단순화했습니다.

회고

소개 페이지는 기술적 복잡도를 늘리기보다 방문 목적, 검색 노출과 문의 전환에 필요한 최소 구성을 선택해야 합니다.

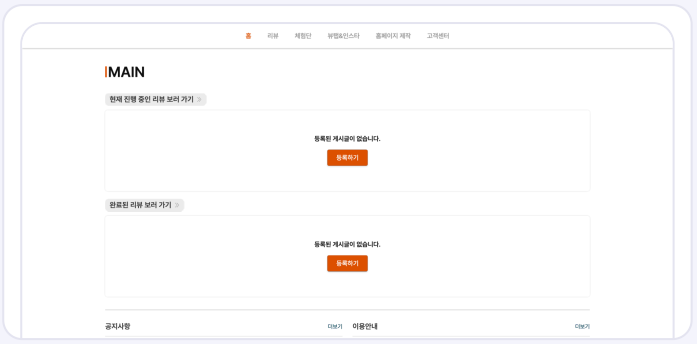
바로가기 — https://welcome.net-place.co.kr

SI PROJECT · 2024

넷플레이스 마케팅 서비스 시스템

2024

화면



목적

지역 마케팅 서비스의 온라인 창구를 구축하기 위해 진행했습니다.

역할 · 기술스택

React 18 · NestJS · MySQL · Docker Compose

React 18 | NestJS | MySQL | Docker Compose

성과 01

수작업 신청 흐름을 24시간 온라인 서비스로 전환

배경·원인

마케팅 상품 안내와 신청을 담당자 문의와 수작업으로 처리해 정보 누락·중복 위험이 있었고, 운영자는 신청 내용을 반복 입력해야 했습니다.
고객과 처리 상태를 공유하기도 어려웠으며 신청 가능 시간도 영업시간에 제한됐습니다.

고민·선택 이유

고객 화면과 업무 API를 React와 NestJS로 분리해 독립적인 개발·배포 경계를 만들고, 관리자 채널도 같은 API와 데이터 계약을 재사용하도록 구성했습니다.
화면 변경이 업무 규칙에 직접 영향을 주지 않고, 고객·관리자 채널에서 신청 데이터를 중복 구현하지 않게 하기 위한 선택이었습니다.

결과

고객이 시간에 관계없이 회원가입·로그인 후 상품을 조회하고 서비스를 신청할 수 있게 됐습니다.
신청 정보가 일관된 형식으로 저장되어 누락·중복과 운영자의 반복 입력을 줄였고, 고객·관리자 채널이 동일한 신청 데이터를 공유하게 했습니다.

회고

온라인화는 화면을 만드는 데서 끝나지 않고, 누락·중복을 막는 데이터 구조를 먼저 정한 뒤 고객 신청과 운영 처리를 하나의 흐름으로 연결해야 합니다.

성과 02

JWT 기반으로 고객·관리자 권한 경계 분리

배경·원인

고객과 관리자가 동일한 API를 사용하므로 로그인 여부만 확인해서는 내부 운영 기능과 데이터 접근 범위를 안전하게 구분할 수 없었습니다.

고민·선택 이유

고객·관리자 인증 경로를 분리하고 API에서 역할 기반 접근 제어를 적용했습니다.
JWT는 HttpOnly 쿠키로 전달해 브라우저 스크립트가 토큰에 직접 접근하지 못하게 하면서도 두 채널이 동일한 인증 계약을 사용할 수 있게 했습니다.

결과

고객과 관리자가 같은 업무 API를 재사용하면서도 각 역할에 허용된 기능만 접근하도록 경계를 분리했습니다.

회고

인증은 로그인 성공보다 사용자 유형별 권한과 데이터 접근 경계를 먼저 설계해야 합니다.

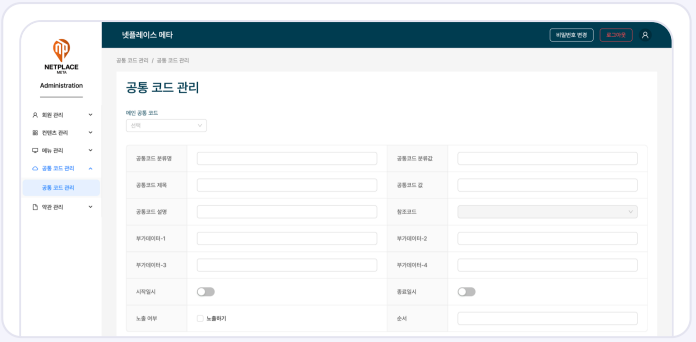
바로가기 — https://net-place.co.kr

SI PROJECT · 2024

넷플레이스 마케팅 서비스 관리자 시스템

2024

화면



목적

마케팅 서비스 운영(고객·서비스 현황 관리)을 위한 백오피스로 진행했습니다.

역할 · 기술스택

React 18 · NestJS · MySQL

React 18 NestJS MySQL

성과 01

고객 서비스와 분리된 관리자 운영 채널 구축

배경·원인

고객 신청을 조회·처리하고 고객·상품·담당자·상태를 관리할 내부 채널이 필요했습니다.
고객에게 노출하면 안 되는 운영 기능과 일반 관리자·최고 관리자의 권한도 분리해야 했습니다.

고민·선택 이유

고객 화면 안에 관리자 기능을 조건부로 넣지 않고 별도의 React 애플리케이션으로 분리했습니다.
고객 채널과 배포·노출 경계를 분리하면서도 동일한 API 계약은 재사용하고, 정보 밀도가 높은 운영 화면은 Ant Design으로 일관되게 구성하기 위한 선택이었습니다.

