경력기술서 — 길현준

게임듀오

DEV팀 Server Developer

2025.01 — 현재

모바일 게임 개발 및 퍼블리싱 회사. 12개 라이브 서비스 공통 백엔드 플랫폼 설계, 데이터 파이프라인 구축, 공통 패키지 체계 운영

주요 성과

- 마케팅 데이터 조회 비용 급증(전체 스캔 과금 구조). 파티셔닝 튜닝 분석 후 Write IOPS 증가 문제로 기각, gRPC 기반 분산 읽기(Storage Read API) 전환. 비용 82% 절감(TB당 $6.25→$1.1).
- 배치 처리의 수집 기간 한계(60일). 서버리스 아키텍처 전환. 수집 범위 6배 확장(60일→360일), 처리 시간 2시간→5분 단축.
- 7개 프로젝트 간 코드 중복과 컨벤션 불일치. 공통 패키지 체계 설계(NestJS 유틸 10모듈 + 게임서버 Kit 2패키지). 업그레이드 자동화로 배포 시간 3시간→15분 단축.
- 3개 광고 플랫폼 데이터 분산에 따른 마케팅 조회 지연(18초). 테이블 정규화 및 인덱스 튜닝. 조회 성능 97% 개선(18초→0.5초).
- 게임 확률 공시 의무화 규제 리스크. DynamicModule 기반 확률 패키지를 다중 게임에 통합 + CDK 기반 감사 로그 파이프라인. 통합 테스트 커버리지 94%+ 확보(12 suites, 115 tests).

프로젝트

Glider Sheet Audit Log 시스템

2025.01 ~ 2025.04

게임 운영 중 데이터 변경 이력 추적 불가로 인한 밸런스 문제 대응 지연 해결

- **6개 게임 간 데이터 변경 이력 추적 불가** — UUID 기반 크로스 환경/프로젝트 엔티티 추적 Git-like 버전 관리 시스템 설계, 6개 게임 간 데이터 일관성 보장
- **엔티티 변경 기록의 수동 관리 부담** — Auditable 데코레이터 + TypeORM Subscriber 패턴으로 이벤트 소싱 기반 변경 자동 기록 파이프라인 설계, 엔티티 변경 이력 자동 추적 체계 구축
- **다중 환경 간 엔티티 충돌 감지 및 병합 부재** — 상위/하위 엔티티 충돌 감지와 유니크 제약조건을 고려한 3-Way Merge 엔진 구현, 6개 게임 다중 환경 간 버전 병합 자동화
- **버전 간 변경 사항 비교 비효율** — Base Audit 유무에 따른 증분 비교와 스냅샷 비교의 이중 전략 Diff 엔진 설계, 비용 효율적 버전 비교 체계 확립
- **PK 의존 엔티티 추적의 환경 간 불일치** — 공유 식별자 기반 크로스 환경 엔티티 추적 체계 설계로 PK 의존 탈피, 마이그레이션/비교/병합 시 정확한 엔티티 추적 보장

> **기술 선택 이유**
> TypeORM EntitySubscriberInterface 동작과 제약을 별도 분석 후 채택. Auditable 데코레이터 + Subscriber 조합으로 엔티티 변경을 표준화된 Audit 파이프라인으로 자동 수집하는 AOP 방식 설계.

