PLAQUETTE · DISPONIBLE · VERSION 0.2.0

Code Explorer

Un agent qui relit vos fichiers à chaque question ne connaîtra jamais votre code. Code Explorer lui en donne la carte.

Code Explorer analyse votre dépôt une fois pour toutes et en construit un graphe de connaissance : fichiers, symboles, appels, dépendances, historique. Développeurs et agents IA l'interrogent ensuite en quelques millisecondes, au lieu d'ouvrir des dizaines de fichiers pour suivre une seule chaîne d'appels. Tout reste sur votre machine.

État : Version 0.2.0 publiée le 9 septembre 2026, code source public, binaires Linux, macOS et Windows. 1 007 tests automatisés dans le dépôt.

À qui ça sert

Les équipes qui héritent d'un code volumineux

Reprendre une base de plusieurs centaines de milliers de lignes sans son auteur : le graphe montre l'architecture réelle, pas celle de la documentation.

Ceux qui font travailler des agents IA

Claude Code, Codex, Cursor et tout client compatible MCP interrogent le graphe au lieu de remplir leur fenêtre de contexte de code brut.

Les responsables techniques

Chiffrer l'impact d'un changement avant de l'engager : appelants, appelés, tests concernés, portée transitive.

Ce que ça change concrètement

1 **Répondre à « qu'est-ce que ça casse ? » en une requête**

Sur le dépôt d'Ollama, retrouver tout ce qui dépend d'une fonction demandait de lire 174 fichiers, soit environ 730 000 tokens. Avec le graphe, la même réponse tient dans environ 18 000 tokens et arrive en quelques dizaines de millisecondes.

2 **Un graphe typé, pas une pile de fichiers**

Plus de 50 types de nœuds et 27 types de relations (appels, importations, héritage, propriété) : la question « qui appelle quoi, et par où » a une réponse exacte, pas une approximation textuelle.

3 **14 langages avec le même outil**

JavaScript, TypeScript, Python, Java, C, C++, C#, Go, Rust, Ruby, PHP, Kotlin, Swift et Razor, analysés par tree-sitter. Un dépôt polyglotte donne un seul graphe.

4 **Branché sur les agents que vous utilisez déjà**

30 outils exposés par le protocole MCP, en mode stdio ou HTTP, avec une commande d'installation par client : Claude Code, Codex, Cursor, VS Code, Claude Desktop.

5 **Entièrement local**

Un binaire unique, l'index écrit dans votre projet, aucun code envoyé à un tiers. Les fonctions qui appellent un modèle de langage sont facultatives et utilisent votre propre clé.

Comment ça marche

Quatre étapes, une seule fois par dépôt, puis des requêtes instantanées tant que le code ne change pas.

1

Votre dépôt

Code source, configuration, documentation, tests et historique Git, lus sur place.

2

Analyse

tree-sitter découpe 14 langages : symboles extraits, appels et références résolus, dépendances reconstituées.

3

Graphe de connaissance

Plus de 50 types de nœuds, 27 types de relations, instantané persistant écrit dans le dossier .codeexplorer du projet.

4

Requêtes et agents

Ligne de commande ou 30 outils MCP : contexte d'un symbole, analyse d'impact, recherche hybride, points chauds.

L'analyse se relance de manière incrémentale quand le code bouge ; les requêtes, elles, ne coûtent plus que quelques millisecondes.

Chiffres vérifiables

Chaque chiffre ci-dessous est daté et provient du dépôt ou d'une mesure publiée. Rien n'est extrapolé.

730 000 → 18 000

tokens pour savoir ce qui dépend d'une fonction, sur le dépôt d'Ollama (863 fichiers)

Source : README du dépôt. Les tokens sont estimés à quatre caractères par token, et le chiffre «

×36 en médiane

réduction du contexte mesurée sur 11 dépôts réels et 10 langages, jusqu'à ×164 sur Kubernetes

Source : README du dépôt, tableau de mesures par projet.

sans » ne compte que les fichiers concernés : c'est une borne basse.

14 langages · 30 outils MCP

périmètre d'analyse et surface offerte aux agents

Source : README et CHANGELOG de la version 0.2.0 (9 septembre 2026).

1 007 tests

tests automatisés exécutés par cargo test sur l'ensemble du projet

Source : README du dépôt, commande cargo test --workspace.

Pour démarrer

Des binaires prêts à l'emploi sont publiés pour Linux, macOS et Windows sur la page des versions du dépôt. Pour compiler depuis les sources, il faut Rust 1.75 ou plus récent.

1. Obtenir le binaire

Depuis les sources, une seule commande de compilation.

```
git clone https://github.com/phuetz/code-explorer.git
cd code-explorer
cargo build --release -p code-explorer-cli
# binaire : target/release/code-explorer
```

2. Indexer le projet

L'index est écrit dans le projet lui-même, rien ne sort de la machine.

```
cd /chemin/vers/votre/projet
code-explorer analyze .
```

3. Interroger le graphe

```
code-explorer context  handleLogin
code-explorer impact   PaymentService
code-explorer query    "où vérifie-t-on l'authentification"
code-explorer cypher   "MATCH (n:Function) RETURN n.name LIMIT 10"
```

4. Brancher votre agent

Une commande par client, ou le greffon Claude Code.

```
code-explorer mcp-install --client claude --scope project
code-explorer mcp-install --client codex  --scope global
code-explorer mcp-install --client cursor --scope project

claude plugin marketplace add phuetz/code-explorer
claude plugin install code-explorer@code-explorer
```

Nous installons et paramétrons l'outil sur vos dépôts dans le cadre de la prestation « reprendre la maîtrise d'un patrimoine logiciel », y compris sur du code que vous ne pouvez pas sortir de chez vous.

Ce qui n'est pas encore prêt

Nous préférons l'écrire noir sur blanc plutôt que de le faire découvrir après la signature.

- Le gain de contexte dépend de la densité des liens autour du symbole interrogé : ×164 sur Kubernetes, ×6 sur Jellyfin.
- Indexer un très gros dépôt prend du temps : environ 38 minutes pour les 3,6 millions de lignes de Kubernetes sur une machine à 24 cœurs, contre 3 secondes pour un dépôt de 863 fichiers.
- Les fonctions qui font appel à un modèle de langage restent facultatives et supposent votre propre clé d'API.

Licence, dépôt et contact

Licence

Business Source License 1.1 : usage interne libre, revente sous forme de service à des tiers soumise à licence. Bascule automatique en Apache 2.0 le 31 août 2030.

Où le trouver

- github.com/phuetz/code-explorer (https://github.com/phuetz/code-explorer)
- Documentation du projet (https://phuetz.github.io/code-explorer/)
- → Fiche produit sur ce site
- → La prestation « reprendre la maîtrise »

Éditeur

AGILE UP — Structure d'ingénierie logicielle indépendante, Vincennes.

Contact

contact@agile-up.com
agile-up.com