

# Faculté des Sciences de Sfax

Département Informatique



## Rapport de Projet de Test Logiciel

*Automatisation Avancée*

**Sujet :**

Automatisation Intelligente du Portail `data.gov.tn` :  
Intégration LLM et Gestion Jira

**Réalisé par :**

Mohamed Reguigui

Firas Taktak

1<sup>ère</sup> Année Ingénieur

**Enseignant :**

M. Taher Labidi

**Année Universitaire : 2025-2026**

# Table des matières

<b>1</b>	<b>Introduction et Contexte du Projet</b>	<b>3</b>
1.1	Présentation du Projet . . . . .	3
1.2	Cahier des Charges et Objectifs . . . . .	3
1.3	Architecture Technique . . . . .	3
<b>2</b>	<b>Innovations : Bypass CAPTCHA et IA</b>	<b>5</b>
2.1	Stratégie de Contournement du ReCAPTCHA . . . . .	5
2.1.1	Phase 1 : Résolution via DrissionPage . . . . .	5
2.1.2	Phase 2 : Injection de Cookies dans Selenium . . . . .	5
2.2	Intégration LLM (Gemini) et Jira . . . . .	6
<b>3</b>	<b>Analyse Détaillée des Suites de Tests</b>	<b>7</b>
3.1	Suite de Tests : Authentification & Sécurité . . . . .	7
3.1.1	Flux Complet Connexion/Déconnexion . . . . .	7
3.1.2	Tentative de Connexion Invalide . . . . .	8
3.1.3	Injection SQL sur le Login . . . . .	8
3.1.4	Tentative XSS (Cross Site Scripting) . . . . .	9
3.2	Suite de Tests : Formulaire de Contact . . . . .	10
3.2.1	Disponibilité du Formulaire de Contact . . . . .	10
3.2.2	Interaction avec les Champs (Sans Envoi) . . . . .	10
3.2.3	Limites : Champs Vides . . . . .	11
3.2.4	Limites : Chaînes Longues et Caractères Spéciaux . . . . .	11
3.3	Suite de Tests : Moteur de Recherche . . . . .	12
3.3.1	Recherche Basique (Mot-clé Valide) . . . . .	12
3.3.2	Persistance de la Recherche avec Pagination . . . . .	12
3.3.3	Recherche Avancée (Opérateurs Booléens) . . . . .	13
3.3.4	Recherche : Cas Extrêmes (Caractères Spéciaux) . . . . .	13
3.4	Suite de Tests : Consultation des Données . . . . .	14
3.4.1	Détails d'un Dataset (End-to-End) . . . . .	14
3.4.2	Accessibilité du Lien de Téléchargement . . . . .	14
3.4.3	Fil d'Ariane (Breadcrumb) . . . . .	15
3.4.4	Présence des Métadonnées . . . . .	15

3.5	Suite de Tests : Performance et Charge . . . . .	16
3.5.1	Performance : Recherches Multiples . . . . .	16
3.5.2	Stress : Recherches Concurrentes/Rapides . . . . .	16
3.5.3	Performance : Chargement Pages Statiques . . . . .	16
3.6	Suite de Tests : Compatibilité et Responsive . . . . .	18
3.6.1	Layout Responsive : Page d'Accueil . . . . .	18
3.6.2	Smoke Test Cross-Browser . . . . .	18
3.6.3	Transitions Responsive . . . . .	19
3.7	Suite de Tests : Reproduction de Bugs (Non-Régression) . . .	20
3.7.1	Bug #3 : Lien API Brisé . . . . .	20
3.7.2	Bug #4 : Lien "Mes Tâches" Fantôme . . . . .	20
3.7.3	Bug #2 : Langue Page Email . . . . .	21
3.8	Suite de Tests : Contenu Statique et Monitoring . . . . .	22
3.8.1	Pages Statiques Multiples . . . . .	22
3.8.2	FAQ : Recherche . . . . .	22
3.8.3	FAQ : Récupération Contenu . . . . .	22
3.8.4	Monitoring Automatique . . . . .	23
<b>4</b>	<b>Conclusion et Perspectives</b>	<b>24</b>
4.1	Points Forts du Projet . . . . .	24

# Chapitre 1

## Introduction et Contexte du Projet

### 1.1 Présentation du Projet

Ce projet s'inscrit dans le cadre du module "Test Logiciel et vise à auditer la qualité et la robustesse du portail gouvernemental de données ouvertes **data.gov.tn**.