TypeScript NestJS TypeORM MySQL

마케팅 통합 플랫폼

2025.02 ~ 현재

마케팅 전용 DB 분리 기반 위에 Google Ads/Meta/TikTok 캠페인·소재·지표를 단일 시스템에서 관리하는 통합 마케팅 플랫폼

- **3개 광고 플랫폼 캠페인 관리 분산** — Google Ads/Meta/TikTok 캠페인 생성·배포·수정과 리텐션 지표 관리를 단일 플랫폼에서 처리하는 API 개발, 통합 캠페인 자동화 체계 마련
- **NAS-S3 간 에셋 동기화 신뢰성 부족** — SQS 서버-워커 분리, outbox 자동 발행, asset_nas_sync 상태추적 설계, 에셋 복원 신뢰도 보장
- **Meta 에셋 동기화 성능 병목** — ORM 단건 저장을 벌크 처리로 전환, Image 동기화 72% 단축(25.7s→7.1s), DB 트랜잭션 95% 단축(10~16s→0.3~0.5s)
- **소재 처리 Lambda 비용 부담** — Graviton2 ARM64 아키텍처 전환, 소재 처리 Lambda 비용 20% 절감
- **마케팅 데이터 조회 비용 급증** — BigQuery Storage Read API 도입으로 gRPC 스트리밍 기반 고성능 데이터 읽기 전환, BigQuery 비용 82% 절감(TB당 $6.25→$1.1)
- **100+ 컬럼 테이블 조회 성능 저하(18초)** — 테이블 3개(메인/시계열/예측) 정규화, 인덱스+커서 페이지네이션 적용, 조회 성능 97% 개선(18s→0.5s)

> **기술 선택 이유**
> BigQuery Storage Read API로 비용 82% 절감. GCP Pub/Sub → AWS Lambda/SQS 하이브리드 파이프라인으로 이벤트 기반 처리 전환, Outbox 패턴으로 API 비차단 비동기 발행 설계.

AWS Lambda Apache Arrow BigQuery GCP Pub/Sub Google Ads API Meta API MySQL NestJS S3 SQS BigQuery (Storage Read API) TikTok API TypeORM TypeScript gRPC

AWS Lambda 마이그레이션 & Event-Driven Architecture

2025.06 ~ 2025.08

마케팅 지표 60일 → 360일 확장에 따른 배치 작업 한계 해결 및 배치 처리 서버리스 전환

- **전체 서버리스 전환 시 운영 안정성 리스크** — API 서버(EC2)는 유지하고 배치/Job 처리만 Lambda로 분리하는 하이브리드 아키텍처 설계, 워크로드별 최적 리소스 활용 달성
- **배치 처리의 수집 기간 한계(60일) 및 처리 시간 2시간** — SQS+Lambda+EventBridge 기반 Event-Driven 비동기 처리 전환, 처리 시간 2h→5m 단축, 수집 범위 6배 확장(60일→360일)
- **비동기 전환 시 데이터 정합성 보장 필요** — 예약·지연 발행 기반 트랜잭션 아웃박스 패턴 적용, 이벤트 기반 아키텍처에서 데이터 정합성 보장
- **Lambda 스로틀링, CloudWatch 비용 증가, 빌드 OOM 발생** — Batch Size 벌크 처리, CloudWatch 로그 경량화, 빌드 OOM 조치, Lambda 운영 안정화 및 비용 절감
- **Lambda 대량 실행 시 DB 커넥션 고갈** — RDS Proxy 풀링 기반 커넥션 관리 도입, DB 커넥션 고갈 문제 해결

기술 선택 이유

배치·이벤트성 작업이 서버 본체에 묶여 독립 배포 불가한 제약 해소를 위해 Lambda 선택. SQS로 비동기화하여 트래픽 변동 시 처리 지연과 확장성 한계 해결.

TypeScript | NestJS | AWS Lambda | SQS | SNS | EventBridge | RDS Proxy | AWS CDK

Cloud Data 동기화 시스템

2025.08 ~ 현재

환경 간 동적 게임 데이터 불일치 문제를 해결하기 위한 S3 기반 동기화 및 DDL 자동 관리 시스템

- **환경 간 동적 게임 데이터 불일치** — S3 기반 개발/스테이징/프로덕션 환경 간 동기화 시스템 구축, 환경 간 데이터 일관성 보장
- **환경 간 스키마 불일치 수동 관리 부담** — PK 컬럼 타입 동적 결정, 컬럼 타입 불일치 감지·MODIFY, 인덱스 자동 생성·RENAME DDL 자동 관리 엔진 구현, 스키마 동기화 자동화
- **Cloud Data 적재 및 S3 업로드 성능 병목** — 병목 구간 분석 및 저장/업로드 최적화, 대용량 처리 지연 해소
- **동기화 작업 불안정(타임아웃, 스케줄링 충돌)** — job 분리, 스케줄링 방식 전환, timeout 조정, non-blocking 처리 적용, 운영 안정성 개선
- **CloudData 운영 장애 4건(캐시, 메타데이터, 키, 옵션 누락)** — Redis 캐시 무효화, DDL SKIP 메타데이터, 복사 키 삭제, excludeCloudData 전달 누락 수정, CloudData 안정화
- **시트 마이그레이션 시 excludeCloudData 옵션 미전파로 전체 삭제 리스크** — POST 경로 4곳에 excludeCloudData 옵션 전달 보강, Cloud Data 전체 삭제 리스크 차단

기술 선택 이유

S3 Lifecycle 정책(30일 Glacier IR, 90일 만료)으로 비용 최적화. 이벤트성 실행에서 스케줄링 방식으로 전환하여 동기화 누락 위험 감소.

