

Refactorizarea verificabilă a unui formular

Fișă publică de proiect · exerciții și criterii de acceptare

Un formular demonstrativ are validare amestecată cu afișarea și accesul la date. Folosești un asistent pentru o schimbare mică și păstrezi comportamentul prin teste.

Livrabile

- Context de cod anonimizat
- Teste pentru cazuri-limită
- Modificare separată
- Fișă de code review

Criterii de acceptare

- Testele trec înainte și după
- Nu apar secrete în prompt
- Accesul altui utilizator este refuzat

Asistent de cod evaluat prin comportament, nu prin încredere

Un asistent poate propune repede cod plauzibil și totuși poate greși contractul, dependențele sau permisiunile. Evită să tratezi metafora unui coleg drept dovadă de competență. Pentru un formular simplu, descrie comportamentul observabil și cere o schimbare mică. Revizuirea trebuie să urmărească intrări invalide, accesul la datele altui utilizator și efectele secundare. Explicația generată nu înlocuiește rularea testului.

Exemplu de lucru

Cerință: formularul acceptă email valid și mesaj 10–500 caractere.
Cazuri: email invalid; mesaj gol; mesaj prea lung; intrare validă.
Restricții: nu schimba schema bazei și nu trimite email real în test.
Livrabil: modificare minimă + teste + limitări declarate.

Interpretarea rezultatului

O implementare este acceptată dacă toate cazurile sunt verificate și nu introduce expediere reală sau schimbări de schemă. „Ar trebui să funcționeze” este o ipoteză. Păstrează logul comenzii de test și diferența de cod pentru a putea reproduce verificarea.

Exercițiu de consolidare

Adaugă cerința ca mesajul să nu poată conține numai spații. Verifică dacă testul nou eșuează înaintea corecturii.

1. Modelul mental: un junior rapid și încrezător

Adaugă cerința ca mesajul să nu poată conține numai spații. Verifică dacă testul nou eșuează înaintea corecturii.

Verificare orientativă

O implementare este acceptată dacă toate cazurile sunt verificate și nu introduce expediere reală sau schimbări de schemă. „Ar trebui să funcționeze” este o ipoteză. Păstrează logul comenzi de test și diferența de cod pentru a putea reproduce verificarea.

De evitat

Codul care arată plauzibil poate rata exact cazul folosit în producție.

Context minim suficient pentru o modificare de cod

Nu lipi întregul proiect când problema este într-o funcție. Oferă semnătura, un apel reprezentativ, comportamentul actual și cel dorit. Păstrează numele tipurilor și versiunile relevante, dar înlocuiește secretele și datele clienților. Un fragment fără context poate produce un patch incompatibil; prea mult context poate ascunde informația decisivă. Arată și ceea ce nu trebuie modificat, mai ales contractele publice și limitele de autorizare.

Exemplu de lucru

Funcție: `normalizeEmail(string $email): string`
Actual: transformă în litere mici, păstrează spațiile.
Dorit: elimină spațiile de la capete și normalizează literele.
Apel: `normalizeEmail(' ANA@EXAMPLE.TEST ')`
Așteptat: `ana@example.test`
Nu modifica: formatul răspunsurilor API sau schema bazei.

Interpretarea rezultatului

Contextul este suficient pentru a discuta transformarea și testul, fără a expune configurația de mail. Normalizarea nu înseamnă validare: un șir poate fi normalizat și totuși să nu fie adresă validă. Păstrează aceste responsabilități distincte în cod și teste.

Exercițiu de consolidare

Adaugă un șir gol și unul cu spații în interior. Stabilește ce ține de normalizare și ce ține de validarea formularului.

2. Context de cod: cât și cum lipești

Adaugă un șir gol și unul cu spații în interior. Stabilește ce ține de normalizare și ce ține de validarea formularului.

