BOUKHIMA LILIA & Mahrez BOUCHELOUCHE

# Rapport – Tests End-to-End avec Playwright

**Projet :** Tests de l'application *TodoList*
**Application testée :** https://alexdmr.github.io/l3m-2023-2024-angular-todolist/

## 1. Installation et Configuration

### 1.1 Installation de Playwright

L'installation de Playwright a été effectuée à l'aide de la commande :

npm init playwright@latest

Cette commande génère automatiquement :

- un dossier **`tests/`** contenant les scénarios ;
- un fichier **`playwright.config.ts`** pour la configuration ;
- les navigateurs nécessaires (Chromium, Firefox, WebKit).

### 1.2 Adoption du modèle CBT (Component-Based Testing)

La structure adoptée suit l'architecture **CBT**, permettant de séparer clairement :

- **Les actions** → implémentées dans le *Page Object Model*
- **Les vérifications** → regroupées dans des fonctions dédiées
- **Les scénarios** → écrits en langage naturel, lisibles et expressifs

Cette méthodologie améliore la lisibilité, la maintenance et la réutilisabilité du code.

## 2. Modèle de Données

**Fichier : `data/todolist.data.ts`**

```
export type TodoListData = readonly TodoItem[];

export interface TodoItem {
  readonly label: string;
  readonly completed: boolean;
```

```
}

export function areTdlSimilar(
  tdl1: TodoListData,
  tdl2: TodoListData
): boolean {
  if (tdl1.length !== tdl2.length) return false;

  for (let i = 0; i < tdl1.length; i++) {
    if (tdl1[i].label !== tdl2[i].label ||
        tdl1[i].completed !== tdl2[i].completed) {
      return false;
    }
  }
  return true;
}
```

### Description

- **`TodoItem` :** représente une tâche (texte + état).
- **`TodoListData` :** liste immuable de tâches.
- **`areTdlSimilar`** : permet de comparer deux listes pour vérifier qu'elles sont identiques.

## 3. Actions – Page Object Model

**Fichier :** `pages/TodoListPage.ts`

### Actions disponibles

| Action | Méthode | Description |
|---|---|---|
| Naviguer | `naviguer()` | Charge l'application |
| Ajouter | `ajouterTache(texte)` | Ajoute une nouvelle tâche |
| Cocher | `cocherTache(index)` | Coche une tâche |
| Supprimer | `supprimerTache(index)` | Supprime une tâche |
| Filtrer actifs | `cliquerFiltreActifs()` | Affiche les tâches actives |

| | | |
|---|---|---|
| Filtrer complétés | `cliquerFiltreCompletees()` | Affiche les tâches complétées |
| Filtrer tous | `cliquerFiltreTous()` | Affiche toutes les tâches |

### Exemple d'implémentation

```
export class TodoListPage {
  readonly page: Page;
  readonly input: Locator;
  readonly listeTaches: Locator;

  constructor(page: Page) {
    this.page = page;
    this.input = page.locator('input[placeholder*="faire"]');
    this.listeTaches = page.locator('ul li');
  }

  async ajouterTache(texte: string) {
    await this.input.fill(texte);
    await this.input.press('Enter');
    await this.page.waitForTimeout(300);
  }

  async supprimerTache(index: number) {
    const item = this.listeTaches.nth(index);
    await item.hover(); // Révèle le bouton X
    const deleteBtn = item.locator('button.destroy').first();
    await deleteBtn.click();
  }
}
```

**Principe :** le Page Object représente uniquement *comment agir*, jamais *quoi vérifier*.

## 4. Fonctions de Vérification

**Fichier :** `verifs/todolist.verif.ts`

### 4.1 Extraction des données

```
export async function extractTdlFromPage(page: Page): Promise<TodoListData> {
  const jsonText = await page.locator('pre').first().textContent();
  if (!jsonText) throw new Error('JSON non trouvé');

  const data = JSON.parse(jsonText);
  return data.items.map((item: any) => ({
    label: item.label,
    completed: item.done
```

```
  }));
}
```

Cette fonction lit le JSON affiché sur la page (Étape 1 du site).

### 4.2 Vérifications disponibles

| Fonction | Description |
|---|---|
| `verifierLeNombreDeTaches(page, n)` | Vérifie le nombre total de tâches dans le JSON |
| `verifierQueLaTodoListContient(page, taches)` | Compare la liste attendue et la liste réelle |
| `verifierLeNombreDeTachesVisibles(page, n)` | Vérifie les tâches réellement visibles |

### Exemple

```
export async function verifierQueLaTodoListContient(
  page: Page,
  tachesAttendues: TodoListData
) {
  const tachesReelles = await extractTdlFromPage(page);

  expect(tachesReelles.length).toBe(tachesAttendues.length);
  expect(areTdlSimilar(tachesReelles, tachesAttendues)).toBeTruthy();
}
```

## 5. Scénarios de Test

**Fichier :** `tests/todolist.spec.ts`

### Scénarios écrits

1. Ajouter une tâche simple
2. Filtrer les tâches actives
3. Supprimer une tâche
4. Cocher une tâche
5. Afficher uniquement les tâches complétées
6. Afficher toutes les tâches avec le filtre “Tous”

### Exemple de scénario

```
test('devrait ajouter une tâche avec un texte simple', async ({ page }) => {
  // Arrange
  const texte = 'Acheter du lait';

  // Act
  await todoPage.ajouterTache(texte);

  // Assert
  await verifierLeNombreDeTaches(page, 1);
  await verifierQueLaTodoListContient(page, [
    { label: 'Acheter du lait', completed: false }
  ]);

  const compteurTexte = await todoPage.obtenirTexteCompteur();
  expect(compteurTexte).toMatch(/\b1\b/);
});
```

Chaque test respecte la structure **Arrange – Act – Assert**.

## 8. Résultats d’Exécution

### Commande

npx playwright test

### Résultat

Running 6 tests using 4 workers

✓ devrait ajouter une tâche avec un texte simple
✓ devrait afficher seulement les tâches actives avec le filtre
✓ devrait supprimer une tâche
✓ devrait cocher une tâche et mettre à jour le JSON
✓ devrait afficher seulement les tâches complétées
✓ devrait afficher toutes les tâches avec le filtre Tous

6 passed (42.0s)

**Tous les tests sont passés avec succès.**

## 9. Dépôt GitHub

**Lien du dépôt :**

**Contenu :**

- Code source des tests
- Configuration Playwright
- Documentation et instructions d’exécution

## Conclusion

Ce projet a permis de :

- Découvrir et maîtriser Playwright pour les tests E2E
- Appliquer efficacement le modèle **CBT**
- Produire des scénarios lisibles en langage naturel
- Séparer clairement actions, vérifications et scénarios
- Améliorer la qualité et la maintenabilité des tests

Les tests couvrent l’ensemble des fonctionnalités principales de l’application *TodoList*.