TypeScript | NestJS | TypeORM | MySQL | S3

사내 공통 라이브러리 체계

2025.07 ~ 현재

NestJS 유틸 10개 모듈 + 게임서버 Kit 2개 패키지 개발·운영으로 다중 프로젝트 코드 일관성 달성

- **서비스 간 공통 코드 중복 및 유지보수 비용 증가** — core, repository, cache, lock, slack, crypto, smb, hash, type, iac 10개 모듈 라이브러리 체계 설계, 7개 프로젝트 공통 코드 일원화 및 의존성 관리 표준화
- **Repository 계층 코드 비대화(2000+줄) 및 타입 안전성 부족** — Bulk·Audit Log·TypeORM 타입 좁히기 지원, 함수 오버로딩+TypeScript 제네릭 적용, SRP 분리, Repository 모듈 코드 품질 및 재사용성 향상
- **멀티 인스턴스 환경의 동시성 제어 부재** — ElastiCache (Redis) 기반 분산 락 데코레이터 설계, 멀티 인스턴스 환경 동시성 제어 체계 구축
- **7개 프로젝트 패키지 업그레이드 수작업(3시간)** — workflow_dispatch+matrix 병렬 실행, 변경 패키지 CI 테스트 자동화(GitHub Packages), 업그레이드 배포 시간 3h→15m 단축
- **게임서버 공통 코드 5개 브랜치 분산 관리** — Sheet+5개 서브모듈을 @gameduo/glider-sheet로 추출, 5개 브랜치 차이 분석·충돌 해결, 게임서버 Kit 패키지 분리 완료
- **CI 테스트 실행 시간 과다(15m47s)** — ts-jest isolatedModules+Entity 타입 명시 적용, CI 61% 단축(15m47s→6m06s), 81 suites 978 tests 통과

기술 선택 이유

ElastiCache(Redis) 기반 분산 락을 AOP 데코레이터로 구현하여 비즈니스 로직에서 락 코드 분리. Jest 30 VM 격리 대응 시 5가지 옵션 검토 후 poolSize=2 채택.

ElastiCache (Redis) | GitHub NPM | NestJS | Slack API | TypeORM | TypeScript

확률 계산 및 감사 로그 분석 파이프라인 (Glider Probability)

2026.02 ~ 현재

게임 확률 검증 체계 부재에 따른 규제 리스크 해소를 위한 확률 계산 패키지와 CDK 기반 감사 로그 분석 인프라 개발

- **12개 게임 확률 계산 로직 분산 및 규제 대응 부재** — NestJS DynamicModule 기반 5개 함수+Kinesis 로깅을 @gameduo/glider-probability로 패키지화, 12개 게임 확률 패키지 통합
- **확률 감사 로그 분석 인프라 부재** — Kinesis→Firehose(Dynamic Partitioning+Parquet)→S3→Glue→Athena E2E 파이프라인 CDK 코드화, 확률 감사 로그 분석 인프라 조성
- **모킹 기반 테스트의 회귀 신뢰도 부족** — 모킹 삭제, LocalStack+Testcontainers 기반 통합 테스트 교체, 회귀 신뢰도 강화, 12 suites 115 tests 커버리지 94%+ 확보

기술 선택 이유

Kinesis → Firehose(Dynamic Partitioning + Parquet) → S3 → Glue → Athena 파이프라인을 CDK로 코드화. Parquet 컬럼형 포맷으로 Athena SQL 분석 비용 최적화, LocalStack Testcontainers로 모킹 기반 테스트 제거.