Verificare orientativă

Contextul este suficient pentru a discuta transformarea și testul, fără a expune configurația de mail. Normalizarea nu înseamnă validare: un șir poate fi normalizat și totuși să nu fie adresă validă. Păstrează aceste responsabilități distincte în cod și teste.

De evitat

Un repository întreg nu este necesar pentru orice problemă.

Test care detectează o schimbare reală de comportament

Test-first înseamnă să exprimi comportamentul cerut înaintea implementării, apoi să observi eșecul pentru motivul corect. Un test care doar compară o constantă cu ea însăși nu verifică funcția. Pentru normalizarea emailului, construiește intrări care diferențiază versiunea veche de cea nouă. Verifică și că testul nu depinde de rețea sau de un cont extern; altfel un eșec de infrastructură poate fi confundat cu defectul de cod.

Exemplu de lucru

```
PHP demonstrativ:
self::assertSame(
    'ana@example.test',
    normalizeEmail(' ANA@EXAMPLE.TEST ')
);
self::assertSame('', normalizeEmail(' '));
Implementare propusă: return strtolower(trim($email));
```

Interpretarea rezultatului

Prima aserțiune trebuie să eșueze dacă trim lipsește, iar a doua dacă spațiile nu sunt eliminate. Rulează testul și în versiunea defectă, nu doar după patch. Nu folosi acest fragment ca validator complet de email; el verifică transformarea convenită.

Exercițiu de consolidare

Modifică deliberat funcția să elimine toate spațiile, inclusiv din interior. Scrie un test care arată de ce această schimbare depășește contractul.

3. Test-first cu AI

Modifică deliberat funcția să elimine toate spațiile, inclusiv din interior. Scrie un test care arată de ce această schimbare depășește contractul.

Verificare orientativă

Prima aserțiune trebuie să eșueze dacă trim lipsește, iar a doua dacă spațiile nu sunt eliminate. Rulează testul și în versiunea defectă, nu doar după patch. Nu folosi acest fragment ca validator complet de email; el verifică transformarea convenită.

De evitat

Un test care copiază formula greșită nu detectează greșeala.

Regex cu cazuri de frontieră și limită de domeniu

O expresie regulată trebuie evaluată pe potriviri și nepotriviri, nu doar pe exemplul care a inspirat-o. Pentru un identificator intern alegem deliberat un format simplu: prefix ORD, separator și exact patru cifre. Nu reutiliza expresia ca validator de alte coduri fără revizuire. Explică ancorele și lungimea; acestea separă potrivirea întregii valori de găsirea unui fragment în interiorul unui text.

Exemplu de lucru

```
Pattern demonstrativ: /^ORD-[0-9]{4}$/
Acceptă: ORD-0001; ORD-2048
Respinge: ord-0001; ORD-123; ORD-12345; XORD-0001
Transformare separată: trim înaintea validării, dacă regula permite spații externe.
Nu normaliza automat prefixul dacă aplicația cere majuscule exacte.
```

Interpretarea rezultatului

Ancorele cer începutul și sfârșitul șirului conform motorului folosit. Verifică particularitățile motorului pentru newline final dacă datele pot conține linii noi. O potrivire nu dovedește existența comenzii; aceasta se verifică în sursa de date autorizată.

Exercițiu de consolidare

Adaugă ORD-0001 urmat de newline și documentează rezultatul motorului concret. Ajustează ancorarea dacă regula cere sfârșit absolut.

4. Transformări de date și regex

Adaugă ORD-0001 urmat de newline și documentează rezultatul motorului concret. Ajustează ancorarea dacă regula cere sfârșit absolut.

Verificare orientativă

Ancorele cer începutul și sfârșitul șirului conform motorului folosit. Verifică particularitățile motorului pentru newline final dacă datele pot conține linii noi. O potrivire nu dovedește existența comenzii; aceasta se verifică în sursa de date autorizată.

De evitat

Regex nu este un parser universal pentru formate structurate.

