# 구매하기 구현

## 1 UI 및 기능

# 주문의 요구사항

- 단일 상품의 주문 가능
- 여러 상품의 묶음 주문도 가능

개별 상품 주문 정보를 담는

OrderItem 테이블과

실제 주문 정보를 담는

Order 테이블을 만들자

OrderItem은 Cart와 같은 구조이면 됨

- userId
- productId
- quantity
- price
- amount
- (할인정보)

Cart와 OrderItem에 들어가는 amount에 대해서

Cart에서는 price가 안중요
quantity만 있으면 amount 계산해서 보여줄 수 있지만(계산 편의를 위해 담음)

OrderItem에는 생성하는 당시 할인 정보가 담긴 price 저장이 필요

Orders는

orderItemIds: string

receiver: string

address: string

phone: string

createdAt: Date

status: number (주문취소 -1, 주문대기0,  결제대기1, 결제완료2, 배송대기3, 배송중4, 배송완료5, 환불대기6, 환불완료7, 반품대기8, 반품완료9)

Table 생성

```
model OrderItem {
  id Int @id @default(autoincrement())
  productId Int
  price Int
  quantity Int
  amount Int
}
```

```
model Orders {
  id Int @id @default(autoincrement())
  userId String
  orderItemIds String
  receiver String?
  address String?
  phoneNumber String?
  createdAt DateTime @default(now())
  status Int
}
```

주문하기 진입점

- 상품상세(생성)
- 장바구니(생성)
- my 주문내역(조회 및 업데이트)

페이지

- 주문 목록 페이지(my 에 넣자)
- (스킵) 주문 상세 페이지(받는이, 주소, 전화번호 입력)

결제는 따로 구현하지 않음

만약 구현하고 싶다면,
온라인 결제 연동 서비스들 참고

토*페이먼츠 / 네**페이 / 카**페이 / Pay*** 등..

온라인 결제 연동 서비스들은

웹사이트가 고객이기에 최대한 수월하게 API를 연동할 수 있도록

공식문서와 API가 잘 되어있음

(마치 OAuth 연동하듯이..)

# api 구현

add-order: post orderItem 들을 생성하고 createMany or create

get-order: order 목록을 조회(필요하다면 filter로 status 별)

update-order: orderItem의 option을 변경하여 order까지 업데이트하거나 order의 status를 업데이트 (아마도 admin 처럼 기능을 주면 될듯)

/my 주문내역 페이지

일자순으로 주문내역을 나열

Cart와 유사하지만 Cart들을 묶어서 하나의 주문으로 노출 필요

주문에서는 상태를 노출해줄 필요 있음

결제는 구현 안할 것

대신 X(주문취소), 결제완료 처리 기능을 넣자

# 정리

## 구매하기 UI 및 기능 구현

| | |
|---|---|
| 요구사항 | 단건 주문 / 다건 주문 |
| 테이블 3개 | 3중 JOIN |
| 주문 생성 | 상품상세 / 장바구니 |
| 주문 내역 | 마이페이지 |