오픈소스 답게 소프트웨어 설계하기

Designing software as open source

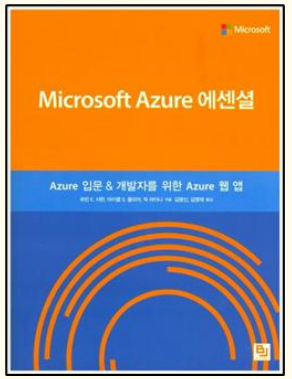
김영재
LINE

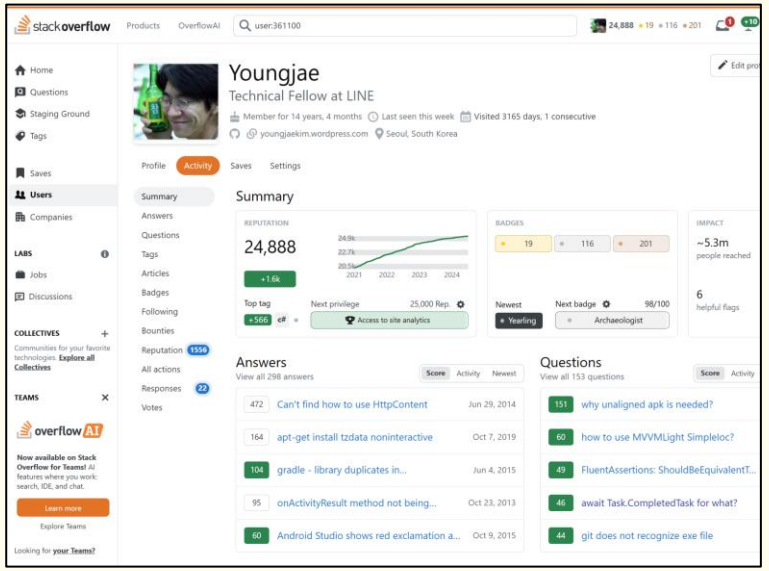
00 자기소개

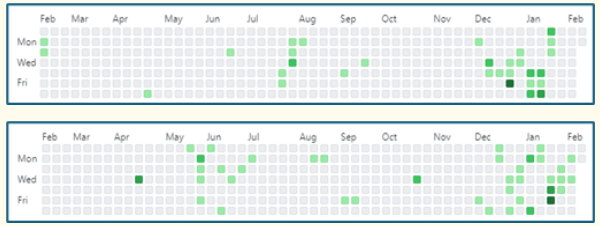
- LINE 기술임원, 일본 出前館(Demae-can) CPO
  - LINE, ABC Studio
  - 일본 최대 푸드 딜리버리 서비스

- Naver CLOVA Logistics, Edge ML lead
  - 50+ AI/ML Products
  - 200+ Process Automation
- Bapul CTO (acquired to Naver/LINE)
  - No.1 Edu-tech Startup
- Sensor/Embedded 연구원

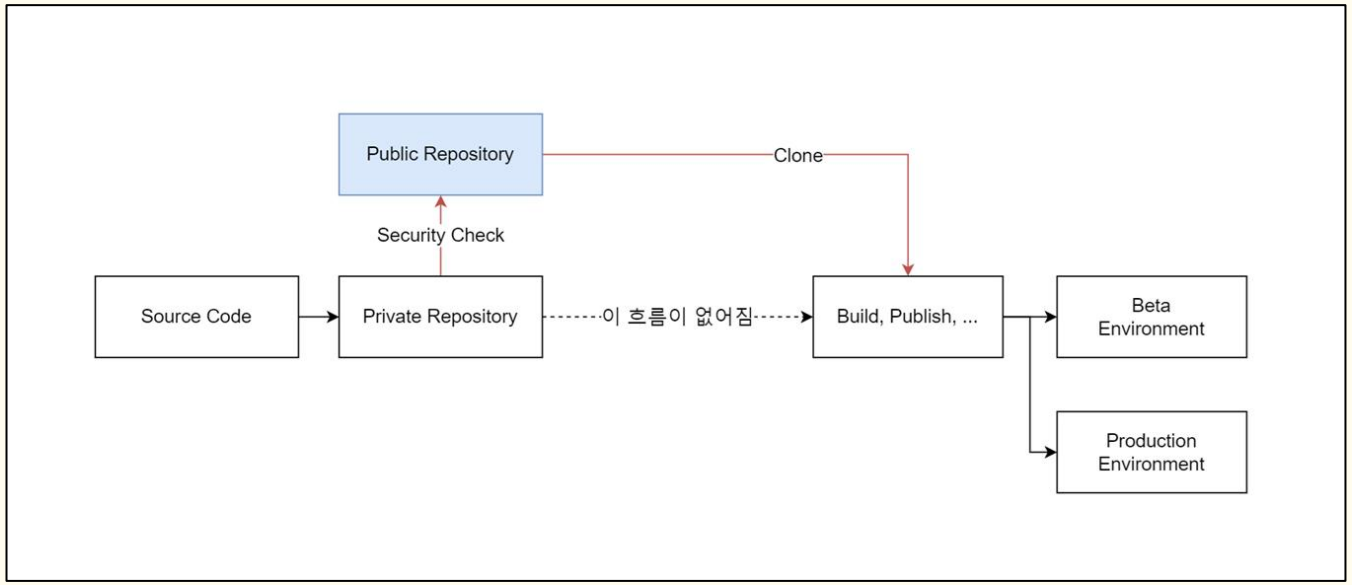
- 오픈소스
  - 위 기업들의 20여개의 소프트웨어 디렉팅, 개발







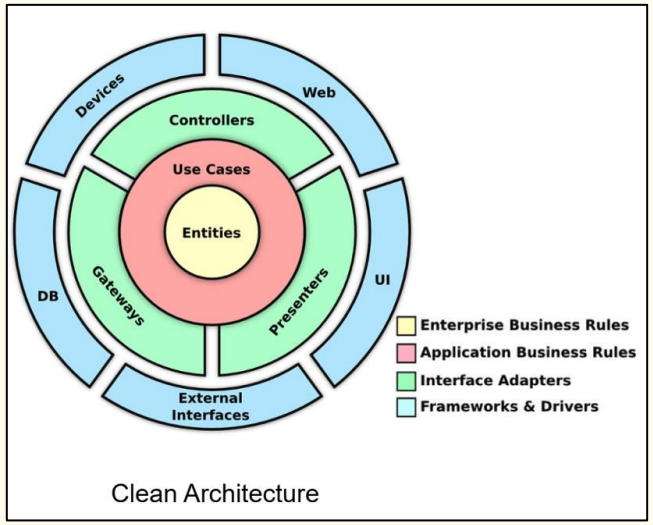
00 회사 소프트웨어를 오픈소스로 할 때의 장점

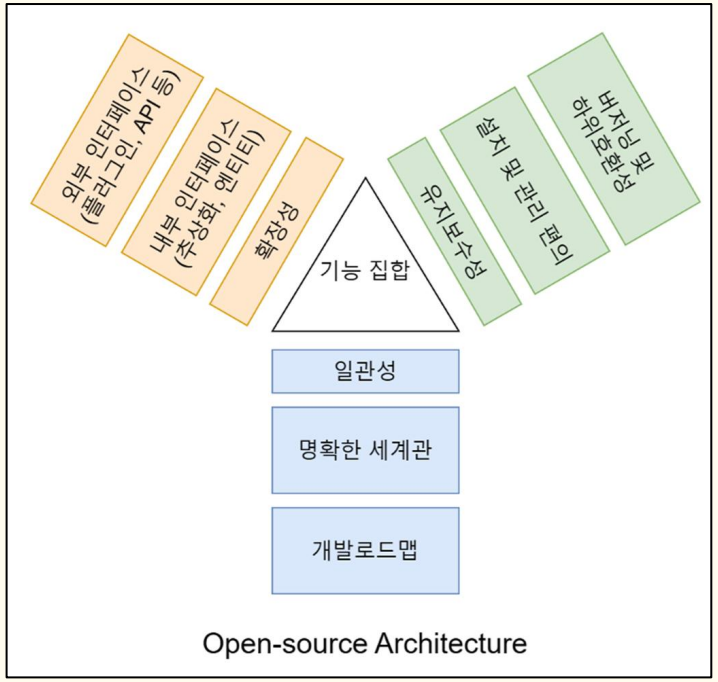


- 아키텍처의 독립성 – 환경 종속을 줄여서 범용성 확보
- 소프트웨어 사용의 자유도 상승 – 관계사간 지적재산권 법률검토 시간 단축
- 보안 향상 – 첫 커밋부터 안전하게

