Centrale Méditerranée

3A - Filière R%D : Projet de recherche

Rapport technique

Conception d'un outil de mesure et d'analyse des performances d'un site web sous forme d'extension navigateur

Arthur GAUTIER
Gaspard LEFORT
Pierre LE MANCQ
Elio DOTTI
Séo HOLLEVILLE

Année 2025-2026

LABORATOIRE
D'INFORMATIQUE
& DES SYSTÈMES
UMR 7020

Référent scientifique : M. François Brucker

Résumé

Ce projet de R&D vise la conception d'une extension navigateur dédiée à la mesure et à l'analyse des performances web en conditions réelles d'usage. L'objectif est de proposer un outil plus pédagogique et plus accessible que les solutions d'audit classiques, tout en conservant des indicateurs techniquement pertinents pour le diagnostic.

Le prototype développé, *Traffic Monitor*, interroge périodiquement une API serveur et agrège les informations de trafic HTTP (méthode, point d'accès, code de statut, durée). Les données sont transformées en indicateurs de synthèse (temps de réponse moyen et maximal, taux de succès/erreur, volume de requêtes, distribution des statuts et méthodes), puis visualisées dans une interface compacte intégrée au navigateur.

La démarche méthodologique combine une revue de littérature sur les métriques de performance web, la conception d'une architecture logicielle légère (popup + logique d'analyse), et une phase d'expérimentation via un serveur de test local. Les résultats montrent la faisabilité d'un suivi continu de la performance, utile pour identifier rapidement des points de ralentissement côté serveur ou réseau. Le module de signaux de sécurité (détection de connexions suspendues de type slowloris) illustre en outre l'intérêt d'une approche unifiée performance/sécurité.

Le prototype reste cependant contraint par la qualité des données remontées par l'API et par le périmètre des endpoints disponibles. Les perspectives portent sur l'extension à davantage de métriques orientées utilisateur (par exemple Core Web Vitals côté client), la comparaison systématique avec des outils de référence, et l'amélioration de l'aide à l'interprétation pour des utilisateurs non experts.

Mots-clés

Performance web ; Extension navigateur ; Mesure réseau ; HTTP ; Analyse de trafic ; Core Web Vitals ; Expérience utilisateur ; Slowloris ; Supervision applicative.

Glossaire

API *Application Programming Interface* : interface permettant l'échange de données entre l'extension et le serveur.

Core Web Vitals Ensemble d'indicateurs de performance orientés expérience utilisateur (ex. LCP, CLS).

Endpoint Point d'accès HTTP (route) exposé par un service web, tel que `/api/traffic`.

Latence Délai observé entre l'envoi d'une requête et la réception d'une réponse.

Polling Mécanisme d'interrogation périodique d'un service pour récupérer des données actualisées.

Slowloris Technique d'attaque applicative consistant à maintenir de nombreuses connexions incomplètes pour saturer un serveur.

Taux de succès Proportion de requêtes aboutissant à un code HTTP de succès (2xx).

TTFB *Time To First Byte* : temps entre l'émission de la requête et la réception du premier octet de réponse.

Table des matières

1 Introduction

La performance web constitue un levier déterminant pour l'expérience utilisateur, le référencement naturel et la sobriété numérique. Des temps de réponse élevés dégradent la satisfaction et peuvent augmenter l'empreinte énergétique d'un service en prolongeant les temps de calcul, de transfert réseau et d'affichage [5, 8, 4].

Dans ce contexte, les outils existants (Lighthouse, WebPageTest, GTmetrix) offrent des analyses riches, mais leurs résultats sont parfois difficiles à interpréter pour un public non spécialiste, en particulier lorsqu'il faut relier un score global à des causes techniques précises [1]. Le projet présenté ici répond à ce besoin en proposant un prototype d'extension navigateur centré sur des métriques lisibles et actionnables.

La problématique étudiée est la suivante : *comment mesurer efficacement les performances d'un site web dans des conditions réelles d'utilisation et identifier simplement les facteurs principaux responsables des ralentissements ?*

L'hypothèse de travail est qu'une extension basée sur l'analyse continue du trafic HTTP et sur des indicateurs de synthèse peut fournir une mesure à la fois fiable, pédagogique et exploitable pour le diagnostic. Le prototype *Traffic Monitor* a été conçu pour tester cette hypothèse.

