12

O‘N IKKINCHI AMALIY MASHG‘ULOT

Korxona ilovalarini dasturlash · 2 soat · 12-mavzu

Pest testlari va kod tahlili

Jamoa loyihasi bosqichi · 5 ta PHP topshiriq · o‘qituvchi bahosi

Pest | Factories | Coverage | Larastan | Code review

MASHG‘ULOT MAQSADI

Amaliy mashg‘ulot oxirida siz quyidagilarni bajara olasiz...

1. **Sozlash** — PostgreSQL test bazasini, `tests/Pest.php` va qamrov o‘lchash (PCOV) muhitini
2. **Yozish** — biznes-qoidalar uchun unit va integratsion testlarni: holat o‘tishlari, rezerv, jo‘natish
3. **Tekshirish** — REST API ni: 201/401/403/422, JSON tuzilmasi va rollar matritsasi
4. **O‘lchash** — qamrovni va uni kamida 70 % ga (servislar — 85 %) yetkazishni
5. **Joriy qilish** — pre-commit hook, PR shabloni va main uchun branch himoyasini

Qanday bajariladi va qanday baholanadi?

Ish tartibi

1. Saytga login va parolingiz bilan kiring: **emro.uz/korxona/kirish.html**
2. **12-amaliy** sahifasini oching (oxirgi slayddagi QR kod).
3. Yo‘riqnoma bo‘yicha jamoa loyihasida bosqichni bajaring, **commit** va **push** qiling.
4. PHP topshiriqlarni yeching: ▶ **Sinab ko‘rish** → ✓ **Topshirish** (yashirin testlar).
5. Loyiha havolasini (GitHub repo, PR yoki tag) topshiring — o‘qituvchi **baho** qo‘yadi.

Baholash mezonlari

- **Natija** — loyiha bosqichi mezonlar ro‘yxati bo‘yicha bajarilgan.
- **Topshiriqlar** — PHP topshiriqlar testlardan o‘tgan.
- **Kod sifati** — tushunarli nomlar, PSR-12, kichik va mazmunli commitlar.
- **Jamoaviy hissa** — har bir a’zo o‘z qismini tushuntira oladi.
- **Muddat** — muddatdan keyin topshirilgan ish “kechikkan” deb belgilanadi.

01

Test muhiti va biznes-qoida testlari

omborpro_test · phpunit.xml · tests/Pest.php · StockService va OrderStatus testlari

phpunit.xml

RefreshDatabase

StockService

PHPUNIT.XML

Test muhiti o‘zgaruvchilari

```
<php>
    <env name="APP_ENV" value="testing"/>
    <env name="BCRYPT_ROUNDS" value="4"/>
    <env name="DB_CONNECTION" value="pgsql"/>
    <env name="DB_DATABASE" value="omborpro_test"/>
    <env name="CACHE_STORE" value="array"/>
    <env name="QUEUE_CONNECTION" value="sync"/>
    <env name="MAIL_MAILER" value="array"/>
    <env name="SESSION_DRIVER" value="array"/>
</php>
```

▶ NATIJA

```
$ docker compose exec db createdb -U omborpro omborpro_test
$ docker compose exec app php artisan test
  Tests:    46 passed (131 assertions)
  Duration: 3.82s
```

omborpro_test

alohida baza — ishchi ma’lumotga tegmaydi

CACHE_STORE=array

har test toza kesh bilan

MAIL_MAILER=array

xatlar hech qayerga ketmaydi

BCRYPT_ROUNDS=4

parol xeshlash tezlashadi — testlar tezroq

TESTS/PEST.PHP

Umumiy sozlama va yordamchi

```php
use Illuminate\Foundation\Testing\RefreshDatabase;
use Tests\TestCase;

uses(TestCase::class, RefreshDatabase::class)->in('Feature');

function validOrderPayload(array $override = []): array
{
    $product = Product::factory()->create(['price' => '100000.00']);
    $stock = Stock::factory()->for($product)
        ->create(['quantity' => 50]);

    return array_replace([
        'counterparty_id' => Counterparty::factory()->create()->id,
        'warehouse_id' => $stock->warehouse_id,
        'items' => [['product_id' => $product->id, 'quantity' => 2]],
    ], $override);
}
```

▶ NATIJA

```
validOrderPayload(['items' => []])
→ ['counterparty_id' => 12, 'warehouse_id' => 3, 'items' => []]
```