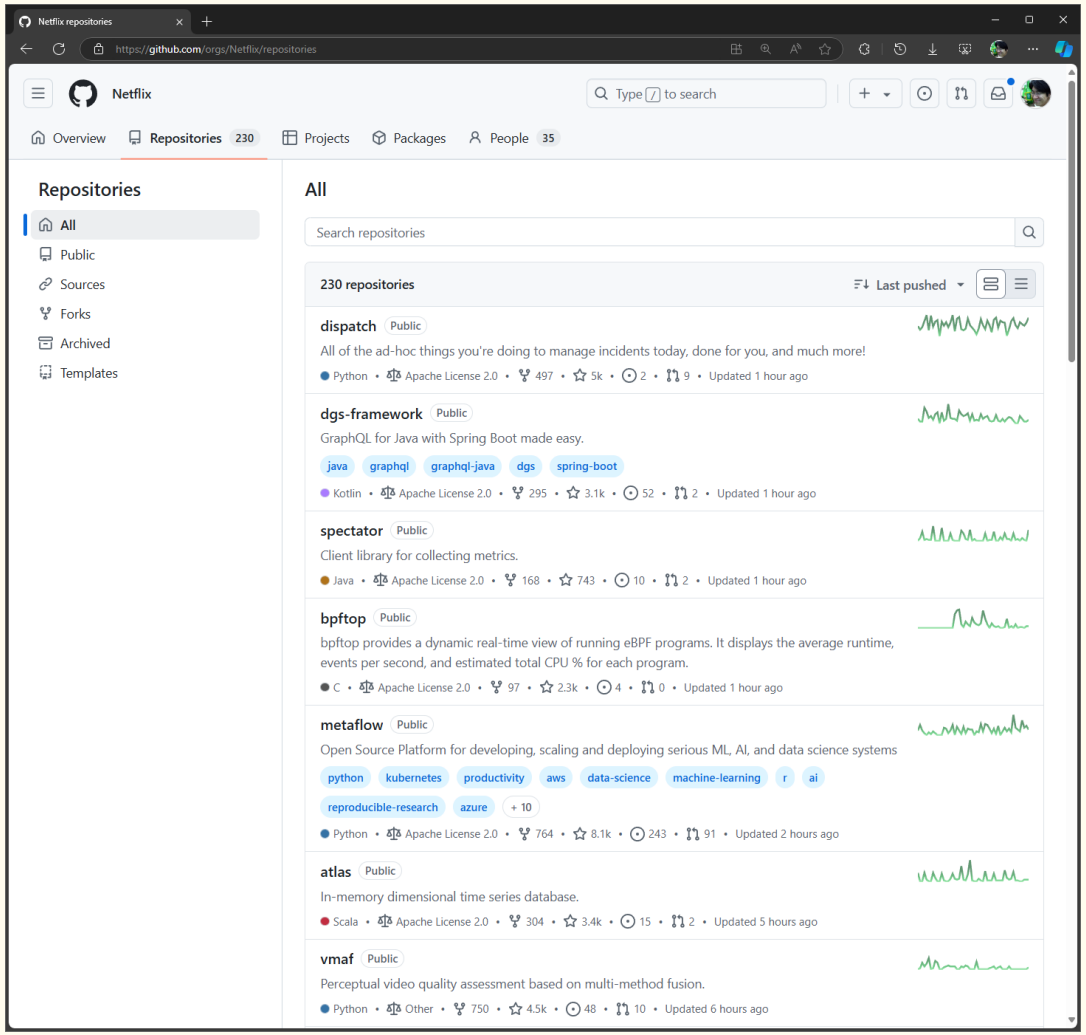
00 아키텍처의 중앙으로 기능이 모인다

- 비즈니스 로직은 최대한 바깥으로 밀어낸다
  - 외부 의존성도 최대한 바깥으로
  - 당연한 것 같은데 회사에서는 그렇지 않은 경우가 많음



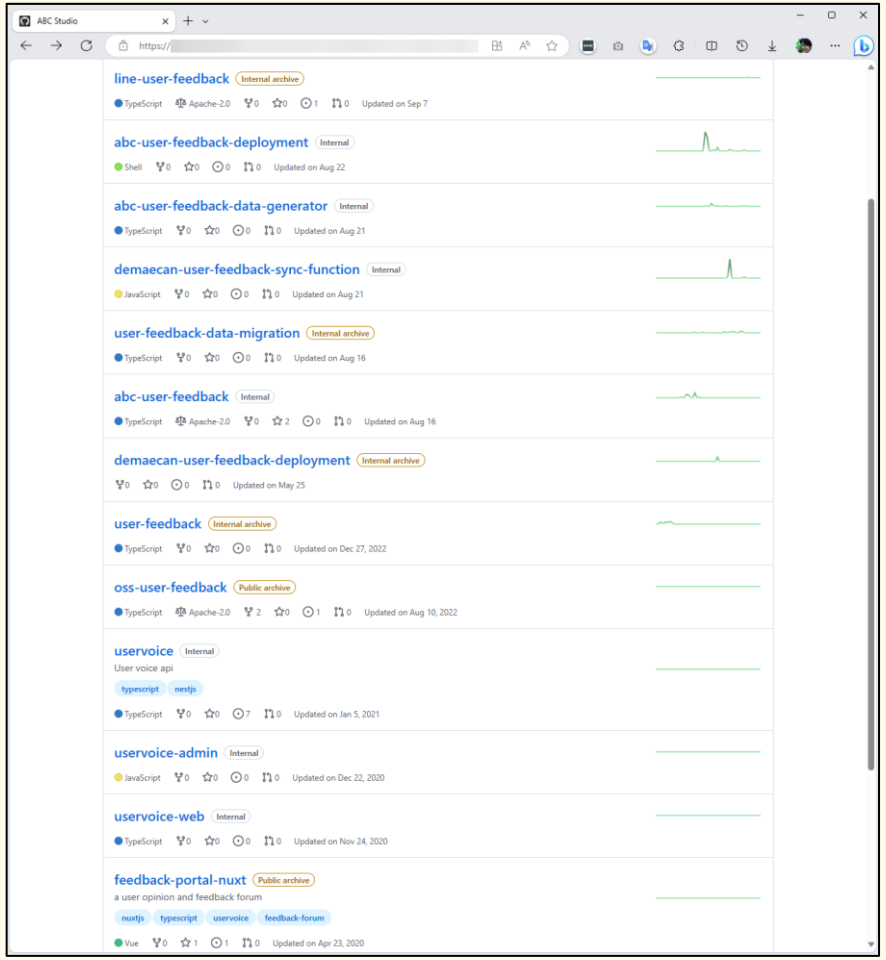


00 마치 공구상자 모음 같아진다



Netflix OSS – https://github.com/orgs/Netflix/

00 시행착오가 꽤 있었다



수많은 Archived 처리

=

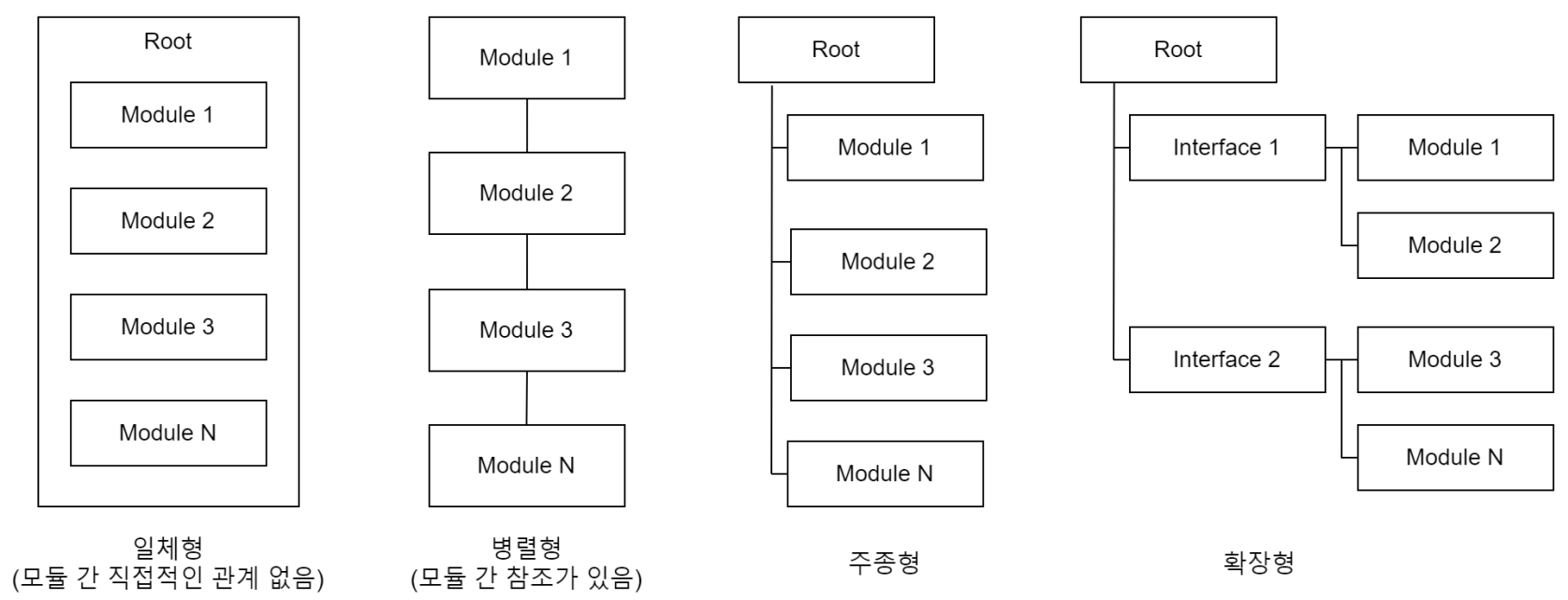
빈집만 가득…

01

좋은 구조로 꾸준히 가꾸어나가는 방법

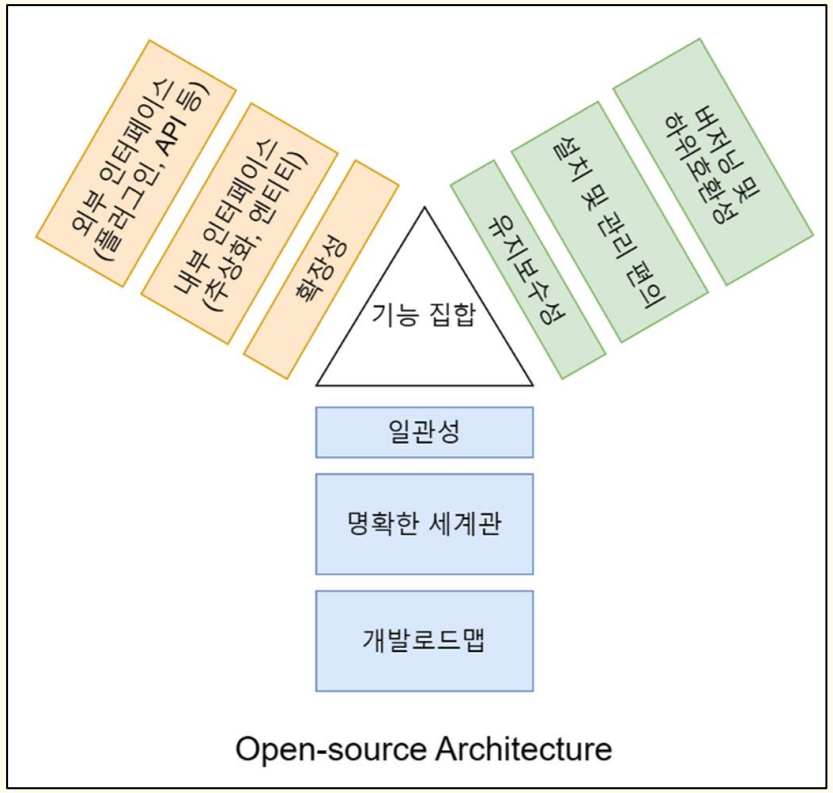
01 소프트웨어 유형별 구조

- 처음부터 명확하게 정의하면 유틸리티 라이브러리
- 데이터 의존 라이브러리
- API/CLI 애플리케이션
- 최종 사용자가 있는 애플리케이션



01 오픈소스가 지속가능하기 위한 품질속성(quality attributes)

- 일관성
- 유지보수성
- 확장성

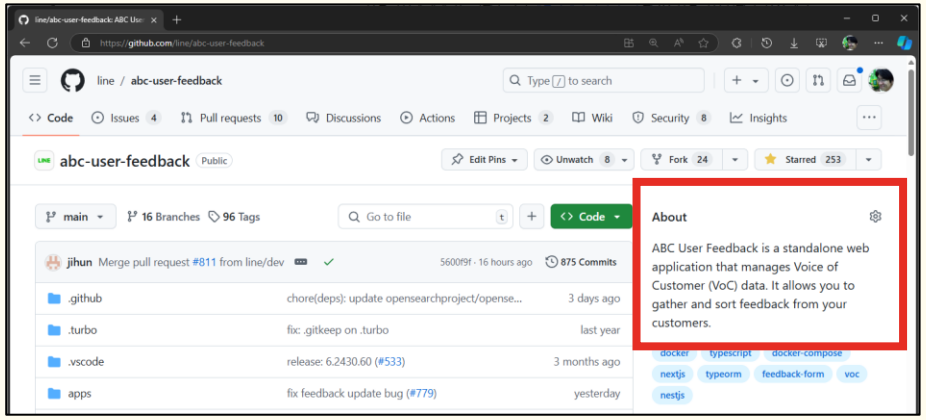


External Quality	Brief Description
Availability	The extent to which the system's services are available when and where they are needed
Installability	How easy it is to correctly install, uninstall, reinstall, and update the application
Integrity	The extent to which the system protects against data loss and corruption
Interoperability	How easily the system can interconnect and exchange data with other systems
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system
Internal Quality	**Brief Description**
Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent components can be used in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, or other extensions
Verifiability	How readily developers and testers can confirm that the software was implemented correctly

https://medium.com/analysts-corner/those-other-requirements-quality-attributes-and-their-inescapable-tradeoffs

01 일관성 - 명확한 세계관 수립

- 이 소프트웨어를 사용함으로써 얻을 수 있는 가치와 가능성
- 추천: 이상한 이름 짓지 맙시다 (기능이 곧 이름, 이름이 곧 기능!)