Le présent rapport détaille : (i) la méthodologie et l'architecture retenues (section 2), (ii) les résultats obtenus lors de tests sur serveur local (section 3), (iii) les limites identifiées et les perspectives d'amélioration (sections 4 et 5).

2 Matériel et méthodes

2.1 Démarche générale

La démarche suivie reprend les étapes de la note de cadrage : revue de littérature, sélection d'indicateurs, conception de l'architecture, développement d'un prototype, puis phase de test et d'analyse. Le choix a été fait de privilégier une approche incrémentale avec un périmètre fonctionnel réduit mais opérationnel.

2.2 Architecture du prototype

Le prototype est implémenté sous forme d'extension Chromium en JavaScript. Son fonctionnement repose sur trois briques principales :

— une interface utilisateur (popup) pour la configuration, l'affichage du trafic et la visualisation des indicateurs ;

— une logique d'analyse agrégée pour extraire les tendances (endpoints dominants, taux de succès, lenteurs) ;

— une API serveur exposant les flux de trafic et métriques, interrogée en *polling*.

L'extension interroge `/api/traffic` toutes les 100 ms en mode connecté, puis met à jour les cartes de synthèse et le graphe des temps de réponse. Un endpoint de santé (`/api/health`) est utilisé pour valider la connectivité. Les données d'authentification (clé API) sont stockées localement via `chrome.storage.local`. Le détail du contrat d'API est fourni en Annexe A.

2.3 Indicateurs retenus

La sélection des indicateurs vise un compromis entre simplicité de compréhension et utilité technique. Les mesures présentées dans l'interface sont :

— temps de réponse moyen et maximal ;

— taux de succès et taux d'erreur ;

— volume total de requêtes et rythme de requêtes par minute ;

— répartition des méthodes HTTP et des codes de statut ;

— classement des endpoints les plus sollicités et des requêtes les plus lentes.

Ces indicateurs s'inscrivent dans la continuité des recommandations issues de la littérature sur la performance perçue, les tests de charge et la planification de capacité [7, 3, 6].

2.4 Protocole expérimental

Un serveur local de test (Node.js/Express) a été utilisé pour simuler le trafic entrant et exposer les endpoints minimaux : `/api/traffic`, `/api/metrics` et `/api/health`. Le protocole comprend :

1. démarrage du serveur de test et connexion de l'extension ;
2. génération manuelle de trafic via des requêtes HTTP répétées ;
3. observation en temps réel des évolutions de métriques ;
4. vérification de la cohérence des agrégations affichées.

En complément, une interface de signaux de sécurité de type slowloris a été intégrée. Les endpoints associés (`/api/slowloris`, `/api/block-ip`, `/api/unblock-ip`, `/api/blocked-ips`) restent optionnels et ne sont pas fournis par le serveur de test standard. La procédure de test reproductible est détaillée en Annexe B.

3 Résultats

3.1 Résultats fonctionnels

Le prototype atteint les objectifs fonctionnels fixés pour une première itération :

— connexion à une API de trafic avec authentification par clé ;

— affichage en continu des requêtes récentes (méthode, endpoint, statut, durée, horodatage) ;

— calcul et affichage d'indicateurs agrégés de performance ;

— visualisation de l'évolution des temps de réponse sous forme de graphe ;

— test de connectivité via endpoint de santé.

La fréquence de polling retenue permet une perception quasi temps réel de l'activité serveur, avec une interface suffisamment compacte pour un usage en contexte de diagnostic rapide.

3.2 Capacité de diagnostic

Les expérimentations sur serveur local mettent en évidence trois apports principaux :

— **Détection des lenteurs :** les pics de durée sont rapidement visibles dans le graphe et la liste des requêtes les plus lentes ;

— **Identification des zones chaudes :** les endpoints les plus sollicités sont isolés, facilitant la priorisation des optimisations ;

— **Lecture de la qualité de service :** la combinaison taux de succès/taux d'erreur fournit un indicateur simple de stabilité.

Ces observations confirment l'intérêt d'une agrégation continue, complémentaire aux audits ponctuels réalisés par des outils de benchmark [1, 2].

