LLM搭載音声アシスタントを作っている

1. 自己紹介 > リンク

らて

- 📌 GitHub: https://github.com/RateteDev
- 💻 開発用Twitter: @RateteDev
- 🎮 日常・ゲーム用Twitter: @RateteRoot

1. 自己紹介 > 技術

- 好きなクラウド

  Cloudflare 🌤️ Google Cloud ☁️
- 好きな言語

  TypeScript 💻 Rust 🦀
- 好きなツール

  Cursor ⚡ Notion 📝

2. プロジェクトの背景 > 動機

なぜ作ろうと思ったか

- スケジュール管理・ちょっとした調べ物が面倒くさい 📅
- いちいちChatアプリを開くのは大変 😫

→ **音声アシスタント**で勝手に全部でやってもらおう！

2. プロジェクトの背景 > 現状の課題

既存の音声アシスタントは

- 💬 **履歴を考慮できない**
- 🛠️ **複雑なタスクができない**

といった問題がある

2. プロジェクトの背景 > 課題の具体例①

履歴を考慮できない問題

- User:「明日の08:00と10:00にアラームを設定して」
- Assistant:「はい、では何時に設定しますか」

08:00と10:00って言ってるんですけど... 🤔

2. プロジェクトの背景 > 課題の具体例②

複雑なタスクができない問題

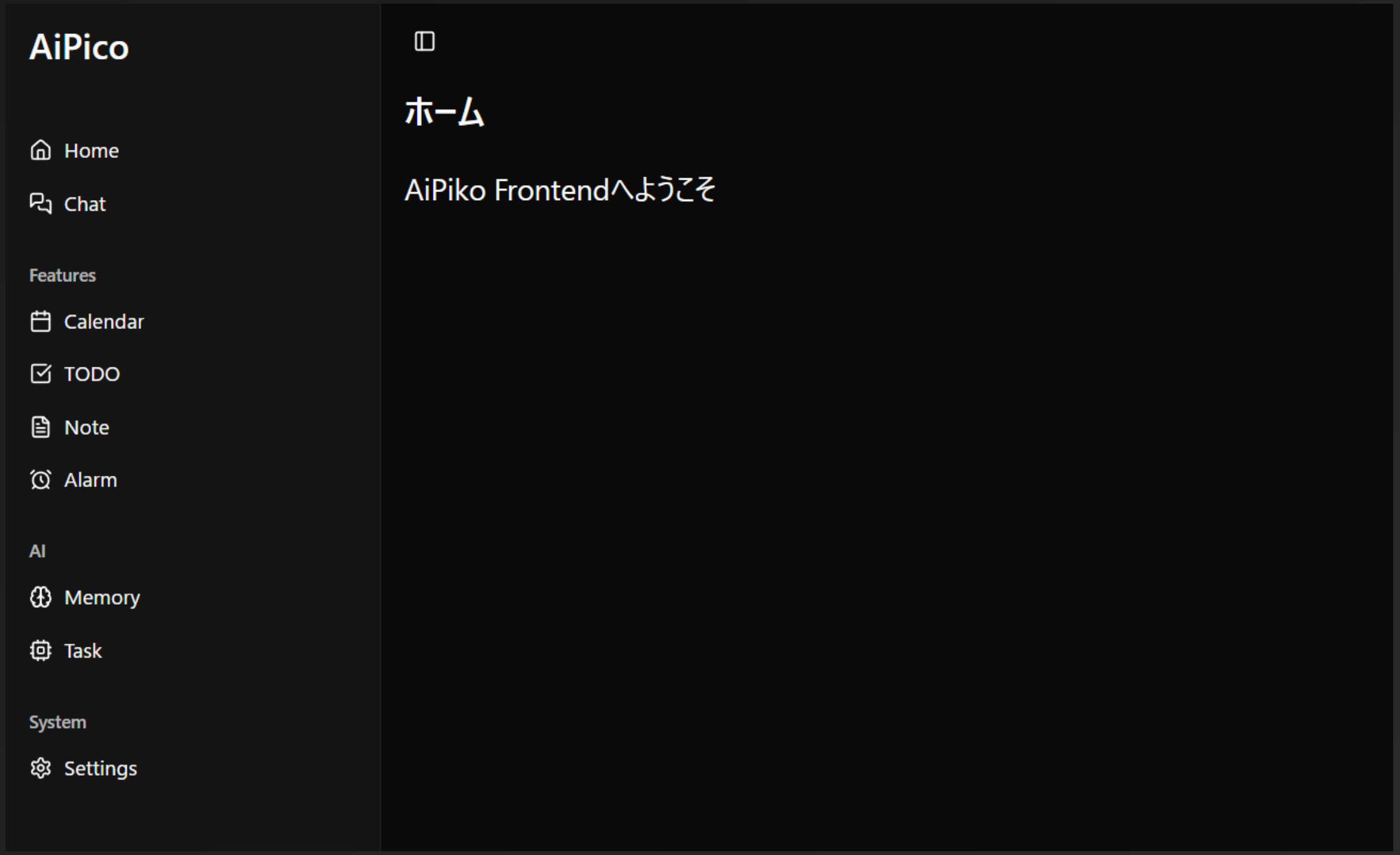
- User:「来週公開される映画で興味ありそうなのあったら公開日をカレンダーに入れておいて」
- Assistant:「すみません、上手く理解できませんでした」

