김지훈

Frontend Engineer

2001년생

wlgns6921@gmail.com

010-5664-5041

https://github.com/wlgns5041

https://velog.io/@wlgns5041/posts

배움과 **개선**을 멈추지않는 **책임감** 있는 개발자 김지훈 입니다.

Summary

기획 단계부터 **개발, 배포, 운영**까지 서비스를 완성해본 경험을 갖춘 프론트엔드 개발자입니다.

실제 사용자 환경을 고려하여 **UI를 개선**하고 **문제를 해결하는 과정**에 가치를 두며,

협업과 **지속적인 학습**을 통해 더 나은 제품을 만들고 배우는 것을 목표로 성장하고 있습니다.

Skills

Languages	TypeScript, JavaScript
Frameworks	React
Styling & CSS	Tailwind CSS, CSS(Responsive Design)
Deployment	Nginx, Vercel, Docker, Jenkins, AWS
Tools	Git, Figma, Notion, Slack

2025.02 Education

성결대학교 정보통신공학과 학사 졸업 — 2019.03 ~ 2025.02

학점 3.71 / 4.5

Awards

프로그래밍 설계 경진대회 우수 — 2020.06

웹서버 프로그래밍 교내 경진대회 최우수 — 2023.06

2025.03 Project 1

펫토리(PETORY) - 반려동물 일정 및 공유 돌봄 웹

친구/가족/돌보미와 함께 반려동물의 일정을 공유 및 관리할 수 있는 웹서비스

개발 환경 | 2025.03 - 2025.12 / 2명 (BE 1, FE 1)

포지션 | Frontend

기술 스택 | React, JavaScript, CSS(Responsive Design)

개발 내용

1 React 구조 설계

- 페이지 / 레이아웃 / 공통 컴포넌트 분리로 모듈화된 설계 구조
- Axios Interceptor 기반 전역 인증 및 에러 처리 로직 구현
- JWT 기반 인증 흐름 설계 및 보호 라우팅 적용

2 상태관리 및 데이터 흐름 설계

- 서비스 레이어 분리를 통한 관심사 분리(SoC) 및 데이터 관리 구조화
- React Router 기반 역할 및 권한 제어
- 페이지, 서비스, 상태, UI 업데이트 기반 일관된 데이터 흐름 확립

3 React Query를 적용한 데이터 및 성능 최적화

- 캐싱된 데이터 사용을 이용한 상태관리 적용 후
  페이지 로딩시간 약 40% 감소, API 호출 빈도 70~80% 감소
- 알림 데이터 자동 갱신(SEE 연동), 재조회 최소화
- 화면 전환 시 즉각적인 데이터 표시로 UX 및 성능 개선

4 UI / UX 설계 및 실서비스 운영 관점 개선

- 모바일 반응형 레이아웃 구성 및 사용자 피드백 기반 인터랙션 개발
- Modal 중심의 입력 흐름 관리(멀티 스텝 폼)로 인한 사용성 향상
- useTheme를 이용한 전역 관리(다크모드)로 환경 대응력 강화

Github | https://github.com/SJ-Petory

배포 URL | https://www.petory.site

OverView |

2025.12 Project 2

개인 포트폴리오 웹사이트

나 자신과 프로젝트 경험을 소개하고 개발 역량을 보여주는 웹 기반 포트폴리오

개발 환경 | 2025.12 - 진행 중 / Solo

포지션 | Frontend

기술 스택 | React, TypeScript, Tailwind CSS

개발 내용

1 TypeScript 기반 컴포넌트 구조 및 타입 안전성 확보

- TypeScript로 컴포넌트 props 및 상태를 명확히 정의해 타입 안정성 강화
- 컴포넌트 분리 및 상태 최소화를 통해 UI 렌더링 안정성 및 유지보수성 확보
- 타입 오류를 CI 단계에서 조기 감지해 런타임 버그 가능성 감소

2 React 기반 렌더링 성능 최적화 (CSR)

- CSR 기반 구조를 통해 사용자 입력 변화에 즉각적으로 반응하도록 흐름 설계
- useTransition, useDeferredValue를 활용해 화면 갱신과 입력 처리 분리
- Virtualized Rendering 적용으로 대규모 UI 리스트에서 DOM 부하 최소화

3 코드 스플리팅 / Lazy Loading 기반 초기 로딩 최적화

- React.lazy, Suspense 기반 섹션/라우트 단위 코드 스플리팅
- 이미지 등 후순위 자원에 사용 시점(On-Demand) 지연 로딩 적용
- 초기 번들 크기 감소로 첫 화면 로딩 속도 개선

4 Tailwind 기반 UI 시스템 및 인터랙션 성능 개선

- Tailwind 유틸리티 기반 스타일 시스템으로 UI 일관성과 생산성 향상
- GPU 가속 기반 애니메이션으로 부드러운 인터랙션 구현
- Framer Motion으로 주요 인터랙션의 모션을 최적화

Github |

배포 URL |

OverView |