

RAPPORT D'AUDIT FLASH

Qualité de la suite de tests Playwright

Projet anonymisé · monorepo TypeScript

40/100	24	80
SCORE QUALITÉ GLOBAL	ERREURS CRITIQUES	SIGNALEMENTS

PRESTATION	Audit Flash Playwright — 490 € HT (diagnostic sous 48 h ouvrées)
PÉRIMÈTRE	repo interne anonymisé
MOTEUR D'ANALYSE	@doctest/audit-engine · scan déterministe (20 règles Playwright best practices)
DATE DU RAPPORT	28 mai 2026

Sommaire

- 1 Synthèse exécutive & impact CI
- 2 Score par dimension & constats dans le code
- 3 Anti-patterns prioritaires
- 4 Plan de remédiation — quick wins
- 5 Prochaines étapes proposées
- 6 Confidentialité, périmètre & méthodologie

1 Synthèse exécutive & impact CI

Score qualité global : 40 / 100. Suite fragile, refonte partielle recommandée. (inférieur au seuil recommandé de 75/100).

L'analyse du dépôt (8 fichier(s) spec, 36 test(s) détecté(s)) met en évidence un score global de 40/100, inférieur au seuil recommandé de 75/100. Nous avons identifié 24 erreur(s) critique(s), 20 avertissement(s) et 36 recommandation(s), soit 80 signalements au total.

Indicateurs clés

40/100	24	20	36
SCORE GLOBAL	ERREURS CRITIQUES	AVERTISSEMENTS	RECOMMANDATIONS

Lecture des résultats

- Les erreurs critiques peuvent provoquer des échecs intermittents ou des faux positifs / faux négatifs, ce qui dégrade la confiance accordée à la CI.
- Les avertissements signalent des fragilités qui alourdissent la maintenance dès que l'interface ou les parcours évoluent.
- Les recommandations (tags, conventions, données) permettent de filtrer les campagnes et de piloter la suite à mesure qu'elle grossit.

Ce que ça coûte aujourd'hui

- Temps CI perdu : non mesuré — aucun run CI fourni pour ce Flash. Le chiffrage des re-runs demande les 5 derniers runs, ou l'Audit Flakiness Approfondi.
- Confiance : non observée sur des runs. La priorisation repose sur les anti-patterns statiques.
- Risque livraison : les faux rouges ne sont pas quantifiés sans historique CI.

Impact métier. Le score sert à situer la suite ; la valeur de cet audit est l'ordre de priorité. Corriger les erreurs critiques en premier restaure le signal CI sans réécrire toute la suite.

2 Score par dimension & constats dans le code

Le tableau ci-dessous synthétise les six dimensions évaluées. Les sous-sections détaillent les signalements détectés, avec extrait de code et correction recommandée.

Dimension	Score	Erreurs	Avert.	Info	Total
Locators (sélecteurs robustes)	30/100	7	5	—	12
Assertions web-first / waits	0/100	14	5	—	19
Isolation parallèle	91/100	—	2	—	2
Fixtures POM	44/100	3	8	—	11
Tags & nommage	21/100	—	—	35	35
Données / factories	98/100	—	—	1	1
Total	40/100	24	20	36	80

2.1 Locators (sélecteurs robustes) · 12 signalements (7 erreurs, 5 avertissements)

Erreur — packages/audit-engine/src/rules/assertions.test.ts:17:39

Locator CSS détecté — fragile car couplé au DOM. Préférer getByTestId / getByRole.

```
const ctx = spec(`const v = await page.locator('.x').isVisible();
\nexpect(v).toBe(true);`);
```

Correction. Remplacer par page.getByTestId('xxx') ou page.getByRole('role', { name: '...' }).

Erreur — packages/audit-engine/src/rules/assertions.test.ts:29:36

Locator CSS détecté — fragile car couplé au DOM. Préférer getByTestId / getByRole.

```
const ctx = spec(`await expect(page.locator('.x')).toBeVisible();`);
```

Correction. Remplacer par page.getByTestId('xxx') ou page.getByRole('role', { name: '...' }).

Erreur — packages/audit-engine/src/rules/locator.test.ts:19:15

Locator CSS détecté — fragile car couplé au DOM. Préférer getByTestId / getByRole.

```
await page.locator('.btn-primary').click();
```

Correction. Remplacer par page.getByTestId('xxx') ou page.getByRole('role', { name: '...' }).

+ 9 autres signalements dans cette dimension, planifiés en section 4.