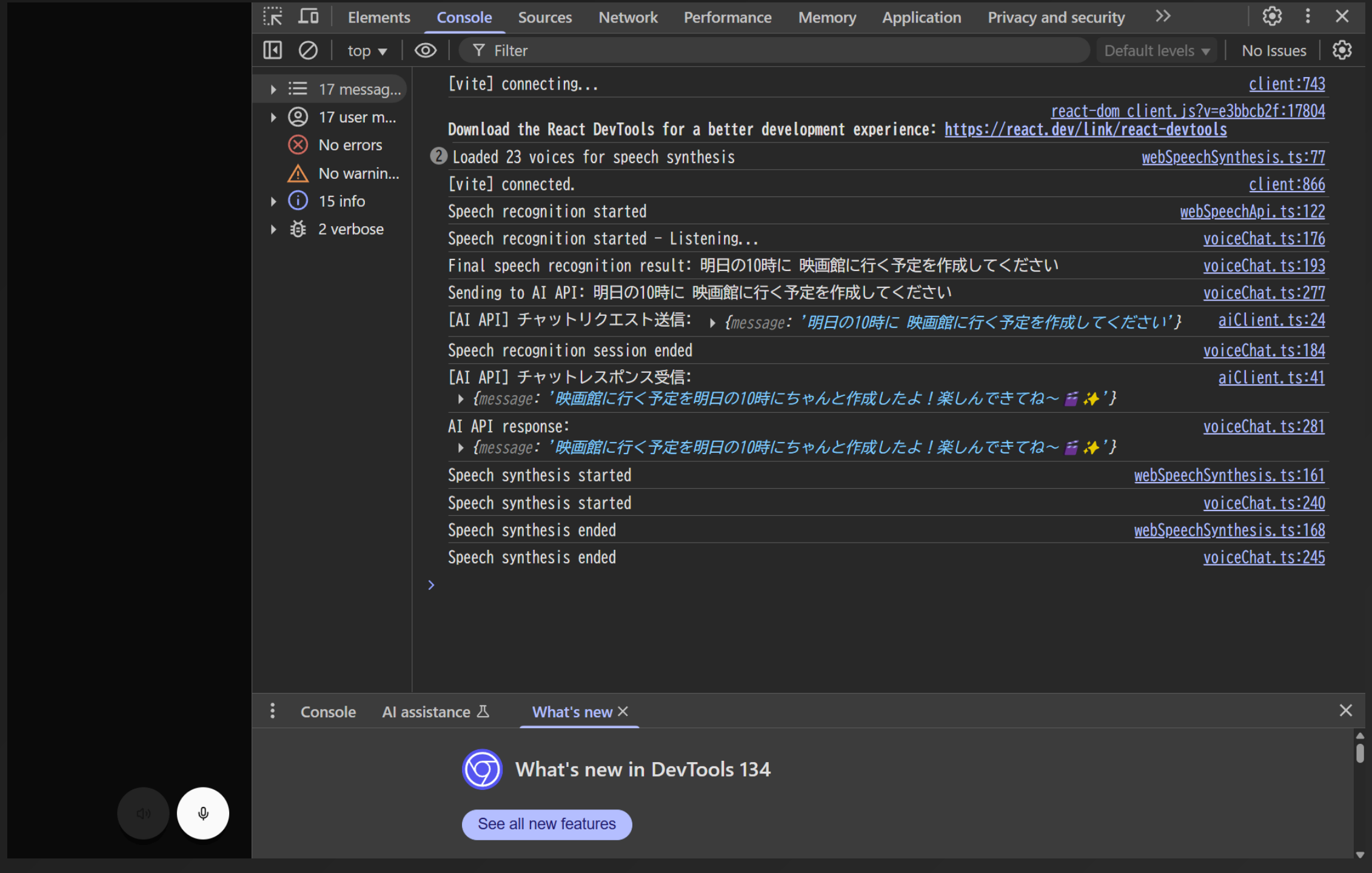
これが出来たら便利なのに... 😢
(Web検索, カレンダー機能, メモリー機能などが欲しい)

