4

TO‘RTINCHI MA’RUZA

Korxona ilovalarini dasturlash · 4-mavzu

Biznes-logika va servis qatlami

Servis qatlami · Tranzaksiyalar · Qulflash · Invariantlar · Hujjat oqimi

Servis qatlami | DB::transaction | lockForUpdate | Invariantlar | Holat mashinasi

ISINISH

Omborda `UN-001` dan 5 dona qoldi. Ikki menejer bir soniyada 4 va 3 donalik buyurtmani rezerv qilsa, nima bo‘ladi?

Hozircha javob bermang. Har bir menejerning so‘rovi serverda qanday qadamlar bilan bajarilishini tasavvur qiling.

1

Ikkala so‘rov ham “mavjud qoldiq 5 ta” deb o‘qisa, ikkalasi ham rezervga ruxsat beradimi?

2

Rezerv yozilgandan keyin hisob-faktura yaratishda xato chiqsa, rezerv qolib ketadimi?

3

Bunday xatoni kim va qachon sezadi: dasturchi, omborchi yoki buxgalter?

O‘QUV MAQSADLARI

Bu ma’ruza oxirida siz quyidagilarni bajara olasiz…

1. **Tushuntirish** — biznes-logika nima uchun controller yoki modelda emas, servis qatlamida yashashini
2. **Qo‘llash** — `DB::transaction()` ni bir nechta jadvalga yozadigan biznes-amallarda
3. **Farqlash** — pessimistik (`lockForUpdate`) va optimistik (version, shartli UPDATE) qulflashni
4. **Aniqlash** — poyga holati (race condition) va deadlock xavfi bor kod bo‘laklarini
5. **Himoyalash** — invariantlarni uch qatlamda: servis tekshiruvi, domen istisnosi va baza CHECK cheklovi
6. **Loyihalash** — buyurtma → rezerv → hisob-faktura → jo‘natish hujjat oqimini va ombor harakatlari jurnalini
7. **Yozish** — `OrderStatus` enumi asosida holat mashinasini va taqiqlangan o‘tishlarni rad etishni
8. **Hisoblash** — QQS va jami summani butun tiyinda, `float` siz

01

Biznes-logika qayerda yashaydi

Semiz controller · servis qatlami · use-case metodlari · qayta ishlatish

Servis qatlami | Controller | Use-case | DI

QAYERGA YOZAMIZ?

Semiz controller va servis qatlami

Noto‘g‘ri: hamma narsa controllerda

- `OrderController::reserve()` qoldiqni o‘qiydi, tekshiradi, `stocks` ni yangilaydi
- API controllerida xuddi shu kod nusxalanadi
- Importda yoki fon vazifasida bu qoidani chaqirib bo‘lmaydi
- Test uchun HTTP so‘rov, sessiya, CSRF kerak

To‘g‘ri: yupqa controller + servis

- Controller: kirish → `$orders->reserve($order)` → javob
- Qoida bitta joyda: `OrderService`, `StockService`
- Web, API, import, job, artisan — bir xil metodni chaqiradi
- Servisni to‘g‘ridan-to‘g‘ri unit/feature testda chaqirish mumkin

Model — ma’lumot va kichik qoidalar (accessor, cast, munosabatlar); servis — bir nechta modelni o‘zgartiradigan jarayon.

YUPQA CONTROLLER

Controller faqat chaqiradi va javob beradi

```php
use App\Models\Order;
use App\Services\OrderService;
use Illuminate\Http\RedirectResponse;

final class OrderController extends Controller
{
    public function reserve(
        Order $order,
        OrderService $orders,
    ): RedirectResponse {
        $orders->reserve($order);

        return redirect()
            ->route('orders.show', $order)
            ->with('status', 'Buyurtma rezerv qilindi');
    }
}
```

▶ NATIJA

```
POST /orders/124/reserve
→ 302 Found, Location: /orders/124
→ flash: “Buyurtma rezerv qilindi”
```

`Order $order`

route model binding: `{order}` parametridan model topiladi

`OrderService $orders`

servis konteyneri uni bog‘liqliklari bilan yaratadi

`$orders->reserve()`

barcha qoida va tranzaksiya — servis ichida

Qoldiq yetmasa?

servis istisno tashlaydi; uni global handler xabarga aylantiradi (4-bo‘lim)

