

# 황현규

이메일: [joseph95501@gmail.com](mailto:joseph95501@gmail.com)

휴대폰: 010-9049-9550

포트폴리오: <https://01joseph-hwang10.github.io>

Github: <https://github.com/01Joseph-Hwang10/>

주니어 개발자 황현규입니다. 창업과 스타트업에 관심이 많습니다.

창업 아이템을 내 손으로 직접 구현하기 위해 개발을 시작했으며, 나의 아이디어를 직접 구현하여 세상에 가치를 제공할 때 느끼는 보람이 매우 커, 학부 신입생 때부터 교내외의 스타트업, 다수의 해커톤 및 창업 활동에 참여하여 팀의 핵심 인원으로서 활약하였습니다.

"최소 자원으로 최대 효용"이라는 린 스타트업의 철학에 깊은 공감을 하며, 업무를 진행할 때도 어떻게 하면 최소한의 자원으로 최대의 효과를 이끌어낼 수 있을지 끊임없이 고민합니다. 또한 항상 "왜?" 라는 질문을 던지며, 무엇을 하는지에 대한 뚜렷한 목적을 가지고 업무를 진행합니다.



## Start-ups

스타트업과 창업에 관심이 많습니다. 심층 인터뷰 등의 시장조사를 통해 잠재적 Pain Point를 찾고 린 스타트업 방법론에 기반해 MVP를 개발해나가는 과정을 즐깁니다.



## Lean Programming

최소의 자원으로 최대의 효과를 추구합니다. 기능을 구현할 때는 그 목적을 항상 검토하고, 구현에 있어 불필요한 프로세스가 있는지를 유심히 살펴봅니다.



## Code Aesthetics

가독성이 높은 코드, 유지보수가 용이한 코드를 지향합니다. OOP를 비롯한 다양한 디자인 패턴을 적재적소에 활용해 처음 보는 코드라도 금방 이해하고 기여할 수 있게끔 노력합니다.

## 학력 사항

### 울산과학기술원 (2021.03 - 휴학 중)

- 컴퓨터공학과 학사과정 (주전공)
- 경영과학부 학사과정 (부전공)

## 경력 사항

트립빌더 (@tripbuilder)

- 2021.07 - 2021.12
- 개발팀 / Front-end Developer
- AI 기반 여행 계획 추천 모바일 앱 **TripBuilder** 개발 참여

## 랩이즈 (@labis)

- 2021.12 - 2022.11
- 개발팀 / Development Lead & Co-Founder
- 하드웨어 기반의 블록 코딩 교육 솔루션 **LearnQue** 기획 및 개발

## 쉐퍼드23 (@shepherd23)

- 2022.07 - 현재 재직중
- 개발팀 / Product Manager & Software Engineer
- E-Commerce 쇼핑몰을 위한 개인화 상품 추천 플러그인 **PickHound** 기획 및 개발

## 업무 경험



팁

아래 링크에서 더 많은 경력 사항을 확인하세요.

<https://01joseph-hwang10.github.io>

### => OOP에 기반한 **Admin Dashboard** 코드 리팩터링

**Frontend** **Improvement** @shepherd23

- OOP와 **React** 기본 혹은 활용에 불필요한 Redux 코드를 제거하고, 컴포넌트화를 통해 코드의 재사용성과 가독성을 높이는 등 클린코드 작성을 위해 노력함

### => **Cafe24 API Auth Code** 추출 자동화

**Automation** @shepherd23

- 실제 카페24 API를 활용해 개발을 진행하기 위해서는 카페24 API의 Authorization Code Flow 인증 과정을 거쳐야 함.
- 이에 작업을 위해 매번 브라우저를 열고 로그인 후 auth code를 추출하는 과정을 반복해야 하는 불편함이 존재했음.
- 이러한 비효율을 없애기 위해 Selenium을 활용해 로그인 및 auth code 추출 과정을 자동화함.
- => 개발 생산성 크게 향상. 더 많은 Iteration을 가능하게 함.