3.3 Bilan par objectif

Objectif	**Niveau d'atteinte**
Identifier des indicateurs clés de performance	Atteint
Concevoir un protocole de mesure simple	Atteint
Développer un prototype d'extension fonctionnel	Atteint
Tester sur différents types de sites	Partiellement atteint
Proposer une visualisation claire des résultats	Atteint

Le point partiellement atteint concerne la diversité des environnements testés. À ce stade, l'évaluation repose principalement sur un serveur local de simulation et non sur une campagne exhaustive multi-sites.

4 Discussion

4.1 Interprétation des résultats

Les résultats obtenus valident l'hypothèse principale : une extension navigateur peut fournir une lecture claire et utile de la performance web à partir de métriques simples et de traces réseau agrégées. L'approche présente un bon niveau de réactivité et permet de relier directement certaines dégradations à des endpoints spécifiques.

Le prototype se positionne comme un outil d'aide au diagnostic rapide plutôt que comme une plateforme complète de benchmarking. Il est particulièrement pertinent en phase de développement, de recette technique ou de surveillance applicative locale.

4.2 Comparaison aux outils existants

Par rapport aux outils généralistes d'audit, l'approche retenue privilégie :

— l'observation continue plutôt qu'un score ponctuel ;

— des indicateurs directement reliés au trafic HTTP réel ;

— une interface minimaliste orientée lecture opérationnelle.

En contrepartie, le prototype couvre un spectre plus restreint que des solutions matures (absence d'analyse front-end avancée, instrumentation limitée côté rendu navigateur), ce qui est cohérent avec le périmètre d'un premier jalon R&D.

4.3 Limites et menaces à la validité

Plusieurs limites doivent être explicitées :

— **Dépendance à l'API serveur :** la qualité de l'analyse dépend directement de la granularité et de la fiabilité des données remontées ;

— **Portée des tests :** la campagne expérimentale reste centrée sur un environnement local ;

— **Biais de charge :** la génération de trafic n'émule pas toutes les conditions réelles (variabilité réseau, charge utilisateur massive, hétérogénéité matérielle) ;

— **Mesures perçues utilisateur :** certains indicateurs UX (LCP, INP) ne sont pas encore intégrés de manière robuste.

Ces limites sont classiques dans une phase de prototypage et confirment la nécessité d'une itération ultérieure orientée validation externe et comparaison systématique [6, 3].

5 Conclusion

Ce projet a permis de concevoir et d'implémenter un prototype d'extension navigateur capable de mesurer et d'analyser des performances web à partir de données de trafic HTTP. Les objectifs principaux de la note de cadrage sont globalement atteints : définition d'indicateurs pertinents, mise en œuvre d'un protocole de mesure simple, réalisation d'un outil fonctionnel et proposition d'une restitution visuelle exploitable.

Les résultats confirment la pertinence d'une approche légère, embarquée dans le navigateur, pour un diagnostic rapide des ralentissements. Le prototype facilite l'identification des points de tension (endpoints lents, hausse d'erreurs, variations de latence) et constitue une base crédible pour une démarche d'amélioration continue.

Les perspectives de travail sont les suivantes :

— élargir la campagne de tests à des sites variés et à des scénarios de charge plus réalistes ;

— intégrer davantage de métriques orientées utilisateur final (Core Web Vitals complets) ;

— enrichir la comparaison avec des outils de référence afin d'évaluer la robustesse des mesures ;

— améliorer les mécanismes d'explication pour renforcer la valeur pédagogique auprès d'utilisateurs non experts.

Dans l'ensemble, le projet démontre qu'une extension navigateur peut constituer un support pertinent pour relier performance, observabilité et sensibilisation aux enjeux techniques et environnementaux du web.

Références

[1] André Afonso, Nuno Ferreira et Rui Abreu. « Correlating Lighthouse Scores with Real User Metrics ». In : *Proceedings of the IEEE/ACM International Conference on Web Engineering (ICWE)*. Piscataway, NJ : IEEE, 2020, p. 112-127. doi : 10.1007/978-3-030-50578-3_8.

[2] Ankit Bajaj et Rahul Garg. « A Hybrid Framework for Web Performance Measurement Using Selenium and Navigation Timing API ». In : *International Journal of Advanced Research in Computer Science* 10.2 (2019), p. 63-69. doi : 10.26483/ijarcs.v10i2.6378.