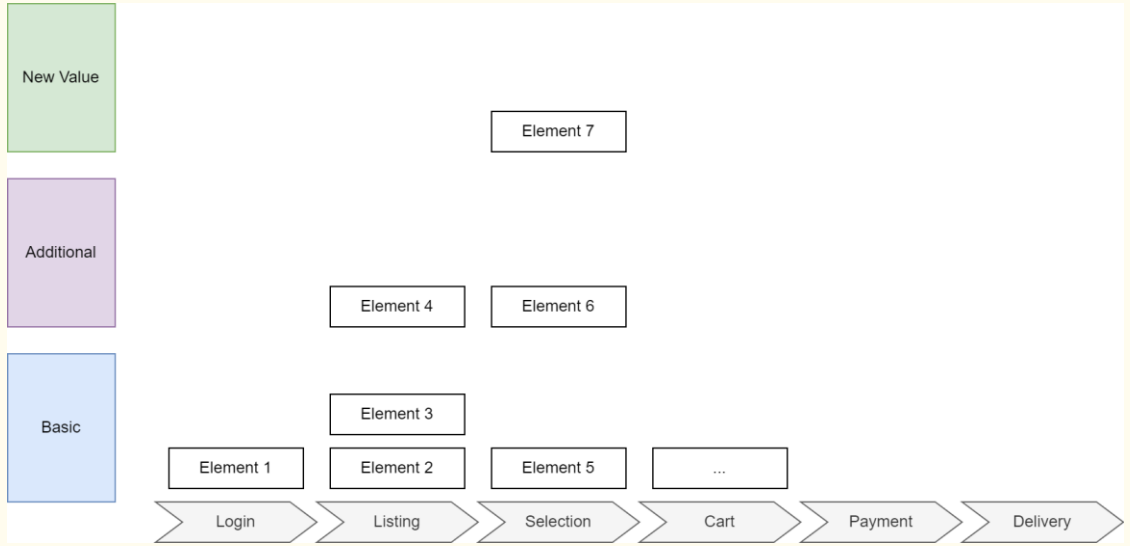


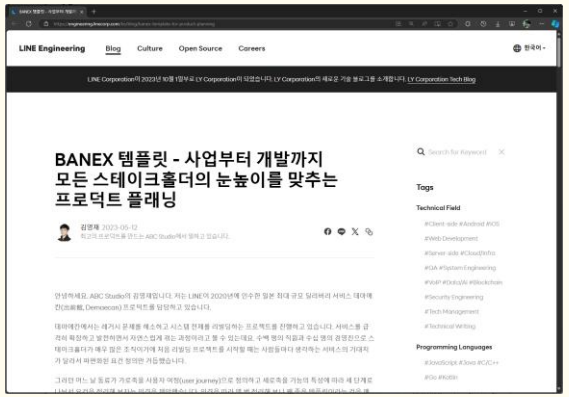
훌륭한 소프트웨어는
소개말에서 세계관을
명확하게 정의

- <u>데스크톱 소프트웨어의 예</u>: Microsoft PowerToys is a set of utilities for power users to tune and streamline their Windows experience for greater productivity.
- <u>SDK의 예</u>: Flutter is Google's SDK for crafting beautiful, fast user experiences for mobile, web, and desktop from a single codebase.
- <u>데이터베이스의 예</u>: OrientDB is an Open Source Multi-Model NoSQL DBMS with the support of Native Graphs, Documents, Full-Text search, Reactivity, Geo-Spatial and Object Oriented concepts.

01 BANEX - 로드맵 수립에 유용한 템플릿 #1

- 좋은 아키텍처는 흔들림 없는 프로젝트 로드맵에서 시작 (결국 사람이 하는 일)
- 여럿이 모여서 중요도와 우선순위를 합의
  - Basic – 당연히 있어야 하는 기능
  - Additional – 있으면 좋은 기능
  - New Value – 미래를 위해 필요한 기능

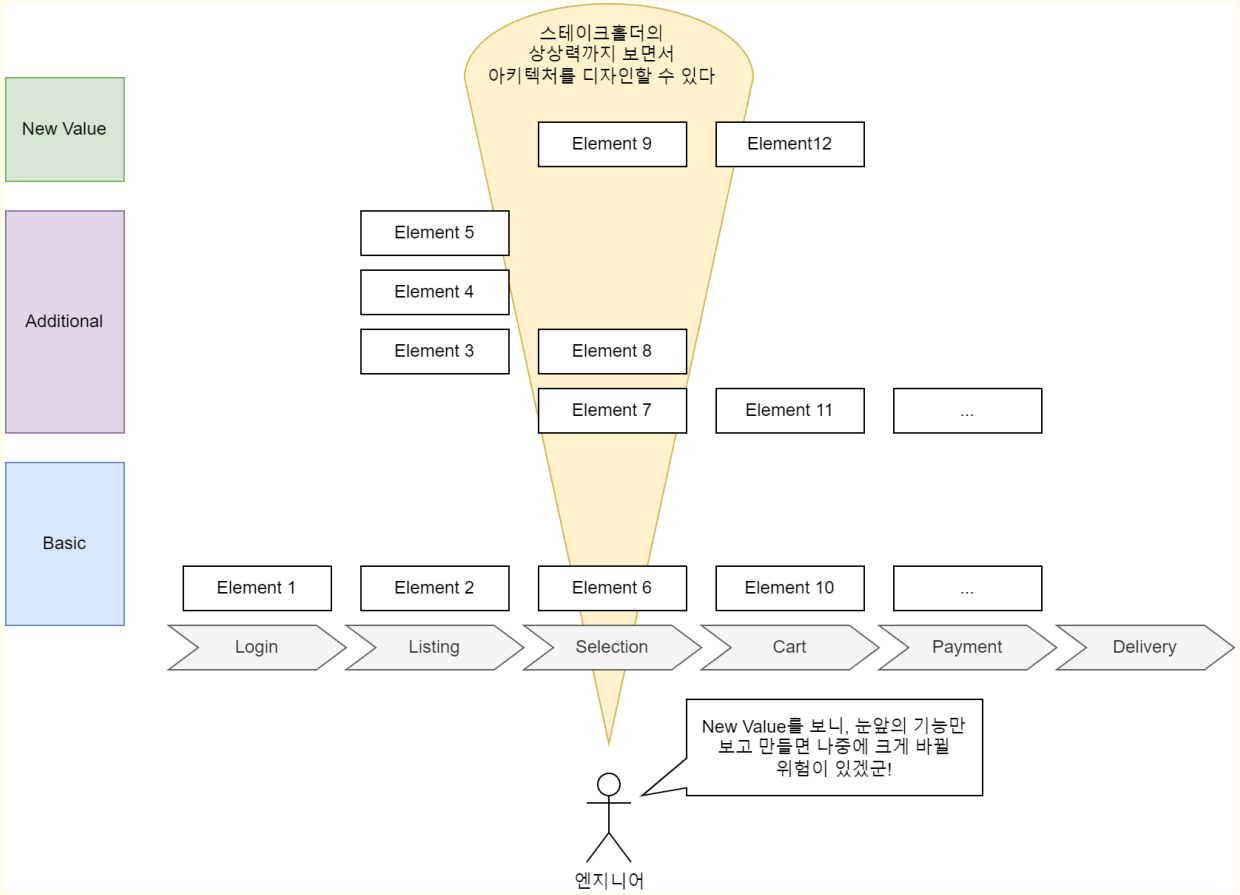




https://engineering.linecorp.com/ko/blog/banex-template-for-product-pl

01 BANEX - 로드맵 수립에 유용한 템플릿 #2

- 미래의 발전 방향을 보면서 아키텍처를 수립할 수 있다



01 확장성

- ‘무엇을 조금만 수정하면 내가 원하는 대로 쓸 수 있을까?’에 대한 대답
- **생태계 구성의 첫걸음** (특히 우리나라 소프트웨어에서 약한 부분)

01 유지보수성

- 쉬운 설치 경험은 당연
- 강조하고 싶은 것: 쉬운 설정

- 좋은 훈련: 로직 없이 설정파일부터 작성

```yaml
github-pr-webhook-to-slack-notification:
    github:
        url: https://sample.github.com
        username: sample-name
        authtoken: sample-key
    slack:
        url: sample-webhook.slack.com
        authtoken: sample-slack-key
    template:
        default: sample-template.txt
```