->in('Feature')

Feature testlar Laravel TestCase va toza baza bilan

validOrderPayload()

har test o‘z ma’lumotini yaratadi — bog‘liqlik yo‘q

array_replace

kerakli maydonni almashtirib, xato so‘rov yasash oson

INTEGRATSION TEST

ship(): qoldiq, rezerv va jurnal

```php
beforeEach(function () {
    $this->stock = Stock::factory()->create([
        'quantity' => 10, 'reserved' => 4,
    ]);
});

it('jo‘natish quantity va reserved ni kamaytiradi', function () {
    $s = $this->stock;
    app(StockService::class)->ship($s->warehouse_id, $s->product_id, 4);

    expect($s->fresh())
        ->quantity->toBe(6)
        ->reserved->toBe(0);
    $this->assertDatabaseHas('stock_movements', [
        'product_id' => $s->product_id,
        'type' => 'out', 'quantity' => 4,
    ]);
});
```

▶ NATIJA

```
 PASS  Tests\Feature\StockServiceTest
✓ it jo‘natish quantity va reserved ni kamaytiradi     0.07s
```

beforeEach

har test uchun yangi qoldiq yozuvi

->quantity->toBe(6)

Pest higher-order tasdiqlari: xossalar zanjiri

'type' => 'out'

har bir qoldiq o‘zgarishi jurnalga yoziladi — invariant

02

API testlari va qamrov

OrderApiTest · 201/401/403/422 · rollar matritsasi · PCOV · --min

API test

Coverage

PCOV

API TEST

Menejer buyurtma yaratadi; mass assignment ishlamaydi

```php
it('menejer buyurtma yaratadi', function () {
    $manager = User::factory()->create()->assignRole('menejer');

    $this->actingAs($manager, 'api')
        ->postJson('/api/v1/orders', validOrderPayload([
            'status' => 'shipped',          // e’tiborsiz qolishi kerak
        ]))
        ->assertCreated()
        ->assertJsonPath('data.status', 'draft');

    $this->assertDatabaseCount('order_items', 1);
});

it('tokensiz so‘rov 401', function () {
    $this->postJson('/api/v1/orders', [])->assertUnauthorized();
});
```

▶ NATIJA

```
✓ it menejer buyurtma yaratadi                0.18s
✓ it tokensiz so‘rov 401                      0.02s
```

`'status' => 'shipped'`

mass assignment hujumi — holat `draft` qolishi shart

`assertCreated()`

201 — yangi resurs

`assertDatabaseCount`

javob bilan birga baza holati

COMPOSER SKRIPTLARI

Bir xil buyruqlar: test, qamrov, lint

```
{
    "scripts": {
        "test": "pest --parallel",
        "test:cov": "pest --coverage --min=70",
        "lint": [
            "pint --test",
            "phpstan analyse --no-progress --memory-limit=1G"
        ],
        "hooks": "git config core.hooksPath .githooks"
    }
}
```

▶ NATIJA

```
$ composer test:cov
  Services/OrderService ............................ 91.3 %
  Services/StockService ............................ 96.0 %
  Total: 78.4 %
```

"test:cov"

chegaradan past — chiqish kodi xato

"lint"

massiv — buyruqlar ketma-ket, birinchi xatoda to‘xtaydi

"hooks"

yangi a’zo bir buyruq bilan hookni ulaydi

MAQSADLAR

Qamrov va testlar bo‘yicha minimal talablar

Qism	Minimal talab	Tekshiruv
`app/Services`	≥ 85 % qamrov	`--coverage` hisoboti
Jami loyiha	≥ 70 % qamrov	`composer test:cov` yashil
Holat o‘tishlari	Barcha ruxsat va taqiqlar dataset da	Unit test
Qoldiq invariantlari	reserve, ship, cancel + yetmaslik holati	Integratsion test
Buyurtmalar API	201, 401, 403 (matritsa), 422 (3 holat)	Feature test
Arxitektura	Controllerlarda `DB` yo‘q, `dd()` yo‘q	Arch test