APP/SERVICES

OrderService — use-case metodlari

```php
namespace App\Services;

final class OrderService
{
    public function __construct(
        private readonly StockService $stock,
        private readonly InvoiceService $invoices,
    ) {}

    public function create(OrderData $data): Order { /* 2-mavzu */ }
    public function confirm(Order $order): Order { /* ... */ }
    public function reserve(Order $order): Order { /* ... */ }
    public function invoice(Order $order): Invoice { /* ... */ }
    public function ship(Order $order): Order { /* ... */ }
    public function cancel(Order $order): Order { /* ... */ }
}
```

private readonly

PHP 8.1 konstruktor xossalari: bog‘liqlik bir marta beriladi

Bitta metod — bitta amal

`reserve()` faqat rezerv qiladi, `invoice()` faqat hisob-faktura yaratadi

StockService

qoldiqni faqat u o‘zgartiradi — invariantlar bir joyda

▶ NATIJA

```
app/Services/OrderService.php    — jarayonni boshqaradi
app/Services/StockService.php    — stocks + stock_movements
app/Services/InvoiceService.php  — invoices, QQS
```

Qaytariladigan qiymat

yangilangan model; xato bo‘lsa — domen istisnosi

QOIDALAR

Yaxshi servisning 5 belgisi

1 HTTP ni bilmaydi

`Request`, `session()`, `redirect()` yo‘q — kirish: model yoki DTO.

2 Tranzaksiya chegarasi

Bir amalning barcha yozuvlari bitta `DB::transaction` ichida.

3 Domen istisnolari

Xato holat — `InsufficientStockException`, `false` yoki `null` emas.

4 Holatsiz (stateless)

Xossalarda faqat bog‘liqliklar; so‘rov ma’lumoti metod parametrida.

5 Qayta ishlatiladi

Web, API, import, job va testdan bir xil chaqiriladi.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

1-bo‘lim yakuni: Biznes-logika qayerda yashaydi

1

Nima uchun rezerv qoidasi controllerda emas, servisda yozilishi kerak?

Savol 1

2

Servis metodiga nima uchun `Request` obyektini uzatmaymiz?

Savol 2

3

`OrderService` qoldiqni to‘g‘ridan-to‘g‘ri yangilashi kerakmi?

Savol 3

02

Tranzaksiyalar: hammasi yoki hech narsa

ACID · DB::transaction · qayta urinish · commit dan keyingi hodisalar

ACID | DB::transaction | ROLLBACK | afterCommit

ACID

Tranzaksiya kafolatlari

Tranzaksiya — bir nechta SQL buyrug‘ini bitta bo‘linmas birlikka birlashtirish: `BEGIN` … `COMMIT` yoki `ROLLBACK`.

Atomarlik

Rezerv, jurnal va holat — uchalasi ham saqlanadi yoki birortasi ham saqlanmaydi.

Izchillik

CHECK, FOREIGN KEY, UNIQUE cheklovlari buzilgan holat COMMIT bo‘lmaydi.

Izolyatsiya

Parallel so‘rov yarim tayyor o‘zgarishni ko‘rmaydi (PostgreSQL: `READ COMMITTED`).

Chidamlilik

COMMIT dan keyin ma’lumot disk/WAL da — server qayta ishga tushsa ham saqlanadi.

Izolyatsiya avtomatik ravishda “poyga”dan himoya qilmaydi: `READ COMMITTED` da ikki tranzaksiya bir xil qiymatni o‘qiy oladi (3-bo‘lim).

ROLLBACK g‘oyasi: nusxa va qaytarish

```php
$stock = ['UN-001' => 10, 'UN-002' => 3];
function transaction(array &$db, callable $work): void
{
    $backup = $db;                          // BEGIN
    try {
        $work($db);                         // COMMIT
    } catch (DomainException $e) {
        $db = $backup;                      // ROLLBACK
    }
}
transaction($stock, function (array &$db) {
    $db['UN-001'] -= 4;
    if ($db['UN-002'] < 5) {
        throw new DomainException('UN-002 yetarli emas');
    }
    $db['UN-002'] -= 5;
});
echo json_encode($stock);
```

▶ NATIJA

```
{"UN-001":10,"UN-002":3}
```

`$backup = $db`

baza ham ichki tarzda eski holatni saqlaydi (MVCC)

