초격차 패키지 Online.

안녕하세요
백엔드 개발자를 위한
DevOps 입문 강의를 진행할
이성미, 양재현입니다.

Chap1 | 백엔드 아키텍처 맛보기 (멀티모듈, MSA, DDD)

Chap2 | 백엔드 개발자를 위한 클라우드 인프라 운영 - Amazon

Chap3 | 백엔드 개발자를 위한 컨테이너 서비스 - Docker

Chap4 | 백엔드 개발자를 위한 빌드자동화 - Jenkins

Chap5 | [프로젝트] 웹 개발 프로젝트

Chap6 | 아키텍처 적용한 MVP 프로젝트 배포하기

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

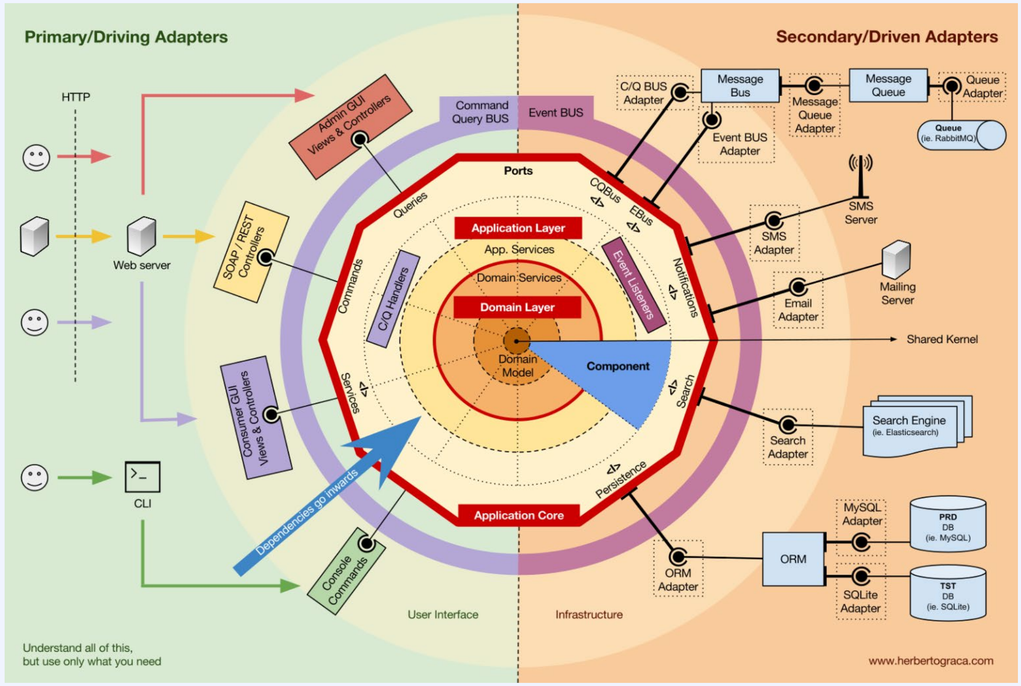
1 DDD(Domain Driven Design)이해하기
2 MSA 기반의 서비스 이해
3 멀티모듈로 구성하는 코드 (java base)
4 사례를 통해 보는 DDD, MSA, 멀티모듈

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

1 DDD(Domain Driven Design)이해하기

DDD란

DDDDomain Driven Design, 도메인 주도 설계란 2003년 에릭에반슨의 도서에서 처음 알려진 모델로, 도메인 설계나 개발 작업의 중심에 도메인 모델을 두고 반복적으로 변경 및 진화시켜서 프로그램 구현하는 개발 방법론



원문 : 도메인 주도 설계 첫걸음(위키북스)

비즈니스 도메인이란?

- 기업의 **주요 활동 영역**을 정의
- 회사가 고객에게 제공하는 서비스
  - Fedex ⇒ 배송서비스
  - StarBucks ⇒ 커피

- 기업은 여러 개의 도메인을 운영할 수 있다.
  - 아마존: 소매와 클라우드 모두 제공
  - 우버 : 차량공유회사이면서 음식 배달 및 자전거 공유 서비스도 제공
  - 노키아: 목제가공, 고무 제조, 통신, 모바일 등의 다양한 분야

원문 : 도메인 주도 설계 첫걸음(위키북스)

하위 도메인이란?

- 비지니스 도메인의 목표 달성을 위해서는 여러가지 하위 도메인을 운영해야 한다.
- 비지니스 할동의 세분화된 영역
- 하위 도메인은 고객에게 제공하는 서비스 단위로 비즈니스 도메인을 만든다. 각각의 하위 도메인은 회사의 비즈니스 도메인에서 목표를 달성하기 위해 서로 상호작용해야 한다.
  - 스터벅스
    - 비즈니스 도메인: 커피 판매
    - 하위 도메인: 부동산을 구매/임대, 직원 고용, 재정 관리 등

원문 : 도메인 주도 설계 첫걸음(위키북스)

비즈니스 도메인 적용 예(1/2)

FC 티켓 판매 및 유통회사는 모바일 앱을 통해 주변 공연을 찾아 추천해주는 사이트로 회원의 개인정보를 민감(암호화)하게 관리해야 하고 회원 각자가 참석했던 공연 정보를 보존해야 한다.