03

Pre-commit hook va Pull Request orqali kod tahlili

.githooks · PR shabloni · branch himoyasi · jamoaviy review

pre-commit

PR shabloni

Code review

O‘RNATISH

Hookni ulash va sinab ko‘rish

```
mkdir -p .githooks
cp docs/pre-commit.sh .githooks/pre-commit
chmod +x .githooks/pre-commit
composer hooks              # git config core.hooksPath .githooks

# sinov: ataylab uslub xatosi
echo "<?php use Foo\Bar;" > app/Tmp.php
git add app/Tmp.php && git commit -m "hook sinovi"
rm app/Tmp.php && git restore --staged app/Tmp.php
```

▶ NATIJA

```
Pint...
  × app/Tmp.php    no_unused_imports, blank_line_after_opening_tag
Uslub xatosi: ./vendor/bin/pint ni ishga tushiring
(commit bekor qilindi)
```

`chmod +x`

bajariladigan bo‘lmasa Git hookni ishga tushirmaydi

composer hooks

har bir a’zo bir marta bajaradi

ataylab xato

hook haqiqatan ishlashini isbotlash

PR SHABLONI

.github/pull_request_template.md

```
## Nima o‘zgardi va nima uchun
Closes #

## Qanday tekshirildi
- [ ] Yangi qoida uchun Pest testlari bor
- [ ] `composer test:cov` yashil (qamrov pasaymadi)
- [ ] `composer lint` xatosiz

## Ko‘rib chiquvchi uchun
- [ ] Biznes-qoidalar va invariantlar (qoldiq, holatlar)
- [ ] Tranzaksiya, `lockForUpdate`, `afterCommit`
- [ ] Policy va validatsiya, `$fillable`
- [ ] N+1 yo‘q, kerakli indekslar bor
- [ ] Migratsiya qaytariladigan
```

Closes #

issue avtomatik yopiladi va bog‘lanadi

Qanday tekshirildi

muallif o‘zi tekshirganini tasdiqlaydi

Ko‘rib chiquvchi uchun

ma’ruzadagi tekshiruv ro‘yxati

▶ NATIJA

GitHub: har bir yangi PR tavsifi shu shablon bilan ochiladi

BRANCH HIMOYASI

GitHub da main ni himoyalash

1

Settings → Branches

`main` uchun himoya qoidasini (ruleset) qo‘shish

2

Require a pull request

to‘g‘ridan-to‘g‘ri push yo‘q; kamida 1 tasdiq

3

Dismiss stale approvals

yangi commitdan keyin qayta ko‘rib chiqish

4

Require conversation resolution

barcha izohlar hal qilinmaguncha merge yo‘q

5

Status checks

CI ulangach (14-mavzu) — yashil tekshiruvlar majburiy

LOYIHA BOSQICHI

Natija va topshirish

Natija: loyiha biznes-qoidalari va API si Pest testlari bilan himoyalangan, qamrov o‘lchanadi va belgilangan chegaradan past emas, pre-commit hook uslub va tur xatolarini commitdan oldin ushlaydi, o‘zgarishlar Pull Request orqali kod tahlilidan o‘tadi.

Baholash mezonlari

- ✓ Testlar alohida `omborpro_test` PostgreSQL bazasida; `RefreshDatabase` ulangan; `php artisan test` yashil.
- ✓ `OrderStatus` uchun barcha ruxsat etilgan va taqiqlangan o‘tishlar dataset bilan testlangan.
- ✓ `StockService` uchun reserve, ship, cancel va qoldiq yetmasligi (holat o‘zgarmasligi bilan) testlari bor.
- ✓ Buyurtmalar API: 201, 401, rollar matritsasi (403), uchta 422 holati va mass assignment testi.
- ✓ Qamrov: `app/Services` ≥ 85 %, jami ≥ 70 % — `composer test:cov` natijasi README da.
- ✓ `.githooks/pre-commit` Pint, Larastan va unit testlarni ishlatadi; `composer hooks` bilan ulanadi.
- ✓ PR shabloni va `main` uchun branch himoyasi sozlangan.
- ✓ Har bir a’zo kamida bitta PR ni ko‘rib chiqqan: ≥ 3 mazmunli izoh va muallif javoblari.

Telefon kamerasi bilan skanerlang

Loyiha havolasini amaliy sahifasidagi «Loyiha bosqichi» bo‘limida topshiring.

📤 Squash merge qilingan Pull Request havolasini yuboring (testlar, hook va shablon bilan); PR da qamrov natijasi, ko‘rib chiquvchi izohlari va ularga javoblar ko‘rinib tursin.

