



# 문현돈

## 프론트엔드 개발자

포트폴리오

---

# 목차

- 1) About Me
- 2) 주요 업무 내역
- 3) 개인 프로젝트



# About Me

## 검증된 역량

Angular, 기존 코드 베이스 단기간 학습 후 프로젝트에 바로 투입되어 실무 기여

프로젝트 요구 사항에 맞는 기술과 설계 방식을 빠르게 학습하고, 이를 효율적으로 구현해 성과 창출

## 다양한 실무 경험

신규 서비스 성공적으로 출시해 Swit 서비스 기능 확장

레거시 코드 리팩토링을 통해 시스템 안정성 및 성능 향상

주요 버그를 신속하게 해결해 사용자 경험 및 서비스 안정성 개선

## 코딩과 공부가 즐거운 개발자

회사에서 사용하지 않는 React, Astro 등 새로운 프레임워크 학습 및 프로젝트에 적용

2021년부터 학습한 내용을 글로 정리하고, 흥미로운 주제는 블로그에 기록하여 다른 사람들과 공유



# About Me

## 경력

### Swit Technologies

Frontend Web Developer

April, 2022 – June, 2024

## 기술 스택

### 프로그래밍 언어

JavaScript, TypeScript

### 프레임워크 및 라이브러리

React, Angular, Next.js, Astro, React Query, RxJS, TailwindCSS

### 기타

Git, GitHub, Jira, Figma

## 학력

### 연세대학교

신학, 영어영문학 (2021.08 졸업)

### 외국어

### 영어

TOEIC: 990점

TOEFL: 117점

### 스페인어

DELE: B2

---

# 주요 업무 내역

- 1) 레거시 Chat 컴포넌트 개선
- 2) 업무 자동화 플러그인
- 3) OKR 목표관리 플러그인



# 레거시 Chat 컴포넌트 개선

개발 기간: 2023.12 - 2024.05

인원: 프론트엔드 2명, 백엔드 1명, 기획 1명

## 목표

- Swit의 여러 Chat 컴포넌트 통합

## 주요 업무:

- 컴포넌트 간 기능 비교, 정리
- 추상 클래스 활용해 컴포넌트 내부 로직 외부로 분리
- 컴포넌트 간 상이한 UI를 주입 받는 방식으로 변경
- 소켓 재연결 이슈 해결



# 레거시 Chat 컴포넌트 개선

## 레거시 컴포넌트 분석

- DM, 채널, Task 댓글 컴포넌트 간 API 엔드포인트, 메시지 데이터 형식, 디자인 등 차이점 정리 및 문서화
- 정리된 내용을 토대로 기획자, 백엔드 개발자와 컴포넌트 간 차이점을 어떻게 극복할지 결정



## 레거시 Chat 컴포넌트 개선

컴포넌트 내부 로직 외부에서 주입

- 추상 클래스를 활용해 채팅 컴포넌트가 반드시 갖추어야 할 기능 정의
- DM, 채널, Task 별로 추상 클래스를 상속 받은 하위 클래스에 실제 메소드 구현





## 레거시 Chat 컴포넌트 개선

컴포넌트 UI 외부에서 주입

- 메시지, 모달, 메뉴 구성 등 Chat 컴포넌트 내부에 정의되어 있는 UI 중 차이가 있는 것 정리
- 해당 UI 중 하드 코딩 되어 있는 부분 제거
- Chat 컴포넌트를 사용하는 곳에서 각자 해당 UI를 주입하는 방식으로 대체



# 레거시 Chat 컴포넌트 개선

## 소켓 재연결 이슈

- 메인 페이지, 사이드 패널이 채널을 보고 있을 때 토큰 만료 시 사이드 패널 소켓 스트림 끊김
- 이후 사이드 패널 닫을 시, 메인 페이지 채널 소켓 끊김

## 작업 사항

- 사이드 패널 소켓 클린업 로직에서 불필요하게 스트림 끊는 코드 제거
- 사이드 패널 소켓 아이디 제거 방식 수정
  - 변경 전: indexOf, splice 메소드로 기존 배열 수정
  - 변경 후: filter 메소드로 새 배열 생성