핵심 하위 도메인:

- 추천 엔진
- 데이터 익명화
- 모바일 앱

일반 하위 도메인:

- 암호화 - 모든 데이터 암호화
- 회계
- 정산
- 인증 및 권한 부여

지원 하위 도메인:

- 음악 스트리밍 서비스와 연동
- 소셜 네트워크와 연동
- 참석 공연 모듈

비즈니스 도메인 적용 예(2/2)

프로젝트 관리 애플리케이션은 특정회사에서 운영하는 여러가지 프로젝트를 통합 관리하는 프로그램으로 진행중인 프로젝트를 목록별로 기간, 진행사항, 투입인력, 비용을 관리합니다.

핵심 하위 도메인:

- **프로젝트 구분 관리**

일반 하위 도메인:

- WBS
- To Do
- 인력 관리
- 비용 관리

지원 하위 도메인:

- 캘린더 연동
- Notification 연동(slack)

도메인 전문가

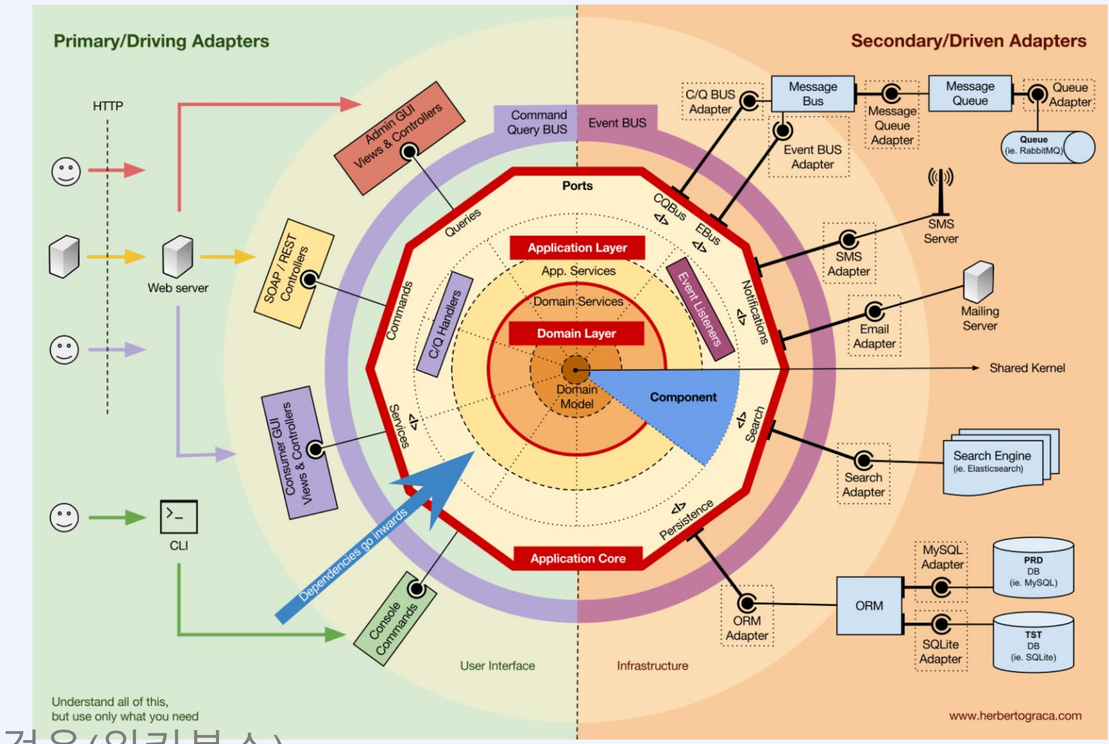
도메인 전문가는?

우리가 모델링하고 코드로 구현할 비즈니스의 모든 복잡성을 알고 있는 주제 전문가다.
도메인 전문가는 회사의 비즈니스 도메인 또는 하나 이상의 하위도메인에 대한 심층적인 지식을 가지고 있으며 프로젝트 성공에 매우 중요한 역할을 한다.

원문 : 도메인 주도 설계 첫걸음(위키북스)

소프트웨어 개발 환경의 DDD란

DDDDomain Driven Design, 도멘인 주도 설계란 2003년 에릭에반슨의 도서에서 처음 알려진 모델로, 도메인 설계나 개발 작업의 중심에 도메인 모델을 두고 반복적으로 변경 및 진화시켜서 프로그램 구현으로 이어지도록 하자는 것이다.