2.2 Assertions web-first / waits · 19 signalements (14 erreurs, 5 avertissements)

Erreur — packages/audit-engine/src/rules/assertions.test.ts:37:1

test('should do nothing') ne contient aucune assertion (web-first ou POM verify*).

```
test('should do nothing', ...)
```

Correction. Ajouter au moins une assertion finale : await expect(locator).toBeVisible() ou xxxPage.expectXxx().

Erreur — packages/audit-engine/src/rules/locator.test.ts:18:1

test('foo') ne contient aucune assertion (web-first ou POM verify*).

```
test('foo', ...)
```

Correction. Ajouter au moins une assertion finale : await expect(locator).toBeVisible() ou xxxPage.expectXxx().

Erreur — packages/audit-engine/src/scan-repo.test.ts:17:11

page.waitForTimeout() détecté — anti-pattern majeur (sleep arbitraire, cause de flake).

```
await page.waitForTimeout(2000);
```

Correction. Remplacer par await expect(locator).toBeVisible() ou await expect.poll(...).

+ 16 autres signalements dans cette dimension, planifiés en section 4.

2.3 Isolation parallèle · 2 signalements (2 avertissements)**Avertissement — packages/audit-engine/src/scan-repo.test.ts:10:1**

Variable de module mutable détectée: sharedUser. Pollution potentielle entre tests parallèles.

```
let sharedUser;
```

Correction. Encapsuler dans une fixture worker-scoped (test.extend) ou un factory par test.

Avertissement — packages/audit-engine/src/scan-repo.test.ts:61:1

Variable de module mutable détectée: tmpRepo. Pollution potentielle entre tests parallèles.

```
let tmpRepo = '';
```

Correction. Encapsuler dans une fixture worker-scoped (test.extend) ou un factory par test.

2.4 Fixtures POM · 11 signalements (3 erreurs, 8 avertissements)

Erreur — packages/audit-engine/src/rules/fixtures.test.ts:17:13

Instanciation directe d'un Page Object dans une spec — bypass des fixtures.

```
it('flags new XxxPage(page) in spec', () => {
```

Correction. Définir le POM dans fixtures/test.fixture.ts via test.extend() puis l'injecter dans le test.

Erreur — packages/audit-engine/src/rules/fixtures.test.ts:18:41

Instanciation directe d'un Page Object dans une spec — bypass des fixtures.

```
const ctx = spec(`const loginPage = new LoginPage(page);`);
```

Correction. Définir le POM dans fixtures/test.fixture.ts via test.extend() puis l'injecter dans le test.

Erreur — packages/audit-engine/src/scan-repo.test.ts:22:22

Instanciation directe d'un Page Object dans une spec — bypass des fixtures.

```
const cartPage = new CartPage(page);
```

Correction. Définir le POM dans fixtures/test.fixture.ts via test.extend() puis l'injecter dans le test.

+ 8 autres signalements dans cette dimension, planifiés en section 4.

2.5 Tags & nommage · 35 signalements (35 informations)

35 scénario(s) ne portent pas de tag conventionnel (@smoke, @critical, @regression, @auth). Exemple dans packages/audit-engine/src/rules/assertions.test.ts:37 :

```
test('should do nothing', ...)
```

Recommandation. Ajouter un tag dans le titre : @smoke / @critical / @auth / @regression / @slow.

2.6 Données / factories · 1 signalement (1 information)

Information — packages/audit-engine/src/scan-repo.test.ts:23:11

Données form inline ({ firstName: 'John', ... }) — risque de pollution + JDD figé.

```
await cartPage.fillShipping({ firstName: 'John', lastName: 'Doe', zip: '12345' });
```

Correction. Extraire vers fixtures/data/*.factory.ts avec Faker + suffixe unique, ou passer par xxxPage.fillAndSubmit*().

3 Anti-patterns prioritaires

Note de lecture. Ce tableau consolide les signalements à traiter en premier. Les extraits de code et les corrections détaillées figurent dans la section 2.

Priorité	Règle	Localisation	Dimension (réf.)	Sévérité
P1	locator/css-fragile	packages/audit-engine/src/rules/assertions.test.ts:17	§2.1 Locators (sélecteurs robustes)	Erreur
P1	locator/css-fragile	packages/audit-engine/src/rules/assertions.test.ts:29	§2.1 Locators (sélecteurs robustes)	Erreur
P1	assert/missing	packages/audit-engine/src/rules/assertions.test.ts:37	§2.2 Assertions web-first / waits	Erreur
P1	fixture/new-page-instance	packages/audit-engine/src/rules/fixtures.test.ts:17	§2.4 Fixtures POM	Erreur
P1	fixture/new-page-instance	packages/audit-engine/src/rules/fixtures.test.ts:18	§2.4 Fixtures POM	Erreur
P1	locator/css-fragile	packages/audit-engine/src/rules/locator.test.ts:19	§2.1 Locators (sélecteurs robustes)	Erreur
P1	locator/css-fragile	packages/audit-engine/src/rules/locator.test.ts:29	§2.1 Locators (sélecteurs robustes)	Erreur
P1	assert/missing	packages/audit-engine/src/rules/locator.test.ts:18	§2.2 Assertions web-first / waits	Erreur
P1	locator/css-fragile	packages/audit-engine/src/scan-repo.test.ts:16	§2.1 Locators (sélecteurs robustes)	Erreur
P1	wait/manual-timeout	packages/audit-engine/src/scan-repo.test.ts:17	§2.2 Assertions web-first / waits	Erreur

