Cloudflareによる
「汎用型RAG ChatBotアプリ」制作

自己紹介

1. ITエンジニアではありません

・コロナ禍の在宅ワーク期間、通勤の時間が浮いたので、趣味でPythonの学習を開始。

・セキュリティとか、ネットワークとか、今だによく分からない。

・新規事業に於けるPoCレベルのものは自作可能。

2. データサイエンティストではありません

・Kaggleは、コロナ禍の在宅ワーク期間、時間があったので趣味の一環として遊びました。

・数学を勉強したのは、大学院時代が最後(20年以上前)で、高度な数学的な素地がある訳ではない。

・半導体技術者時代に「シックスシグマ」に触れて以来、データの活用自体には興味があった。

3. 主なキャリアパスは、半導体開発設計、コンサルタント、新規事業開発とマーケティングです

・グローバルメーカーで、開発・設計・品質管理・生産技術管理等に従事。

・コンサルタント時代は、金融業界、保険業界、メディア業界、IT業界で、様々なPJに従事。

・直近の10年以上は、新規事業開発(本業・副業含む)とマーケティング、海外市場開拓に従事。

なぜ、Cloudflareに目を付けたか？

1. そもそも、僕がしたいことは？

・新規事業開発のPSFフェーズに於けるPoCを自作したい。（このフェーズでは、出戻りが多いため）

・できれば、無料(or低料金)で、爆速で制作し、想定ジョブに対する仮説検証を行いたい。

2. Cloudflareとは？

・無料でもPoC前提の「フロントエンド、軽量バックエンド、簡易DB、ストレージ」を単体で完結できる。

・GitHubと連携可能で、Commitしたら、すぐに自動デプロイ。

・Zero Trust等の認証機能もある。

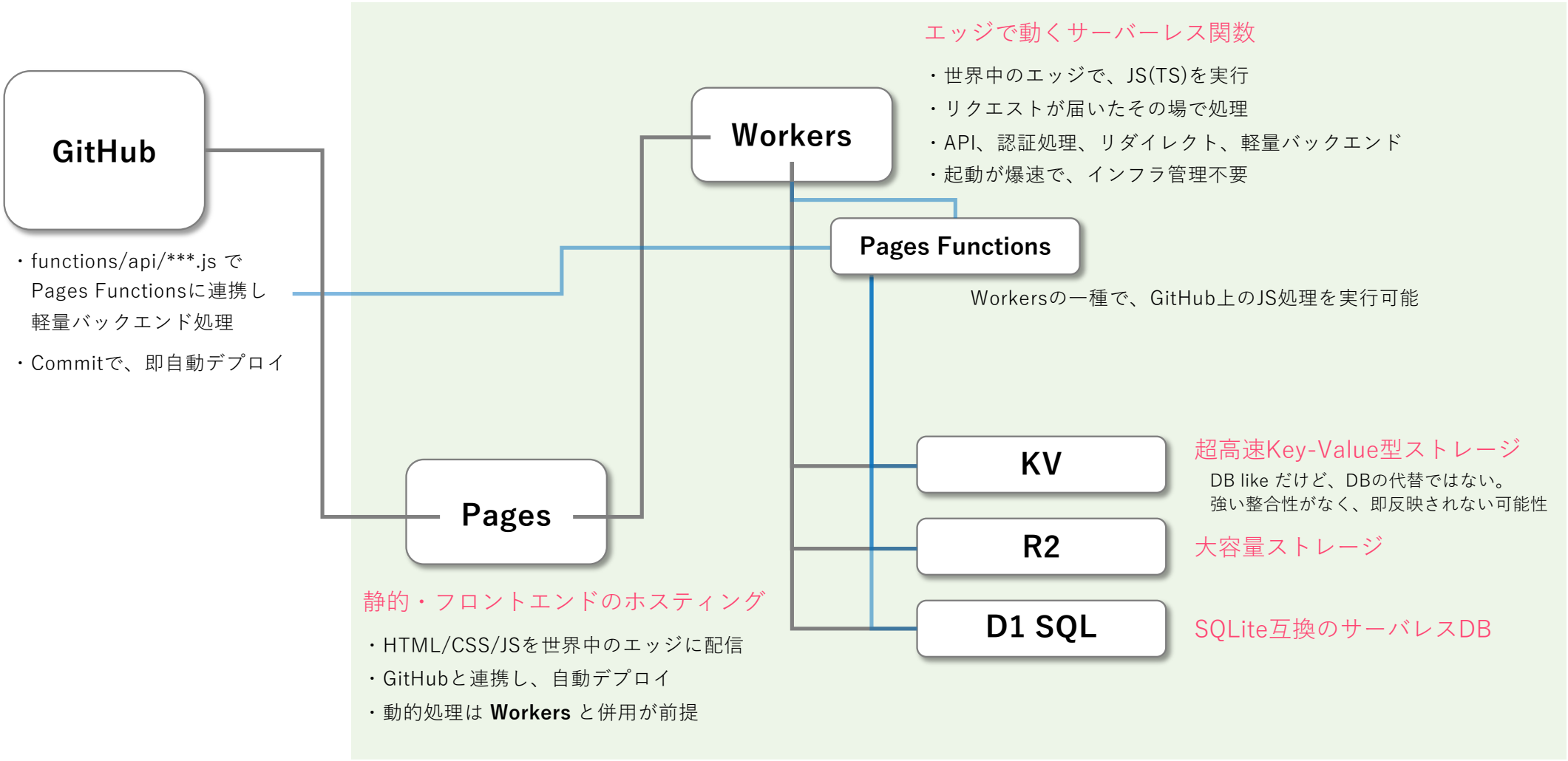
・生成AI API連携ももちろん可能。

3. デメリット/制約条件は？

・Pythonは正式サポートされておらず、JavaScript/TypeScriptが前提。

・軽量バックエンドなので、機械学習関連のタスクは不可（Google Cloud Runで処理するAPI連携は可能）

Cloudflareのイメージ



今回制作する「汎用型RAG ChatBotアプリ」の要件(概要)

