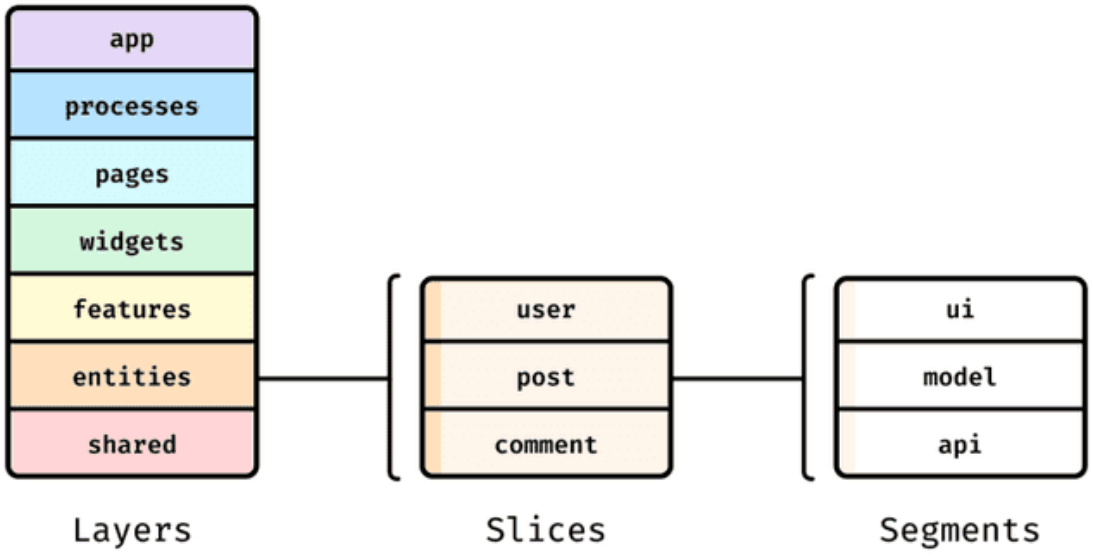
MovieTok 프로젝트 코드 분석

FSD (기능 분할 설계) 아키텍처

프로젝트를 진행하면서, 대부분 모듈 간의 느슨한 결합과 높은 응집력을 제공해야 하며 쉽게 확장할 수 있는 아키텍쳐를 필요로 합니다.

위의 같은 문제를 해결할 수 있는 **기능 분할 설계(Feature-Sliced Design, FSD) 아키텍처**에 대해 설명합니다.

먼저 **레이어(layer), 슬라이스(slice), 세그먼트(segment)**의 세 가지 개념으로 구분한다.



레이어

레이어는 최상위 디렉토리이자 애플리케이션 분해의 첫 단계이다.

레이어 수는 최대 7개로 일부는 선택 사항이지만, 표준화 되어 있다.

레이어들은 코드베이스를 조직화하고, 모듈화되고 유지보수 용이한 확장 가능한 아키텍처를 촉진하는 데 도움을 준다.

- ▼ src/
  - ▼ app/

    애플리케이션 로직이 초기화 되는 곳 (프로바이더, 라우터 설정 등)
  - ▼ processes/

    여러 단계로 이루어진 등록과 같이 여러 페이지에 걸쳐 있는 프로세스를 처리한다.
  - ▼ pages/

    애플리케이션 페이지가 포함된다.
  - ▼ widgets/

    페이지에 사용되는 독립적인 컴포넌트
  - ▼ features/

    비즈니스 가치를 전달하는 사용자 시나리오와 기능을 다룬다.
  - ▼ entities/

    비즈니스 엔티티
  - ▼ shared/

    특정 비즈니스 로직에 종속되지 않은 재사용 가능한 컴포넌트와 유틸리티

기능 분할 설계의 주요 특징 중 하나는 **계층 구조**이다.

features 레이어가 **entities 레이어**보다 더 위에 있기 때문에 entities 레이어는 features 레이어의 기능을 **사용할 수 없다.**

슬라이스

각 레이어에는 애플리케이션 분해의 두 번째 수준인 슬라이스라는 하위 디렉토리가 있다.

슬라이스에서 연결은 **추상적인 것이 아니라 특정 비즈니스 엔티티에 대한 것**이다.

- src/
  - app/
    - providers
    - styles
    - index.tsx
  - processes/
  - pages/
    - home
    - profile
    - movie
  - widgets/
    - headers
    - footers
    - items
    - lists
  - features/
    - user
    - auth
  - entities/

▼ movie

▼ user

▼ shared/

특정 비즈니스 로직에 종속되지 않은 재사용 가능한 컴포넌트와 유틸리티

세그먼트

각 슬라이스는 세그먼트로 구성된다.

세그먼트는 목적에 따라 슬라이스 내의 코드를 나누는 데 도움 된다.

- api - 필요한 서버 요청 (request)
- ui - 슬라이스의 UI 컴포넌트
- model - 비즈니스 로직
- lib - 슬라이스 내에서 사용되는 보조 기능
- config - 슬라이스에 필요한 구성값이지만 구성 세그먼트는 안 필요함.
- consts - 필요한 상수.

공개 API

각 슬라이스와 세그먼트에는 공개 API가 있으며, 공개 API는 index.js 또는 index.ts 파일이다.

Axios 설정

shared 디렉터리에 axios 폴더를 만들고, axios 설정을 진행하였다.

```
import axios from "axios";
import { TMDB_BASE_URL } from "../../constants/const/const";

export const TMDBAxiosInstance = axios.create ({
    baseURL: TMDB_BASE_URL,
    timeout: 1000,
})

TMDBAxiosInstance.interceptors.request.use(function (config)
    return config;
}, function (error) {
    return Promise.reject(error);
});

TMDBAxiosInstance.interceptors.response.use(function (respons
    return response.data;
}, function (error) {
    return Promise.reject(error);
});
```