Les 70 signalements restants sont regroupés par dimension en section 2 et pris en compte dans le plan de remédiation en section 4.

4 Plan de remédiation — quick wins

Actions à fort rapport valeur / effort, ordonnées par priorité. Elles visent d'abord les erreurs critiques qui brouillent le signal CI.

#	Action	Périmètre	Effort	Impact
1	Remplacer <code>page.waitForTimeout()</code> par <code>expect(...).toBeVisible()</code>	1 occurrences	Faible	Élevé
2	Remplacer locators CSS par <code>getByRole</code> / <code>getByTestId</code>	7 occurrences	Faible	Élevé
3	Migrer <code>await locator.isVisible()</code> puis <code>expect(bool)</code> vers <code>await expect(locator).toBeVisible()</code>	5 occurrences	Faible	Moyen
4	Ajouter une assertion finale (web-first ou POM verify*) aux tests sans <code>expect</code>	13 tests	Moyen	Moyen
5	Migrer <code>new XxxPage(page)</code> vers fixtures <code>test.extend()</code>	3 occurrences	Moyen	Moyen

Objectif. Traiter les actions 1 à 3 élimine les 24 erreur(s) critique(s) et ramène la suite au-dessus du seuil de 75/100.

5 Prochaines étapes proposées

Pas d'appel gratuit pour vendre une suite : vous avez le diagnostic, vous choisissez comment exécuter le plan.

Autonomie (0 €)	Votre équipe applique le plan de la section 4 en interne à partir des localisations et corrections fournies.
Playwright Stabilization Sprint	Suite naturelle de cet audit : correction d'un lot de tests fixé à l'avance et validation dans votre CI. Demi-journée à 490 € HT (3 à 5 tests) ou 1 200–1 800 € HT (10 à 20 tests).
Setup Test Architect (4 500–9 000 € HT)	Si la suite doit être refondée : socle Playwright TypeScript, Page Object Model, fixtures, parcours critiques, pipeline CI et passation.
Prestations complémentaires	Retainer QA Playwright (1 200–2 000 € HT / mois), Audit Flakiness Approfondi sur 30 runs CI (dès 1 490 € HT), ou Health-check trimestriel (690 € HT) à 90 jours.

6 Confidentialité, périmètre & méthodologie

Confidentialité et traitement des données

- Scan code-only, en lecture seule : aucun fichier .env, aucun secret, aucune donnée de production.
- Pas de rétention : le clone du dépôt est supprimé dès la génération du rapport, sans conservation au-delà de la mission.
- Aucun lock-in : ce rapport et vos tests restent votre propriété, exécutables sans Doctest.

Ce que couvre cet Audit Flash (490 € HT)

- Prestation Audit Flash 490 € HT — périmètre strict :
- 1 dépôt Git en lecture seule (suite Playwright)
- Analyse jusqu'à 200 tests détectés dans le repo
- Rapport PDF : score, anti-patterns, quick wins, plan de remédiation
- Restitution écrite dans le rapport (pas de correctifs code)
- Exclusions :
- Modification du code, PR, correctifs ou implémentation
- Ingestion des runs CI / classement flaky (offre Approfondi)
- Repos ou packages supplémentaires hors 1 périmètre convenu
- Toute correction ou implémentation fait l'objet d'un devis séparé (options B, C ou D du rapport, ou Setup / Sprint).

Méthodologie

L'audit repose sur un scan statique et déterministe du code de test, relu par un SDET senior Playwright. Chaque signalement est classé selon trois niveaux :

- Erreur — défaut pouvant provoquer un échec intermittent ou un faux vert.
- Avertissement — fragilité qui augmente le coût de maintenance ou le risque d'instabilité.
- Information — écart aux conventions, utile pour filtrer et faire grandir la suite.

Source

@doctest/audit-engine · scan déterministe (20 règles Playwright best practices) · 80 signalements analysés.