기여도: *changmin님 초기 구현체를 패키지 구조화. P0 버그 수정, 테스트 안정화(94%+ 커버리지), CDK 인프라 구축, 12개 프로젝트 적용은 단독 수행.*

TypeScript NestJS AWS CDK Kinesis Firehose S3 Athena Glue Parquet LocalStack Testcontainers

외부 활동

Amazon Q Developer를 활용한 개발 생산성 향상 외부 발표 2025.10

Games on AWS 2025 고객 세션 발표

개발자 1인이 10일 만에 시간당 병렬 처리 Job 수 기준 데이터 파이프라인 용량 270배 확장한 사례 공유

제이앤피메디

개발2팀 Docs Squad Backend Engineer

2023.06 — 2024.08

누적 160억 투자 유치 임상시험 데이터 관리 솔루션 스타트업 (MSA 기반)

주요 성과

- 월 5건+ 발생하던 전자서명 처리 실패(임상시험 Deadlock). FK 공유 락 교착 패턴 재현·규명. 실패 0건 해소, Deadlock 근본 원인 제거.
- VDR 프로젝트 일정 지연 및 MVP 출시 압박. BE 2명 + FE 1명 팀으로 3주 긴급 투입, Event-Driven PDF 변환과 권한 기반 워크플로우 구현. MVP 출시 일정 준수.
- 이메일 발송 실패 사후 인지(수시간 지연). SES+SNS 이벤트 파이프라인 구축. 반송·바운스 감지 수시간→수초 단축, Slack 즉시 알림 체계 수립(기술 블로그 게재).

프로젝트

Maven Docs 전자 동의서 시스템

2023.07 ~ 2023.09

Cyan(사내 Node.js 프레임워크) 기반 임상시험 전자 동의서 수집 비효율성과 전자서명 규제 준수 요구사항 해결

- **임상시험 동의서 개별 발송의 비효율** — 임상시험별 참여자 그룹 관리 및 일괄 동의서 배치 발송 시스템 구현, 동의서 발송 프로세스 자동화
- **배치+S3 이벤트 기반 아키텍처의 이벤트 흐름 비가시성** — EDA+아웃박스 패턴으로 변경, 이벤트 흐름 가시화 및 로컬 테스트 환경 정비

> **기술 선택 이유**
> 동기 서비스 간 결합도 제거를 위해 EDA + 아웃박스 패턴으로 전환. 다양한 입력 방식(텍스트·체크박스·서명) 확장을 위해 플러그인 구조 설계.

기여도: *EDA 전환 설계, 플러그인 구조 구현, 대상자 일괄 등록 + Signer 테이블 분리 마이그레이션 수행. BE 2명 체제.*

TypeScript | Express.js | Cyan | Aurora Serverless v2 | Knex.js

Maven Docs 시스템 안정성 개선

2023.08 ~ 2023.12

임상시험 전자서명 알림에서 발생하는 데이터베이스 Deadlock으로 인한 치명적 알림 누락 해결

- **운영 환경 Deadlock 발생** — 외래키 생성 시 공유락→배타락 전환 타이밍 이슈 근본 원인 분석, Deadlock 근본 원인 규명 및 재발 방지
- **동기 처리 방식의 알림 병목 및 장애 전파** — AWS SNS → SQS → Lambda Event-Driven Architecture 설계, 비동기 파이프라인 완성으로 알림 안정성 확보

> **기술 선택 이유**
> FK child INSERT 시 S-Lock → X-Lock 승격 교착 패턴이 근본 원인. INSERT 전 SELECT FOR UPDATE로 X-Lock 선점하여 교착 순서 자체를 제거.

기여도: *Deadlock 재현·분석·해결 방안 도출 단독 수행. DBeaver로 교착 시나리오를 SQL로 재현하고 INNODB 잠금 테이블로 확인.*

TypeScript | Express.js | Cyan | Slate.js | SNS | SQS | Lambda | React

Maven Mailing 시스템 고도화

2023.12 ~ 2024.02

이메일 발송 시스템의 발송 상태 추적 부족과 데이터 정합성 문제 해결

- **이메일 발송 실패 사후 인지(수시간 지연)** — SES 구성세트 → SNS → 이벤트 처리 파이프라인으로 발송/반송/바운스 상태 추적, Datadog 로깅 + Slack 실시간 알림 구축, 반송·바운스 감지 수시간→수초 단축

> **기술 선택 이유**
> AWS SES 비동기 특성상 발송 성공 여부를 동기적으로 수신 불가. 구성세트에 SNS 이벤트 대상을 추가하여 발송/반송/바운스 이벤트를 비동기 구독.