Refactor cu interfață stabilă și efecte secundare explicite

Refactorizarea schimbă organizarea codului păstrând comportamentul stabilit. Înainte de mutarea logicii într-un serviciu, notează răspunsurile, erorile și efectele secundare. Nu combina refactorul cu o schimbare de tarif sau de validare. Pentru calculul unui total, funcția pură este mai ușor de verificat decât o metodă care citește cererea HTTP și scrie în bază. Păstrează citirea intrării și persistența în straturi distincte.

Exemplu de lucru

Înainte: controllerul citește cantitate, preț și calculează total.
După: `calculateTotal(int $qty, int $priceBani): int`
Exemplu: `calculateTotal(3, 1250) → 3750 bani`
Controller: validează intrarea, apelează funcția, formatează răspunsul.
Serviciu: nu accesează Request, DB sau email.

Interpretarea rezultatului

3.750 bani înseamnă 37,50 lei. Folosirea întregilor evită anumite erori de reprezentare ale valorilor monetare, dar rotunjirea și limitele numerice trebuie tot definite. Testele existente trebuie să păstreze același răspuns înainte și după reorganizare.

Exercițiu de consolidare

Adaugă o regulă nouă de discount într-o modificare separată. Explică de ce această regulă nu aparține refactorului inițial.

5. Refactor asistat

Adaugă o regulă nouă de discount într-o modificare separată. Explică de ce această regulă nu aparține refactorului inițial.

Verificare orientativă

3.750 bani înseamnă 37,50 lei. Folosirea întregilor evită anumite erori de reprezentare ale valorilor monetare, dar rotunjirea și limitele numerice trebuie tot definite. Testele existente trebuie să păstreze același răspuns înainte și după reorganizare.

De evitat

Amestecarea refactorizării cu schimbări de comportament complică verificarea.

Revizuire orientată pe contract și acces la date

Nu începe review-ul cu numele variabilelor. Urmărește întâi dacă solicitantul poate accesa resursa și dacă operația acceptă numai câmpurile permise. Un patch generat poate fi lizibil și totuși poate permite modificarea proprietarului unei înregistrări. Revizuirea trebuie să lege fiecare risc de un caz concret și de un control în cod. Cere teste pentru accesul refuzat, nu doar pentru utilizatorul proprietar.

Exemplu de lucru

```
Patch suspect: $record->update($request->all());
Riscuri de verificat: owner_id, role, status intern.
Variantă controlată: $data = $request->validate(['title' => ['required', 'string', 'max:120']]);
Apoi: verificare policy și update numai cu $data.
Test: utilizatorul B nu poate modifica resursa lui A.
```

Interpretarea rezultatului

Lista câmpurilor validate nu înlocuiește autorizarea, iar autorizarea nu validează conținutul titlului. Ai nevoie de ambele controale. Dacă testul utilizatorului B trece cu 200, problema este de acces, chiar dacă titlul salvat este valid.

Exercițiu de consolidare

Adaugă în cererea de test un owner_id diferit. Verifică separat că accesul este refuzat și că proprietarul nu se modifică.

6. Code review pe cod generat

Adaugă în cererea de test un owner_id diferit. Verifică separat că accesul este refuzat și că proprietarul nu se modifică.

Verificare orientativă

Lista câmpurilor validate nu înlocuiește autorizarea, iar autorizarea nu validează conținutul titlului. Ai nevoie de ambele controale. Dacă testul utilizatorului B trece cu 200, problema este de acces, chiar dacă titlul salvat este valid.

De evitat

Autentificarea nu înseamnă automat autorizare pe fiecare resursă.

Migrare și operație financiară cu plan de revenire

O schimbare care șterge date sau mută bani cere mai mult decât un patch corect sintactic. Definește domeniul de aplicare, verificarea înainte de execuție și ce înseamnă succesul. Pentru date, o copie de siguranță este utilă numai dacă restaurarea poate fi făcută. Pentru o plată, repetarea operației poate produce efecte ireversibile, deci identificatorul operației și idempotența trebuie proiectate înaintea retry-ului. Exemplul este un plan de laborator, fără operații reale.