`$db['UN-001'] -= 4`

birinchi o‘zgarish bajarildi…

`throw`

…lekin ikkinchi qadam xato — hammasi bekor

Natija

`UN-001` yana 10: yarim tayyor holat qolmadi

DB::transaction — closure va qayta urinish

```php
use Illuminate\Support\Facades\DB;

public function confirm(Order $order): Order
{
    $order = DB::transaction(function () use ($order) {
        $order = Order::lockForUpdate()->findOrFail($order->id);
        $this->moveTo($order, OrderStatus::Confirmed);
        $order->update(['total' => $order->items()->sum('total')]);

        return $order;                    // COMMIT
    }, attempts: 3);                      // deadlock — qayta urinish

    OrderConfirmed::dispatch($order);     // COMMIT dan keyin
    return $order;
}
```

▶ NATIJA

```
BEGIN
select * from "orders" where "id" = 124 limit 1 for update
update "orders" set "status" = 'confirmed', ... where "id" = 124
update "orders" set "total" = 2650000.00, ... where "id" = 124
COMMIT
```

`DB::transaction(fn)`

istisno bo‘lsa — ROLLBACK va istisno qayta tashlanadi

`attempts: 3`

faqat deadlock/parallellik xatosida qayta bajaradi

`return $order`

closure qaytargan qiymat `transaction()` natijasi

dispatch tashqarida

yoki hodisa `ShouldDispatchAfterCommit` ni amalga oshiradi

TUZOQLAR

Tranzaksiyadagi keng tarqalgan xatolar

Xato	Oqibat	To‘g‘ri yo‘l
Closure ichida istisnoni yutish (`catch` + log)	ROLLBACK yo‘q, yarim ma’lumot saqlanadi	Log yozing va `throw $e;` qiling
Tranzaksiya ichida HTTP/SMS/e-mail	Qulflar uzoq ushlanadi, ROLLBACK SMS ni qaytarmaydi	Tashqi chaqiruv — COMMIT dan keyin yoki navbatda
Hodisa/job commit dan oldin	Tinglovchi hali yo‘q buyurtmani o‘qiydi	`ShouldDispatchAfterCommit`, `->afterCommit()`
Har bir yozuv alohida tranzaksiyada	Atomarlik yo‘q	Bitta use-case — bitta tashqi tranzaksiya
Juda uzun tranzaksiya (import 10 000 qator)	Qulflar, deadlock, xotira	Bo‘laklab (`chunk`) qayta ishlash

Ichma-ich `DB::transaction` savepoint yaratadi; tashqi ROLLBACK ichkisini ham bekor qiladi.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

2-bo‘lim yakuni: Tranzaksiyalar: hammasi yoki hech narsa

1

`DB::transaction` closure ichida istisno tashlansa, nima bo‘ladi?

Savol 1

2

`attempts: 3` qaysi holatda qayta urinadi?

Savol 2

3

Nima uchun SMS yuborishni tranzaksiya ichida qilmaslik kerak?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · Oson · 10 ball

1-topshiriq: Hammasi yoki hech narsa

Ombor qoldig‘i `['SKU' => son]` massivida, amallar esa `[sku, delta]` ro‘yxatida berilgan (`delta` musbat — kirim, manfiy — chiqim). Amallarni ketma-ket qo‘llang. Agar SKU qoldiqda bo‘lmasa yoki **biror amaldan keyin** qoldiq manfiy bo‘lsa — tranzaksiya bekor: asl qoldiq o‘zgarmagan holda qaytariladi va xato bergan SKU ko‘rsatiladi. Hamma amal muvaffaqiyatli bo‘lsa — yangi qoldiq.

Imzo: `function atomar(array $qoldiq, array $amallar): array`

Qaytaradi: `['ok' => bool, 'qoldiq' => array, 'xato' => ?string]` — muvaffaqiyatda `xato = null`, ROLLBACK da `qoldiq` — asl massiv.