기여도: *SES 구성세트 → SNS → 이벤트 처리 파이프라인 설계·구현. 기술 블로그 게재 및 사내 세미나 발표 주도.*

TypeScript | Express.js | AWS SES | Nodemailer

Maven Auth 시스템 고도화

2024.01 ~ 2024.02

기존 1:1 사용자-조직 구조로 인한 사용자 불편과 시스템 확장성 한계 해결

- **1:1 사용자-조직 구조의 확장성 한계** — 1:N 구조로 관계 확장하여 다중 조직 관리 지원, 다중 조직 관리 기능 제공
- **구조 변경 시 기존 API 호환성 단절 리스크** — 기존 API 무중단 마이그레이션 설계, 서비스 연속성 유지

> **기술 선택 이유**
> 임상시험 도메인 특성상 연구자가 여러 조직에 동시 참여하는 요구사항 수용을 위해 계정-조직 1:N 재설계. 로그인 후 조직 선택 흐름 추가로 컨텍스트 전환 지원.

기여도: *스키마 마이그레이션, 인증 흐름 수정, 하위 서비스(Docs/Billing/Mailing) 계정-조직 참조 로직 일괄 수정까지 풀스택 변경.*

TypeScript | Express.js | Cyan | Aurora Serverless v2

Maven TMF 신규 프로젝트 개발 (프론트엔드)

2023.06 ~ 2024.08

임상시험 필수 문서(TMF) 관리 시스템의 프론트엔드 신규 개발

- **임상시험 필수 문서(TMF) 관리 시스템 부재** — Admin/Dashboard 페이지에서 문서 업로드·분류·버전 관리·감사 추적(Audit Trail) 기능 구현, TMF 문서 관리 프론트엔드 체계 구축

> **기술 선택 이유**
> 3개월 내 출시 일정 제약으로 핵심 기능(문서 업로드·분류·버전 관리·감사 추적)만 MVP 범위로 정의하고 집중 개발.

기여도: *BE 1명 + FE 2명(본인 포함) 체제에서 FE 담당. Admin/Dashboard 페이지의 문서 업로드·분류·감사 추적 기능 구현.*

React TypeScript Jotai

Maven VDR MVP 개발

2024.06 ~ 2024.07 (3주)

진행 중이던 VDR 프로젝트의 개발 일정 지연과 MVP 출시 압박 상황에서의 긴급 투입

- **PDF 변환 처리의 동기 방식 병목** — S3 업로드 트리거 → Lambda 경량화 변환 → SNS/SQS 알림 Event-Driven 체계 설계, 비동기 PDF 변환 파이프라인 구축
- **의료진별 문서 접근 권한 체계 부재** — 역할 기반 문서 접근 및 승인 권한 관리 워크플로우 제작, 권한 기반 문서 관리 체계 구축
- **VDR 프로젝트 일정 지연 및 MVP 출시 압박** — 3주 긴급 투입, 핵심 기능 우선순위 선정 및 집중 개발, 일정 내 MVP 출시 달성

> **기술 선택 이유**
> 동기 PDF 변환 시 타임아웃 발생 문제를 구조적으로 해결하기 위해 EDA 기반 비동기 변환 파이프라인(S3 트리거 → Lambda 변환 → SNS/SQS 알림) 설계.

기여도: *BE 2명 중 1명. EDA 기반 문서 처리 파이프라인 설계·구현, 대용량 PDF 비동기 변환 워커 구현. 3주 긴급 투입으로 MVP 출시 달성.*

TypeScript Lambda S3 SNS SQS

Maven Billing 구독 관리 시스템

2023.06 ~ 2024.08

Maven 전체 서비스의 구독/플랜/라이센스 관리 시스템 개발

- **조직별 맞춤 요금제 제공 불가** — ORG별 커스텀 플랜 Internal API 구현, 조직별 맞춤 요금제 제공 체계 구축
- **갱신 불가 플랜의 구독 갱신·재구독 정책 미적용** — 갱신 불가 플랜 구독 갱신·재구독 차단 로직 작성, 비즈니스 정책 정합성 보장

> **기술 선택 이유**
> 구독 만료 후 비동기 정리 처리를 위해 event queue 도입. 갱신 불가 플랜의 재구독 차단을 bundles.is_renewable 스키마 속성으로 비즈니스 정책을 데이터 레벨에서 강제.