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Oson · 10 ball

4-topshiriq: Factory sequence modeli

`Factory::sequence()` ga o‘xshash test yozuvlari generatorini yozing. `$n` ta yozuv yarating; `i`-yozuv (0 dan) = `['id' => i + 1]` + `$asos` ning `$holatlar[i % count]` bilan birlashtirilgan (`array_merge`) nusxasi. `$holatlar` bo‘sh bo‘lsa — faqat `$asos`. `$asos` da `id` bo‘lsa ham, `id` har doim birinchi kalit va tartib raqamiga teng. `$n < 0` — `InvalidArgumentException`.

Imzo: `function ketma_ketlik(int $n, array $asos, array $holatlar): array`

Qaytaradi: `list<array>` — har bir yozuv birinchi kaliti `id` bo‘lgan massiv.

MISOL · CHAQIRUV

```
ketma_ketlik(3, ['unit' => 'dona', 'is_active' => true],
[['unit' => 'kg'], ['unit' => 'quti']])
```

MISOL · NATIJA

```
[{"id":1,"unit":"kg","is_active":true},{"id":2,"unit":"quti","is
_active":true},{"id":3,"unit":"kg","is_active":true}]
```

Telefon kamerasi bilan skanerlang

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

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · O‘rta · 20 ball

5-topshiriq: assertJsonPath modeli

API test tasdig‘ini modellashtiring. `$javob` — JSON dan olingan massiv, `$kutilgan` — `nuqtali yo‘l => kutilgan qiymat` (masalan, `'data.0.sku' => 'UN-001'`, `'meta.total' => 25`). Har bir yo‘l bo‘yicha qiymatni toping va qat’iy (`===`) solishtiring. Natija — xatolar ro‘yxati `$kutilgan` tartibida: yo‘l topilmasa `"<yo‘l>: yo‘q"`, qiymat farq qilsa `"<yo‘l>: kutilgan <A>, keldi <B>"`, bunda A va B — `json_encode(qiymat, JSON_UNESCAPED_UNICODE)`. Qiymati `null` bo‘lgan mavjud kalit — “topilgan” hisoblanadi.

Imzo: `function json_yol_tekshir(array $javob, array $kutilgan): array`

Qaytaradi: `list<string>` — xatolar; hammasi to‘g‘ri bo‘lsa `[]`.

MISOL · CHAQIRUV

```
json_yol_tekshir(['data' => [['sku' => 'UN-001', 'price' => '150000.00']], 'meta' => ['total' => 25]], ['data.0.sku' => 'UN-001', 'meta.total' => 25])
```

MISOL · NATIJA

```
[]
```

Telefon kamerasi bilan skanerlang

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

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · O‘rta · 20 ball

6-topshiriq: Pre-commit: tekshiriladigan PHP fayllar

Hook `git diff --cached --name-status` chiqishini oladi: har bir qator — holat harfi va yo‘l(lar) TAB bilan ajratilgan. `A` (qo‘shilgan), `M` (o‘zgargan), `T` (turi o‘zgargan) — yo‘l; `R<son>` va `C<son>` (nomi o‘zgargan / nusxalangan) — **yangi** (ikkinchi) yo‘l; `D` (o‘chirilgan) — e’tiborsiz. Faqat `.php` bilan tugaydigan va `vendor/` yoki `storage/` bilan boshlanmaydigan fayllar olinadi. Bo‘sh qatorlar e’tiborsiz. Natija — noyob, o‘sish tartibida. Boshqa holat harfi — `InvalidArgumentException`.

Imzo: `function staged_php_fayllar(string $gitOutput): array`

Qaytaradi: `list<string>` — tekshiriladigan fayllar yo‘llari.

MISOL · CHAQIRUV

```
staged_php_fayllar("M\tapp/Services/StockService.php\nA\ttests/Feature/StockServiceTest.php\nD\tapp/Old.php\nM\tREADME.md")
```

MISOL · NATIJA

```
["app/Services/StockService.php","tests/Feature/StockServiceTest.php"]
```

Telefon kamerasi bilan skanerlang

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

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Qiyin · 30 ball

7-topshiriq: Pull Request merge ga tayyormi?