MISOL · CHAQIRUV

```
atomar(['UN-001' => 10, 'UN-002' => 3], [['UN-001', -4], ['UN-002', -2]])
```

MISOL · NATIJA

```
{"ok":true,"qoldiq":{"UN-001":6,"UN-002":1},"xato":null}
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

03

Parallel so‘rovlar va qulflash

Poyga holati · lockForUpdate · shartli UPDATE · optimistik qulf · deadlock

Race condition

lockForUpdate

Optimistik qulf

Deadlock

Qulfsiz rezerv: ikki so‘rov — manfiy qoldiq

```php
$stock = ['quantity' => 5, 'reserved' => 0];

// Ikki menejer bir vaqtda o‘qiydi (qulf yo‘q)
$a = $stock['quantity'] - $stock['reserved'];   // A ko‘radi: 5
$b = $stock['quantity'] - $stock['reserved'];   // B ko‘radi: 5

if ($a >= 4) {
    $stock['reserved'] += 4;                    // A rezerv qiladi
}
if ($b >= 3) {
    $stock['reserved'] += 3;                    // B ham!
}
echo "reserved={$stock['reserved']}, mavjud=",
    $stock['quantity'] - $stock['reserved'];
```

▶ NATIJA

```
reserved=7, mavjud=-2
```

`$a va $b`

ikkala so‘rov ham COMMIT dan oldingi bir xil qiymatni o‘qidi

`if ($b >= 3)`

tekshiruv eskirgan ma’lumotga asoslangan

`mavjud=-2`

invariant buzildi: 2 dona “havodagi” tovar sotildi

PESSIMISTIK QULF

lockForUpdate — ikkinchi so‘rov kutadi

```php
DB::transaction(function () use ($warehouseId, $productId, $qty) {
    $stock = Stock::where('warehouse_id', $warehouseId)
        ->where('product_id', $productId)
        ->lockForUpdate()               // SELECT ... FOR UPDATE
        ->firstOrFail();

    // B shu yerda A ning COMMIT ini kutib turadi
    $available = $stock->quantity - $stock->reserved;
    if ($available < $qty) {
        throw new InsufficientStockException(
            $productId, $qty, $available);
    }
    $stock->increment('reserved', $qty);
});
```

▶ NATIJA

```
A: select * from "stocks" where ... limit 1 for update   → 5
B: select ... for update                                  → kutadi…
A: update "stocks" set "reserved" = "reserved" + 4 ...; COMMIT
B:                                                        → 1
B: InsufficientStockException → ROLLBACK
```

`lockForUpdate()`

qator COMMIT/ROLLBACK gacha boshqalar uchun yopiq

`DB::transaction`

qulf faqat tranzaksiya ichida ishlaydi

`increment()`

`reserved = reserved + ?` — baza tomonida hisoblanadi

`sharedLock()`

`FOR SHARE`: o‘qishga ruxsat, o‘zgartirishga yo‘q

QULFSIZ USULLAR

Shartli UPDATE va optimistik qulf

```php
// 1) Tekshiruv va o‘zgartirish — bitta atomar UPDATE
$affected = Stock::query()
    ->where('warehouse_id', $warehouseId)
    ->where('product_id', $productId)
    ->whereRaw('quantity - reserved >= ?', [$qty])
    ->increment('reserved', $qty);

if ($affected === 0) {
    throw new InsufficientStockException($productId, $qty);
}

// 2) Optimistik qulf: `version` ustuni (qo‘shsangiz)
$ok = Stock::whereKey($stock->id)
    ->where('version', $stock->version)
    ->update(['reserved' => $stock->reserved + $qty,
              'version' => $stock->version + 1]);
// $ok === 0 — kimdir oldinroq yozgan: qayta o‘qing
```

▶ NATIJA