기여도: *ORG별 커스텀 플랜, event queue 기반 인벤토리 자동 삭제, 갱신 가능 여부 판별 로직 구현. 다수 PR 직접 기여.*

TypeScript Express.js Cyan Aurora Serverless v2 Knex.js

외부 활동

AWS SES 이벤트 로그를 통한 이메일 발송 모니터링 기술 블로그 — 2024.03

AWS SES 이벤트 로그 기반 이메일 발송 모니터링 시스템 개발 사례 공유

메일링 시스템 이메일 발송 결과 수신 기능 도입 사내 세미나 — 2024.02

메일링 시스템 이메일 발송 결과 수신 기능 도입 과정 및 아키텍처 공유

AWS SAA 스터디 스터디 — 2023.09 ~ 2023.12

제이앤피메디 사내 스터디, 클라우드 아키텍처 심화 학습

메드고

개발팀 Backend Engineer

2022.04 — 2023.05

누적 30만 다운로드 비대면 진료 및 약배달 플랫폼 스타트업

주요 성과

- Express.js 단일 파일(app.js) 10,000줄 레거시 유지보수 한계. 3단계 점진적 전환(단일 파일→레이어드→NestJS) + JS→TS 마이그레이션. TDD 80% 달성, 주요 릴리스 구간 장애 0건 유지.
- 수동 처방전 입력 병목(약사 건당 3~5분). Naver Clova OCR + 식약처 API 연동으로 처방전→복약지도 자동화. 약사 처리 시간 90% 단축(3~5분→30초).
- 진료·조제·배달 상태 반영 지연(수십 초). Socket.IO 실시간 동기화 + Web Push 알림 도입. 상태 반영 1초 미만으로 단축, 환자 대기 불만 해소.

프로젝트

Auth 서버 신규 개발

2022.04 (3주)

서브도메인 간 세션 충돌 문제를 해결하기 위한 JWT 기반 통합 인증 시스템 신규 개발

- **서브도메인 간 세션 충돌로 인한 인증 불안정** — NestJS + JWT 기반 토큰 인증 시스템 설계 및 구현, 통합 인증 시스템 구축
- **세션 방식의 서브도메인 간 사용자 전환 불가** — 세션→토큰 방식 마이그레이션, 서브도메인 간 원활한 사용자 전환 구현

> **기술 선택 이유**
> 서브도메인 간 쿠키 기반 세션 충돌을 JWT 무상태 토큰으로 구조적 해결. Redis 대신 MySQL 키 관리 — 트래픽 규모 대비 Redis는 오버엔지니어링으로 판단하여 인프라 단순화.

기여도: *3주간 단독 설계·구현. 웹(의사/약사/관리자 포털) + 모바일 앱 통합 인증 체계 전반.*

NestJS TypeScript JWT MySQL

공통 모듈 개발

2022.05 ~ 2022.06

서비스별 독립 관리로 인한 코드 중복과 일관성 부족 문제를 공통 모듈 통합으로 해결

- **서비스별 권한 검증 로직 중복 및 보안 일관성 부족** — 사용자 권한 확인 미들웨어 공통 모듈화, 보안성 강화 및 권한 검증 일원화

NestJS TypeScript

의사/약사 웹 서비스 고도화

2022.04 ~ 2023.05

의사·약사용 백오피스 시스템의 실시간 상태 반영 부재와 수동 알림 체계 문제를 Socket.io 및 Web Push 도입으로 해결

- **중복 예약 발생** — 의사ID+시간대 복합 유니크 키 적용, 중복 예약 원천 차단
- **진료·조제·배달 상태 반영 지연(수십 초)** — Socket.IO Room 기반 실시간 상태 동기화로 이벤트 즉시 반영, 상태 반영 1초 미만으로 단축, 환자 대기 불만 해소
- **외부 서비스(배달·결제) 장애 전파 리스크** — 배달(후다닥) 및 결제 외부 서비스 통합 API 게이트웨이 설계, Circuit Breaker 패턴 적용, 외부 서비스 장애 전파 차단

> **기술 선택 이유**
> Socket.IO Room 단위 이벤트 격리로 진료·조제·배달 상태 실시간 동기화. 외부 서비스 장애 전파 방지를 위해 API 게이트웨이 + Circuit Breaker 통합. DB 복합 유니크 키로 중복 예약을 데이터 레벨에서 차단.