L'objectif est de mettre en œuvre une stratégie de test automatisé complète, respectant les principes du test "Boîte Noire" (Black Box Testing), tout en intégrant des technologies de pointe pour résoudre des défis modernes tels que les CAPTCHA et la génération de tests par IA.

### 1.2 Cahier des Charges et Objectifs

Conformément aux exigences, le projet couvre :

- La création de plus de **20 cas de tests** (positifs et négatifs)
- L'exécution de **4 suites de tests** : Performance, Cross-browser, Responsive Design, et Stress Tests
- L'utilisation de techniques sophistiquées comme l'analyse des valeurs limites et les transitions d'état
- La rédaction automatisée de rapports de bugs vers **Jira**

### 1.3 Architecture Technique

L'architecture repose sur une structure modulaire **Page Object Model (POM)** :

- **Pages** : Encapsulation de la logique (`auth_page.py`, `contact_page.py`)
- **Tests** : Séparation par type (Fonctionnel, Performance, Responsive, Stress)
- **Utils** : Outils de monitoring UI et de gestion de CAPTCHA
- **Docker** : Orchestration d'une grille Selenium (Hub + Nodes Chrome/-Firefox/Edge) pour les tests parallèles

# Chapitre 2

## Innovations : Bypass CAPTCHA et IA

### 2.1 Stratégie de Contournement du ReCAPTCHA

L'un des défis majeurs des tests automatisés sur les sites gouvernementaux est la présence de protections anti-robots (Google reCAPTCHA). Une approche standard avec Selenium échoue systématiquement car elle est détectée comme non-humaine.

Nous avons développé une solution hybride innovante détaillée dans `RecaptchaSolver` et `captcha_login_helper.py`.

#### 2.1.1 Phase 1 : Résolution via DrissionPage

Au lieu d'utiliser Selenium pour le login, nous utilisons **DrissionPage**, une bibliothèque capable de contrôler le navigateur de manière plus furtive.

1. Le script navigue vers la page de login
2. Il utilise un module de reconnaissance audio (`speech_recognition` + `ffmpeg`) pour résoudre le challenge audio du CAPTCHA
3. Une fois le CAPTCHA validé, le login est effectué

#### 2.1.2 Phase 2 : Injection de Cookies dans Selenium

Une fois authentifié dans DrissionPage, nous extrayons les cookies de session (`login_cookies.pkl`). Ces cookies sont ensuite injectés dynamiquement dans le WebDriver Selenium standard au début des tests.

**Résultat :** Les tests Selenium démarrent directement en état "Connecté", contournant totalement la nécessité de résoudre le CAPTCHA à chaque test.

## 2.2 Intégration LLM (Gemini) et Jira

Pour aligner le projet avec les tendances actuelles du QA (Quality Assurance) :

- **Génération de Tests :** Le script `generate_test_cases.py` utilise l'API Gemini pour transformer des User Stories en scénarios de tests concrets
- **Jira Automation :** Le script `jira_automation_enhanced.py` détecte chaque échec de test Pytest et crée automatiquement un ticket détaillé dans Jira contenant les logs et les étapes de reproduction

# Chapitre 3

## Analyse Détaillée des Suites de Tests

Ce chapitre détaille chaque cas de test implémenté, en expliquant son alignement avec les principes du test boîte noire.

### 3.1 Suite de Tests : Authentification & Sécurité

#### 3.1.1 Flux Complet Connexion/Déconnexion

<b>ID</b> : test_complete_login_logout_flow Transition d'État	<b>Type</b> : Fonctionnel /
--	-----------------------------

**Objectif** : Vérifier qu'un utilisateur peut se connecter avec des identifiants valides (après bypass CAPTCHA) et se déconnecter proprement.

**Principe Black Box** : Ce test valide la **Transition d'État**. Le système doit passer de l'état "Invité" à "Connecté", puis revenir à "Invité". On ne teste pas le code de session PHP/Python, mais le comportement visible.

**Scénario** :

1. Injecter les cookies de session pré-authentifiés (Bypass CAPTCHA)
2. Vérifier l'accès au Dashboard
3. Cliquer sur le bouton de déconnexion
4. Confirmer la déconnexion dans la modale

**Résultat Attendu** : L'assertion `is_logged_in()` doit passer de `True` à



False. L'utilisateur est redirigé vers la page d'accueil.