```
update "stocks" set "reserved" = "reserved" + 4
  where "warehouse_id" = 1 and "product_id" = 7
  and quantity - reserved >= 4
→ 1 qator (muvaffaqiyat) yoki 0 qator (yetarli emas)
```

`whereRaw(..., [$qty])`

qiymat binding orqali — SQL injection yo‘q

`increment() → int`

Query Builder yangilangan qatorlar sonini qaytaradi

`version`

Laravel da tayyor optimistik qulf yo‘q — shart bilan yoziladi

Qachon qaysi?

issiq qator (qoldiq) — pessimistik/shartli; kam to‘qnashuv — optimistik

Qulflar tartibi

Noto‘g‘ri: tasodifiy tartib

- A: `UN-001` ni qulfladi, `UN-002` ni kutmoqda
- B: `UN-002` ni qulfladi, `UN-001` ni kutmoqda
- PostgreSQL: `deadlock detected` (40P01) — biri to‘xtatiladi
- Foydalanuvchi tasodifiy “500” xatosini ko‘radi

To‘g‘ri: doim bir xil tartib

- Qatorlar `product_id` bo‘yicha saralanadi: `orderBy('product_id')`
- A va B ikkalasi ham avval `UN-001` ni so‘raydi — B shunchaki kutadi
- `attempts: 3` — kamdan-kam deadlockda avtomatik qayta urinish
- Tranzaksiya qisqa: hisob-kitob oldin, qulf va yozish oxirida

Qoida: bir nechta qatorni qulflaydigan har bir kod qulflarni bitta, hujjatlashtirilgan tartibda oladi.

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

3-bo‘lim yakuni: Parallel so‘rovlar va qulflash

1

Nima uchun `lockForUpdate()` tranzaksiyasiz foyda bermaydi?

Savol 1

2

Shartli UPDATE da qoldiq yetmaganini qanday bilamiz?

Savol 2

3

Deadlockning oldini olishning eng oddiy qoidasi qaysi?

Savol 3

04

Invariantlar va domen istisnolari

Qoldiq qoidalari · domen istisnolari · pul tiyinda · CHECK cheklovi

Invariant | DomainException | CHECK | Tiyin

Qoldiq invariantlari

Har bir amaldan keyin quyidagilar rost bo‘lishi shart. Bittasi buzilsa — tranzaksiya ROLLBACK.

1 quantity ≥ 0

Omborda manfiy tovar bo‘lmaydi.

2 0 ≤ reserved ≤ quantity

Rezerv bor tovardan oshmaydi: mavjud = quantity − reserved ≥ 0.

3 Rezerv sharti

`reserve(n)` faqat `mavjud ≥ n` bo‘lsa.

4 Jo‘natish

`out(n)`: `quantity -= n`, `reserved -= n` — birga.

5 Jurnal

Har bir o‘zgarish — bitta `stock_movements` yozuvi.

6 Jami summa

`orders.total` = qatorlar `total` yig‘indisi.

Invariantni qo‘riqlaydigan obyekt

```php
final class StockRow
{
    public function __construct(public int $qty, public int $res) {}
    public function available(): int
    {
        return $this->qty - $this->res;
    }
    public function reserve(int $n): void
    {
        if ($n <= 0 || $n > $this->available()) {
            throw new DomainException("Mavjud: {$this->available()}");
        }
        $this->res += $n;
    }
}
$s = new StockRow(10, 7);           // 7 tasi allaqachon band
try { $s->reserve(4); }
catch (DomainException $e) { echo $e->getMessage(), ' / ', $s->res; }
```

▶ NATIJA

```
Mavjud: 3 / 7
```

`available()`

formula bitta joyda — hamma shu metodni chaqiradi

`$n <= 0`

nol va manfiy miqdor ham qoida buzilishi

DomainException

biznes-rad etish; xabarda kontekst bor

Natija

10 − 7 = 3 mavjud; 4 ta so‘ralganda rad etildi, `res` 7 bo‘lib qoldi

DOMEN ISTISNOLARI

Istisno va uni bitta joyda ko‘rsatish

```php
// bootstrap/app.php
use App\Exceptions\InsufficientStockException;
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Http\Request;

return Application::configure(basePath: dirname(__DIR__))
    ->withExceptions(function (Exceptions $exceptions) {
        $exceptions->dontReport(InsufficientStockException::class);
        $exceptions->render(function (
            InsufficientStockException $e,
            Request $request,
        ) {
            return $request->expectsJson()
                ? response()->json(['message' => $e->getMessage()], 422)
                : back()->withErrors(['stock' => $e->getMessage()]);
        });
    })->create();
```

▶ NATIJA

```
Web: ← orqaga, @error('stock'): “Mahsulot #7: so‘ralgan 4, mavjud 1”
API: 422 {"message": "Mahsulot #7: so‘ralgan 4, mavjud 1"}
```

extends DomainException

`InsufficientStockException` — biznes-qoida rad etdi

`render()`

istisnoni web/API javobiga aylantiradi — controllerda `try` kerak emas

`dontReport()`

kutilgan rad etishlar xatolar logini to‘ldirmaydi

422

Unprocessable Content: so‘rov to‘g‘ri, lekin qoida bajarilmadi

Pul: float emas, butun tiyin