원문 : 도메인 주도 설계 첫걸음(위키북스)

DDD 적용 이유

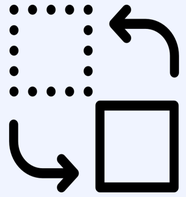
소프트웨어 개발 시 발생할 수 있는 복잡도

개발초기 1단계:
요구사항 추출 , 분석 , 설계 구현까지 진행
*처음에는 작은 단위로 설계

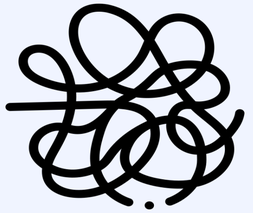
2단계:
계속되는 추가 요구사항이 발생
* 도메인이 복잡해짐

3단계:
도메인에 대한 이해 부족 상태에서 업무에 관련된 로직이 계속 추가되고 하나의 큰 덩어리로 얽히게 된다.

4단계:
하나의 큰 덩어리로 얽힌 서비스는 클래스 파악이 어렵고, 유지보수가 어려움.



빈번한 수정



유지보수 어려움

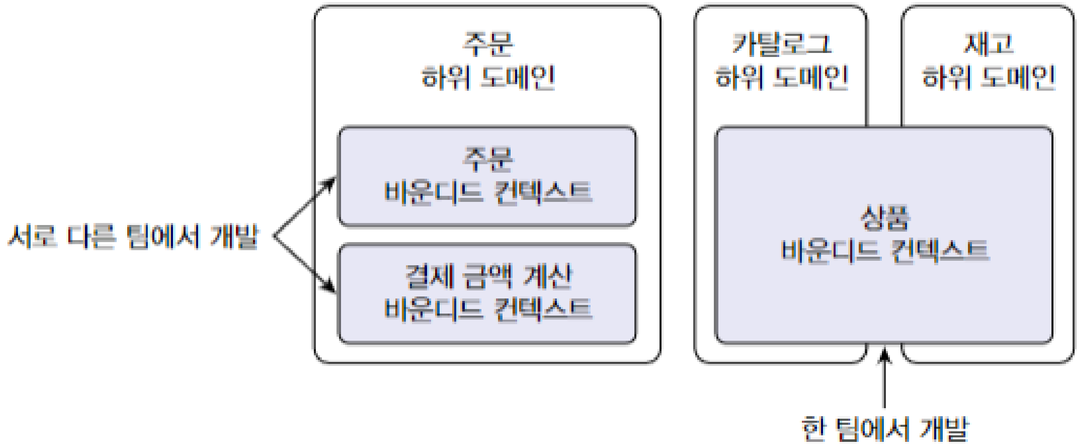
소프트웨어 프로젝트에서 지식 공유 흐름

- 소프트웨어 프로젝트에는 도메인 전문가, 프로젝트 소유자, 에지니어, UI/UX 디자이너, 프로젝트메니저, 테스터, 분석가 등 다양한 역할의 이해관계자의 협업 필요
- 이해관계자 간의 커뮤니테이션 문제를 없애고 상호 이해할 수 있고 모든 문서와 코드에 이르기까지 동일한 표현과 단어로 구성된 **유비쿼터스 언어** 즉, 단일화 된 언어체계를 구축

DDD 바운디드 컨텍스트

도메인은 여러 하위 도메인으로 구분되기 때문에 한 개의 모델로 여러 개의 하위 도메인을 표현하기 어려움.

각 하위 도메인은 명시적으로 구분되는 경계를 이용해 섞이지 않도록 하는데, 이렇게 구분되는 경계를 갖는 컨텍스트를 **바운디드 컨텍스트(bounded context)**라 함



도메인 주도 설계의 장단점

도메인 주도 설계의 장점	도메인 주도 설계의 단점
– S/W Life-Cycle 동안 용이한 커뮤니케이션 – 모듈화/캡슐화 기반 유연성 향상 – 현재 상황에 적합한 S/W 개발	– 도메인 전문가 참여 필수 요구 – 기존 도메인의 관행 개선 어려움 – 기술적으로 복잡한 프로젝트에 부적합

DDD 기반 프로젝트는
직관적인 도메인 연관 관계로 명확하게 설계 및 기능 구현이 가능하나,
도메인 전문성이 필수이며 프로젝트가 도메인에 종속되어 개선이 어려울 수 있으므로 도메인 전문가와의 적극적인 소통을 통해 효과적인 프로젝트 진행 필요가 있습니다.

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

2 MSA 기반의 서비스 이해

MSA(MicroService Architecture)란?