[3] Michael Butkiewicz, Harsha V. Madhyastha et Vyas Sekar. « Understanding Website Complexity: Measurements, Metrics, and Implications ». In : *Proceedings of the ACM Internet Measurement Conference (IMC)*. New York, NY : ACM, 2011, p. 313-328. doi : 10.1145/2068816.2068846.

[4] Vlad C. Coroama et Lorenz M. Hilty. « Assessing Internet Energy Intensity: A Review of Methods and Results ». In : *Environmental Impact Assessment Review* 45 (2014), p. 63-68. doi : 10.1016/j.eiar.2013.12.004.

[5] John A. Hoxmeier et Chris DiCesare. « System Response Time and User Satisfaction: An Examination of Four Models of Information Systems ». In : *Journal of Management Information Systems* 17.1 (2000), p. 135-149. doi : 10.1080/07421222.2000.11045641.

[6] Victor Ionescu, Alin Peculea et Vasile Dadarlat. « A Comparative Performance Analysis of Load Testing Tools ». In : *Proceedings of the ACM Symposium on Applied Computing (SAC)*. New York, NY : ACM, 2015, p. 751-756. doi : 10.1145/2695664.2695910.

[7] Daniel A. Menascé et Virgílio A. F. Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Upper Saddle River, NJ : Prentice Hall, 2002.

[8] Fiona Fui-Hoon Nah. « A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? » In : *Behaviour & Information Technology* 23.3 (2004), p. 153-163. doi : 10.1080/01449290410001669914.

[9] Gustavo Pinto, Fernando Castor et Yu David Liu. « Mining Questions About Software Energy Consumption ». In : *Proceedings of the IEEE Working Conference on Mining Software Repositories (MSR)*. New York, NY : ACM, 2014, p. 22-31. doi : 10.1145/2597073.2597110.

Annexe A – Contrat d'API du prototype

Le prototype *Traffic Monitor* repose sur une API HTTP simple. Toutes les routes, sauf mention contraire, attendent l'en-tête `X-Traffic-Key`.

Endpoints requis

— `GET /api/traffic` : retourne la liste des requêtes observées avec les champs `timestamp`, `method`, `path`, `statusCode`, `duration`.

— `GET /api/metrics` : retourne les métriques agrégées utilisées par l'interface (temps moyen/max, taux de succès/erreur, volume, cadence).

— `GET /api/health` : endpoint de santé sans authentification permettant le test de connectivité.

Endpoints optionnels (volet sécurité)

— `GET /api/slowloris` : informations de risque et connexions suspectes.

— `POST /api/block-ip` : blocage d'une adresse IP.

— `POST /api/unblock-ip` : déblocage d'une adresse IP.

— `GET /api/blocked-ips` : liste des IP actuellement bloquées.

Le serveur de test local fourni dans le projet implémente uniquement les endpoints requis, ce qui permet de valider le cœur fonctionnel sans dépendre des fonctionnalités de sécurité avancées.

Annexe B – Protocole de test reproductible

Cette annexe décrit la procédure minimale permettant de reproduire les tests présentés dans ce rapport.

B.1 Préparation

1. Se placer dans le dossier du serveur de test.
2. Installer les dépendances Node.js.
3. Démarrer le serveur local (port 3000).

B.2 Configuration de l'extension

1. Ouvrir la page d'administration des extensions du navigateur Chromium.
2. Activer le mode développeur.
3. Charger le projet en mode *Load unpacked*.
4. Dans le popup, renseigner :

   — URL serveur : `http://localhost:3000`

   — Clé API : `trafficapikey`

B.3 Exécution et collecte

1. Cliquer sur *Connect* pour démarrer le polling.
2. Générer du trafic en naviguant vers des routes locales.
3. Observer les métriques agrégées et le graphe des latences.
4. Utiliser *Refresh* pour forcer une mise à jour immédiate.

B.4 Critères de validation

Le test est considéré valide si :

— des entrées apparaissent dans le flux trafic ;

— les compteurs évoluent avec la charge générée ;

— les requêtes lentes sont correctement identifiées ;

— le test de santé (`/api/health`) confirme la connectivité.