```php
function toTiyin(string $amount): int
{
    [$som, $frac] = explode('.', $amount . '.00');
    return (int) $som * 100 + (int) substr($frac . '0', 0, 2);
}

var_dump(0.1 + 0.2 == 0.3);
echo (int) (19.99 * 100), "\n";        // float: 1 tiyin yo‘qoldi
echo toTiyin('19.99'), "\n";            // satrdan — aniq
echo toTiyin('1250000.5'), "\n";
$vat = intdiv(toTiyin('19.99') * 3 * 12 + 50, 100);
echo $vat, " tiyin QQS\n";
```

▶ NATIJA

```
bool(false)
1998
1999
125000050
720 tiyin QQS
```

`19.99 * 100`

ikkilik kasr `1998.999…` — `(int)` kesadi

`toTiyin('19.99')`

`decimal:2` cast satr qaytaradi — satrdan aylantiramiz

`intdiv(x * 12 + 50, 100)`

QQS 12 %, yarmidan yuqoriga yaxlitlash, faqat butun sonlar

Manfiy summa?

bu soddalashtirilgan funksiya faqat musbat summalar uchun

BAZA — OXIRGI CHEGARA

CHECK cheklovi migratsiyada

```php
public function up(): void
{
    DB::statement(<<<'SQL'
        ALTER TABLE stocks
        ADD CONSTRAINT stocks_quantity_chk CHECK (quantity >= 0),
        ADD CONSTRAINT stocks_reserved_chk
            CHECK (reserved >= 0 AND reserved <= quantity)
    SQL);
}

public function down(): void
{
    DB::statement('ALTER TABLE stocks
        DROP CONSTRAINT stocks_quantity_chk,
        DROP CONSTRAINT stocks_reserved_chk');
}
```

▶ NATIJA

```
php artisan tinker
> DB::update('update stocks set reserved = quantity + 1 where id =
1');
Illuminate\Database\QueryException  SQLSTATE[23514]: Check
violation:
  new row for relation "stocks" violates check constraint
"stocks_reserved_chk"
```

DB::statement()

Laravel 12 Blueprint da CHECK uchun fluent metod yo‘q

<<<'SQL'

nowdoc: satr ichida o‘zgaruvchi almashtirilmaydi

23514

servisni chetlab o‘tgan kod ham invariantni buza olmaydi

down()

migratsiya orqaga qaytariladigan bo‘lsin

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

4-bo‘lim yakuni: Invariantlar va domen istisnolari

1

OmborPro da mavjud qoldiq qanday hisoblanadi va qaysi invariant uni himoyalaydi?

Savol 1

2

Nima uchun servis tekshiruvi bor bo‘lsa ham bazada CHECK cheklovi kerak?

Savol 2

3

`(int) (19.99 * 100)` nima uchun 1999 emas?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · Oson · 10 ball

2-topshiriq: Rezerv va mavjud qoldiq

Qoldiq qatori `['quantity' => int, 'reserved' => int]` ko‘rinishida. `miqdor` dona rezerv qiling. Mavjud qoldiq — `quantity - reserved`. `miqdor ≤ 0` bo‘lsa — `InvalidArgumentException`; `miqdor` mavjud qoldiqdan katta bo‘lsa — `DomainException` (qoldiq o‘zgarmaydi).

Imzo: `function rezerv(array $stock, int $miqdor): array`

Qaytaradi: `['quantity' => int, 'reserved' => int, 'available' => int]` — rezervdan keyingi holat.

MISOL · CHAQIRUV

```
rezerv(['quantity' => 10, 'reserved' => 2], 5)
```

MISOL · NATIJA

```
{"quantity":10,"reserved":7,"available":3}
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

05

Hujjat oqimi va holat mashinasi

Buyurtma → rezerv → hisob-faktura → jo‘natish · OrderStatus · ombor harakatlari jurnali

Hujjat oqimi

OrderStatus

Holat mashinasi

stock_movements

HUJJAT OQIMI

Buyurtmadan jo‘natishgacha

1

Yaratish → draft

Menejer `OrderService::create()` — qatorlar va jami summa; qoldiq o‘zgarmaydi.

2

Tasdiqlash → confirmed

Mijoz bilan kelishildi; narxlar qotiriladi, `OrderConfirmed` hodisasi.

3

Rezerv → reserved

`StockService::reserve()` har bir qator uchun: `reserved += n`, jurnalga `reserve`.

