서동원

풀스택 웹 개발자 | me@won2dev.xyz

Blog Github Linkedin

직접 부딪치고 끝까지 구현하며 문제를 해결하는 개발자가 되고자 노력했습니다.
다양한 현장에서 업무를 경험하며 '작업을 책임지고 마무리하는 자세'를 체득했고, 반복적인 문제 해결과 개선 과정에서 논리적 사고와 협업의 중요성을 배웠습니다.
부트캠프에서 백엔드 개발을 처음 접했을 때 모든 것이 낯설었지만, 하루 12시간 이상 학습하며 기능 하나가 정상 동작할 때마다 몰입과 성취를 경험했습니다.

-

아이디어가 생기면 설계부터 직접 주도합니다. 기획·개발·결제·운영·CS까지 혼자 끌고 가며 실서비스를 만들어 운영 중이며, 제품 전체를 책임지는 감각을 실전에서 익혔습니다. 기능 구현에 머물지 않고 부하 테스트와 장애 상황을 직접 재현하며 운영 환경까지 검증하고, AI 도구를 활용해 더 빠르게 더 멀리 가는 방법을 만들어가고 있습니다.

프로젝트

바른이봇 — 디스코드 커뮤니티 자동화 봇

2025.12~ 진행 중 (7개월)

서비스 페이지

모달 기반 인앱 설정으로 웹 대시보드 없이 디스코드 내에서 모든 기능을 제어하는 자동화 봇

역할 및 책임

- 기획·설계·개발·운영·CS 전담 1인 사업
- Discord Bot 전체 기능 구현 및 AWS 인프라 운영
- 홈페이지(Next.js), NicepayPG 결제 연동, 라이선스 자동 발급·검증 구조 설계 및 구현

주요 기능

- 전 기능 Discord Modal 기반 인앱 제어 — 웹 대시보드 없이 설정 완결
- 입장/퇴장/부스트/로그/티켓/TTS/자동 번역 서버 운영 자동화
- AWS Polly TTS, AWS Translate 실시간 자동 번역
- i18n 다국어 지원 — 서버별 언어 설정(한국어/영어) 분리
- 고루틴 기반 병렬 처리 + sync.Mutex로 TTS 재생 상태 동시성 제어
- Soft/Hard 2단계 Rate Limit — 1분 3회 초과 시 쿨다운, 10분 8회 초과 시 30분 차단 및 보안 로그 저장
- 플랜별 라이선스 검증 — MongoDB 기반 활성화/만료 관리, 플랜별 일일 사용량 제한

사용 기술

- Backend: Go 1.26.3, AWS Polly, AWS Translate, Firebase, MongoDB
- Frontend: Next.js
- Infra: AWS, Docker, Docker Compose, GitHub Actions

팀 구성

- 1인 / 전체 기획·개발·운영·CS

고루틴 취소 누락으로 인한 패널 UI 충돌

- 문제상황: 3초 뒤 원래 메시지로 복귀하는 고루틴 실행 중 다른 버튼 조작 시 이전 패널로 강제 복귀되는 현상
- 원인분석: 고루틴 취소 로직이 일부 핸들러에만 적용되어 신규 인터랙션 발생 시 이전 고루틴이 살아있는 채로 메시지를 덮어씀
- 해결방안: context.CancelFunc를 채널/메시지 ID 복합 키로 관리하는 전역 맵 구현, 모든 버튼·패널 진입 시 기존 고루틴 선제 취소
- 결과: 전 기능 패널 전환 시 UI 충돌 완전 제거, 메모리 누수 없는 안정적 고루틴 관리

외부 API 장애 대응 및 알림 폴백 구조 구현

- 문제상황: Discord API 지연 또는 장애 발생 시 장애 알림이 전달되지 않아 운영자가 문제를 인지하지 못하는 위험 존재
- 원인분석: 운영 알림이 Discord 단일 채널에 의존하여 Discord 자체 장애 시 모니터링 체계가 무력화됨
- 해결방안: 주기적 Health Check를 통해 Discord API 응답 시간과 상태를 모니터링하고, 장애 감지 시 자동 상태 보고서를 생성하도록 구현. Discord 알림 전송 실패 시 Slack Webhook으로 자동 전환되는 알림 폴백 구조 적용
- 결과: Discord 장애 상황에서도 운영 알림 수신 가능, 단일 알림 채널 장애로 인한 모니터링 공백 제거