# 업무 자동화 플러그인

서비스명: Swit Automation

개발 기간: 2023.01 - 2023.06

인원: 프론트엔드 1명, 기획 1명, 디자인 1명,  
외부 백엔드 개발 업체

## 목표

- Zapier와 비슷한 오토메이션 기능을 Swit에 유료 플러그인 형태로 구현

## 주요 업무

- API 초안 설계
- Server Driven UI 방식으로 오토메이션 생성 페이지 설계
- 재사용 가능한 컴포넌트 시스템 개발



# 업무 자동화 플러그인

## API 초안 설계

- 외부 업체에 전달할 API 초안 필요
- 초기 기획 및 디자인 토대로 클라이언트와 서버 간 데이터 교환 방식 결정
- Server Driven UI 방식을 지원하는 API 초안 작성
- 초안 작성 후 빠르게 테스트 서버 구축 및 UI 프로토타입 구현



# 업무 자동화 플러그인

## Server Driven UI 설계

- 다수의 서비스 및 이벤트를 지원할 수 있도록 Server Driven UI 방식 채택
- 1차 배포 이후 추가 개발 없이 빠르게 새로운 서비스 및 이벤트 지원 가능
- 추후에 사용자가 직접 써드파티 서비스 및 이벤트 추가할 수 있도록 대비



# 업무 자동화 플러그인

재사용 가능한 컴포넌트 시스템 개발

- 오토메이션 플로우를 생성하기 위해 필요한 공통 컴포넌트 파악
  - 예: 텍스트 입력 컴포넌트, 날짜 선택 컴포넌트, Swit 유저 선택 컴포넌트
- TypeScript로 컴포넌트 스키마 작성, 외부업체와 공유
  - 예: 컴포넌트 안내 문구, 필수값 여부, 선택 데이터 형태

# 업무 자동화 플러그인

트리거, 액션 선택 시 API 응답값을 토대로  
하단에 정보 입력 필드 렌더링

재사용 가능한 공통 컴포넌트를 통해 여러  
이벤트 지원

- 예: 워크스페이스, 채널 선택 필드는  
동일한 select 컴포넌트 활용


The image displays a configuration interface for a workflow automation plugin, divided into two main sections: '트리거' (Trigger) and '액션' (Action).

**트리거 (Trigger) Section:**

- 서비스 (Service):** A dropdown menu with 'Swit' selected.
- 트리거 (Trigger):** A dropdown menu with '메시지가 생성되면' (When message is created) selected.
- 계정 (Account):** A dropdown menu with 'hyundonmoon@gmail.com' selected. A note below reads: '별을 사용하려면 로그인이 필요합니다.' (Login is required to use this account).
- 워크스페이스 (Workspace):** A dropdown menu with '선택하세요' (Select) selected.
- 채널 (Channel):** A dropdown menu with '선택하세요' (Select) selected.

**액션 (Action) Section:**

- 서비스 (Service):** A dropdown menu with 'Gmail' selected.
- 액션 (Action):** A dropdown menu with '메일 보내기' (Send email) selected.
- 계정 (Account):** A dropdown menu with 'hyundonmoon@gmail.com' selected. A note below reads: '별을 사용하려면 로그인이 필요합니다.' (Login is required to use this account).
- 수신자 (Recipient):** A text input field with the placeholder '내용을 입력하세요' (Enter content). A note below reads: '\*ID를 입력하면 이전 단계의 문자열 필드를 참조할 수 있습니다.' (If you enter an ID, you can refer to the string field in the previous step).
- 참조 (Reference):** A text input field with the placeholder '내용을 입력하세요' (Enter content). A note below reads: '\*ID를 입력하면 이전 단계의 문자열 필드를 참조할 수 있습니다.' (If you enter an ID, you can refer to the string field in the previous step).
- 숨은 참조 (Hidden Reference):** A text input field with the placeholder '내용을 입력하세요' (Enter content). A note below reads: '\*ID를 입력하면 이전 단계의 문자열 필드를 참조할 수 있습니다.' (If you enter an ID, you can refer to the string field in the previous step).