애플리케이션을 독립된 소프트웨어 컴포넌트, 즉 서비스로 분할하는 것

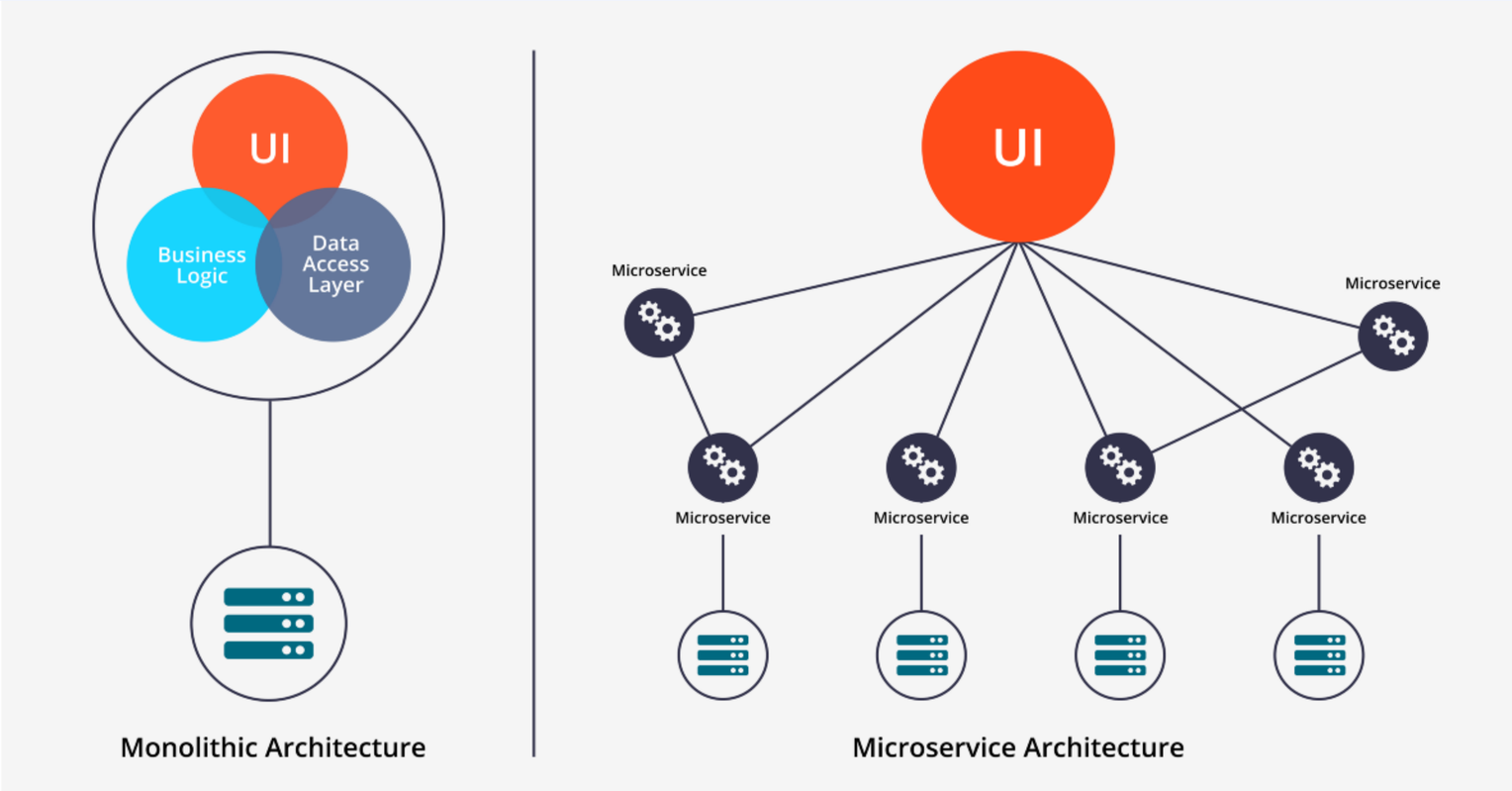
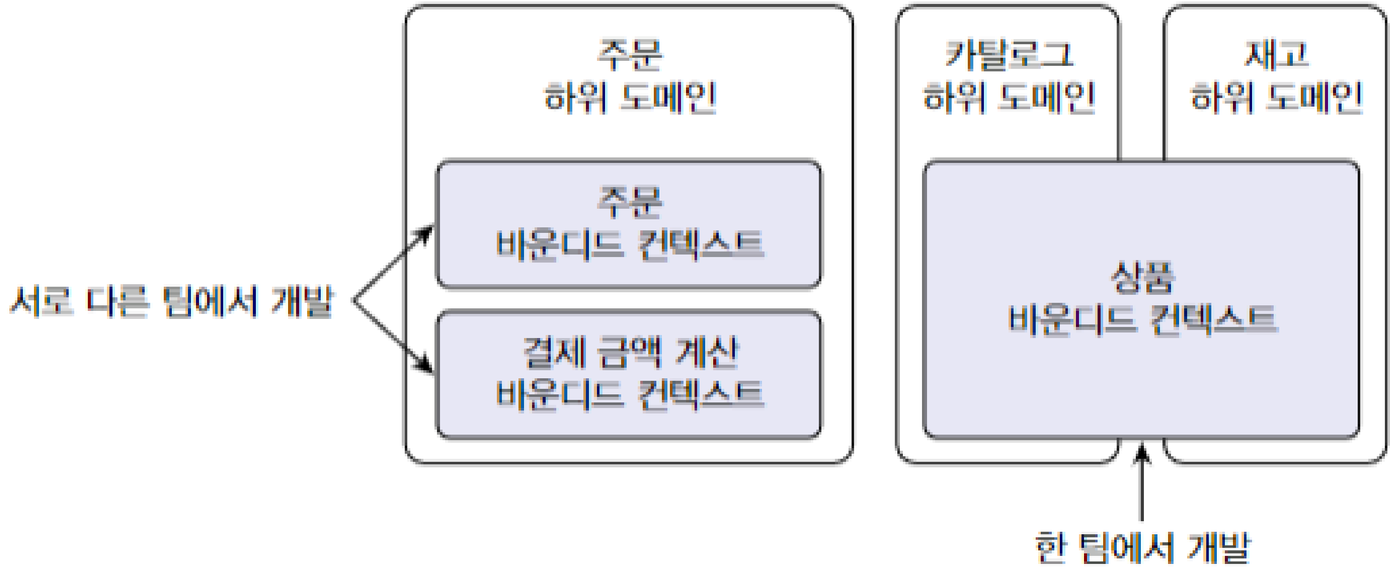