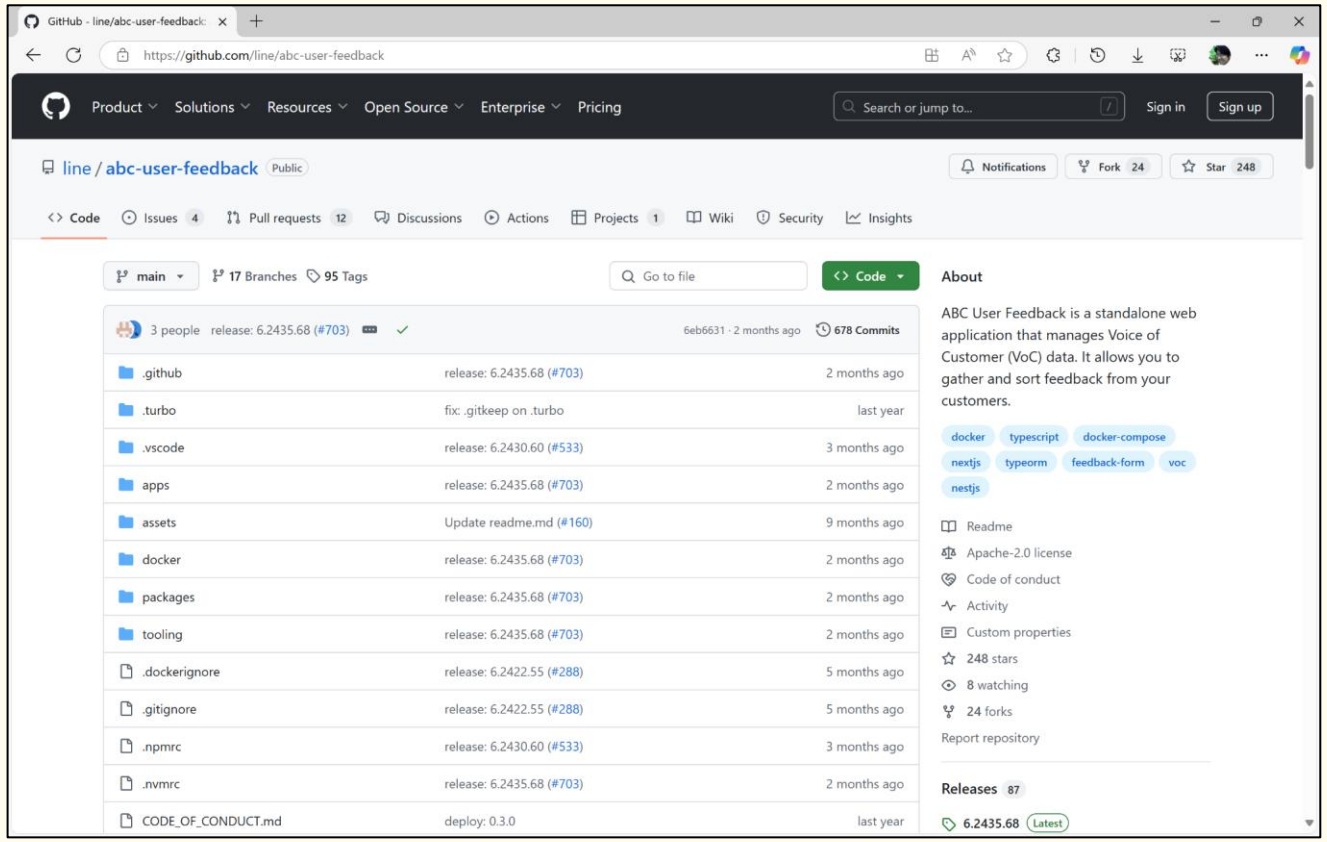
어느 자동화 스크립트의 설정파일 예시

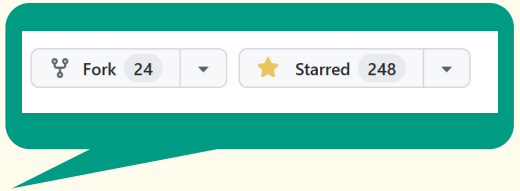
02

ABC User Feedback 사례

02 ABC User Feedback - 사용자 의견 수집 도구

- 5년간 만들고 있음



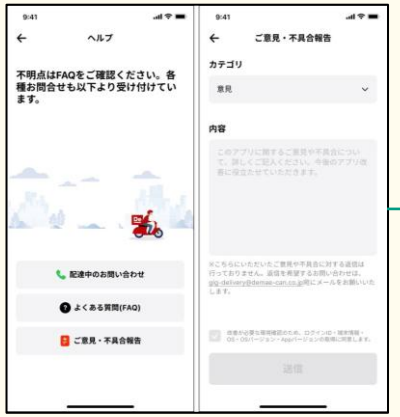


BOSCH

+스타트업 10여곳

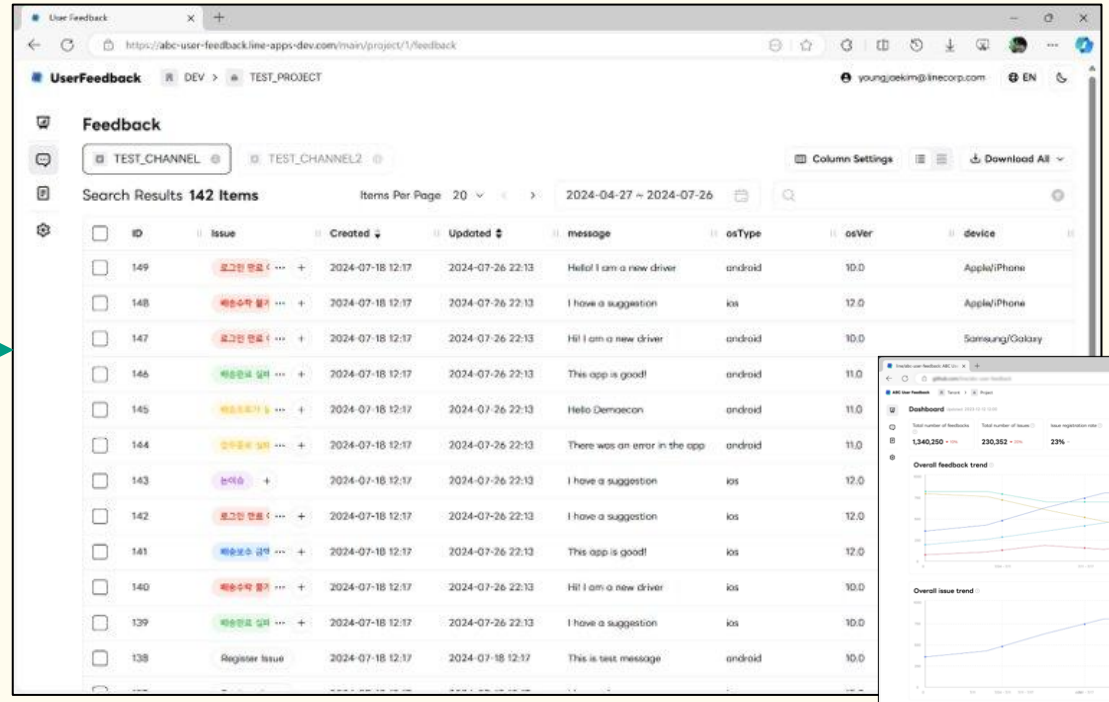
https://github.com/line/abc-user-feedback