### 3.1.2 Tentative de Connexion Invalide

<b>ID :</b> test_login_failure_with_invalid_credentials Sécurité / Cas Négatif	<b>Type :</b>
---	---------------

**Objectif :** Vérifier que le système refuse l'accès avec des identifiants incorrects.

**Principe Black Box : Partitionnement d'Équivalence :** On teste la classe des entrées "invalides".

**Scénario :**

1. Naviguer vers /login
2. Saisir User : "invalid\_user", Pass : "invalid\_pass"
3. Soumettre le formulaire

**Résultat Attendu :** L'utilisateur reste sur la page login. Un message d'erreur est détecté. is\_logged\_in() reste False.

### 3.1.3 Injection SQL sur le Login

<b>ID :</b> test_login_with_sql_injection_attempts Boundary	<b>Type :</b> Sécurité /
--	--------------------------

**Objectif :** Vérifier la robustesse du formulaire contre les injections SQL basiques.

**Principe Black Box : Boundary Testing (Sécurité) :** Tester les limites de la sanitization des entrées.

**Scénario :**

1. Itérer sur une liste de payloads : ' OR '1'='1, admin' -, etc.
2. Tenter le login avec chaque payload comme mot de passe

**Résultat Attendu :** Aucun payload ne doit permettre la connexion. Le système ne doit pas crasher (Erreur 500).

### 3.1.4 Tentative XSS (Cross Site Scripting)

**ID** : test\_xss\_attempts\_in\_contact\_form\_check **Type** : Sécurité  
/ Input Validation

**Objectif** : Vérifier que les champs de formulaire acceptent ou rejettent les scripts malveillants.

**Principe Black Box : Validation des Entrées** : S'assurer que le système gère les caractères spéciaux HTML.

**Scénario** :

1. Aller sur la page Contact
2. Remplir le message avec `<script>alert('XSS')</script>`
3. Vérifier la réaction du champ (sans soumission pour éviter le spam)

**Résultat Attendu** : Le champ doit soit sanitizer l'entrée, soit l'accepter comme texte brut, sans exécuter le script.

## 3.2 Suite de Tests : Formulaire de Contact

### 3.2.1 Disponibilité du Formulaire de Contact

<b>ID :</b> test_contact_page_load_success Smoke	<b>Type :</b> Fonctionnel /
---	-----------------------------

**Objectif :** Vérifier que la page de contact est accessible.

**Principe Black Box : Smoke Testing :** Vérification basique de la disponibilité.

**Scénario :**

1. Naviguer vers l'URL de contact
2. Vérifier le titre de la page et l'URL

**Résultat Attendu :** L'URL contient "contact" et le code HTTP est 200.

### 3.2.2 Interaction avec les Champs (Sans Envoi)

<b>ID :</b> test_contact_form_fields_interactions Fonctionnel / UI	<b>Type :</b>
---	---------------

**Objectif :** Vérifier que tous les champs (Nom, Email, Sujet, Message) sont inscriptibles.

**Principe Black Box : Test d'Interface :** Vérifier l'interaction utilisateur sans déclencher le backend.

**Scénario :**

1. Identifier chaque élément input
2. Envoyer des clés (send\_keys) de test

**Résultat Attendu :** Aucune exception levée lors de la saisie. Les valeurs sont présentes dans les champs.

### 3.2.3 Limites : Champs Vides

<b>ID :</b> test_contact_form_with_empty_values_check Fonctionnel / Cas Négatif	<b>Type :</b>
--	---------------

**Objectif :** Vérifier le comportement du formulaire avec des données vides.

**Principe Black Box : Cas aux Limites :** Tester la limite inférieure (longueur 0).

**Scénario :**

1. Laisser tous les champs vides
2. Vérifier si des messages de validation HTML5 apparaissent (attribut **required**)

**Résultat Attendu :** Le navigateur ou le JS doit empêcher la soumission ou marquer les champs en erreur.

### 3.2.4 Limites : Chaînes Longues et Caractères Spéciaux

<b>ID :</b> test_boundary_values_in_contact_form_check Fonctionnel / Boundary	<b>Type :</b>
--	---------------