기여도: *BE 2인 체계에서 Socket.IO 실시간 동기화, Web Push 알림, API 게이트웨이, 진료 예약 중복 방지 전 기능 직접 구현.*

Express.js NestJS TypeScript EJS MySQL Socket.io Web Push

처방전 자동 인식 및 복약지도 시스템

2022.11 ~ 2022.12

약사의 수동 처방전 입력 과정을 OCR 연동으로 자동화

- **처방전 수동 입력 병목** — Naver Clova OCR API 연동으로 처방전 이미지에서 텍스트 자동 추출, 처방전 입력 자동화
- **복약지도 수동 작성 비효율** — 공공데이터(식약처 API) 의약품 정보 활용한 자동 복약지도 생성, 약사 건당 처리 시간 90% 단축(3~5분→30초)

> **기술 선택 이유**
> 한국어 의약품 용어 인식률을 고려하여 Naver Clova OCR 선택. 의약품 정보 DB 자체 구축 대신 식약처 공공데이터 API 활용으로 유지보수 부담 제거.

기여도: *2개월간 단독 구현. 이미지 수신 → OCR 추출 → 식약처 API 매칭 → 복약지도 생성 파이프라인 전 구간.*

NestJS TypeScript Naver Clova OCR MySQL

백엔드 아키텍처 전환

2023.03 ~ 2023.05

Express.js 단일 파일(app.js) 구조에서 NestJS 모듈 아키텍처로 전환

- **Express.js 단일 파일(app.js) 10,000줄 레거시 유지보수 한계** — 3단계 점진적 전환: app.js 단일 파일 → 레이어드 아키텍처(Controller/Service/Repository) → NestJS 프레임워크 도입, 모듈 아키텍처 이관 완료
- **JavaScript 기반 코드의 타입 안정성 부족** — JavaScript → TypeScript 마이그레이션, 타입 안정성 강화
- **테스트 부재로 인한 회귀 리스크** — TDD 도입 및 단위 테스트 작성, 테스트 커버리지 80% 달성

기술 선택 이유

Express 단일 app.js 10,000줄 레거시의 유지보수 한계 해결을 위해 NestJS DI·모듈 시스템 도입. 3단계 점진적 전환(단일 파일 → 레이어드 → NestJS)으로 리스크 최소화. TDD 도입으로 전환 중 회귀 방지.

기여도: *단독 주도. 3단계 전환 전 구간, JS→TS 마이그레이션, TDD 도입, 팀 내 코드 컨벤션 확립 모두 직접 수행.*

NestJS TypeScript TypeORM Jest MySQL

심플한

개발팀 Backend Engineer

2021.07 — 2022.03

고객 맞춤형 소프트웨어 솔루션 기업

주요 성과

- 학습 관리(출석, 수료, 진도율) 수작업 의존. LMS 풀스택 개발(QR 출석, PDF 수료증, Excel 일괄 등록). 학기당 300명+ 수강생 관리 전 과정 디지털화.
- 레거시 PHP 시스템의 확장성 한계. iframe 기반 Spring Boot 앱 임베드 + 가중 평균 매칭 알고리즘 구현. JOB Agent 서비스 출시.

프로젝트

학습 관리 시스템(LMS) 개발

2021.07 ~ 2022.03

교육 기관의 학습 관리(출석, 수료, 진도율)가 수작업에 의존하여 운영 효율이 낮고 오류가 빈번한 문제 해결

- **출석 관리 수작업 및 중복 출석 발생** — 날짜+사용자ID 복합 유니크 키로 중복 방지하는 QR 기반 실시간 출석 처리 API 구현, 출석 관리 자동화 및 중복 출석 원천 차단
- **수강생 개별 등록의 비효율** — Apache POI 기반 행별 순차 처리로 데이터 정합성을 보장하는 Excel 일괄 등록 시스템 개발, 학기당 300명+ 수강생 일괄 등록 지원
- **수료증 수동 발급 부담** — JasperReports 기반 동적 PDF 수료증 자동 생성 API 구현, 수료증 발급 자동화

> **기술 선택 이유**
> 교육기관 납품 프로젝트로 Java/Spring Boot 표준 선택. QR 출석 중복 방지를 DB 레벨 복합 유니크 키(날짜+사용자ID)로 원천 차단. JasperReports 템플릿 기반 PDF 수료증 대량 발급.

