현재 시스템 구조

- 단일 모놀리식 애플리케이션
- 기능 : 잔액충전, 콘서트 조회, 콘서트 일정 조회, 좌석 조회, 좌석 예약, 결제

MSA 형태로 서비스 분리

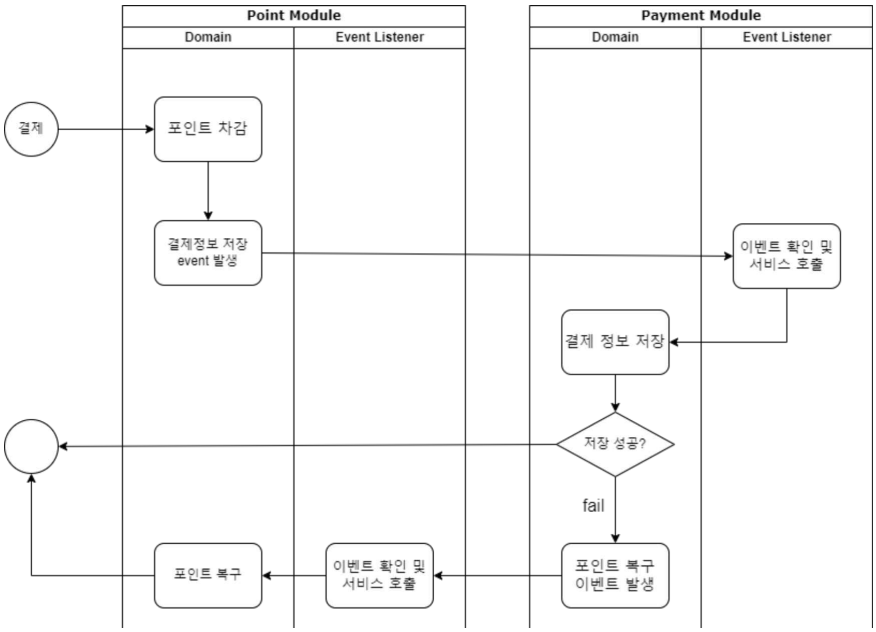
- 유저 서비스 : 유저 정보 관리
- 포인트 서비스 : 잔액 관리
- 콘서트 서비스 : 콘서트 정보 및 좌석 관리
- 결제 서비스 : 결제 관리
- 대기열 서비스 : 대기열 관리

트랜잭션 처리의 한계

위처럼 서비스를 분리하면 확장성이나 유지보수 면에서 장점을 얻을 수 있지만 서로 참조가 안되기 때문에 생기는 단점이 있습니다.
결제 기능 같이 여러 서비스를 트랜잭션을 묶어 처리하던 기능을 따로 처리를 해줘야 하며, 데이터의 일관성에도 문제가 생길 수 있습니다.

트랜잭션 처리 방안

위에 문제점은 Saga 패턴을 사용하여 해결할 수 있습니다.
Saga패턴이란, 각 서비스가 각자의 트랜잭션을 수행하고, 다음 동작할 서비스에 이벤트를 보내는 형태의 패턴으로, 실패 시 보상 트랜잭션을 통해 이전 상태로 복구하는 형태입니다.
여러 서비스를 사용하는 결제 기능을 예시로 보겠습니다.



위와 같이 서로 어떠한 참조도 없는 상태로 Event 를 통해서만 동작을 하도록 되어있습니다.