# OKR 목표관리 플러그인

서비스명: Swit Goals

개발 기간: 2022.06 - 2022.11

인원: 프론트엔드 2명, 백엔드 2명, 디자인 1명

## 목표

- OKR 목표 관리 기능을 Swit에 유료 플러그인 형태로 개발

## 주요 업무

- OKR 상세 페이지 개발
- OKR 상세 페이지 내 댓글창 개발
- OKR 어드민 콘솔 개발

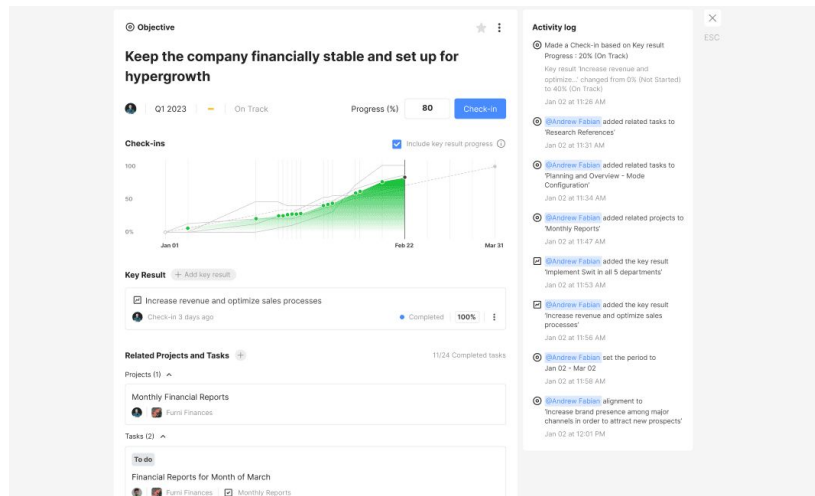


# OKR 목표 관리 플러그인

OKR 진행 상황, 변경 사항, 담당자 등 여러 정보를 담은 상세 페이지 개발

기본 정보, 변경 내역, 진행 상황 차트 등 기능 별로 컴포넌트 개발

- 동일한 기능이 다른 곳에서 필요할 경우 재사용 가능
- 기획 변경에 빠르게 대응 가능



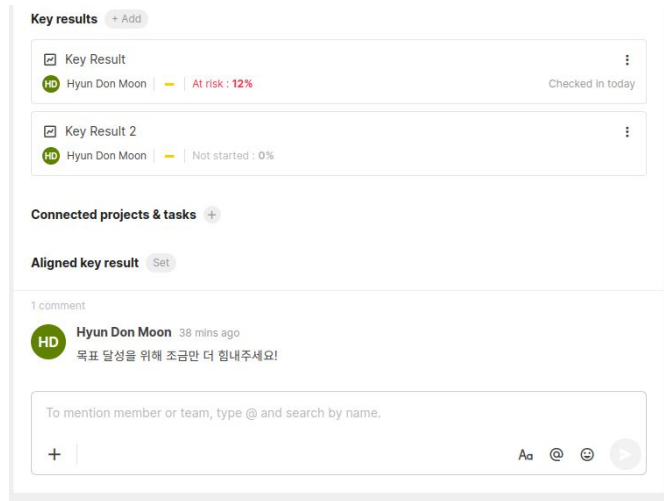
# OKR 목표 관리 플러그인

## OKR 상세 페이지 댓글 시스템

- 기존 메시지 댓글 컴포넌트 재활용
- 새로운 채팅 컴포넌트로 인한 코드 중복 및 에러 방지

## 실시간 댓글 업데이트용 소켓

- 기존 소켓 로직 참고해 OKR 댓글용 소켓 연결



---

# 개인 프로젝트

- 1) URL 단축기
- 2) 블로그



# URL 단축기

기술 스택: Next.js, TypeScript,  
TailwindCSS, Zod, React Query

배포 링크: [ziplink.at](https://ziplink.at)

## 목표

- 이력서와 블로그에 단축된 URL을 적용해, 긴 링크로 인한 불편함 해소
- React, Next.js, Zod, React Query 등 학습 내용 프로젝트에 적용



# URL 단축기

## URL 단축 기능

- nanoid 라이브러리를 활용해 6자리 고유 코드를 생성
  - 약 37,000개의 코드를 생성해야 ID가 충돌할 확률이 1%에 불과
  - 데이터베이스에서 ShortCode 필드를 유니크 속성으로 설정해 충돌을 추가적으로 방지
- 단축된 링크 방문 시, 코드를 데이터베이스에서 찾은 후, 그에 매칭되는 기존 URL로 redirect