결과

고객·상품·신청 업무와 담당자 배정, 상태 변경을 하나의 관리자 채널에서 처리할 수 있게 했습니다.
일반 관리자는 운영 업무를, 최고 관리자는 관리자 계정과 권한까지 관리하도록 구분했으며 실제 운영 환경에 배포했습니다.

회고

고객 채널과 관리자 채널은 데이터 계약은 공유하되 UI, 권한과 배포 경계를 분리해야 각 채널의 변화가 서로를 불필요하게 제약하지 않습니다.

성과 02

반복되는 목록·검색·폼을 공통 구조로 표준화

배경·원인

고객·상품·신청 관리 화면마다 목록, 검색, 필터, 페이지네이션과 등록·수정 폼이 반복됐습니다.
화면별로 독립 구현하면 상태 처리 방식이 달라지고 신규 화면을 추가할 때 같은 코드를 다시 작성해야 했습니다.

고민·선택 이유

공통 테이블·검색·폼 컴포넌트를 만들고 화면별 설정과 데이터만 주입하도록 구성했습니다.
검색·페이지네이션·API 호출 상태는 커스텀 훅으로 분리해 UI와 데이터 처리 책임을 나누고 화면 간 동작을 통일했습니다.

결과

반복되는 운영 UI의 조작 방식을 일관되게 유지하고, 신규 관리 화면의 추가·수정 범위를 줄였습니다.

회고

관리자 화면은 개별 페이지를 빠르게 늘리기보다 반복되는 목록·검색·폼 패턴을 먼저 공통화해야 장기적인 변경 비용을 줄일 수 있습니다.

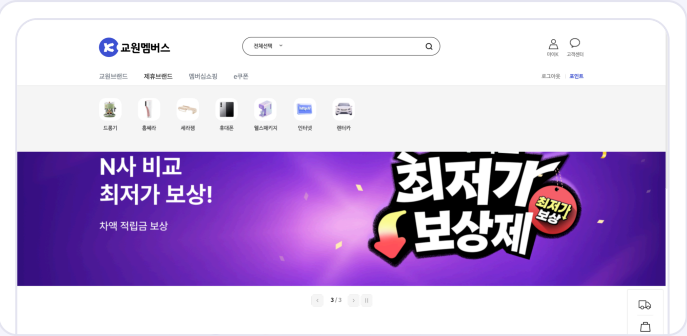
바로가기 — https://admin.net-place.co.kr

PROFESSIONAL PROJECT · 2020

K멤버스

2020.01 – 2021.02

화면



목적

교원그룹 임직원과 회원이 그룹 상품과 제휴 상품을 포인트로 이용하는 통합 멤버십 쇼핑몰로, 렌터카 상품과 상담 신청 흐름을 새로 제공했습니다.

역할 · 기술스택

Spring Framework · MyBatis · Oracle · JSP · JavaScript · jQuery

Spring Framework | MyBatis | Oracle | JSP | JavaScript | jQuery

성과 01

오프라인 계약을 포함한 렌터카 상담·포인트 운영 흐름 구축

배경·원인

K멤버스는 임직원과 회원이 이용하는 교원그룹 통합 멤버십 쇼핑몰이었지만, 상담 후 오프라인 계약으로 매출이 발생하는 렌터카 상품의 판매 흐름은 없었습니다.
상담 접수부터 담당자 배정·계약 확인·포인트 지급까지 기존 상품보다 복잡한 운영 절차가 필요했습니다.

고민·선택 이유

기획·운영 조직이 정의한 신청 접수 → 상담사 배정 → 처리 완료 상태를 사용자 화면과 백오피스 로직으로 연결했습니다.
포인트는 신청 시점이 아니라 실제 계약이 확인되는 처리 완료 시점에 지급해 취소 가능성이 있는 오프라인 계약과 포인트 정책의 정합성을 유지했습니다.

결과

쇼핑몰 안에서 렌터카 상품을 탐색하고 상담을 신청한 뒤, 운영자가 담당자 배정·처리 상태·포인트 지급까지 관리할 수 있는 새로운 판매 흐름을 마련했습니다.

회고

기존 서비스에 성격이 다른 상품을 추가할 때는 화면보다 계약·취소·보상 정책이 바뀌는 상태를 먼저 모델링해야 합니다.

성과 02

고정 월과 가변 계열사를 함께 다룬 매출현황 구조

배경·원인

운영자가 교원그룹 계열사별 매출을 월 단위로 확인해야 했습니다.
월은 1~12월로 고정되어 있지만 계열사 목록은 바뀔 수 있어 두 축을 같은 고정 배열로 처리하기 어려웠습니다.

고민·선택 이유

월 12개는 고정 크기 배열로 두고, 각 월에 계열사별 매출 객체 목록을 담았습니다.
고정 축과 가변 축을 분리해 월별 조회 구조를 유지하면서도 계열사 증감에 데이터 구조를 다시 바꾸지 않게 하기 위한 선택이었습니다.

결과

운영자가 계열사별 월간 매출 상태를 한 화면에서 확인할 수 있게 했고, 계열사 목록이 달라져도 동일한 월별 집계 구조를 유지했습니다.

회고

고정된 시간 축과 변하는 조직 축을 함께 다룰 때는 두 차원을 분리해 모델링해야 변경 범위를 줄일 수 있습니다.

바로가기 — https://www.kwmembers.com

감사합니다

Thank you

해당 포트폴리오는 **https://jihwang.kim**에서 더 많은 정보를 온라인으로 확인 가능합니다.