BFF(Backend For Frontend)

BFF

> 프론트엔드 종류별(웹, 모바일)로 최적화된 백엔드 API를 각각 제공하는 아키텍처 패턴이다.

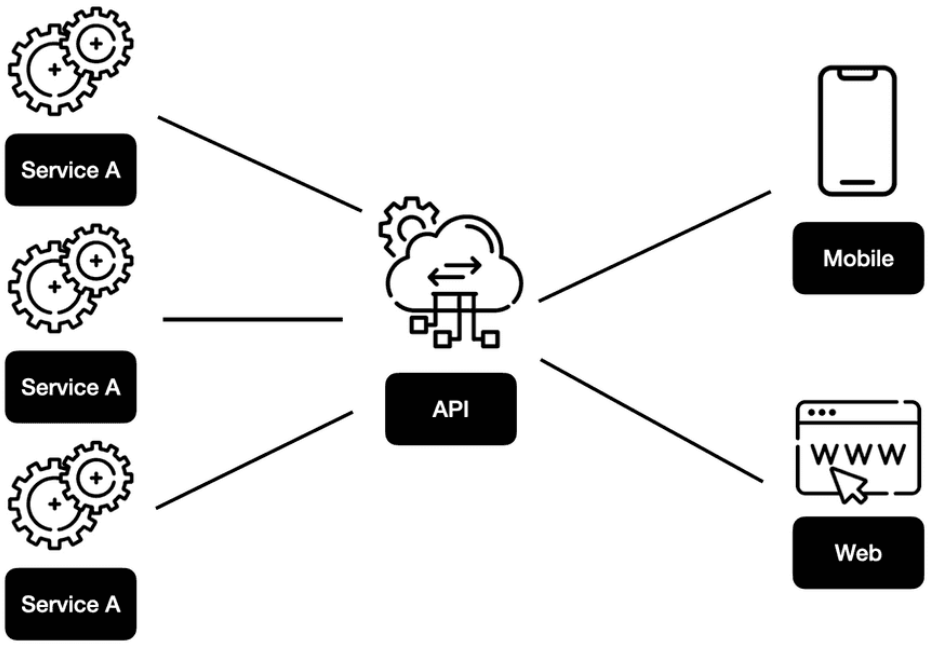
BFF를 사용하는 이유

- 각 프론트엔드의 요구사항이 다를 때 효율적으로 대응하기 위해서 사용한다.
- 공통 백엔드를 그대로 쓰면 불필요한 데이터까지 받아오거나, 응답 구조가 복잡해질 수 있으므로 이를 해결하기 위해 프론트엔드에 맞는 전용 백엔드를 만드는 것이다.

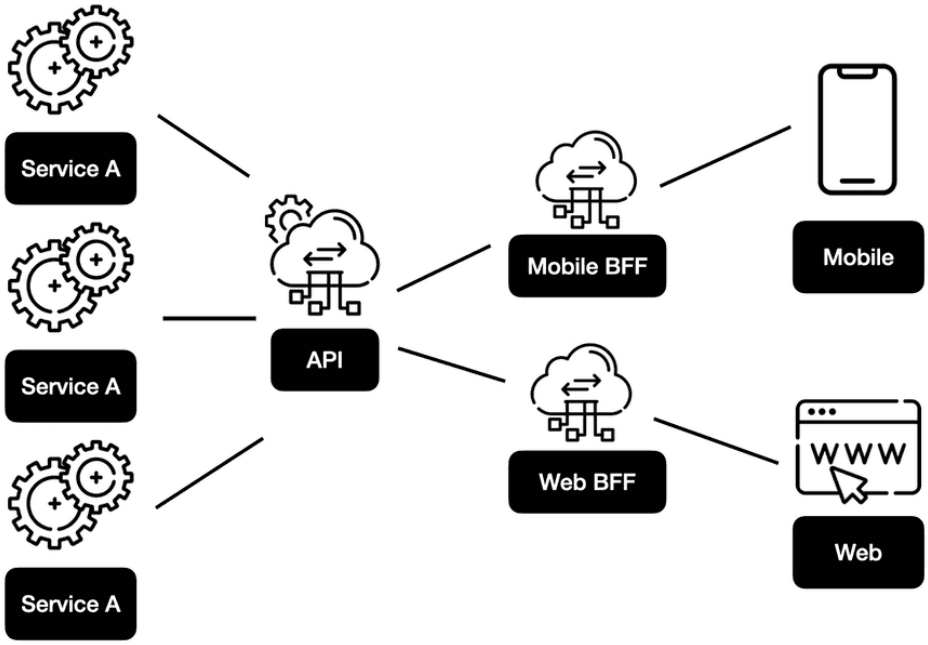
BFF를 어떻게 구성하는가?

- 웹용 BFF, 모바일용 BFF 와 같이 프론트 별로 별도의 서버 혹은 서비스를 둔다.
- BFF는 클라이언트의 요청을 받아 다른 마이크로 서비스나 백엔드와 통신하고, 필요한 형태로 데이터를 가공해서 프론트에게 전달한다.

일반적인 API 구조



BFF 구조



NextJS로 BFF가 가능한 이유

Next.js는 프론트엔드 프레임워크이면서, 동시에 **API Routes** 기능을 통해 **서버리스(Serverless)** 백엔드 API를 만들 수 있음.

- API Routes: Next.js 안에 API 서버처럼 쓸 수 있는 기능
- 서버리스: 요청이 있을 때만 실행되는 백엔드 함수

Next.js에서는 pages/api/*.js 또는 app/api/*/route.ts 경로에 파일을 만들면, 그게 API Endpoint가 되는데,Next.js 앱을 Vercel이나 AWS Lambda 같은 환경에 배포하면, 이 API Route 파일들은 서버를 계속 띄워놓는 게 아니라, 요청이 들어올 때마다 짧게 실행되고 사라지는 백엔드 함수처럼 동작하는데, 이를 서버리스 라고 한다.

BFF의 장점

- 프론트의 요구사항에 딱 맞는 API를 맞춤으로 제공
- 복잡한 백엔드의 로직을 숨기고, 클라이언트는 단순하게 사용할 수 있다.
- 보안, 인증, 응답 포맷 통합 등 중간 계층 역할을 수행할 수 있다.

BFF의 단점

- BFF 서비스가 많아지면, 운영, 관리 복잡도가 증가할 수 있다.
- 코드 중복 가능성이 생긴다. => 프론트엔드마다 별도의 BFF를 두기 때문에 비슷한 로직이 여러 곳에 흩어질 수 있기 때문이다.

출처

카카오페이지 BFF