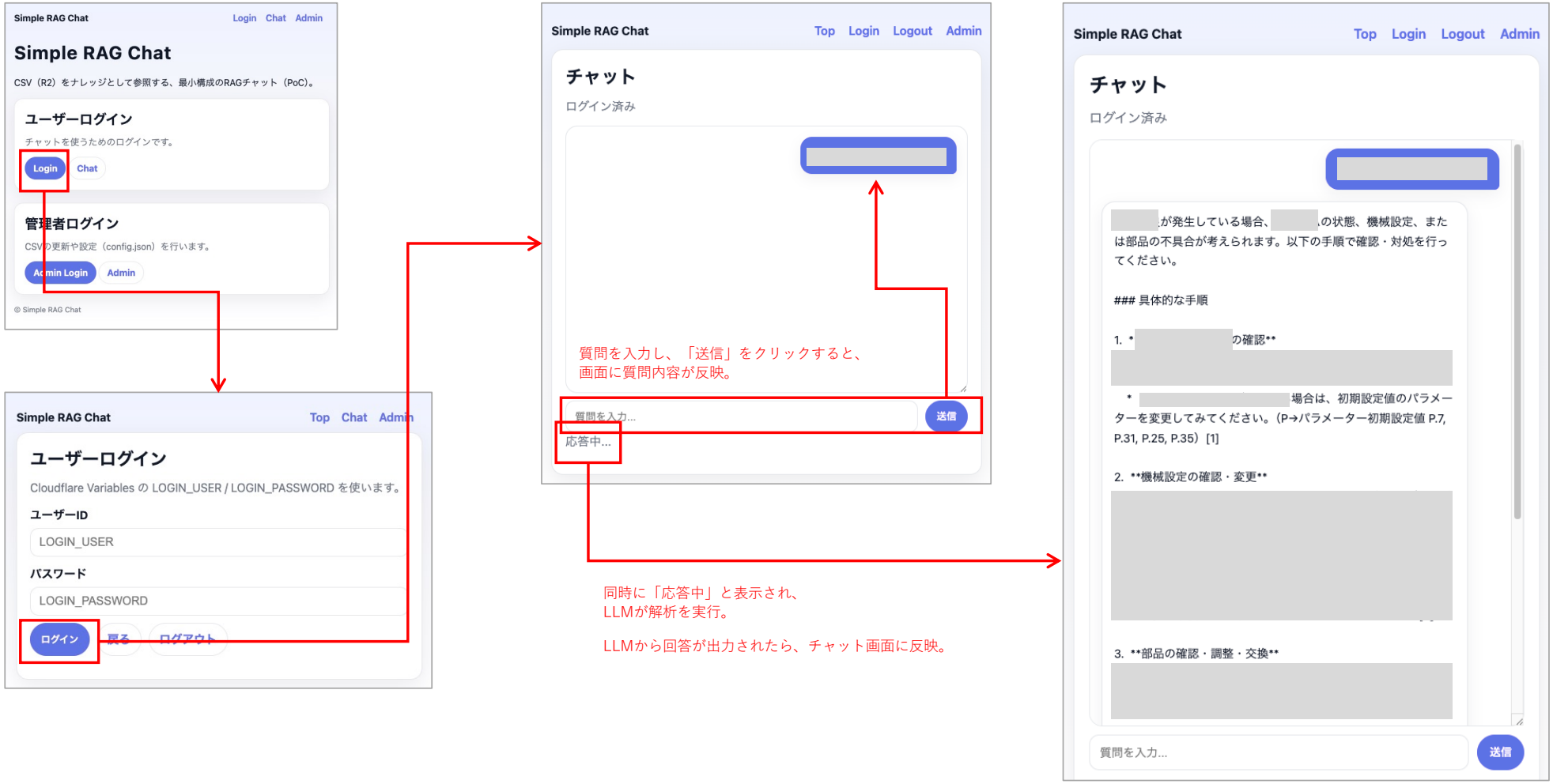
目的：

- Cloudflare 及び 生成による、RAGChatアプリの制作
- GeminiAPI と OpenAI APIを使用
- ベクトル検索ではなく、行単位のスコアリング検索を採用
- 必要最低限のセキュリティ要件を実装
- 管理者画面から、RAGデータの更新・変更・削除が可能(CSV or TXT)
- 管理者画面から、LLMの種類・モデルの変更が可能
- 管理者画面から、RAGや用途に合わせた柔軟なプロンプト設定が可能
- 柔軟な管理画面により、非エンジニアでも汎用的な運用が可能

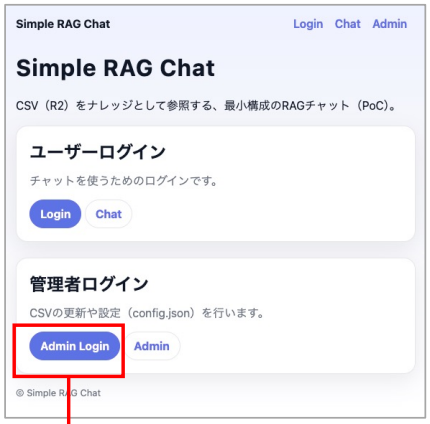
主な要件：

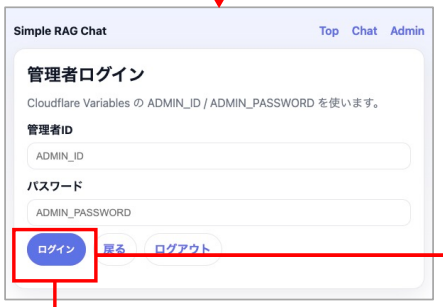
目的	仕組み
RAGデータの保持	- Cloudflare R2によるストレージ利用