**Objectif :** Vérifier la gestion des textes très longs (255+ chars).

**Principe Black Box : Analyse des Valeurs Limites :** Tester la capacité maximale des buffers d'entrée.

**Scénario :**

1. Remplir le champ "Sujet" avec 255 caractères
2. Remplir "Message" avec 1000 caractères

**Résultat Attendu :** L'interface doit accepter la saisie sans lag excessif ni troncature immédiate visible.

## 3.3 Suite de Tests : Moteur de Recherche

### 3.3.1 Recherche Basique (Mot-clé Valide)

**ID :** test\_search\_with\_valid\_keyword\_returns\_results **Type :**  
Fonctionnel / Nominal

**Objectif :** Vérifier qu'une recherche simple retourne des résultats.

**Principe Black Box : Cas Nominal :** Le chemin heureux de l'utilisateur.

**Scénario :**

1. Naviguer vers la page de recherche
2. Saisir "budget"
3. Valider

**Résultat Attendu :** Le compteur de résultats est supérieur à 0. La liste des datasets n'est pas vide.

### 3.3.2 Persistance de la Recherche avec Pagination

**ID :** test\_search\_results\_persist\_after\_pagination **Type :**  
Fonctionnel / Navigation

**Objectif :** Vérifier que les résultats de recherche ne sont pas perdus en changeant de page.

**Principe Black Box : Test de Navigation/État :** S'assurer que les paramètres GET sont conservés.

**Scénario :**

1. Rechercher "data"
2. Cliquer sur le bouton "Suivant" (Page 2)

**Résultat Attendu :** L'URL contient page=2 et le terme de recherche. Des résultats sont toujours affichés.

### 3.3.3 Recherche Avancée (Opérateurs Booléens)

**ID :** test\_search\_with\_boolean\_operators    **Type :** Fonctionnel / Avancé

**Objectif :** Vérifier la gestion des requêtes complexes ("santé et éducation").

**Principe Black Box : Test de Logique Métier :** Vérifier l'interprétation de la requête par le moteur (Solr/Elasticsearch).

**Scénario :**

1. Saisir "santé et éducation"
2. Lancer la recherche

**Résultat Attendu :** Le système ne plante pas. Il retourne des résultats pertinents ou zéro, mais pas d'erreur.

### 3.3.4 Recherche : Cas Extrêmes (Caractères Spéciaux)

**ID :** test\_search\_with\_special\_characters    **Type :** Robustesse

**Objectif :** Vérifier que le moteur ne crashe pas avec des entrées "garbage".

**Principe Black Box : Test de Robustesse :** Injection de bruit.

**Scénario :**

1. Rechercher des caractères spéciaux variés

**Résultat Attendu :** La zone de résultat reste visible. Pas d'erreur 500 Server Error.

## 3.4 Suite de Tests : Consultation des Données

### 3.4.1 Détails d'un Dataset (End-to-End)

**ID :** test\_cross\_browser\_end\_to\_end\_dataset\_details    **Type :** Cross-Browser / E2E

**Objectif :** Vérifier le flux complet : Accueil, Recherche, Détail Dataset.

**Principe Black Box : Test de Parcours Utilisateur :** Valider l'intégration des composants.

**Scénario :**

1. Rechercher "transport"
2. Cliquer sur le premier résultat
3. Vérifier le chargement de la fiche

**Résultat Attendu :** La page détail affiche un titre non vide et une liste de ressources (fichiers).

### 3.4.2 Accessibilité du Lien de Téléchargement

**ID :** test\_dataset\_resources\_downloadable    **Type :** Fonctionnel

**Objectif :** Vérifier que les boutons de téléchargement ("Télécharger") sont cliquables.

**Principe Black Box : Test d'Accessibilité Fonctionnelle.**

**Scénario :**

1. Ouvrir un dataset contenant des ressources
2. Localiser le bouton de téléchargement de la première ressource
3. Tenter le clic

**Résultat Attendu :** L'action est interceptée sans erreur. L'URL cible est valide.

### 3.4.3 Fil d'Ariane (Breadcrumb)

**ID :** test\_breadcrumb\_allows\_return\_to\_search  
Navigation / UI

**Type :**

**Objectif :** Vérifier que l'utilisateur peut remonter l'arborescence via le fil d'ariane.

**Principe Black Box : Test d'Usabilité :** Vérifier la navigation hiérarchique.

**Scénario :**

1. Aller sur un détail de dataset
2. Cliquer sur le lien parent (ex : "Organisations" ou "Recherche")

**Résultat Attendu :** Le navigateur revient à la page précédente ou parente correcte.

### 3.4.4 Présence des Métadonnées

**ID :** test\_dataset\_has\_metadata

**Type :** Contenu

**Objectif :** Vérifier l'affichage des méta-informations (Licence, Auteur, Date).

**Principe Black Box : Vérification de Contenu.**

**Scénario :**

1. Scanner le DOM de la page dataset
2. Chercher les balises contenant les métadonnées

**Résultat Attendu :** Au moins un champ de métadonnée (Organisation, Licence) est présent et visible.



## 3.5 Suite de Tests : Performance et Charge

### 3.5.1 Performance : Recherches Multiples

<b>ID :</b> test_search_performance_multiple_queries Performance	<b>Type :</b>
---	---------------

