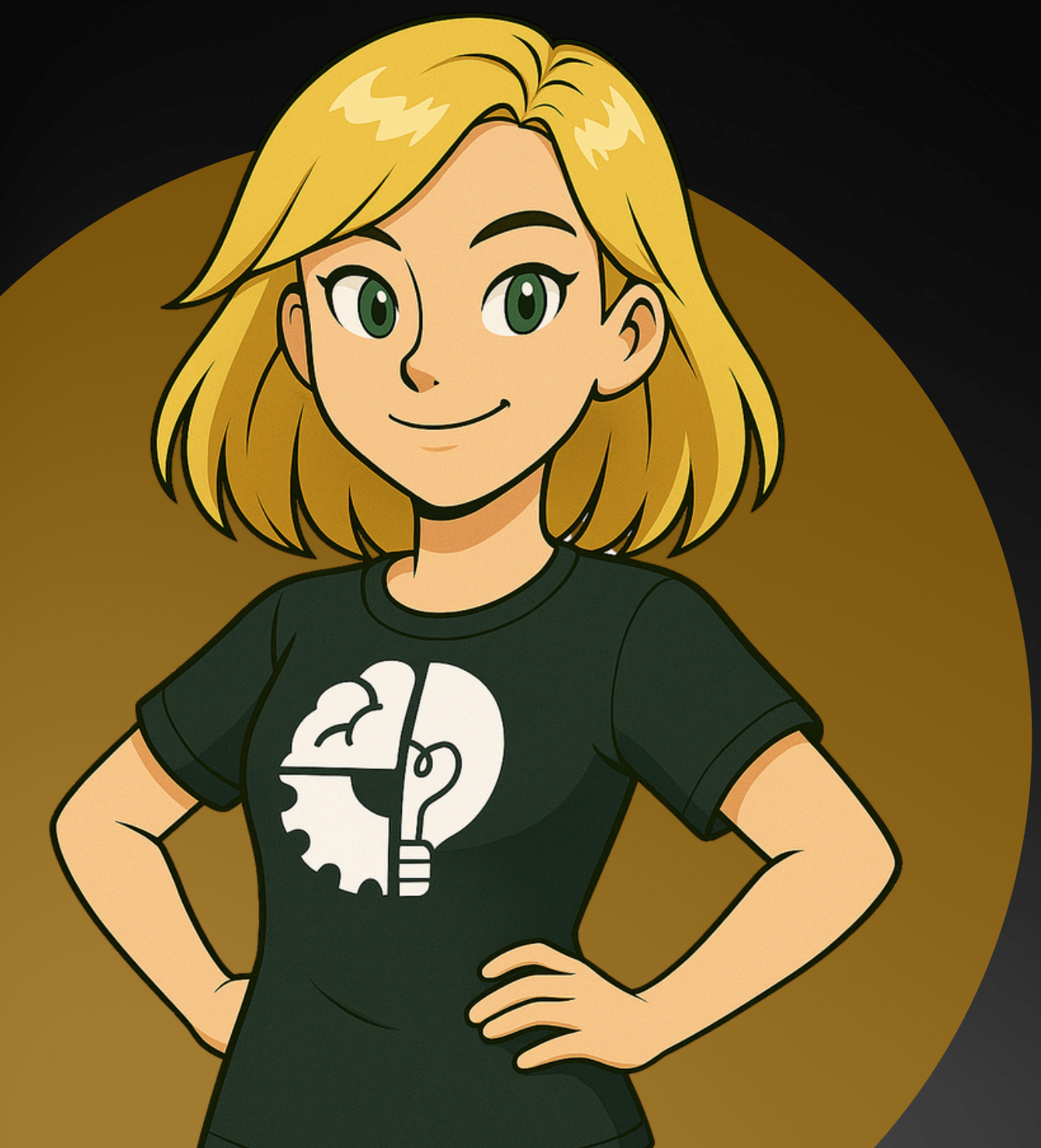
DITES ADIEU À KARMA : MIGREZ VOS TESTS ANGULAR VERS VITEST



Elodie TURLIER

Angular, IA, agilité & storytelling : Forger les outils pour survivre à l'apocalypse digitale

01

POURQUOI MIGRER ?

- **Performance accrue**

Vitest, basé sur Vite, exécute les tests beaucoup plus rapidement que Karma.

- **Courbe d’apprentissage réduite**

Syntaxe moderne proche de Jest.

- **Intégration native avec Vite**

Si ton projet utilise déjà Vite (Angular ≥17), Vitest s’intègre naturellement.

- **Support TypeScript & ESM natif**

Aucun besoin de config custom comme sous Karma ou Jest.

- **Expérience développeur améliorée**

CLI claire, couverture facile, UI interactive disponible.

- **Karma est obsolète**

Karma n’est plus activement maintenu. Angular recommande désormais Vitest.