Exemplu de lucru

Plan: adaugă un câmp nullable; publică aplicația care îl suportă; completează datele în loturi; verifică numărul de rânduri; aplică restricția într-o etapă separată.
Pentru fiecare etapă: intrare, condiție de oprire, dovadă, revenire.
Nu combina: ștergerea vechiului câmp și primul deploy care citește noul câmp.

Interpretarea rezultatului

O versiune veche a aplicației trebuie să poată funcționa în timpul tranziției dacă acesta este obiectivul deploy-ului. Revenirea codului nu repară automat o ștergere de date. Păstrează verificările de integritate distincte de verificarea HTTP 200.

Exercițiu de consolidare

Scrie planul pentru redenumirea unei coloane folosite de un job din coadă. Include cazul în care jobul vechi rulează după deploy.

7. Limite: bani, permisiuni, migrări

Scrie planul pentru redenumirea unei coloane folosite de un job din coadă. Include cazul în care jobul vechi rulează după deploy.

Verificare orientativă

O versiune veche a aplicației trebuie să poată funcționa în timpul tranziției dacă acesta este obiectivul deploy-ului. Revenirea codului nu repară automat o ștergere de date. Păstrează verificările de integritate distincte de verificarea HTTP 200.

De evitat

Accesul la terminal nu înseamnă permisiune nelimitată.

Atelier final: Refactorizare cu acces interzis verificat

Timp de lucru estimat: 40 de minute. Folosește datele de laborator de mai jos și salvează rezultatul separat de orice proiect real.

Date de pornire

Aplicație fictivă: documentul 17 aparține utilizatorului A. B este autentificat, dar nu este proprietar. Formularul acceptă title, iar owner_id nu este modificabil. Titlul are între 1 și 120 de caractere.

1. Pregătire — 5 minute

Extrage intrările și condițiile din enunț. Notează ce rezultat trebuie produs și ce informații nu sunt confirmate. Pregătește fișierele sau tabelul în care vei păstra dovada lucrului.

2. Construire — 15 minute

Pregătește testele pentru A cu titlu valid, A cu titlu gol, B cu titlu valid și A cu owner_id trimis suplimentar. Cere asistentului un patch mic și revizuieste diferența înainte de aplicare.

3. Variație și verificare — 15 minute

Introdu intenționat o validare care acceptă orice câmp și arată care test detectează regresia. Lucrează numai în copia de laborator.

4. Autoevaluare — 5 minute

Compară rezultatul cu reperele de mai jos. Înregistrează cel puțin o eroare identificată sau un caz-limită verificat, plus corectura ori explicația comportamentului observat.

Repere de rezolvare

A poate schimba titlul valid; cererea invalidă nu modifică baza; B primește refuz; owner_id rămâne al lui A. Nu este suficient să verifici numai codul HTTP dacă o scriere neautorizată tot a avut loc.

Ce predai

- Context de cod anonimizat
- Teste pentru cazuri-limită
- Modificare separată
- Fișă de code review

Criterii de acceptare

- Testele trec înainte și după
- Nu apar secrete în prompt
- Accesul altui utilizator este refuzat

Păstrează o versiune inițială și una revizuită, cu explicația diferențelor. Dacă un criteriu nu este îndeplinit, revino la lecția relevantă și repetă verificarea. Durata este orientativă; instalarea mediului și aprofundarea pot necesita timp suplimentar.

Registrul de lucru

Pentru fiecare etapă notează intrarea, rezultatul, dovada verificării, eroarea găsită și corectura. Nu introduce date personale sau secrete în fișa partajată.

Documentație și surse

- Prompt engineering overview — Anthropic
- OWASP Top 10 — OWASP

<https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/overview>

<https://owasp.org/projects/top-ten>