그림: https://www.google.co.kr/url?sa=i&url=https%3A%2F%2Fmedium.com%2Fhengky-sanjaya-blog%2Fmonolith-vs-microservices-b3953650dfd&psig=AOvVaw09NHLNeHXgCxAcDDkzdg9m&ust=1684400826975000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCIDSwpmA_P4CFQAAAAAdAAAAABAc

각 하위 도메인은 바운디드 컨텍스트(bounded context)로 구분하여 개발되고 이 구조가 일반적으로 MSA 구조가 될수 있음



프로젝트 관리 애플리케이션으로 확인하는 DDD

프로젝트 관리 애플리케이션은 특정회사에서 운영하는 여러가지 프로젝트를 통합 관리하는 프로그램으로 진행중인 프로젝트를 목록별로 기간, 진행사항, 투입인력, 비용을 관리합니다.

핵심 하위 도메인:

- **프로젝트 구분 관리**

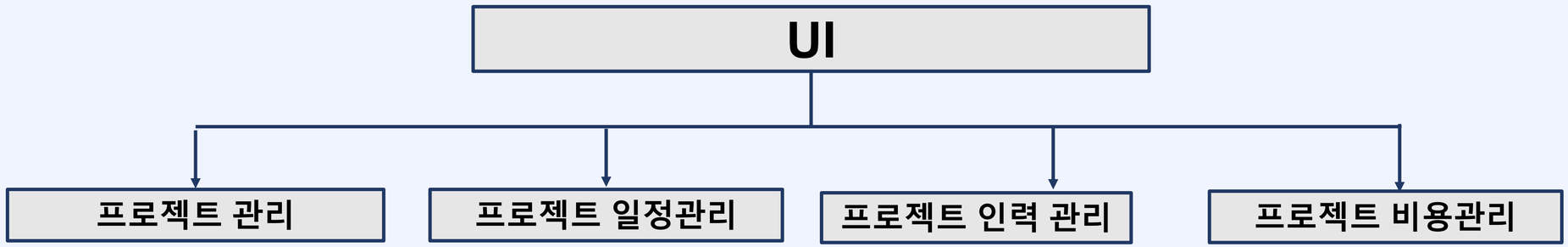
일반 하위 도메인:

- WBS
- To Do
- 인력 관리
- 비용 관리

지원 하위 도메인:

- 캘린더 연동
- Notification 연동(slack)

프로젝트 관리 애플리케이션 샘플 코드(시스템 아키텍처)



MSA 장점

배포 : 서비스별 개별 배포가 가능하고 특정 서비스의 요구사항 만을 반영하여 빠르게 배포 가능
확장 : 특정 서비스에 대한 확장성(scale-out)이 유리. 클라우드 기반 서비스 사용에 적합
장애처리 : 일부 장애가 전체 서비스로 확장될 가능성이 적고, 부분적으로 발생하는 장애에 대한 격리가 수월함
유연성 : 새로운 기술을 적용하기 유연하고 전체 서비스가 아닌 특정 서비스만 별도의 기술 또는 언어로 구현 가능
간단한 구조: 각각의 서비스에 대한 구조 파악 및 분석이 모놀리식 구조에 비해 간단

빠른 배포

Scale Out

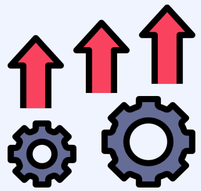
장애 최소화

시스템 유연함

MSA 단점

설계의 어려움 : MSA는 모놀리식에 비해 상대적으로 많이 복잡하고, 서비스가 모두 분산되어 있기 때문에 개발자는 내부 시스템의 통신을 어떻게 가져가야 할지 정해야함 . 또한 통신의 장애와 서버의 부하 등이 있을 경우 어떻게 transaction을 유지할지 결정하고 구현해야함.
성능적 이슈 : 서비스 간 호출 시 API를 사용하므로, 통신 비용이나 Latency에 대해 이슈가 존재함
테스트/데이터 트랜잭션 : 모놀리식에서는 단일 트랜잭션을 유지하면 됐지만 MSA에서는 비즈니스에 대한 DB를 가지고 있는 서비스도 각기 다르고, 서비스의 연결을 위해서는 통신이 포함되기 때문에 트랜잭션을 유지하는것이 어려움
통합 테스트 어려움: 개발 환경과 실제 운영환경을 동일하게 가져가는 것이 어려움
데이터 관리 : 데이터가 여러 서비스에 분산되어 있어 조회하거나 관리하기 어려움