2種類の生成AI APIを使用してBot回答を生成	- Gemini or OpenAI のいずれかを管理画面から選択 - 従量課金されるため、コスト暴走防止のレート制限も実装
RAG検索(前処理)	- 強めの文字正規化(NFKC正規化) - 日本語を想定した簡易キーワード抽出 - n-gram(bigram/2文字)による補助スコアリング - 行単位のスコアリングを実施し、上位行のみを抽出
ユーザーログイン機能	- 環境変数によるログイン認証 - HttpOnly Cookieによるセッション管理
簡易レート制限	- Cloudflare KV によるカウント情報保持 - 総当たり・過剰アクセス対策

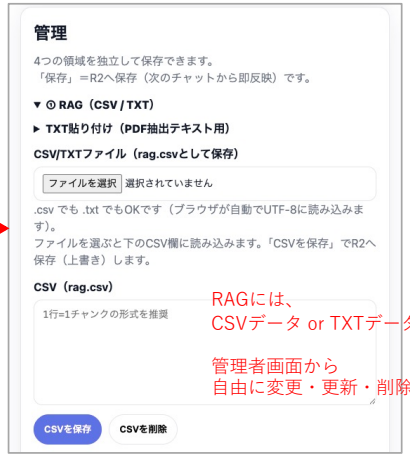
今回制作する「汎用型RAG ChatBotアプリ」のイメージ(チャット)



今回制作する「汎用型RAG ChatBotアプリ」のイメージ(管理画面)