02 ABC User Feedback - 사용자 의견 수집 도구

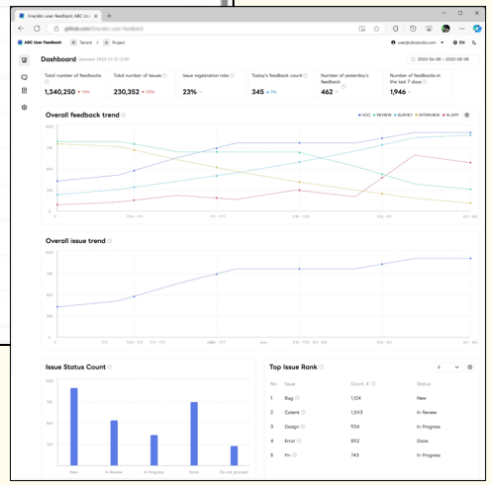


App

API

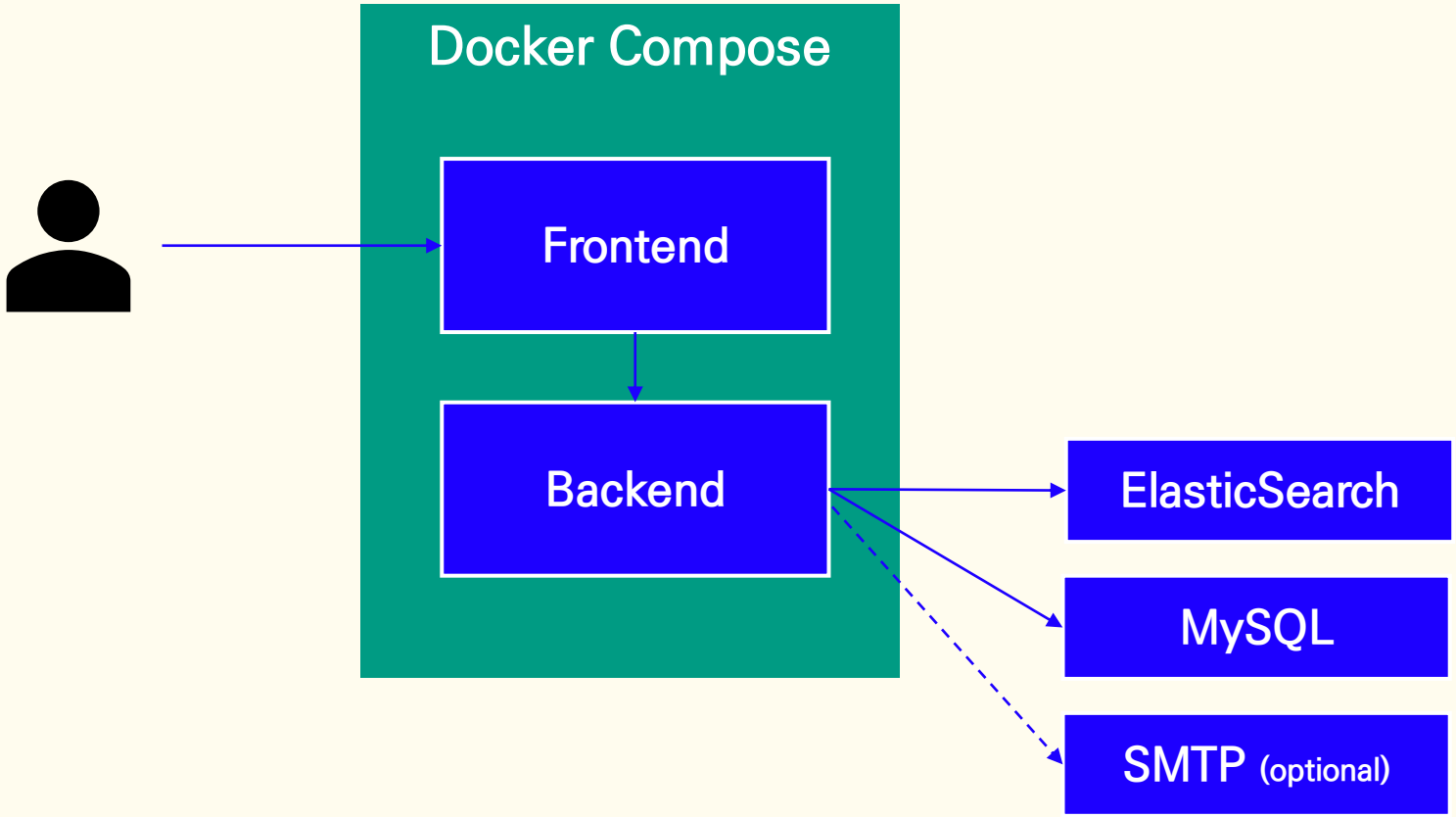


Admin



Dashboard

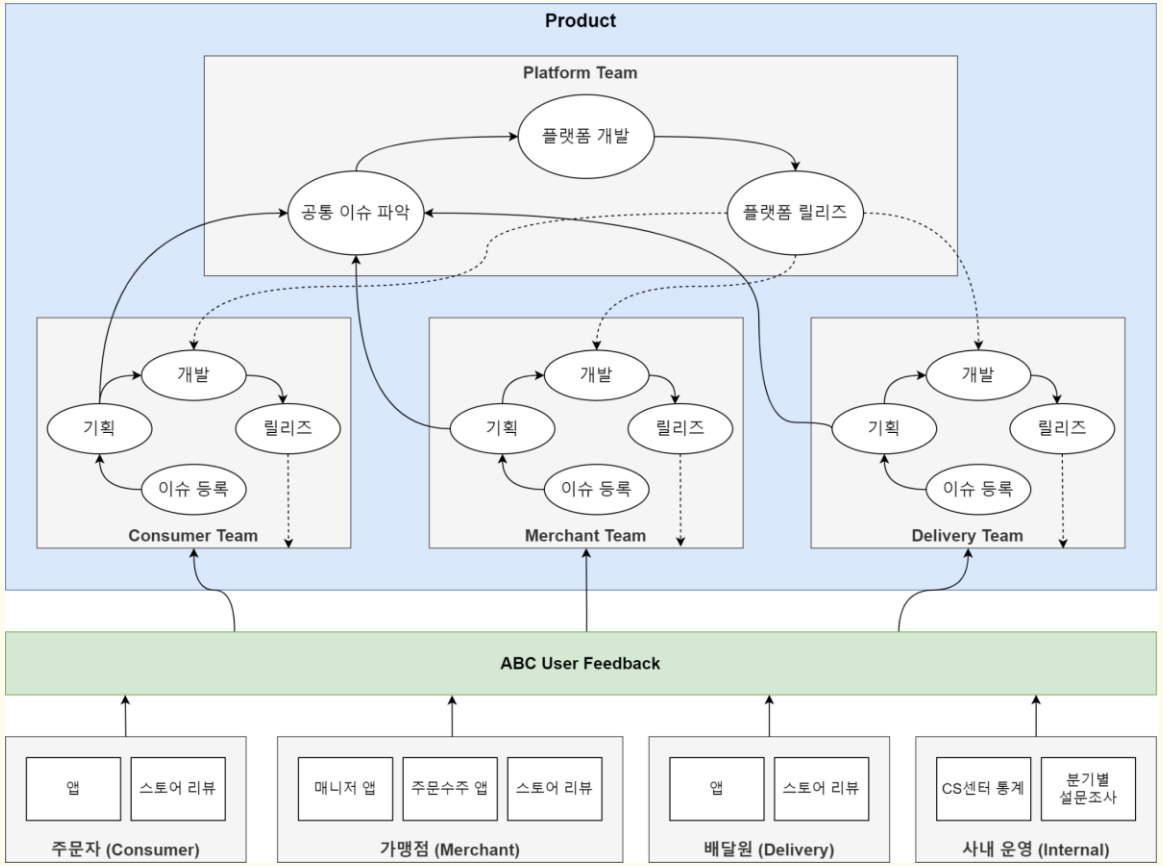
02 대략적인 구조


Docker Compose
Frontend
Backend
ElasticSearch
MySQL
SMTP (optional)