설계어려움



성능

트랜잭션

데이터관리

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

1 DDD(Domain Driven Design)이해하기
2 MSA 기반의 서비스 이해
3 멀티모듈로 구성하는 코드 (java base)
4 사례를 통해 보는 DDD, MSA, 멀티모듈

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

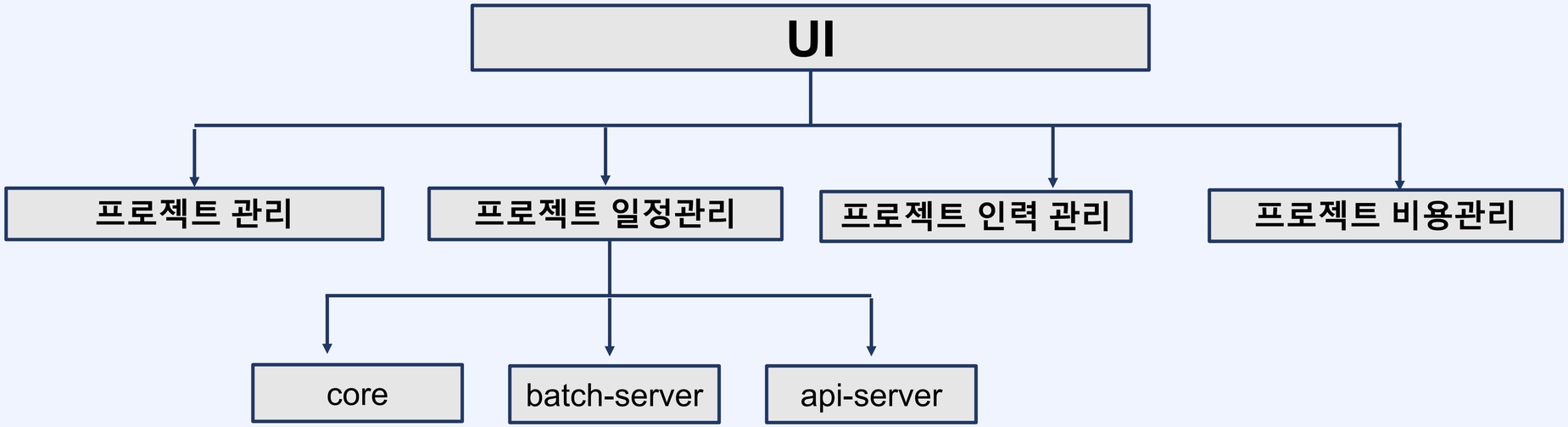
3 멀티모듈로 구성하는 코드(java base)

멀티 모듈이란?

이미지 링크:
http://m.hobbybox.co.kr/product/detail.html?product_no=5344
https://media.bunjang.co.kr/product/213813252_2_1681433618_w360.jpg

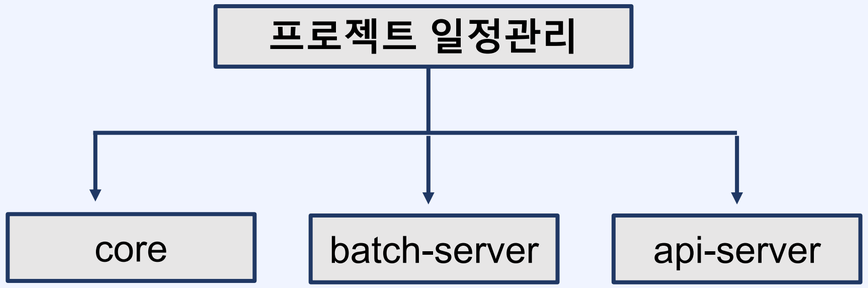
멀티 모듈이란?

- **모듈**이란 독립적으로 배포될 수 있는 코드의 단위
- **멀티 모듈**이란 상호 연결된 여러 개의 모듈로 구성된 프로젝트.
  - 각 모듈은 독립적으로 빌드
  - 멀티모듈 적용시 독립된 코드로 빌드되어 확장 및 유지보수가 용이



프로젝트 일정관리에서 구현한 멀티모듈

- batch-server 반복적으로 처리하는 루틴한 서비스
- api-server REST API(CRUD)를 구현하는 서비스
- core batch-server, api-server에서 공통적으로 사용하는 코드