PR holati: `['approvals' => int, 'changes_requested' => int, 'checks' => [nom => holat], 'coverage' => float, 'base_coverage' => float, 'changed_lines' => int]`; birorta kalit yo‘q — `InvalidArgumentException`. Bloklovchi sabablar (`reasons`) aynan shu tartibda: tasdiqlar `$kerakliTasdiq` dan kam — `tasdiq yetarli emas: X/Y`; `changes_requested > 0` — `o‘zgartirish so‘ralgan`; `checks` bo‘sh — `CI tekshiruvlari yo‘q`; aks holda har bir tekshiruv **nomi bo‘yicha alifbo tartibida**: `pending` — `tekshiruv tugamagan: NOM`, `success` dan boshqa — `tekshiruv muvaffaqiyatsiz: NOM`; `coverage < base_coverage` — `qamrov pasaydi: B → C` (bir xonali, `%.1f`). Ogohlantirish (`warnings`, merge ni to‘smaydi): `changed_lines > 400` — `PR juda katta: N qator`. `mergeable` — sabablar yo‘q.

Imzo: `function pr_tayyorligi(array $pr, int $kerakliTasdiq = 1): array`

Qaytaradi: `['mergeable' => bool, 'reasons' => list<string>, 'warnings' => list<string>]`.

MISOL · CHAQIRUV

```
pr_tayyorligi(['approvals' => 1, 'changes_requested' => 0,
'checks' => ['pint' => 'success', 'pest' => 'success',
'larastan' => 'success'], 'coverage' => 82.4, 'base_coverage' =>
82.0, 'changed_lines' => 180])
```

MISOL · NATIJA

```
{"mergeable":true,"reasons":[],"warnings":[]}
```

Telefon kamerasi bilan skanerlang

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

AMALIY TOPSHIRIQ

Kompyuterda mustaqil · Qiyin · 30 ball

8-topshiriq: Ombor daftari: testlanadigan biznes-qoidalar

`StockLedger` — bitta ombor uchun qoldiq qoidalari (keyin ular Pest bilan testlanadi). `in($sku, $qty)` — kirim. `reserve($sku, $qty)` — faqat `mavjud = quantity - reserved ≥ qty` bo‘lsa. `release($sku, $qty)` — rezervni bo‘shatish, `reserved ≥ qty` bo‘lsa. `ship($sku, $qty)` — rezervdan jo‘natish: `reserved ≥ qty` bo‘lsa `quantity` va `reserved` kamayadi. `$qty ≤ 0` — `InvalidArgumentException`; qoida buzilsa — `DomainException` va holat o‘zgarmaydi. Har bir **muvaffaqiyatli** amal `movements()` ga `"type:sku:qty"` (`in`, `reserve`, `release`, `out`) ko‘rinishida yoziladi. `available($sku)` — mavjud qoldiq (noma’lum SKU — 0). `snapshot()` — `sku => ['quantity', 'reserved', 'available']`, SKU bo‘yicha tartiblangan.

Imzo: `final class StockLedger`

Qaytaradi: `in/reserve/release/ship` — `void`; `available` — `int`; `movements` — `list<string>`; `snapshot` — massiv.

MISOL · CHAQIRUV

```
(function () { $l = new StockLedger(); $l->in('UN-001', 10); $l->reserve('UN-001', 4); $l->ship('UN-001', 3); return [$l->available('UN-001'), $l->snapshot(), $l->movements()]; })()
```

MISOL · NATIJA

```
[6,{"UN-001":{"quantity":7,"reserved":1,"available":6}},["in:UN-001:10","reserve:UN-001:4","out:UN-001:3"]]
```

Telefon kamerasi bilan skanerlang

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

TOPSHIRIQLAR VA TOPSHIRISH

12-amaliy: 8 ta PHP topshiriq

1 **Test natijalari xulosasi**
Oson · 10 ball · ma’ruzada

2 **Mini expect(): tekshiruvlar**
O‘rta · 20 ball · ma’ruzada

3 **Qamrov hisoboti**
O‘rta · 20 ball · ma’ruzada

4 **Factory sequence modeli**
Oson · 10 ball

5 **assertJsonPath modeli**
O‘rta · 20 ball

6 **Pre-commit: tekshiriladigan PHP fayllar**
O‘rta · 20 ball

7 **Pull Request merge ga tayyormi?**
Qiyin · 30 ball

8 **Ombor daftari: testlanadigan biznes-qoidalar**
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.