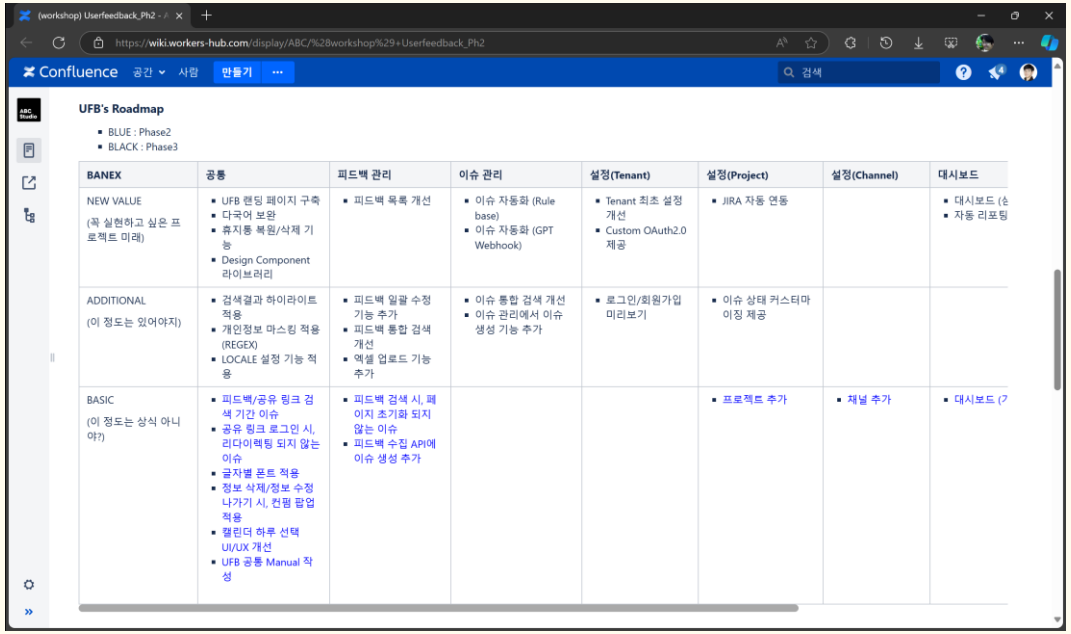
02 좋은 구조가 되도록 꾸준히 신경쓰는 3가지 품질 속성

- 일관성 – 5년간 User Feedback 수집도구라는 정체성 유지
- 유지보수성 – Docker, npx 등 설치 편의와 설정 파라미터 제공
- 확장성 – SSO 인증, API, Webhook 제공