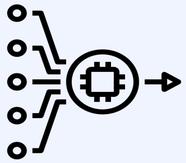


멀티 모듈 코드의 장단점

멀티모듈의 장점
- 재사용과 공유
- 빌드가 쉽고 시간 단축
- 변경으로 인한 영향 최소화
- 의존성을 최소화


재사용


코드 공유


빌드시간 단축

멀티모듈의 단점
- 멀티모듈에 대한 이해 필요
- 여러 모듈을 유지 관리 어려움

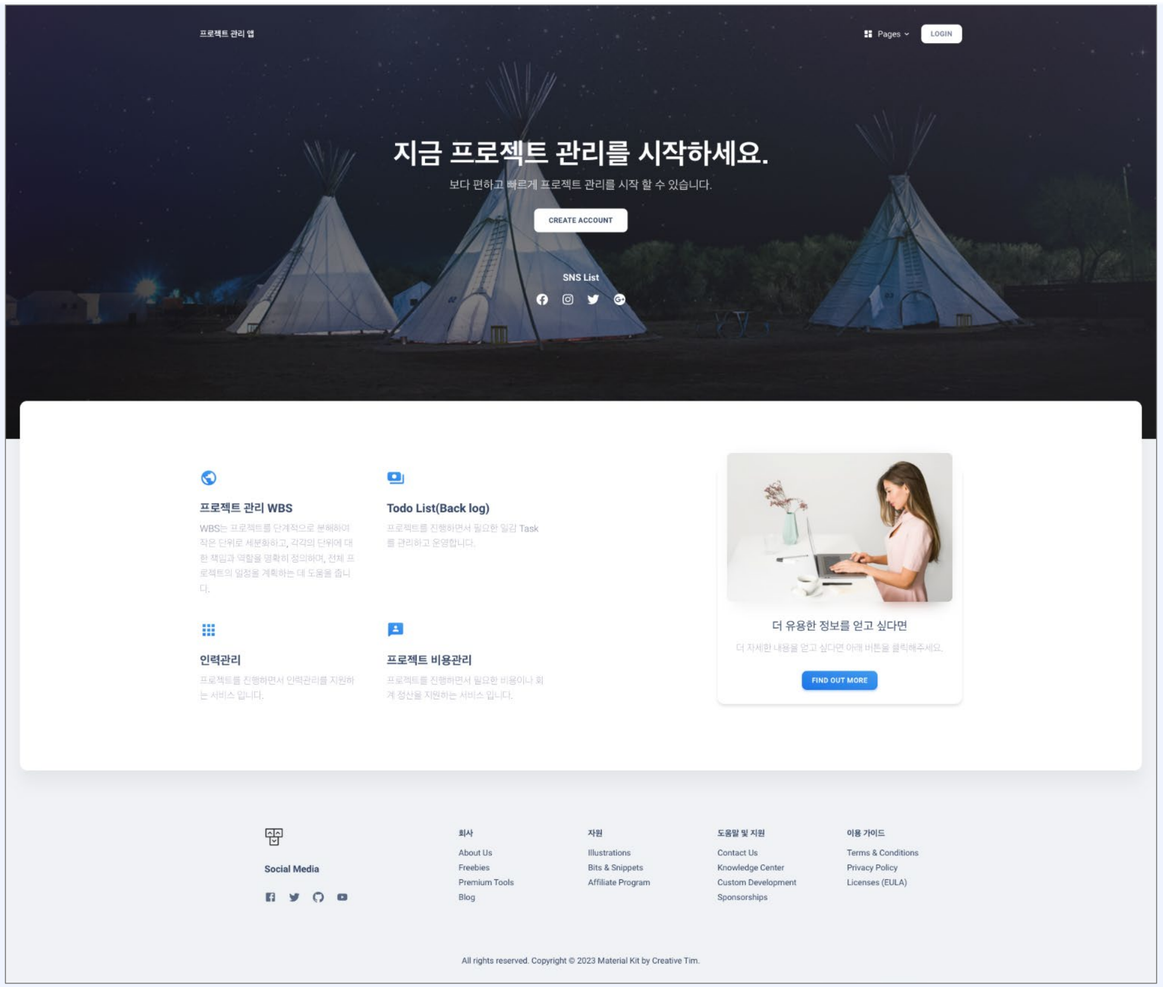
크기가 큰 프로젝트에서 멀티모듈은 필수이나,
크기가 작은 프로젝트에서는 멀티모듈의 장점을 살릴수 없다.

원본: https://hudi.blog

백엔드 아키텍처 맛보기 (DDD,MSA,멀티모듈)