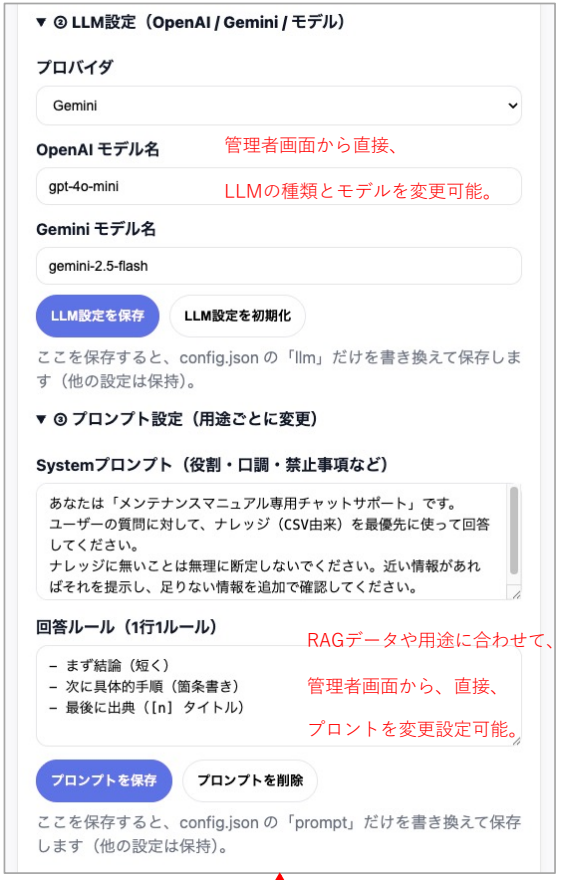






RAGには、
CSVデータ or TXTデータ形式で、

管理者画面から
自由に変更・更新・削除が可能。



今回制作する「汎用型RAG ChatBotアプリ」のRAG検索前処理の概要

1. 強めの文字正規化(NFKC正規化):

検索時には、質問文・RAGデータの双方に以下の正規化を実行。

- Unicode正規化・NFKC正規化(全角・半角の統一等)
- 全角スペースを半角スペースへ変換(連続空白も、1つに圧縮)
- 各種記号を空白に置換

これにより、表記揺れ・全角半角差異・記号有無による検索漏れを抑制

2. 日本語を想定した簡易キーワード抽出:

形態素解析ではなく、以下の方法でキーワードを抽出。

- 助詞や文法語で分割
- 漢字・平仮名・カタカナ・英数字の連続列を正規表現で追加抽出
- 2文字以上の語のみを採用し、文字数が長い語句を優先使用

これにより、日本語特有の複合語を拾いやすくする

3. 2文字 n-gram(bigram)による補助スコアリング:

単語一致だけは拾えない表記揺れ対策として、2文字 n-gram(bigram)のJaccard類似度を補助的に使用（誤ヒットが増えないよう、スコアの重み付は控えめに）

4. 行単位のスコアリング & 上位行(最大20行)のみを抽出し、LLMへ入力:

・質問文全体が行に含まれる場合は強い加算・・・等、スコアリングの重み付けを実施　→ なるべく関連の高い情報をLLMへ
・スコアが0や低い行は、上位20行以内でもLLMへ入力しない(なるべく除外)　→ ノイズとなる情報はLLMへは渡さない