# URL 단축기

학습 내용 프로젝트에 적용해보기

- React 서버 컴포넌트, 서버 액션 사용해보기
  - 클라이언트 컴포넌트에서 서버 액션으로 URL 단축
- React Query로 단축된 링크 목록 불러오기
  - 서버 컴포넌트에서 React Query로 목록을 prefetch
  - 서버 QueryClient를 클라이언트 QueryClient와 합쳐 데이터 사용
- Zod로 런타임 타입 체크



# 블로그

기술 스택: Astro, TypeScript,  
TailwindCSS

배포 링크: [hyundon.dev](https://hyundon.dev)

## 목표

- 웹 성능 개선 기법들을 실제 프로젝트에 적용
- 이해하기 어려웠던 개념이나 추후 다시 보고 싶은 내용을 기록할 공간 구현

# 블로그

웹 성능 관련 학습 내용 적용

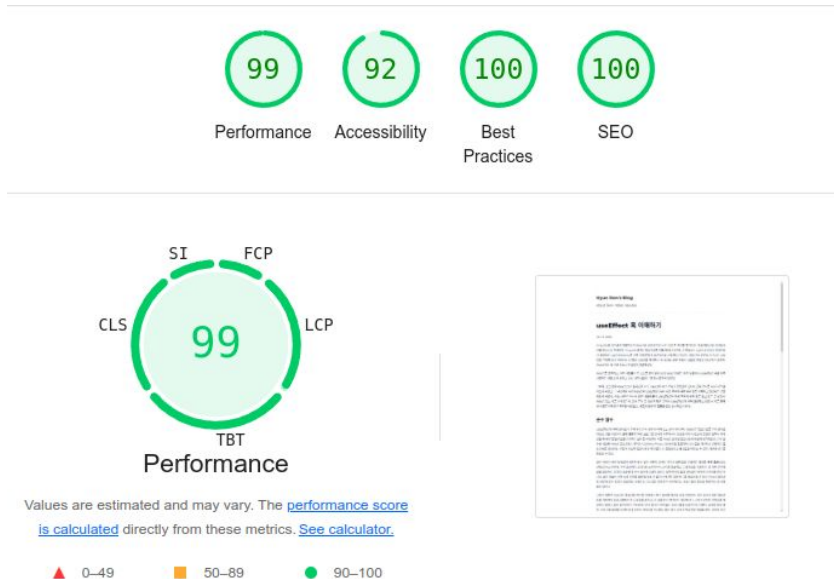
정적 사이트

- SSG에 최적화된 Astro 사용

리소스 로딩 최적화

- preload, prefetch, 이미지 변환 등

GitHub Actions로 Lighthouse 실행해 성능 측정







# 블로그

학습 내용 글로 작성해 더욱 깊이 이해

다른 이에게도 배운 내용을 공유하고자 노력

[useEffect 훅 이해하기](#)

Jan 3, 2025 · 한국어 · React 공식 문서를 읽으며 useEffect 훅의 본질에 대해 생각해 보았습니다.

---

[TLS 핸드셰이크와 암호화](#)

Dec 10, 2024 · 한국어 · TLS 핸드셰이크 절차와 데이터 암호화 방식 알아보기

---

[Digging into React Hook Internals: A few things I learned about useState and other hooks](#)

Nov 7, 2024 · English · Digging into React source code to understand why hook order matters and how useState initial values are managed.

---

[효율적인 리소스 로딩: preload와 prefetch 알아보기](#)

Sep 29, 2024 · 한국어 · preload와 prefetch를 활용하여 웹페이지 로딩 속도를 개선하는 방법을 알아봅니다.

---

[Static imports vs. Dynamic imports in JavaScript](#)

Aug 19, 2023 · English · Comparing static and dynamic imports in JavaScript

---

# 연락처

이메일	<a href="mailto:hyundonmoon@gmail.com">hyundonmoon@gmail.com</a>
GitHub	<a href="https://github.com/hyundonmoon">github.com/hyundonmoon</a>
LinkedIn	<a href="https://linkedin.com/in/hyundonmoon">linkedin.com/in/hyundonmoon</a>
Blog	<a href="https://hyundon.dev">hyundon.dev</a>

---

**감사합니다**

---