### => **Admin Dashboard**의 **Auth Flow** 테스트 효율화

**Frontend** **Improvement** @shepherd23

- **Cafe24 API Auth Code 추출 자동화**를 통해 개발 Iteration의 속도가 향상되었으나, auth code나 access token의 만료 등의 이유로 지속적으로 Auth Code 추출 스크립트를 실행해야 하는 불편함이 존재했음.
- 또한 Auth Flow가 복잡했기 때문에 어느 부분에서 에러가 발생했는지 파악하기 어려웠음.
- 이러한 문제를 해결하고자 **Admin Dashboard**의 Auth Flow를 테스트하기 위해 Cypress를 활용한 테스트 작성함.
- => Iteration 속도 향상. 개발 생산성 향상.

### => **StencilJS** 도입을 통한 번들 크기 및 성능 최적화

**Devops** **Optimization** @shepherd23

- Cafe24로 제작된 쇼핑몰은 이미 많은 Javascript를 사용하며, 여기에 타사의 플러그인 또한 여럿 설치해 사용했음
  - => 큰 번들 사이즈는 엔드 유저의 UX 경험에 악영향을 미칠 수 있음을 인지함
- 기존의 **React** 대신 **StencilJS**를 도입하여 Web Components API의 이점을 활용하며, 번들 사이즈를 60% 이상 줄임
  - => 기능이 동일한 타사 추천 플러그인에 비해 60% 작은 번들 사이즈. (본사 35KB, 타사 85KB)

## => 다수의 IIFE 번들 빌드 프로세스 구성 및 CI/CD 자동화

Devops Automation @shepherd23

- 카페24에 배포해 쇼핑몰에 삽입되어 실행되는 스크립트는 브라우저에서 바로 실행될 수 있는 VanillaJS로써, 페이지마다 하나의 IIFE 번들이 존재해야 하는 제약조건으로 기존 SPA의 번들링 프로세스를 그대로 사용할 수 없는 상황이었음.
- StencilJS CLI와 Rollup, 배시 스크립트를 활용해 여러 개의 Entrypoint를 동시에 IIFE로 빌드하는 프로세스 구성함.
  - Webpack과 비교해, Scope Hosting 기능, 단순한 번들러 설정 및 CLI 활용성 등을 고려하여 Rollup을 도입함.
- Github Actions를 활용해 Banner Manager 의 배포 자동화
- => 복잡한 빌드 및 배포 과정의 인적 자원 소비 최소화

## => 심층 인터뷰를 통한 수요 파악 및 제품 방향성 수립

Project Management Communication @shepherd23

- 약 1달에 걸쳐 다양한 분야에 종사하는 12명의 쇼핑몰 관리자를 대상으로 인터뷰 진행함.
- 인터뷰를 통한 피드백 수집으로 쇼핑몰의 규모에 따른 서비스의 수요, 운영 방식의 차이 등을 파악할 수 있었음.
- => 정리한 피드백을 바탕으로 불필요한 개발 단계는 제거하여 자원의 소비를 최소화

## => 모노리포 구성, 패키지 버저닝 및 프로젝트 간 공통 모듈 공유

Automation Improvement @shepherd23

- Lerna, Python Module을 활용해 Monorepo 구성 (Recommendation API, Bandit Engine, Banner Manager)
- Lerna, standard-version 을 활용한 패키지 버전 관리 (모든 프로젝트)
- 공유가 필요한 코드는 따로 패키지로 분리해 다른 리포지토리에서도 사용 가능하도록 구성 (Recommendation API, Banner Manager)
  - 공유 패키지 빌드 프로세스 구성 및 Github Actions, Google Artifact Registry를 활용해 자동 배포
  - => 비효율적인 복사 붙여넣기 방식의 코드 공유를 최소화.