02 일관성을 위한 실제 활용과, 이를 토대로 BANEX 진행



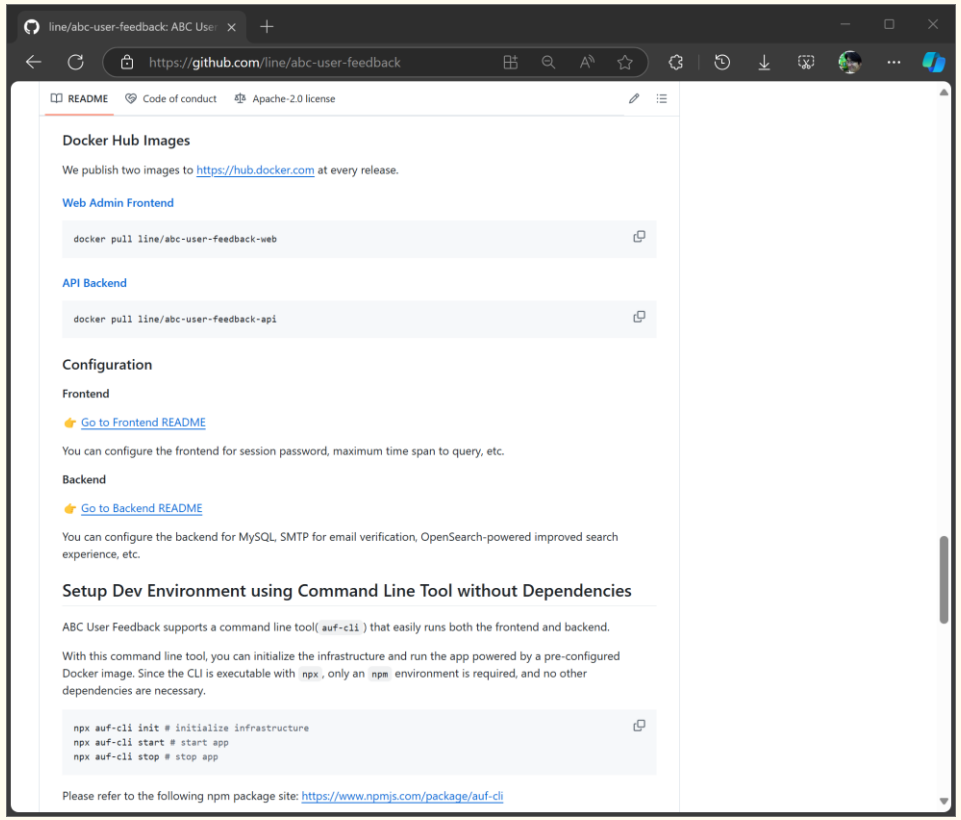
모든 프로덕트에 적용하여 사용자 의견 수집 중



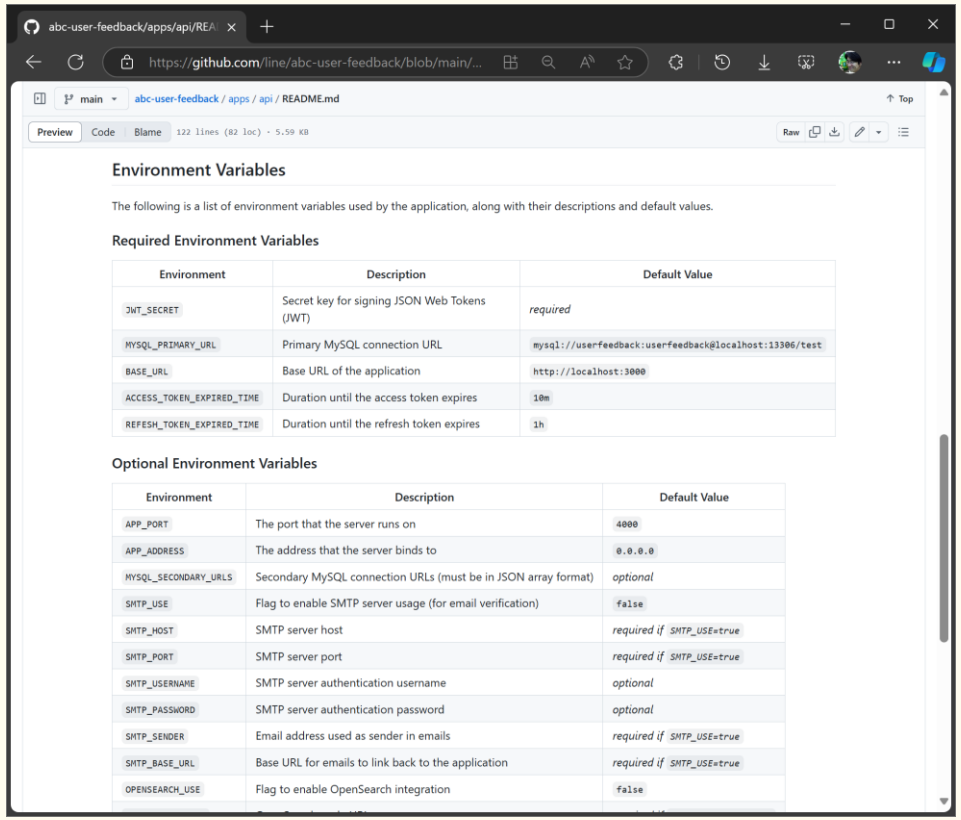
BANEX 3회째 진행

02 유지보수성

- 다양한 설치 편의와 설정 파라미터 설명



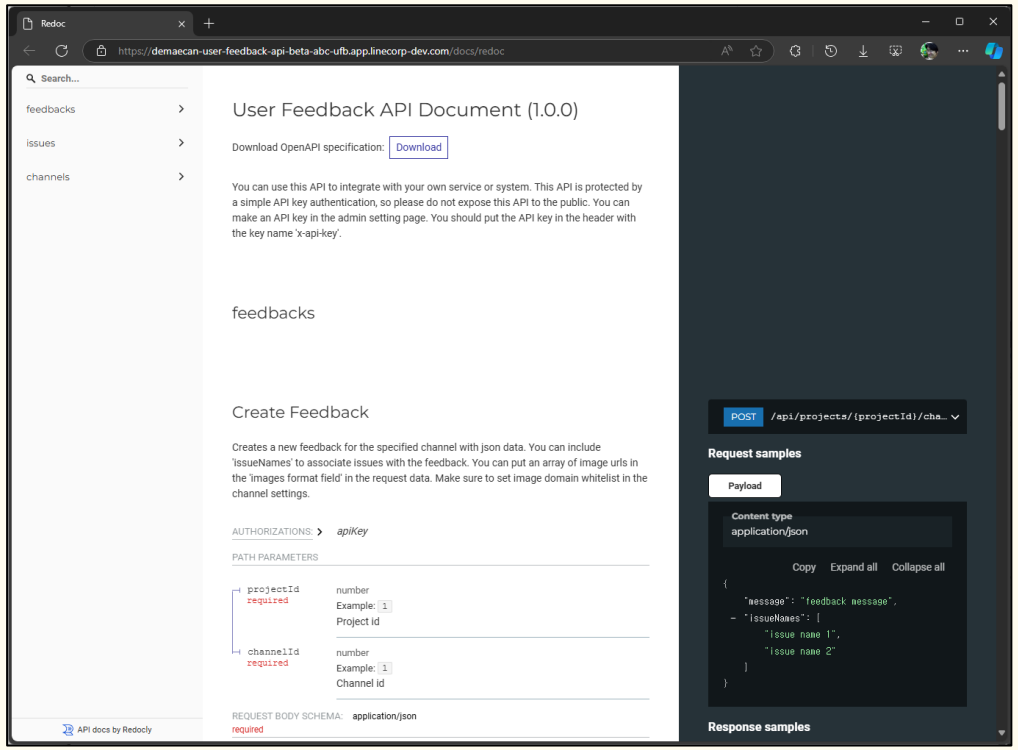
Docker Hub, NPX Script, …



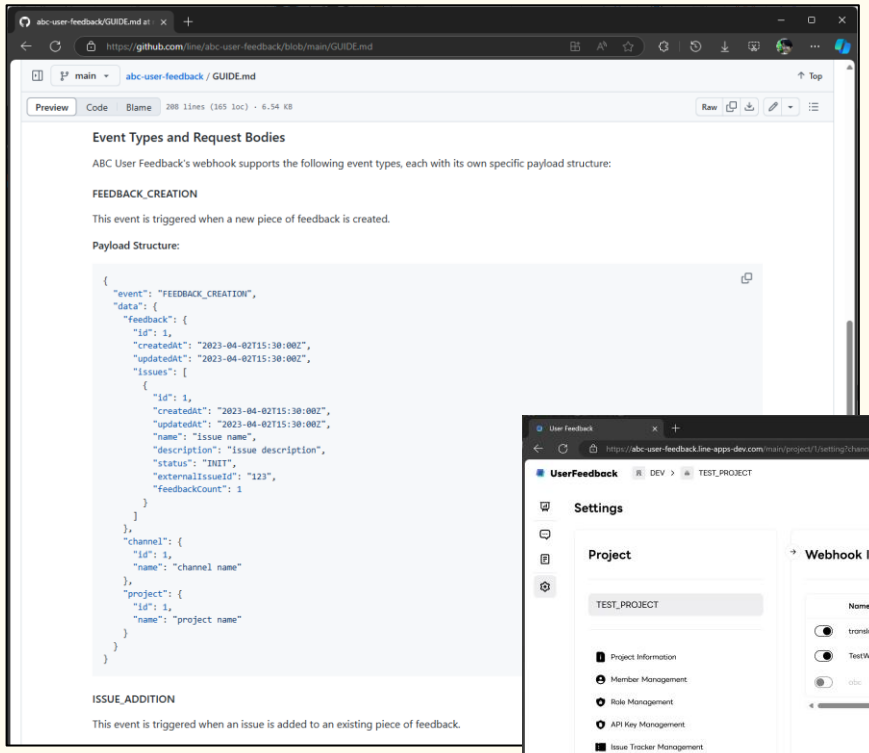