악성 반복 요청으로 인한 AWS API 비용 폭증 위험

- 문제상황: 반복적인 명령어 호출로 AWS Polly/Translate API 비용 폭증 가능성 및 서비스 불안정 위험
- 원인분석: 요청 횟수 제한 로직 부재, 외부 API 호출 구조 직접 노출
- 해결방안: 인메모리 기반 Soft/Hard 2단계 Rate Limit 구현, 보안 이벤트 JSON 로그 일별 파일로 저장
- 결과: API 비용 보호 및 악성 요청 차단, 100서버 운영 중 AWS 인스턴스 CPU 10% 미만 유지

데이터 저장소 장애 대응 및 폴백 구조 구현

- 문제상황: MongoDB 또는 Discord API 장애 발생 시 라이선스 검증, 서버 설정 저장 등 핵심 기능이 중단될 위험 존재
- 원인분석: 단일 저장소 의존 구조에서는 DB 장애 시 서비스 전체 기능이 영향을 받음
- 해결방안: MongoDB를 기본 저장소로 사용하고, 장애 감지 시 Firebase로 자동 전환하는 폴백 구조 구현. Firebase까지 사용할 수 없는 상황에서는 JSON 파일에 임시 저장 후 복구 시 재반영하도록 설계
- 결과: 단일 저장소 장애로 인한 서비스 중단 제거, 저장소 장애 상황에서도 핵심 데이터 유실 없이 서비스 지속 가능

스파르타 로지스틱스 - MSA 기반 B2B 물류 관리 플랫폼

2026.05~ 2026.06 (2개월)

Github

전국 허브 네트워크를 기반으로 기업 간 상품 주문·배송 프로세스를 자동화하는 MSA 플랫폼

역할 및 책임

- User Service 개발: Keycloak 기반 인증/인가, 승인 기반 회원가입, 역할별 권한 관리 구현
- CI/CD 파이프라인 구축: GitHub Actions 기반 빌드·배포 자동화 및 Discord 알림 연동
- 인프라 보안 강화: Gateway 시크릿 헤더 검증으로 서비스 직접 접근 차단

주요기능

- 승인 기반 회원가입 및 역할별 권한 관리 (5단계 RBAC)
- Keycloak 연동 인증/인가 전 흐름 (로그인·토큰 재발급·로그아웃)
- 배송담당자 순번 기반 자동 배정 및 근무 상태 관리
- X-Gateway-Secret 헤더 기반 서비스 간 내부 신뢰 검증

사용 기술

- Backend: Java, Spring Boot, Spring Security, Spring Cloud Gateway, Netflix Eureka, OpenFeign, Keycloak, JWT, PostgreSQL, Redis
- Infra: Docker, Docker Compose, Vultr, Zipkin, GitHub Actions

팀 구성

- 개발자 5인 / User Service(인증·인가·배송담당자 관리) 및 CI/CD 전담

Gateway 우회 직접 접근 보안 취약점

- 문제상황: 서비스 포트 직접 접근 시 헤더 위조로 권한 검사 우회 가능
- 원인분석: 필터가 헤더 출처 검증 없이 신뢰하는 구조
- 해결방안: Gateway에서 X-Gateway-Secret 주입 + 각 서비스 검증 + 서비스 포트 외부 노출 제거
- 결과: Gateway 미경유 직접 접근 차단

FeignClient 서비스 간 헤더 유실 (403/503)

- 문제상황: 서비스 간 내부 호출 시 보안 헤더 미전파로 403 에러 발생
- 원인분석: FeignClient 기본 동작은 현재 요청 헤더를 전파하지 않음
- 해결방안: FeignInterceptor 구현으로 모든 Feign 요청에 보안 헤더 자동 주입
- 결과: 서비스 간 내부 통신 정상화

부하 테스트 및 SAGA 패턴 검증

-

문제상황: 분산 트랜잭션 환경에서 보상 로직 실제 작동 여부 검증 필요

- 원인분석: MSA 특성상 서비스 간 장애 시 데이터 정합성 보장 여부를 코드만으로 확인 불가
- 해결방안: Manus AI 활용 — k6 스크립트 자동 작성 → DB 시딩 → API Gateway 부하 발생, 의도적 장애 유도(503/409)로 SAGA 보상 트랜잭션 작동 검증
- 결과: VU 50명 기준 평균 응답 1.45s, 최대 처리량 15.2 req/s, 로그인·조회 성공률 100 %, SAGA 보상 트랜잭션 정상 작동 확인