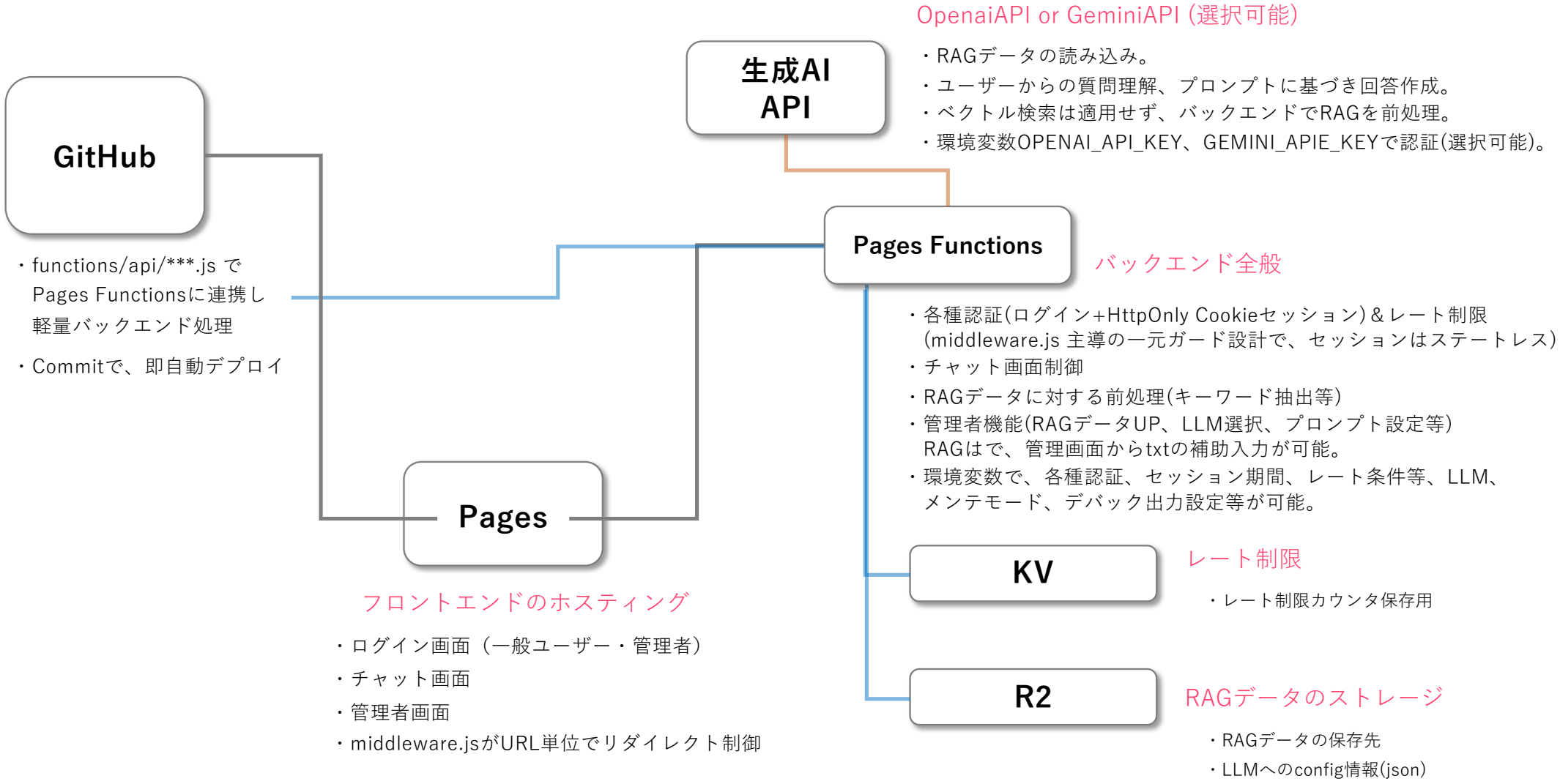
良い点:

・処理が軽く早い、また、トークンの消費が抑えられる。
・functions/api/で再現性高く確実に処理される。
・数千行規模のRAGにも効く。
・様々RAGデータにも適用でき、汎用性が高い。

限界点:

・言い換え表現への耐性は高くない。
・抽象度の高い質問は拾い難い。
・ベクトル検索(Embedding)は未使用で、大容量RAGには不向き。

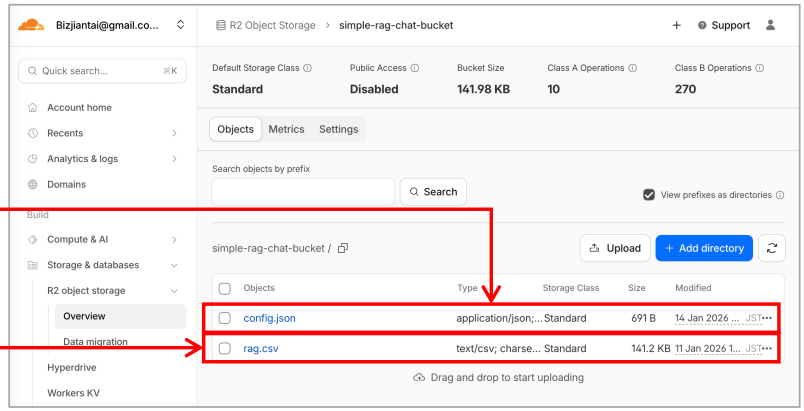
今回制作する「汎用型RAG ChatBotアプリ」の構成図



Cloudflare R2(共有ファイル用ストレージ) と Cloudflare KV(メタ情報格納)