Elodie TURLIER
Angular, IA, agilité & storytelling

02

ÉTAPE 1 : NETTOYER KARMA/JASMINE

Désinstallez les dépendances liées à Karma et Jasmine :

```
npm uninstall karma karma-chrome-launcher karma-jasmine-html-reporter
karma-coverage karma-jasmine jasmine-code @types/jasmine
```

Supprimez les fichiers de configuration spécifiques :

- karma.conf.js
- src/test.ts (ce fichier servait au bootstrap des tests avec Karma/Jasmine)

Elodie TURLIER
Angular, IA, agilité & storytelling

03

ÉTAPE 2 : INSTALLER VITEST + ANALOGJS

Pour intégrer Vitest dans un projet Angular, la méthode la plus simple et recommandée est d’utiliser le package AnalogJS, qui fournit une intégration clé-en-main avec Angular.

Installez le package @analogjs/platform en tant que dépendance de développement :

```
npm install @analogjs/platform --save-dev
```

Elodie TURLIER
Angular, IA, agilité & storytelling

04

ÉTAPE 3 : SETUP AUTO AVEC SCHEMATIC

Lancez le schematic AnalogJS pour configurer automatiquement Vitest :

```
ng generate @analogjs/platform:setup-vitest --project [nom-de-votre-projet]
```

Cette commande :

- Installe les dépendances nécessaires à Vitest et Angular
- Génère la configuration Vite (*vite.config.mts*)
- Crée le fichier de setup des tests (*src/test-setup.ts*)
- Met à jour la configuration de test dans *angular.json* ou *tsconfig.spec.json*

Elodie TURLIER
Angular, IA, agilité & storytelling

05

ÉTAPE 4 : CONFIGURATION TEST GLOBALE

Exemple de fichier *vite.config.mts* :

```ts
/// <reference types="vitest" />
import angular from '@analogjs/vite-plugin-angular';
import { defineConfig } from 'vite';

export default defineConfig(({ mode }) => ({
  plugins: [angular()],
  test: {
    globals: true, // Permet d'utiliser describe/it sans import
    environment: 'jsdom', // Simule un navigateur
    setupFiles: ['src/test-setup.ts'], // Fichier d'initialisation Angular
    include: ['**/*.spec.ts'], // Inclut tous les fichiers de test
    reporters: ['default'],
  },
  define: {
    'import.meta.vitest': mode !== 'production',
  },
}));
```

Elodie TURLIER
Angular, IA, agilité & storytelling

06

ÉTAPE 5 : SETUP TESTBED

Configuration pour Zone.js (détection de changements traditionnelle)

```ts
// src/test-setup.ts
import '@analogjs/vitest-angular/setup-zone';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting,
} from '@angular/platform-browser-dynamic/testing';
import { getTestBed } from '@angular/core/testing';

getTestBed().initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting()
);
```

Elodie TURLIER
Angular, IA, agilité & storytelling

Configuration pour Zoneless (détection de changements sans Zone.js - Angular 18+)

```ts
// src/test-setup.ts
import '@analogjs/vitest-angular/setup-snapshots';
import { provideZonelessChangeDetection, NgModule }
from '@angular/core';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting,
} from '@angular/platform-browser-dynamic/testing';
import { getTestBed } from '@angular/core/testing';

@NgModule({
  providers: [provideZonelessChangeDetection()],
})
export class ZonelessTestModule {}

getTestBed().initTestEnvironment(
  [BrowserDynamicTestingModule, ZonelessTestModule],
  platformBrowserDynamicTesting()
);
```

Elodie TURLIER
Angular, IA, agilité & storytelling

ÉTAPE 6 : ADAPTER LES SCRIPTS

Mise à jour des scripts package.json

```
{
  "scripts": {
    "test": "vitest", // Mode watch
    "test:run": "vitest run", // Exécution unique
    "test:ui": "vitest --ui", // Interface graphique
    "test:coverage": "vitest run --coverage" // Rapport de couverture
  }
}
```

Elodie TURLIER
Angular, IA, agilité & storytelling

09

ÉTAPE 7 : CONVERTIR LE CODE DE TEST

1. Adapter la syntaxe des tests

La plupart des tests écrits avec Jasmine/Karma sont compatibles avec Vitest.

Cependant, il faut :

- S’assurer que les globales (describe, it, expect, etc.) sont bien disponibles (via globals: true dans la config ou en important depuis vitest).
- Remplacer les imports Jasmine spécifiques par ceux de Vitest si nécessaire.

2. Adapter les spies

```
// Jasmine
spyOn(service, 'method').and.returnValue('mock');

// Vitest
spyOn(service, 'method').and.returnValue('mock');
```

Elodie TURLIER
Angular, IA, agilité & storytelling

3. Nettoyer les helpers Jasmine

Supprimez ou remplacez les helpers Jasmine comme jasmine.createSpy, jasmine.any, etc., par les équivalents Vitest (vi.fn(), expect.any(), etc.).

4. Vérifier la compatibilité des assertions

- toBeTrue() → toBe(true)
- toBeFalse() → toBe(false)
- toHaveBeenCalledWith() reste identique

5. Utiliser les bonnes pratiques Vitest

- Utilisez vi.resetAllMocks() dans les hooks beforeEach pour éviter les interférences entre tests.
- Pour restaurer les spies, utilisez mockRestore().