Keycloak 승인 후 로그인 불가 문제

- 문제상황: 승인 완료 후에도 Keycloak에서 로그인 불가
- 원인분석: 회원가입 시 비활성 상태로 생성 후 승인 시 활성화하는 구조에서 반영 누락
- 해결방안: 승인 처리 시 유저 활성화 및 역할별 attribute 업데이트를 단일 흐름으로 통합
- 결과: 승인 후 즉시 로그인 및 권한 정상 반영

6CanDoEat — 음식 주문·결제 플랫폼

2026.04~ 2026.04 (1개월)

Github

광화문 주변 음식점 주문·결제·리뷰를 처리하는 백엔드 서비스

역할 및 책임

- 인증/인가 전담: JWT 기반 인증, Redis role 캐싱, @PreAuthorize + SecurityContext 이중 인가 체계 구축

주요 기능

- JWT + Redis 기반 이중 인가 — @PreAuthorize 역할 검증 + SecurityContext 소유자 검증 2단계 권한 제어
- Redis role 캐싱으로 요청마다 DB 조회 없이 권한 재검증, role 변경 시 캐시 즉시 갱신
- Redis 장애 시 DB Fallback 처리로 가용성 확보

사용 기술

- Backend: Java, Spring Boot, Spring Security, JWT, PostgreSQL, Redis

팀 구성

- 개발자 4인 / 인증·인가 전담

JWT Stateless 포기 및 Redis role 재검증 도입

- 문제상황: 토큰 발급 후 권한 강등 시 토큰 만료 전까지 기존 권한 유지되는 보안 공백 발생
- 원인분석: JWT Stateless 구조상 서버가 토큰 내 권한 정보를 별도 검증하지 않으면 변경 반영 불가
- 해결방안: Stateless 장점 일부 포기, Redis에 role 캐싱 후 매 요청마다 토큰 role과 캐시 role 비교 / 캐시 미스 시 DB Fallback 후 재캐싱 (30분 TTL)
- 결과: 동시 접속 300명 기준 처리량 53% 향상, 평균 응답속도 40% 감소, 응답 실패율 1.49% → 0.06%

@PreAuthorize + 소유자 검증 이중 인가 구현

- 문제상황: 역할 검증만으로는 타 사장님이 다른 가게 리소스에 접근 가능한 보안 취약점 존재
- 원인분석: @PreAuthorize는 역할만 확인하고 리소스 소유자 여부는 검증하지 않음
- 해결방안: Controller에서 @PreAuthorize로 역할 1차 검증 + Service에서 SecurityContext의 userId와 리소스 소유자 ID 대조 2차 검증
- 결과: 역할 우회를 통한 타 사용자 리소스 접근 완전 차단

경비 자동 처리·증빙 자동화 시스템

2025.09~ 2025.09 (1개월)

Slack 기반 경비 자동 처리·증빙 자동화 시스템

주요 기능

- Slack /expense 항목 금액 명령어 기반으로 비용 요청을 자동 수집
- Google Sheet의 정책 기준(항목 제한, 직원별 한도, 월 누적 금액)을 비교해 자동 승인/반려
- 승인된 요청에 대해 고유 saveId·threadId를 생성해 기록 저장
- Slack 스레드에 업로드된 영수증 파일을 자동 감지하여 Google Drive로 업로드
- 영수증 링크를 saveId/threadId 기반으로 기존 요청 데이터와 정확하게 매칭
- 승인 안내 → 증빙 업로드 → 기록 완료까지 전 과정 완전 자동화

사용 기술

- n8n, Slack API, Google Sheets, Google Drive API, JavaScript

팀 구성

- 개발자 1인

파일 증빙 자동 처리 로직 개선

- 문제상황: Slash Command(/expense)는 단일 요청·응답 구조로 동작해 영수증 파일을 동일 흐름에서 처리불가
- 원인분석: /커맨드 실행 후 이벤트가 종료되어, 파일 업로드 이벤트를 같은 워크플로우에서 이어받는 것이 구조적으로 불가
- 해결방안: 승인·저장 워크플로우와 파일 감지 워크플로우를 분리하고, saveId·threadId를 발급해 저장 후 스레드 파일을 Drive 업로드와 매칭
- 결과: 커맨드 입력과 스레드 파일 업로드가 안정적으로 연결되며, 영수증 첨부 → 업로드 → 기록 반영까지 누락 없는 완전 자동화 구현