## => 불필요한 코드 빌드 최소화

Optimization @shepherd23

- 대부분의 CI/CD 빌드 트리거가 Github Actions의 push 이벤트에 의해 발생함.
- 모노리포 상황에서 하나의 패키지에만 변경사항이 있는 경우에도 모든 패키지의 빌드가 발생하는 문제가 있었음.
- Lerna를 이용한 버저닝 관리와 함께 패키지 레지스트리에 배포되어 있는 패키지의 버전과 현재 레포지토리의 패키지 버전을 비교해 변경사항이 있는 경우에만 빌드가 발생하도록 구성함.
  - => 불필요한 빌드 최소화 및 CI/CD 자원 소비 최소화

## => Client Side 에러 트래킹을 위한 로깅 시스템 개발

Feature @shepherd23

- 서버에서 발생하는 에러와 달리 클라이언트 애플리케이션은 에러가 발생해도 팀 내에서는 인지하기 어려움.
- 고객의 오류 문의를 처리하기 위해서는 오류 당시의 자세한 상황을 알아야 하는데, 이를 고객의 설명에만 의존해 파악하기 어려움.
- 이에 Cloud Run에 POST request의 body를 console.log 를 통해 로깅하는 간단한 로그 서버와 이에 대한 클라이언트 라이브러리를 개발함.
  - 클라이언트는 uuid 를 활용해 고유의 에러 ID를 생성하고, 이를 로그 서버에 전송하며, 에러 ID를 에러 메시지와 함께 고객에게 전달.
  - 로그 서버는 Cloud Logging으로 로그를 스트림하고, 개발팀은 위에서 생성한 에러 ID를 통해 Cloud Logging에서 해당 에러를 검색.
- => 고객의 에러 문의에 발행된 에러 ID를 통해 즉각적인 문의 대응을 통한 고객 경험 향상.

## => Jest, Pytest를 활용한 TDD로 개발 흐름 효율화

Backend Devops @shepherd23

- Jest, Pytest를 활용해 주요 엔드포인트 및 서비스 로직에 대한 Unit/E2E 테스트 작성.
- Insomnia 등의 툴을 이용한 수동 테스트 시보다 훨씬 효율적인 개발 흐름을 구성할 수 있었음
  - => 기존 수동으로 테스트를 진행했을 때와 비교해 작업 시간 약 90% 감소. (기존 약 20분 → TDD 도입 후 최대 2분)
  - => 다른 팀과 버그 수정 관련한 불필요한 대화가 오고 가는 빈도 약 80% 감소. (기존 평균 4 ~ 5회 → TDD 도입 후 0 ~ 1회)

## => 사내 내부 문서화 배포를 위한 Private File Server 구축

- 정적 웹페이지를 회사 내부 인원만 열람할 수 있도록 하는 Private File Server 구축.
- [static-webpage-with-auth](#) 로 도커라이즈 하여 여러 도큐멘테이션 페이지에 쉽게 적용할 수 있도록 만들.

## => Google Sheets를 활용한 간이 어드민 대시보드 구축

Feature @shepherd23

- NestJS는 Django처럼 Admin Dashboard를 제공하지 않고, Thrid Party 라이브러리 (e.g. AdminJS) 도 Firestore 를 DB로 사용하는 경우를 고려한 라이브러리는 없었음.
- 이에 Google Sheets, Sheets API, Cloud Functions 를 활용해 사용자 데이터 시각화 및 쇼핑물 관리 기능을 포함하는 간이 대시보드 구축함.
- => UI 구현 필요 없이 간단한 로직 구현만 필요했기에 빠른 대시보드 구현이 가능했음.
- => 엑셀에 기반한 대시보드 구현으로 비개발 인원도 쉽게 지표를 확인하고 쇼핑물을 관리할 수 있었음.

## => Cloud Logging & Slack을 활용한 에러 모니터링 시스템 구축

Devops Feature @shepherd23

- 에러 확인을 위해 주기적으로 GCP 대시보드에 접속하는 일이 매우 번거롭다는 문제가 있었음.
- 이에 GCL 로그 라우터와 Cloud Functions를 활용해 에러가 발생할 때마다 Slack으로 알림을 전송하는 시스템 구축함.
- => 에러 확인을 위해 주기적으로 GCP 대시보드에 접속하는 비효율 제거.
- 4XX 에러나 5XX 코드를 가지는 로그도 라우터에 포함되어 알림이 전송되는 문제가 있었음.
- GCL의 Logging Query Language 를 이용해 알림을 받을 필요가 없는 로그를 필터링하는 기능을 추가함.
- => 불필요한 알림 방지.

