ChatGPTなどの便利なツールはありますが、機密を含むコードを外に出すことはできません。だから現場は従来のやり方に縛られ続けています。

コード内の機密情報

- 生パスワードや認証キーが埋め込まれている
- 関数名・変数名に連携システム名やホスト名などが含まれる

組織的な制約

- 外部AI利用の社内ポリシーの内容が曖昧
- 利用したとしても、万一の漏洩時に説明責任が果たせない

情報漏洩のリスク

- 一度でも外部に流出すれば**重大インシデント**
- セキュリティ監査・コンプライアンス的にも許されない

心理的な影響

- 「便利だから外部に投げてしまえ」という意識低下のリスク
- 結果としてセキュリティ文化を損なう懸念

社内環境だけで動かせるLLMなら、機密を守りながら開発支援が可能です。外に出さずに使えることが大前提です。

実際にローカルでLLMを導入するには選定・評価・運用の壁があります。ここをどう乗り越えるかが次の課題です

セキュリティ面

- ✅ 情報漏洩の心配なし（すべてローカルで完結）
- ✅ 社内ポリシーに準拠した運用が可能

導入・運用面の課題

- ❗ どの言語モデル／ツールを使うか選定が大変
- ❗ 性能評価やチューニングが面倒
- ❗ 外部AIと比べると性能差は残る

Krugleは、コード検索・解析・依存関係の可視化に特化したローカルLLMです。外部に情報を出さず、開発者の相棒として機能します。

開発元：Aragon Consulting Group Inc.（米国）
販売元：クリューグル株式会社（日本）※1

Krugle

安全・安心

オンプレ完結、社内ネットワーク内で稼働

インストール後は外部通信なし

レガシー対応力

Perlなど古い言語も学習済み

既存コード資産や社内ドキュメントをRAG検索

強力な解析機能

コード検索エンジン＋LLMの組み合わせ

依存関係の可視化

開発者フレンドリー

VS Code拡張機能での便利な支援

※1 https://www.krugle.co.jp

全文検索や要約、依存関係の把握、ドキュメント自動生成。今まで数時間かかっていたことが、数分でできるようになります。

AIによるコード解析支援

目的	プロンプト例
全体の解説	このモジュールの役割を説明してください
構造の把握	APIエンドポイントはどこにありますか？
動作の理解	入力チェックはどのように行われていますか？
動作の検索	xxxテーブルにINSERTしている処理を一覧表示してください
ロジックの要約	この関数の目的は？
テストからの理解	この関数のテストを作成してください
ドキュメント生成	ドキュメンテーションコメントを作成してください

コード検索エンジンとの連携

全文検索・セマンティック検索・ベクトル検索を統合

社内のAPI仕様などのドキュメント、他システムのコードも検索対象にできる

→「必要な情報を迷わず探せる」

ライブラリ・フレームワークの参照

主要なライブラリやフレームワークのマニュアルが検索可能な状態で登録済

コードを見ながら最小のタイプ数で即参照可能

→「調べものの時間を削減」

※ すべて VSCode 拡張機能内で完結

コードのQAをナレッジとして共有化。属人化していた暗黙知が形式知になり、引継ぎがスムーズになります。

…ということをしようと考えてます

RAGによるナレッジ化

- Markdownで記述したローカル知識をインデックス化しDBに登録
- 開発中システムの仕様書・設計書
- ADR（Architecture Decision Record：設計判断の記録）
- インフラ情報（サーバ名・DB情報・プロキシ設定など）
- 連携サーバのAPI仕様やコードスニペット
- Krugleで利用するプロンプト例

利用シーン

- 新規参加者が過去の実装理由を素早く把握
- 問い合わせ対応で、過去の議論や仕様を即検索
- 引継ぎ時に「質問すれば答えてくれる」環境を実現

メリット

- 属人化した知識を組織全体で活用
- ドキュメント更新が追いつかなくても検索で補完
- チャット感覚で情報を呼び出せる

暗黙知　⇒　形式知　⇒　組織共有