SlackLens - Slack 기반 장애 자동 처리

Slack 기반 AI 장애 자동 분석·담당자 배정 자동화 시스템

2025.11~ 2025.11 (1개월)

주요 기능

- Slack 메시지를 기반으로 장애 여부·카테고리·우선순위를 AI가 자동 분석
- Google Sheet 팀 정보를 활용해 담당자를 자동 매핑 및 배정
- Notion DB에 장애 이력과 분석 결과를 자동 저장하여 기록 관리
- Slack으로 실시간 알림을 제공해 장애 접수 → 분석 → 배정 과정을 자동화

사용 기술

- n8n, Slack API, Google Sheets, Notion API, JavaScript

팀 구성

- 개발자 1인

장애 메시지 사용자 확인 로직 개선

- 문제상황: Slack 이벤트가 user ID만 제공해 실제 작성자를 즉시 파악하기 어려움
- 원인분석: Slack User Info API가 별도 비동기 흐름으로 동작해 사용자명이 노드 체인에서 소실
- 해결방안: AI 분석 결과와 Slack User Info API 응답을 Merge로 통합해 reporter 정보를 일관되게 유지
- 결과: 장애 기록의 식별성이 개선되고 담당자·운영팀이 메시지를 신속하게 파악 가능

DORANGG - 리그오브레전드 AI 코칭 도우미 웹앱

2025.07~ 2025.08 (2개월)

Github Github

리그오브레전드 전적 기반 AI 맞춤 코칭 및 전략 제공 플랫폼

역할 및 책임

- 서비스 기획 및 제품 설계: 사용자 흐름 정의, 요구사항 정리 및 기능 우선순위 결정
- UI/UX 및 프론트엔드 개발: React 기반 화면 설계 및 WebSocket/REST API 연동,Next.js + Vercel 배포 및 HTTPS 환경 구성
- 백엔드 개발: Spring Boot 기반 REST API 및 DB 설계, Riot API 연동,JWT 인증, Redis 기반 세션 처리 및 실시간 WebSocket 채팅 구현

주요 기능

- 게임 데이터를 기반으로 한눈에 볼 수 있는 요약 위젯과 Daily Briefing 기능 제공
- 실시간 개선 전략을 복잡한 리포트 대신 구체적이고 즉시 적용 가능한 코칭 형태로 추천
- 개인화된 목표 설정과 커뮤니티 기능(오픈채팅 및 게시판) 지원
- 리그오브레전드에 특화된 AI 챗봇으로, 각 유저 전적 기반 맞춤형 전략 및 추천 제공

사용 기술

- Backend: Java, Spring Boot, JPA, Spring Security, JWT, Redis, Spring AI, MySQL, WebSocket/STOMP, Riot API
- Frontend: Next.js, TailwindCSS

팀 구성

-

개발자 3인 / 서비스 기획·설계 및 Spring API·Spring AI, Riot API 연동, 게임 특화 AI 챗봇·실시간 채팅·멤버 관리 기능 구현

설계

- 사용자의 모든 전적을 한눈에 요약해 확인할 수 있는 위젯형 서비스로, 각 위젯에 AI가 연결되어 전적을 분석하고 맞춤형 코칭을 제공함.
- 최근 전적, 승률, 연승/연패, 자주 사용하는 챔피언과 숙련도, 개인 티어 정보를 직관적으로 확인할 수 있으며, 우리 서비스에서만 제공되는 독자적 기능으로, 기존 복잡한 전적 사이트와 달리 즉시 적용 가능한 전략과 코칭을 제공

전적 데이터 기반 AI 코칭 로직 개선

• 문제상황: 일부 유저 전적 데이터에서 AI가 부적절한 전략 추천하는 사례 발견
• 원인분석: 전적 필터링 및 전처리 로직 미흡
• 해결방안: Spring AI(OpenAI API) 기반 분석 로직 개선 및 데이터 전처리 강화
• 결과: AI 코칭 정확도 향상 및 사용자 만족도 향상