## => 코드주석과 Notion을 연계한 사내 도큐멘테이션 작성 효율화

Devops Improvement @shepherd23

- 기존에는 비개발 인원이 개발 상황을 파악하기 위해 개발자에게 직접 문의하거나, 개발자가 이미 코드 주석으로 작성된 내용을 Notion에 다시 작성하는 등의 비효율이 존재했음.
- 이에 Swagger, Typedoc, Sphinx 등의 툴을 활용해 도큐멘테이션을 생성하고, 노션 페이지와 도큐멘테이션 페이지의 웹 링크를 연계하여 중복 작업을 최소화함.
- [static-webpage-with-auth](#) 를 활용해 Cloud Run에 배포하여 사내 인원만 접근할 수 있도록 구성함.

## => 서비스 기획을 위한 심층 인터뷰 진행

Service Planning Communication @labis

- 울산광역시 중구/북구/문수/성남/동구 청소년문화의집 등 울산 내의 많은 방과후 교육 기관을 방문해 데이터 수집함.
  - 시장의 수요 파악을 위해 타겟인 프리랜서 강사 및 청소년문화센터 담당자와의 심층 인터뷰 진행함.
- => 심층인터뷰를 통한 데이터 수집 과정을 통해 인터넷 서칭으로는 찾을 수 없는 실질적인 현장의 Pain Point를 파악할 수 있었음.
- => 이후 [엑셀러레이팅 프로그램 유치에 큰 영향](#)

## => 엑셀러레이팅 프로그램 유치를 위한 사업계획서 및 피치덱 작성

Project Management @labis

- 심층 인터뷰를 통해 수집한 데이터를 바탕으로 설득력 있는 사업계획서 및 피치덱을 작성함
- => [2022 울산청년창업사관학교](#) 엑셀러레이팅 프로그램 유치 성공
- => [창업 300 밸류-UP](#) 엑셀러레이팅 프로그램 유치 성공

아래 링크에서 위의 피치덱을 확인할 수 있습니다.

<https://drive.google.com/file/d/1HczhzUcrE3G31cn9tiu6fXh3vUofvldg/view?usp=sharing>

## => Observer 패턴을 활용한 상태 관리

Frontend Improvement @labis

- Redux에 과도하게 의존적인 코드로 복잡한 블록코드 UI 개발에 비효율이 발생함.

- 이에 이벤트를 관리하는 `EventRegistry` 클래스를 구현하여, `EventRegistry.emit` 을 통해 이벤트를 발생시키고, `EventRegistry.on` 을 통해 이벤트를 구독하는 방식으로 상태 변경 사항을 관리하게끔 상태 관리 방식을 대거 수정함.
- => 불필요한 Redux 사용을 줄이고 코드 유지보수가 효율화

## See Also

- [Observer Pattern @ dbcav3](#)

## => OOP를 활용한 블록 코드의 블록 정의 개발

Frontend Feature @labis

- 각 블록정의는 `AbstractBlock` 을 상속하고 `actionChain` 이라는 메서드를 구현해야하며, 이 메서드는 상속과 참조를 통해 Redux Store로 연결되는 액션의 체인을 구성함.
- => `AbstractBlock` 을 중점으로 상태 관리에 있어 단순함을 유지하며, OOP에 기반해 블록 정의의 확장성을 높임.