**Objectif :** Mesurer le temps de réponse moyen pour une série de mots-clés variés.

**Principe Black Box : Test de Performance :** Mesure des temps de réponse backend.

**Scénario :**

1. Lancer des recherches pour "education", "health", "transport"
2. Mesurer le temps pour chaque requête
3. Calculer la moyenne

**Résultat Attendu :** Le temps moyen doit être inférieur au PERFORMANCE\_THRESHOLD (15s ou 5s selon config).

### 3.5.2 Stress : Recherches Concurrentes/Rapides

<b>ID :</b> test_concurrent_search_load	<b>Type :</b> Stress Test
---	---------------------------

**Objectif :** Vérifier la stabilité du site sous une charge de requêtes rapides.

**Principe Black Box : Stress Testing :** Tester la résilience.

**Scénario :**

1. Exécuter une série de recherches en succession rapide sans pause

**Résultat Attendu :** Le site ne doit pas rejeter les connexions. La zone de résultat doit rester cohérente.

### 3.5.3 Performance : Chargement Pages Statiques

<b>ID :</b> test_static_pages_load_performance	<b>Type :</b> Performance
--	---------------------------

**Objectif :** Mesurer le temps de chargement des pages "À propos", "Conditions", etc.

**Principe Black Box : Benchmarking.**

**Scénario :**

1. Naviguer vers chaque page statique
2. Enregistrer le temps de chargement (Load Event)

**Résultat Attendu :** Les temps sont loggués pour analyse. Pas d'échec si le site est lent, mais une alerte est générée.

## 3.6 Suite de Tests : Compatibilité et Responsive

### 3.6.1 Layout Responsive : Page d'Accueil

**ID :** test\_homepage\_responsive\_layout    **Type :** UI / Responsive

**Objectif :** Vérifier l'adaptation de la Home Page sur Mobile, Tablette et Desktop.

**Principe Black Box :** Test d'Adaptabilité UI.

**Scénario :**

1. Redimensionner le viewport : 375x812 (Mobile), 768x1024 (Tablette), 1920x1080 (Desktop)
2. Vérifier la visibilité des éléments clés à chaque taille

**Résultat Attendu :** La page charge sans erreur. La largeur de la fenêtre correspond à la consigne.

### 3.6.2 Smoke Test Cross-Browser

**ID :** test\_smoke\_cross\_browser    **Type :** Compatibilité

**Objectif :** Assurer le fonctionnement minimal sur Chrome, Firefox et Edge.

**Principe Black Box :** Test de Compatibilité Navigateur : Exécuté via Docker Grid.

**Scénario :**

1. Lancer le test simultanément sur les nœuds Docker Chrome, Firefox, Edge
2. Vérifier le titre de la page d'accueil

**Résultat Attendu :** Le site charge correctement sur les trois moteurs de rendu.

### 3.6.3 Transitions Responsive

<b>ID :</b> test_responsive_element_visibility_transitions <b>Type :</b> UI / Dynamique
--

**Objectif :** Vérifier que les éléments s'adaptent dynamiquement lors du redimensionnement de la fenêtre.

**Principe Black Box :** Test de Fluidité UI.

**Scénario :**

1. Charger en Desktop
2. Redimensionner progressivement vers Mobile

**Résultat Attendu :** Le titre et le contenu restent accessibles après la transition.

## 3.7 Suite de Tests : Reproduction de Bugs (Non-Régression)

### 3.7.1 Bug #3 : Lien API Brisé

**ID :** test\_api\_link\_is\_broken

**Type :** Bug Reproduction

**Objectif :** Confirmer que le lien "API" du footer mène à une page d'erreur JSON brut.

**Principe Black Box :** Test de Confirmation de Défaut.

**Scénario :**

1. Aller sur le catalogue
2. Cliquer sur "API"
3. Lire le contenu de la page

**Résultat Attendu :** Le contenu est exactement {"version": 3}, confirmant l'absence de documentation HTML.

### 3.7.2 Bug #4 : Lien "Mes Tâches" Fantôme

**ID :** test\_mes\_taches\_link\_appears\_and\_is\_broken **Type :** Bug Reproduction / État complexe

**Objectif :** Reproduire l'apparition d'un lien brisé dans le menu après visite de l'API.

**Principe Black Box :** Test de Séquence Complexe.

**Scénario :**

1. Se connecter (Bypass Captcha)
2. Visiter la page API
3. Ouvrir le menu utilisateur
4. Vérifier présence lien "Mes tâches"
5. Cliquer