6. Lancer et ajuster les tests

Elodie TURLIER
Angular, IA, agilité & storytelling

RÉSULTAT FINAL

- **Plus aucune trace de Karma/Jasmine**

Les anciennes dépendances et fichiers de config sont supprimés.

- **Vitest est installé et opérationnel**

Intégré avec AnalogJS, prêt à exécuter tous tes tests Angular.

- **La configuration est claire et centralisée**

→ vite.config.mts, test-setup.ts, tsconfig.spec.json sont à jour.

- **Ton code de test utilise l'API Vitest**

→ vi.spyOn, vi.fn, expect… avec une syntaxe simple et lisible.

- **Une expérience développeur fluide**

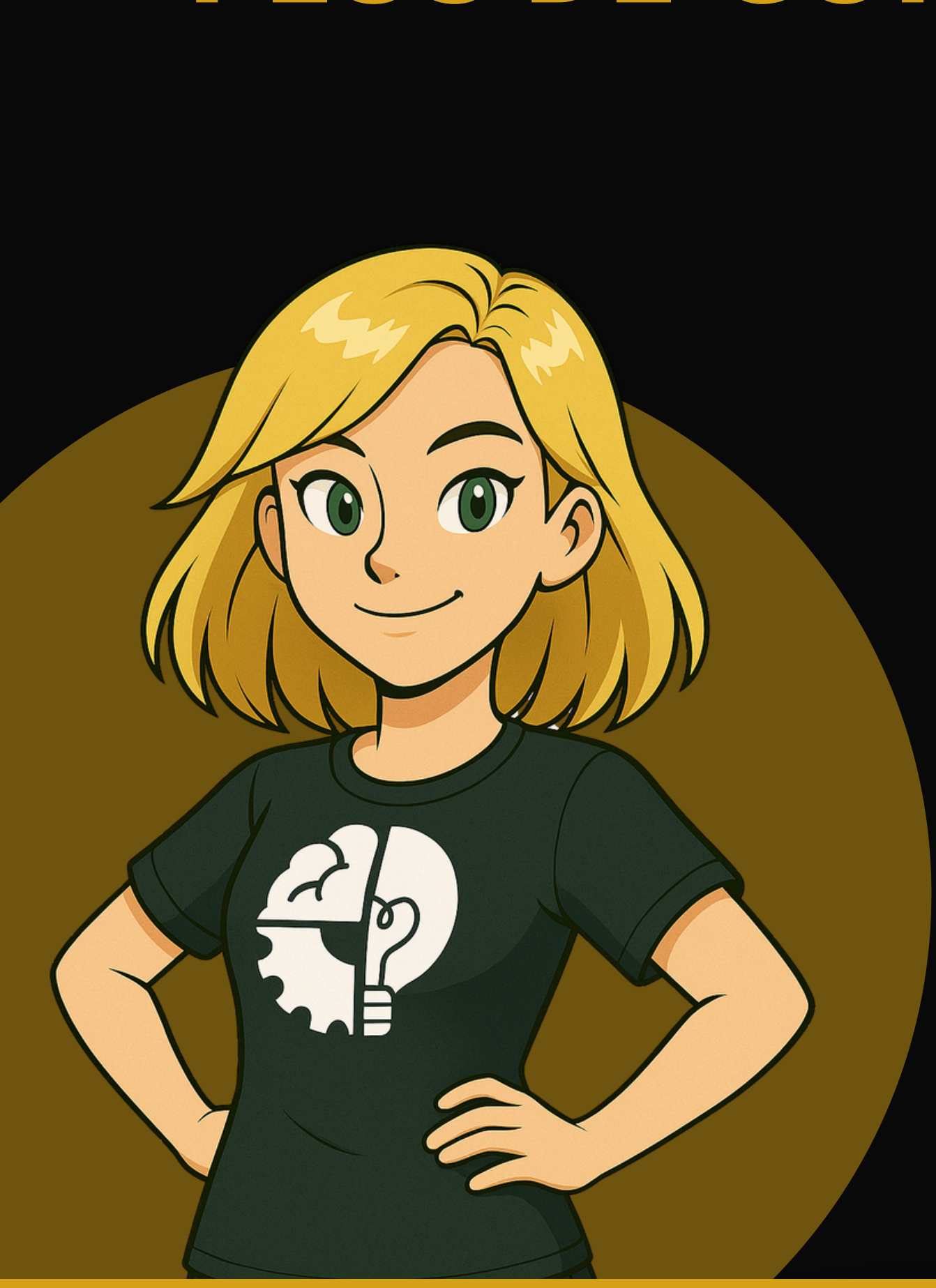
→ Tests rapides, interface graphique, couverture de code facile à activer.

- **Pour en savoir plus**

https://analogjs.org/docs/features/testing/vitest

Elodie TURLIER
Angular, IA, agilité & storytelling

ABONNE TOI POUR PLUS DE CONTENU



Elodie TURLIER

Angular, IA, agilité & storytelling : construire des apps prêtes à survivre à l’apocalypse digitale

Enregistre ou partage si ça t’a été utile