AI 코칭 및 대시보드 불일치 문제

• 문제상황: 특수 상황(연패, 최근 경기 편차 등)에서 Daily Briefing, 대시보드 위젯, 추천 전략/빌드 간 AI 분석 결과와 UI 불일치
• 원인분석: AI 분석 로직과 위젯/추천 기능 동기화 부족
• 해결방안: AI 응답과 위젯/추천 전략·빌드 UI 연동 로직 개선
• 결과: 부적절 전략 추천 케이스 20% 감소, 사용자 만족도 4.1/5 → 4.6/5 상승, AI 코칭 로직 모듈화로 재사용성 및 유지보수성 향상

WebSocket 오픈채팅 성능 문제

• 문제상황: 동시 접속 증가 시 전체 채팅 조회로 메시지 지연 및 DB 과부하 발생
• 원인분석: 메시지 초기 로딩 시 전체 데이터 조회로 DB 비용 증가 및 지연 발생
• 해결방안: 최근 메시지 200개 우선 로딩 + 추가 요청시 200개씩 점진적 과거 조회 구현
• 결과: 메시지 지연 40% 감소, DB 쿼리 횟수 60% 절감, 서버 안정성 확보 및 비용 절감

JWT 인증/로그아웃 관리 문제

• 문제상황: 토큰 만료, 로그아웃, 탈퇴 시 레디스 블랙리스트 관리 불안정
• 원인분석: 토큰 만료와 블랙리스트 관리 로직 미흡
• 해결방안: 레디스 기반 블랙리스트 적용 및 만료/로그아웃/탈퇴 처리 로직 개선
• 결과: 로그아웃/탈퇴 처리 시 인증 오류율 0% 달성, 인증 안정성 및 보안 강화

대시보드 조회 최적화 문제

• 문제상황: 전적 조회 및 AI 추천 시 불필요한 Riot API 호출로 서버 부하 발생
• 원인분석: 프론트에서 모든 요청 동기 처리 및 무분별 호출 구조
• 해결방안: Daily Briefing 하루 단위 캐싱, 위젯/추천 전략은 매치 데이터 변동 시만 재호출 로직 구현
• 결과: Riot API 불필요 호출 제거, DB 캐싱 기반 재호출로 응답속도(p95) 1.2s → 0.6s 단축, API 호출 80% 감소, 코드 모듈화로 재사용성 향상 및 사용자 경험 안정화

UI 구조 및 재사용성 개선

- 문제상황: 기능 확장에 따라 페이지별 UI 구성 방식이 달라 코드 중복 증가
- 분석: 모달·버튼·데이터 카드 등 반복 UI 요소가 개별 구현되어 유지보수 어려움
- 해결방안: 공통 UI 컴포넌트화 및 Props 기반 재사용 구조로 리팩터링
- 결과: UI 코드량 약 30% 감소, 기능 추가 속도 향상 및 유지보수 시간 단축

대외활동 및 수상

K-Digital Training (내일배움캠프) – 실무형 Kotlin & Spring 개발자 양성과정 (7회차)

2025.04~ 2025.08 (5개월)

Spring 기반 백엔드 개발 집중 학습

- REST API, JPA, DB 설계, MVC 패턴 기반 실무 프로젝트 수행
- 팀 단위 협업 및 서비스 배포 경험

[AI/AX] 업무 자동화 교육 (팀스파르타)

2025.09~ 2025.09 (1개월)

- n8n, Slack, Notion, Google Sheet 등 API 연동을 통한 자동화 워크플로우 구현
- AI 분석 및 업무 자동화 프로세스 기획·설계 경험

K-Digital Training - 심화_AI를 활용한 백엔드 아키텍처 심화 과정 7회차 (팀스파르타)

2026.04~ 2026.07 (4개월)

- MSA 아키텍처 기반 분산 시스템 설계 및 운영 경험
- Redis 캐싱, Kafka 메시징을 활용한 성능 최적화 및 대용량 트래픽 처리 실습
- 모니터링·장애 대응 체계 구축과 AI 연계 백엔드 프로젝트 수행

학력 사항

가톨릭상지대학교

철도운전시스템

2021.03~ 2024.02(중퇴)

외국어 및 자격증

PC정비사 2급 2018.12