4

Hisob-faktura → invoiced

`InvoiceService::createForOrder()`: subtotal, QQS 12 %, jami; `HF-2026-000045`.

5

Jo‘natish → shipped

`StockService::ship()`: `quantity -= n`, `reserved -= n`, jurnalga `out`.

6

Bekor qilish → cancelled

Faqat draft/confirmed/reserved dan; reserved bo‘lsa — `release`.

Har bir qadam — `OrderService` ning bitta metodi va bitta tranzaksiya.

Harakat turi va qoldiqqa ta’siri

Tur	quantity	reserved	Shart (invariant)
`in` — kirim	+n	—	n > 0
`reserve` — rezerv	—	+n	quantity − reserved ≥ n
`release` — bo‘shatish	—	−n	reserved ≥ n
`out` — jo‘natish	−n	−n	reserved ≥ n (avval rezerv qilingan)
`adjust` — tuzatish	±n	—	natijada quantity ≥ reserved

Har bir qatorda: `document_type`/`document_id` (Order, Invoice…), `user_id`, `created_at`. Jurnal faqat qo‘shiladi — o‘chirilmaydi.

APP/ENUMS

OrderStatus: o‘tishlar qoidasi enum ichida

```php
// enum OrderStatus: string — Draft, Confirmed, Reserved,
//                          Invoiced, Shipped, Cancelled
public function canTransitionTo(self $to): bool
{
    return in_array($to, match ($this) {
        self::Draft => [self::Confirmed, self::Cancelled],
        self::Confirmed => [self::Reserved, self::Cancelled],
        self::Reserved => [self::Invoiced, self::Cancelled],
        self::Invoiced => [self::Shipped],
        self::Shipped, self::Cancelled => [],
    }, true);
}

public function isFinal(): bool
{
    return $this === self::Shipped || $this === self::Cancelled;
}
```

▶ NATIJA

```
OrderStatus::Draft->canTransitionTo(OrderStatus::Confirmed)  //
true
OrderStatus::Shipped->canTransitionTo(OrderStatus::Cancelled) //
false
OrderStatus::from('reserved')                                  //
Reserved
```

`match ($this)`

barcha holatlar sanab o‘tilgan — yangi holat qo‘shilsa, PHP `UnhandledMatchError` beradi

`in_array(..., true)`

qat’iy taqqoslash: enum obyektlari bo‘yicha

`isFinal()`

yakuniy holatlar: tugmalarni yashirish, tahrirni taqiqlash

`label(), color()`

xuddi shu enumda — interfeys uchun yorliq (6-mavzu)

O‘tishni tekshirish: ruxsat yoki istisno

```php
const NEXT = [
    'draft' => ['confirmed', 'cancelled'],
    'confirmed' => ['reserved', 'cancelled'],
    'reserved' => ['invoiced', 'cancelled'],
    'invoiced' => ['shipped'],
];
function transition(string $from, string $to): string
{
    if (!in_array($to, NEXT[$from] ?? [], true)) {
        throw new DomainException("$from → $to taqiqlangan");
    }
    return $to;
}
$s = transition('draft', 'confirmed');
$s = transition($s, 'reserved');
echo $s, "\n";
try { transition('shipped', 'cancelled'); }
catch (DomainException $e) { echo $e->getMessage(); }
```

▶ NATIJA

```
reserved
shipped → cancelled taqiqlangan
```

NEXT

holat mashinasi — oddiy ma’lumot: holat → ruxsat etilgan holatlar

?? []

yakuniy holatlarda (`shipped`, `cancelled`) ro‘yxat bo‘sh

throw

servisda bu `InvalidStatusTransitionException`

Zanjir

`draft → confirmed → reserved` — har bir qadam alohida tekshiriladi

▶ INTERAKTIV

Bu o‘tish ruxsat etilganmi?

Har bir holat o‘tishini to‘g‘ri qutiga joylashtiring. (Interaktiv slaydlarda bosib bajariladi.)

`draft → confirmed`

`confirmed → reserved`

`reserved → cancelled`

`invoiced → shipped`

`draft → cancelled`

`shipped → cancelled`

`draft → invoiced`

`invoiced → cancelled`

`reserved → confirmed`

`cancelled → draft`

Ruxsat etilgan

Taqiqlangan

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

5-bo‘lim yakuni: Hujjat oqimi va holat mashinasi

1

Qaysi holatlardan buyurtmani bekor qilish mumkin va `reserved` dan bekor qilinganda nima bo‘ladi?