**Résultat Attendu :** Le lien apparaît (Anomalie 1) et mène à une 404/Erreur (Anomalie 2).

### 3.7.3 Bug #2 : Langue Page Email

**ID :** test\_modify\_email\_page\_language\_bug  
Reproduction / Localization

**Type :** Bug

**Objectif :** Vérifier que la page "Modifier Email" passe en Anglais de force.

**Principe Black Box : Test de Localisation.**

**Scénario :**

1. Se connecter
2. Aller sur "Modifier mon E-mail"
3. Vérifier le titre H1

**Résultat Attendu :** Le titre est "Change E-mail" au lieu de "Modifier l'E-mail", confirmant le bug de langue.

## 3.8 Suite de Tests : Contenu Statique et Monitoring

### 3.8.1 Pages Statiques Multiples

<b>ID :</b> test_multiple_static_pages_load Contenu	<b>Type :</b> Fonctionnel /
--	-----------------------------

**Objectif :** Vérifier en boucle le chargement de "About", "Terms", "Licenses".

**Principe Black Box :** Vérification de Disponibilité de Contenu.

**Scénario :**

1. Itérer sur la liste des URLs statiques
2. Vérifier le code 200 et la présence de texte supérieur à 50 caractères

**Résultat Attendu :** Toutes les pages doivent charger un contenu significatif.

### 3.8.2 FAQ : Recherche

<b>ID :</b> test_faq_search_functionality	<b>Type :</b> Fonctionnel
---	---------------------------

**Objectif :** Vérifier que la barre de recherche de la FAQ est utilisable.

**Principe Black Box :** Test Fonctionnel UI.

**Scénario :**

1. Aller sur la FAQ
2. Taper "data" dans le champ de recherche FAQ

**Résultat Attendu :** L'input est présent et interactif.

### 3.8.3 FAQ : Récupération Contenu

<b>ID :</b> test_faq_questions_and_answers_retrieval Contenu / Scraping	<b>Type :</b>
--	---------------

**Objectif :** Vérifier que les questions et réponses sont bien structurées dans le DOM.

**Principe Black Box :** Validation de Structure HTML.

**Scénario :**

1. Extraire toutes les questions (balises spécifiques)
2. Extraire toutes les réponses

**Résultat Attendu :** Les listes ne sont pas vides. Le contenu textuel est lisible.

### 3.8.4 Monitoring Automatique

**ID :** test\_search\_with\_auto\_monitoring  
Monitoring

**Type :** Outil /

**Objectif :** Démontrer l'utilisation du décorateur de monitoring UI automatique.

**Principe Black Box :** Validation de l'Outillage de Test.

**Scénario :**

1. Activer le fixture `auto_setup_monitoring`
2. Effectuer une recherche standard

**Résultat Attendu :** Le test passe et un rapport de changement d'état UI (DOM changes) est généré en arrière-plan.



# Chapitre 4

## Conclusion et Perspectives

Ce projet a permis d'établir une couverture de test significative sur le portail `data.gov.tn`. L'approche hybride combinant **Selenium** pour la robustesse des tests et **DrissionPage** pour l'intelligence du bypass CAPTCHA s'est avérée décisive.

L'ajout de l'IA pour la génération des tests et l'automatisation des tickets Jira positionne ce framework comme une solution moderne, prête pour le CI/CD.

### 4.1 Points Forts du Projet

- **Contournement intelligent du CAPTCHA** via une approche hybride innovante combinant DrissionPage et Selenium
- **Architecture modulaire** basée sur le Page Object Model (POM) pour une maintenance facilitée
- **Couverture exhaustive** avec plus de 20 cas de test couvrant les aspects fonctionnels, de sécurité, de performance et d'ergonomie
- **Intégration d'outils modernes** : LLM (Gemini) pour la génération automatique de tests, Jira pour le suivi des anomalies, Docker pour l'orchestration
- **Tests multi-navigateurs** garantissant la compatibilité sur Chrome, Firefox et Edge
- **Responsive design testing** vérifiant l'adaptabilité sur mobile, tablette et desktop