Table des matiŁres

Introduction et Contexte du Projet

PrØsentation du Projet
Ce projet automatise des tests end-to-end (Selenium + Pytest) et des audits non-intrusifs (requests) sur le site p

Objectif principal : augmenter la capacitØ àdØtecter des anomalies rØelles (liens cassØs, erreurs JavaScript, rØ

Cahier des Charges et Objectifs
- Construire des suites de tests utiles (et non ?hollow?) orientØes bug hunting.
- Collecter automatiquement des preuves quand un test Øchoue (screenshot, HTML, console logs, erreurs JS).
- Faciliter la crØation d?un bug report Jira (automatique optionnel).
- GØnØrer un rapport PDF structurØ (ce document) pour la soutenance / livrable.

Architecture Technique
- ‘pytest‘ : orchestration, markers, exØcution sØlective.
- ‘selenium‘ : tests UI (navigation, interactions, map, responsive).
- ‘requests‘ : audits rapides (headers sØcuritØ, intØgritØ des liens).
- ‘tests/conftest.py‘ : configuration runtime + collecte d?artifacts en cas d?Øchec + intØgration Jira (optionnelle).
- ‘reports/‘ : rapports (PDF/HTML), screenshots et artifacts de bugs.

Innovations

Phase 1 : Collecte d?Artifacts Automatique
Lors d?un Øchec de test, le framework sauvegarde automatiquement :
- ‘screenshot.png‘
- ‘page.html‘ (HTML complet)
- ‘console.json‘ (logs console navigateur, si disponibles)
- ‘js_errors.json‘ (erreurs runtime collectØes cîtØ client, si supportØ)

Phase 2 : IntØgration Jira (optionnelle)
En activant l?option ‘--jira-create-on-fail‘, un ticket Jira peut Œtre crØØ automatiquement, avec piŁces jointes (ca

Remarque : l?intØgration est volontairement simple et contrôlable par variables d?environnement (pas de secrets

Analyse DØtaillØe des Suites de Tests

Suite 1 : Fonctionnel (Navigation, Pages publiques)
Les tests valident les parcours ?publics? (home/about/data/feed) et le chargement de la carte publique (map).

Suite 2 : Performance (Best-effort)
Tests orientØs temps de chargement et mØtriques navigateur (LCP/CLS/TBT) avec throttling (si support CDP).

Suite 3 : CompatibilitØ (Cross-browser)
Smoke tests (àexØcuter selon le navigateur configurØ via '--browser').

Suite 4 : Responsive
Matrice de viewports (mobile/tablet/desktop) avec :
- screenshots pour revue visuelle
- dØtection de scroll horizontal
- heuristiques ?tap targets?
- checks sur overflow de code blocks

Suite 5 : SØcuritØ (Non-intrusif)
Audit des headers HTTP sur pages publiques (HSTS, nosniff, framing protections, etc.) en mode :
- ?audit? (journalise des findings)
- ?strict? via '--audit-strict' (Øchec du test si manque)

Suite 6 : IntØgritØ des liens
Extraction des URLs depuis le DOM et vØrification HTTP (par dØfaut interne uniquement, optionnellement exter

Suite 7 : Erreurs JavaScript / Console
DØtection d?erreurs client-side (console logs + collecteur runtime) sur pages clØs.

Suite 8 : Fuzzing lØger des inputs (Map search)
Tests nØgatifs sur le champ de recherche de la carte (payloads XSS usuels) avec heuristiques non-intrusives (a

Conclusion et Perspectives

Points Forts du Projet
- Tests orientØs ?bug signals? (erreurs console, liens cassØs, headers sØcuritØ).
- Artifacts automatiques en cas d?Øchec => bug report plus rapide.
- IntØgration Jira (optionnelle) pour industrialiser la remontØe de bugs.

AmØliorations Futures
- Ajout de tests accessibilitØ (axe-core) si l?outillage est autorisØ.
- ExØcution planifiØe (monitoring) + historique de tendances (performances, erreurs JS).
- Enrichissement du rapport PDF par extraction automatique des suites/markers et statistiques d?exØcution.