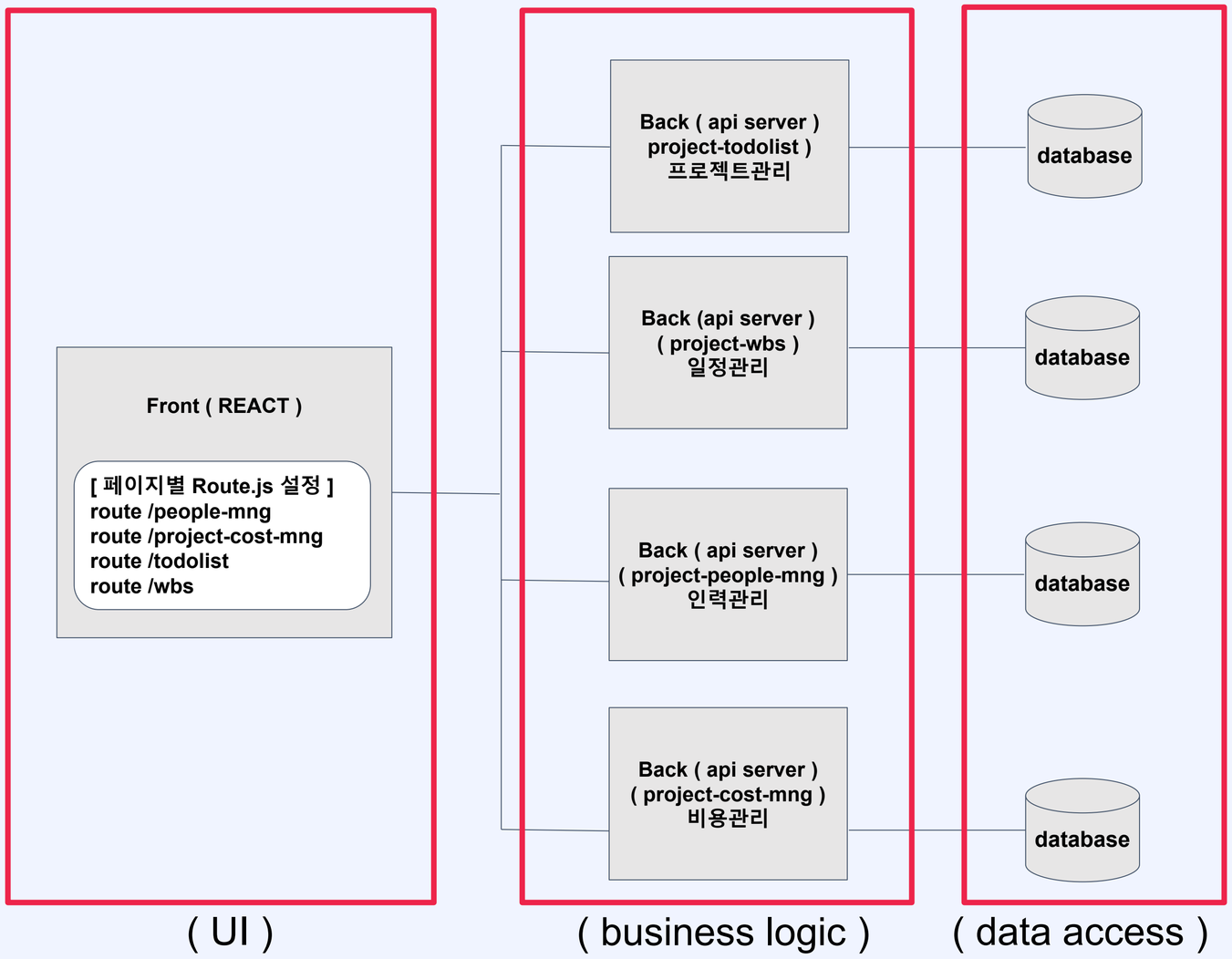
4 사례를 통해 보는 DDD, MSA, 멀티모듈

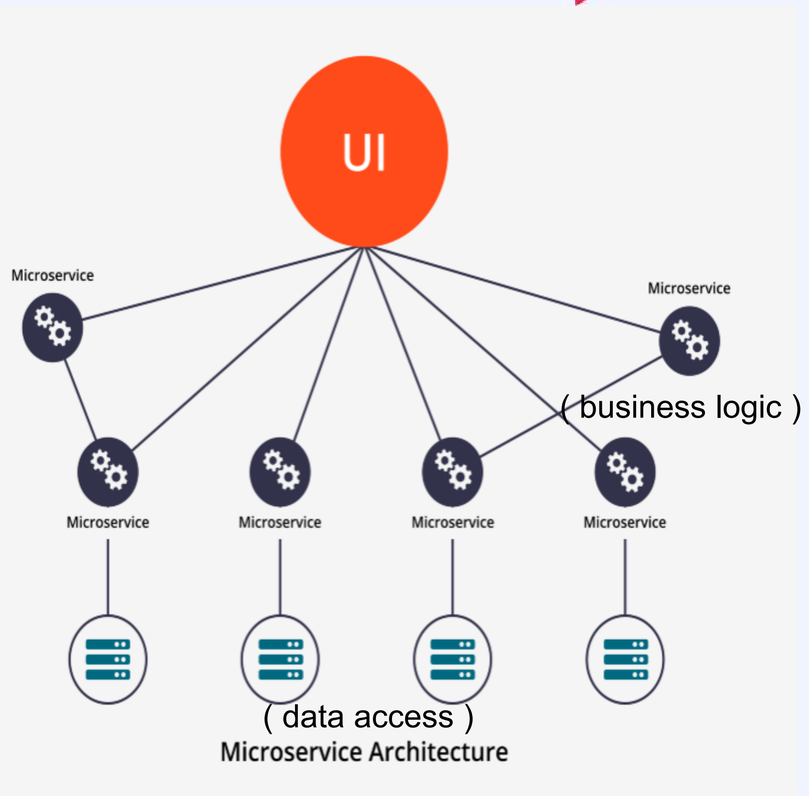
프로젝트 관리 애플리케이션 샘플 코드(메인화면)



https://github.com/azjaehyun/fc-study/tree/main/chapter-1

프로젝트 관리 애플리케이션 샘플 코드(시스템 아키텍처)





미래가 오늘이다.