## See Also

- [Observer Pattern @ dbcav3](#)
- [Block Definition Source Code](#)

## => 상품 정보에 대한 Caching 기능 추가

Backend Optimization @shepherd23

- 추천에 필요한 상품 정보를 Firestore에서 가져오는 과정에서 발생하는 불필요한 읽기 과정을 줄이기 위해 인메모리 캐시 도입.
  - 상품 정보에 대한 캐싱 및 캐시 무효화 기능을 제공하는 NestJS 모듈 `ActionProviderModule` 구현.
- 도큐먼트 읽기 횟수 3천만/일 -> 150만/일 로 감소
- => 매우 큰 비용 절감 효과

## => 점진적인 코드베이스 이관을 통한 DB 마이그레이션

Backend Improvement @shepherd23

- 버저닝, JSDoc Block Tags, `backwardCompatible*` 등의 패턴을 활용한 점진적인 코드베이스 이관으로 일괄 DB 마이그레이션이 불가능한 스키마에 새로운 변경 사항을 반영

## => 상품 추천 속도 향상을 위한 로그 데이터 분석 및 코드 인스펙션

Backend Troubleshooting Optimization @shepherd23

- 특정 쇼핑몰에서 상품 추천 API의 응답 속도가 느린 것을 발견함.
- GCL 로그를 내려받아 `numpy`, `pandas`, `matplotlib` 을 활용해 각종 지표를 살펴보았으나, 위 쇼핑몰을 제외하고는 응답 속도가 예상대로 나타남.
- 이에 상품 추천 API의 코드를 인스펙션하며, `Date.now()` 를 활용해 각 단계의 응답 속도를 측정함.
- => Firestore로의 count 쿼리가 응답 지연의 원인임을 발견하고, count 쿼리를 제거해 응답 지연 문제 해결.

## => Leaky Bucket Policy를 적용한 API의 호출 제한 문제 해결

Devops Feature @shepherd23

- 데이터 요청이 필요한 API의 Leaky Bucket Policy로 인해 1초에 2회 수준으로 API 호출이 제한됨.
- 각 상품의 상세 정보를 하나씩 전부 요청해야 하는 상황에서 단순 API 호출로는 HTTP 429 에러 발생했음
- => LeakyBucketClient 클래스를 구현해 호출의 빈도를 제한하고, 429 에러 발생 시 재시도하는 로직을 구현해 이슈 해결.

## => 긴 프로세스를 담당하는 Worker 인프라 구축

Devops Feature @shepherd23

- 기존에는 모든 서비스를 Google Cloud Run을 이용해서 배포했음.
  - 데이터 요청이 필요한 API의 Leaky Bucket Policy로 인해, 상품 정보를 자사 DB로 가져오는 프로세스가 길어지면서 Cloud Run의 프로세스 제한 시간인 1시간을 초과하는 문제 발생.
- => GCE 인스턴스 그룹과 Cloud Load Balancer를 활용해 긴 프로세스를 담당하는 Worker 인프라를 구축하여 이슈 해결.

- Cloud Build Substitution Variable 과 Docker Build Args를 활용해, Cloud Run과 Worker 인프라 배포 코드 공유.

## => GCE instance group autoscaler 의 instance termination으로 인한 프로세스 중단 문제 해결

Backend Devops Troubleshooting @shepherd23

- GCE instance group autoscaler가 scale down을 하면서 instance를 종료하는 과정에서, 종료되는 instance에 할당된 프로세스를 중단하는 문제가 발생함.
- => NestJS의 `OnApplicationShutdown` 인터페이스를 구현하여 프로세스 종료 시점에 진행중인 작업을 재시작하는 로직을 구현해 이슈 해결.
  - Fire and forget 방식으로 request를 보내 Shutdown process에 영향을 주지 않도록 함.