R2

RAGデータとLLM関連データの格納



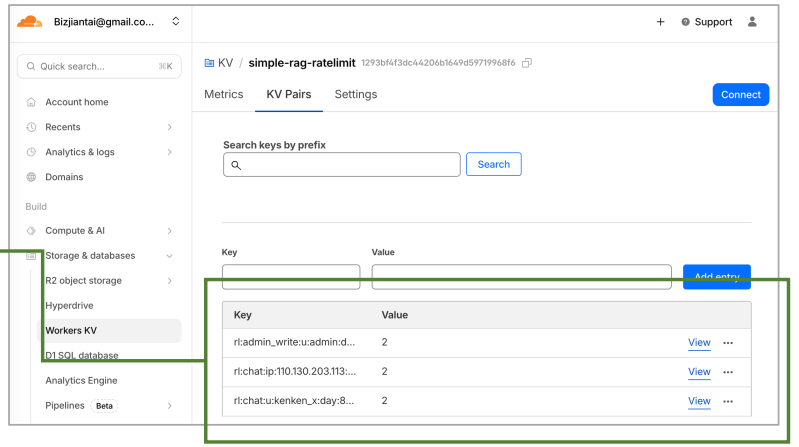
選択されたLLMやプロンプト情報は、
config.jsonとして格納

UPされたRAGデータはrag.csvとして格納

KV

レート制限

rl:admin_write:u:admin:day:86400	2



このレートカウント情報を基に、レート制限を実行している

本アプリのセキュリティのイメージ

1. ユーザーログイン/管理者ログイン
2. セッション + HttpOnly Cookie
3. レート制限

チャット = ユーザーログイン + セッション + HttpOnly Cookie

- ユーザー認証(ID/パスワード)
- SESSION_SIGNING_SECRETによる認証
- SESSION_MAX_AGE_SECによるセッション時間設定
- 回数制限(レート制限)

- RAGチャットの利用

管理画面 = 管理者ログイン + セッション + HttpOnly Cookie

- 管理者認証(ID/パスワード)
- SESSION_SIGNING_SECRETによる認証
- SESSION_MAX_AGE_SECによるセッション時間設定
- 回数制限(レート制限)

- RAGデータの更新・変更・削除
- LLMの選択設定
- プロンプトの編集

目的	仕組み
一般ユーザー/管理者	ログイン + HttpOnly Cookie
ログイン維持	セッション(署名付きCookie)
セッション時間	環境変数設定(24時間)
バックエンド操作認証	SESSION_SIGNING_SECRET 認証
生成API	環境変数設定(API_KEY)
RAGデータアクセス	RAG_CSV_KEY 認証
管理画面のTTL	ADMIN_SESSION_TTL_SEC 制御
レート制限	次頁参照
DEBUG/メンテモード	環境変数による変更可能

回数制限(レート制限)

悪意ある総当たりや過剰アクセスを防ぐため、Cloudflare KVを利用した簡易的なレート制限を実装。

制限はIPアドレス単位で実施され、短時間に一定回数を超えたリクエストは拒否する。
(Cloudflare標準のRate Limittingは使用せず、functions/api/**.js. 及び 環境変数で可変設定)

chat（一般ユーザー）用

Variable 名	内容	デフォルト値
`RL_CHAT_USER_5M`	ユーザー単位：5分あたりの上限	`30`
`RL_CHAT_USER_1D`	ユーザー単位：24時間あたりの上限	`200`
`RL_CHAT_IP_5M`	IP単位：5分あたりの上限	`60`
`RL_CHAT_IP_1D`	IP単位：24時間あたりの上限	`500`

admin（管理画面）用

Variable 名	内容	デフォルト値
`RL_ADMIN_WRITE_USER_1M`	管理者：保存/削除系API（1分）	`10`
`RL_ADMIN_WRITE_USER_1D`	管理者：保存/削除系API（24時間）	`50`
`RL_ADMIN_PREVIEW_USER_1M`	管理者：プレビュー閲覧（1分）	`30`

Cloudflare KVは、上記のレート制限カウンタ用として使用。

当該APIが呼ばれると、Cloudflare KVにアクセス回数(カウンタ)を保存し、一定時間が経過すると自動で期限切れになります。

期限(TTL)は、Functions側のレート制限処理(functions/api/_shared.js)で、用途に応じて秒数を指定しています。

※ あくまでも簡易レート制限であり、厳密な課金防御というより「過剰アクセスの抑止」を目的としています。