Savol 1

2

`status` enumga cast qilingan bo‘lsa, `$order->status === 'reserved'` nima qaytaradi?

Savol 2

3

Nima uchun ombor harakatlari jurnalidagi yozuvlar o‘chirilmaydi?

Savol 3

MA’RUZADAGI MASHQ

Har kim o‘z qurilmasida · 10 daqiqa · O‘rta · 20 ball

3-topshiriq: Buyurtma holatlari zanjiri

Buyurtma `draft` holatida boshlanadi. Hodisalar ro‘yxati ketma-ket qo‘llanadi: `confirm` → `confirmed`, `reserve` → `reserved`, `invoice` → `invoiced`, `ship` → `shipped`, `cancel` → `cancelled`. Ruxsat etilgan o‘tishlar: `draft → confirmed → reserved → invoiced → shipped`; `cancelled` ga faqat `draft`, `confirmed`, `reserved` dan. Noma’lum hodisa — `InvalidArgumentException`, taqiqlangan o‘tish — `DomainException`. Birinchi uchragan xato tashlanadi.

Imzo: `function holat_zanjiri(array $hodisalar): string`

Qaytaradi: Yakuniy holat satri, masalan `'invoiced'`.

MISOL · CHAQIRUV

```
holat_zanjiri(['confirm', 'reserve', 'invoice'])
```

MISOL · NATIJA

```
"invoiced"
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. PHP kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

XULOSA

Asosiy xulosalar

1 Logika — servis qatlamida

Controller yupqa: kirish → `OrderService` metodi → javob; qoida bitta joyda.

2 Bitta amal — bitta tranzaksiya

`DB::transaction` hammasini saqlaydi yoki hammasini ROLLBACK qiladi.

3 Parallellikdan himoya

`lockForUpdate()` yoki shartli UPDATE; qulflar `product_id` tartibida.

4 Invariant uch qatlamda

Servis tekshiruvi, domen istisnosi va bazadagi CHECK cheklovi.

5 Holat mashinasi

`OrderStatus::canTransitionTo()` — taqiqlangan o‘tish `InvalidStatusTransitionException`.

6 Jurnal va pul

Har bir qoldiq o‘zgarishi `stock_movements` da; pul — butun tiyin, `float` emas.

TAKRORLASH

Yakunlashdan oldin sakkizta savol

1. Semiz controllerning asosiy zarari nima?
2. `DB::transaction` ichida istisno tashlansa?
3. Ichma-ich `DB::transaction` nima yaratadi?
4. `lockForUpdate()` qanday SQL qo‘shadi?
5. Deadlockni qanday kamaytirasiz?
6. Mavjud qoldiq formulasi?
7. `shipped → cancelled` mumkinmi?
8. Hodisani COMMIT dan keyin qanday jo‘natish mumkin?

AMALIY MASHG‘ULOT

4-amaliy mashg‘ulot: 5 ta PHP topshiriq va loyiha bosqichi

4 **Hisob-faktura summalari**
Oson · 10 ball

5 **Jurnaldan qoldiqni tiklash**
O‘rta · 20 ball

6 **Hujjatning keyingi raqami**
O‘rta · 20 ball

7 **Ombor: harakatlar va jurnal**
Qiyin · 30 ball

8 **Ko‘p qatorli buyurtmani atomar rezerv qilish**
Qiyin · 30 ball

Telefon kamerasi bilan skanerlang

PHP kod saytda yoziladi va server tomonidan avtomatik tekshiriladi.

Loyiha havolasini ham shu sahifada topshirasiz; o‘qituvchi baho qo‘yadi.

CHIQISH CHIPTASI

Darsni tark etishdan oldin

Javoblaringizni qog‘ozga yozing va o‘qituvchiga topshiring.

1

Ikki menejer bir vaqtda oxirgi tovarni rezerv qilsa, sizning kodingiz buni qaysi qatorda to‘xtatadi?

2

`reserve()` metodida qaysi uchta jadvalga yoziladi va nima uchun ular bitta tranzaksiyada bo‘lishi kerak?

3

Bugungi mavzuda sizga eng qiyin tuyulgan tushuncha qaysi: tranzaksiya, qulf yoki holat mashinasi?