## => 모델 파라미터 압축을 통한 성능 향상

Backend Optimization @shepherd23

- Bandit Model의 파라미터가 너무 커 HTTP 413 에러를 반환하는 문제를 발견함.
- 이에 `np.float32`, `gzip` 을 도입하고 파라미터를 압축, 압축해제하는 로직을 추가함.
- => 모델 파라미터 크기 10배 축소
- => 상품 추천 응답 시간 2배 증가

## => 점진적인 코드베이스 이관을 통한 서비스 장애 방지

Backend Improvement @shepherd23

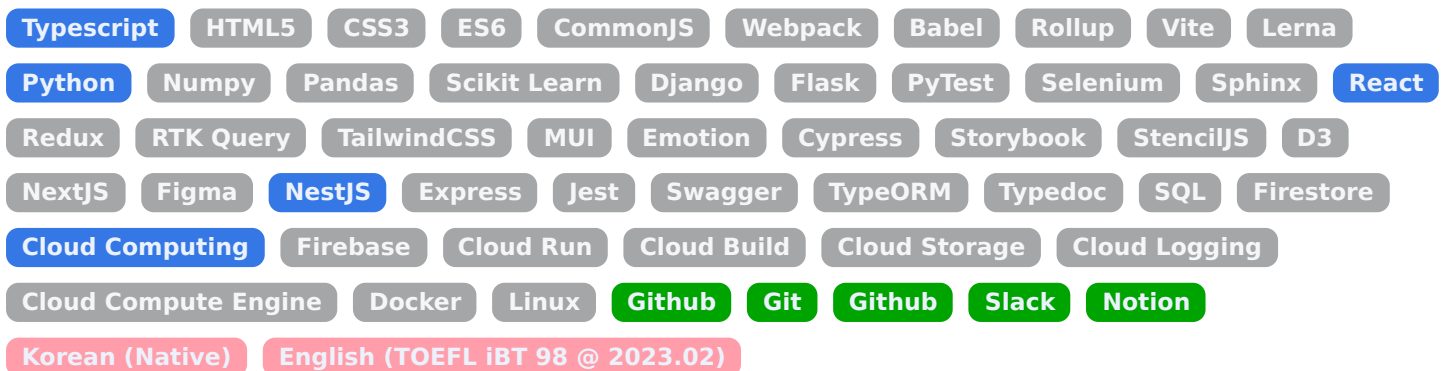
- Bandit 모델의 버전이 많아짐에 따라 각 모델 버전을 engines 폴더 아래에 디저트 이름을 붙여 모듈화함.
- 이와 함께 엔드포인트 구조를 체계적으로 변경하는 과정에서 기존 엔드포인트를 제공하는 `blueprint` 가 사라지면 서비스에 장애가 발생함을 인지함.
- 기존의 코드베이스와의 호환성을 유지하기 위해 기존 코드베이스에 존재하는 엔드포인트를 새로운 엔드포인트로 Redirect 하도록 구현함.
- => 서비스 간의 업데이트 속도 차이로 인해 발생하는 서비스의 일시적인 장애 없이 코드베이스 이관할 수 있었음.

## => 데코레이터 패턴을 활용한 AI 팀의 업무 효율화

Backend Improvement @shepherd23

- 데코레이터 패턴을 활용해 각 엔드포인트의 핸들러 함수를 데코레이터로 감싸 에러 핸들링, 인증 등의 공통된 로직을 분리함.
- => 웹과 관련한 코드를 데코레이터를 통해 추상화하여 웹에 익숙하지 않은 AI 팀의 업무 효율화.
- => 신입 AI 팀원도 별도의 웹 관련 교육 없이 바로 업무에 참여 가능한 수준.

## 스킬셋



## 수상 및 기타

- 2022 울산청년창업사관학교 수료
- 2022 창업유망 300 밸류-Up 프로그램 수료
- 동남권실험실창업혁신단 주관 UNIST-Pre 아이코어 프로그램 장려상

- 2022 울산 스마트 해상물류 창업 오디션 예선 프로그램 수료
- 2022 울산 스마트 해상물류 창업 오디션 본선 프로그램 수료
- 경기콘텐츠진흥원 주관 2021 애자일 해커톤 우수상
- 한국관광공사 & 카카오 주관 2021 관광데이터 활용 공모전 장려상
- 2021 울산지역대학 YOUTH 창업우수 아이디어 경진대회 대상
- 3rd PATHHACK : 쉬운 해커톤 스폰서상
- JUNCTION ASIA 2022 참가자

최종 수정: **2023년 8월 30일에**