친절한 파라미터 설명서

02 확장성

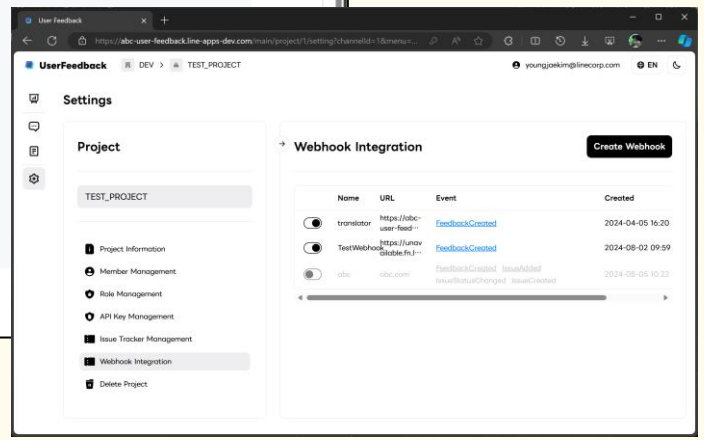
- API, Webhook



API 설명서

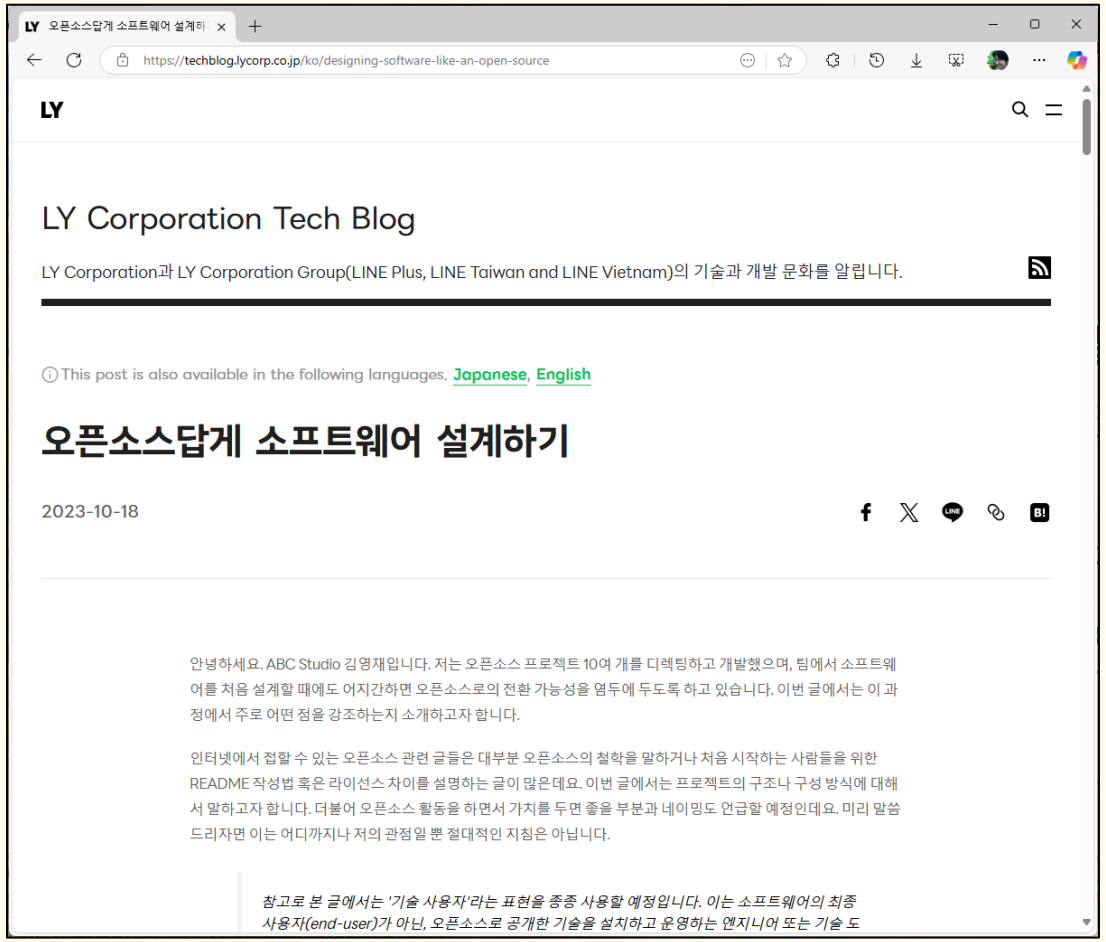


Webhook 설명서



Webhook 설정 UI

더 자세히 읽어보기



https://techblog.lycorp.co.jp/ko/designing-software-like-an-open-source

오픈소스 답게 소프트웨어 설계하기

youngjaekim@linecorp.com