기여도: *10개월간 핵심 기능 직접 구현. QR 출석, Excel 일괄 등록, PDF 수료증, 진도율 계산 등. 6인 팀 풀스택 체계에서 백엔드 전담.*

Java | Spring Boot | MyBatis | MySQL | Apache POI | JasperReports

인천 스마트 그린 산단 통합관제센터

2022.01 ~ 2022.03

산단 관제센터의 회의실 예약·모니터링·SMS 알림이 개별 시스템으로 분산되어 통합 관리가 불가능한 문제 해결

- **회의실 예약 충돌 발생** — 시간+장소 복합 유니크 키로 예약 충돌 방지하는 실시간 예약 API 구현, 예약 충돌 원천 차단
- **예약 확정 알림 동기 처리로 인한 응답 지연** — 비즈뿌리오 SMS API + @Async 비동기 처리 적용, 예약 확정 SMS 알림 비동기 발송 체계 구축

> **기술 선택 이유**
> 회의실 예약 충돌을 DB 복합 유니크 키(시간+장소)로 원천 차단. SMS 발송 시 API 응답 지연 해결을 위해 @Async 비동기 처리 — 트래픽 규모 대비 별도 메시지 큐는 오버엔지니어링으로 판단.

기여도: *4개월간 백엔드 전담. 예약 API, 대시보드 모니터링 API, SMS 비동기 연동 직접 구현.*

Java | Spring Boot | MySQL | 비즈뿌리오 SMS API

미래 서비스 JOB Agent 개발

2021.07 ~ 2022.03

기존 PHP 시스템의 확장성 한계로 새로운 기능 추가가 어렵고, 점수 계산과 상담 관리가 체계화되지 않은 문제 해결

- **레거시 PHP 시스템의 확장성 한계** — iframe 기반 Spring Boot 독립 앱을 기존 PHP 시스템에 임베드하여 레거시 영향 최소화, 레거시 시스템 영향 없이 신규 기능 확장
- **다차원 평가 점수 산정 체계 부재** — 가중 평균 기반 다차원 평가 점수 산정 알고리즘 구현, 복합 점수 계산 자동화

> **기술 선택 이유**
> 레거시 PHP 시스템 비침습 확장을 위해 독립 Spring Boot 앱을 iframe으로 임베드. postMessage API로 크로스 프레임 통신. 다차원 평가를 단일 점수로 산출하기 위한 가중 평균 알고리즘 구현.

기여도: *10개월간 백엔드 전담. iframe 통합 구조 설계, 가중 평균 알고리즘, postMessage 상태 동기화, 상담사 백오피스 직접 구현.*

Java | Spring Boot | MySQL

kfriends 회원 관리 시스템

2021.09 (3주)

5개국 분산 웹사이트에서 국가별 개별 로그인이 필요한 통합 인증 부재 문제 해결

- **5개국 분산 웹사이트의 국가별 개별 로그인 필요** — Spring Security + JWT 기반 중앙화된 통합 인증 서버 개발, 5개국 통합 인증 체계 구축

> **기술 선택 이유**
> 5개국 분산 웹사이트의 개별 로그인 문제를 JWT 기반 중앙 통합 인증 서버로 해결. CORS 다중 도메인 처리로 각국 도메인에서 동일 서버 호출 허용.

기여도: *3주간 단독 구현. 통합 인증 서버 전체.*

Java | Spring Boot | Spring Security | JWT | MySQL

사내 기술 블로그 리뉴얼

2021.09 (3주)

WordPress 기반 블로그의 호스팅 비용 부담과 수동 배포 문제 해결

- **WordPress 호스팅 비용 부담 및 수동 배포** — WordPress → Jekyll 정적 사이트 마이그레이션, 호스팅 비용 100% 절감

Jekyll Liquid Travis CI GitHub Pages