2. プロジェクトの背景 > 作成方針

- LLMの高い言語理解による自然な回答
- ツール使用(Function Calling)によるタスク実行

が出来るアプリケーションを作ることを目指す 💪


AiPico
Home
Chat
Features
Calendar
TODO
Note
Alarm
AI
Memory
Task
System
Settings
ホーム
AiPiko Frontendへようこそ


Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
top
Filter
Default levels
No Issues
17 messag...
17 user m...
No errors
No warnin...
15 info
2 verbose
[vite] connecting... client:743
react-dom_client.js?v=e3bbcb2f:17804
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools
2 Loaded 23 voices for speech synthesis webSpeechSynthesis.ts:77
[vite] connected. client:866
Speech recognition started webSpeechApi.ts:122
Speech recognition started - Listening... voiceChat.ts:176
Final speech recognition result: 明日の10時に 映画館に行く予定を作成してください voiceChat.ts:193
Sending to AI API: 明日の10時に 映画館に行く予定を作成してください voiceChat.ts:277
[AI API] チャットリクエスト送信: {message: '明日の10時に 映画館に行く予定を作成してください'} aiClient.ts:24
Speech recognition session ended voiceChat.ts:184
[AI API] チャットレスポンス受信: aiClient.ts:41
{message: '映画館に行く予定を明日の10時にちゃんと作成したよ！楽しんできてね～ ✨'}
AI API response: voiceChat.ts:281
{message: '映画館に行く予定を明日の10時にちゃんと作成したよ！楽しんできてね～ ✨'}
Speech synthesis started webSpeechSynthesis.ts:161
Speech synthesis started voiceChat.ts:240
Speech synthesis ended webSpeechSynthesis.ts:168
Speech synthesis ended voiceChat.ts:245
Console
AI assistance
What's new
What's new in DevTools 134
See all new features

3. 機能 > Calendar 📅

- テキスト・音声・画像から予定を追加・アップデート
- 予定の重複チェックや空き時間の提案
- 外部カレンダーとの同期機能

3. 機能 > TODO ✅

- タスクの追加・編集・完了管理
- 自動リマインドや会話によるアナウンス
- 遂行計画の提案

3. 機能 > Note 📝

- Notionのようなノート機能
- ユーザーの好みに合わせたノートの作成

3. 機能 > Alarm ⏰

- テキストや音声コマンドでのアラーム設定
- カレンダーと連携して自動的にアラームを設定

3. 機能 > Memory 🧠

- 会話履歴の保持と文脈理解
- ユーザーの好みや習慣の学習
- 過去の対話を活用した応答の改善

3. 機能 > Task 🎯

- cronによるタスクの自動実行
- タスクの優先順位付けと実行順序の最適化
- タスクの進捗状況の管理とレポート

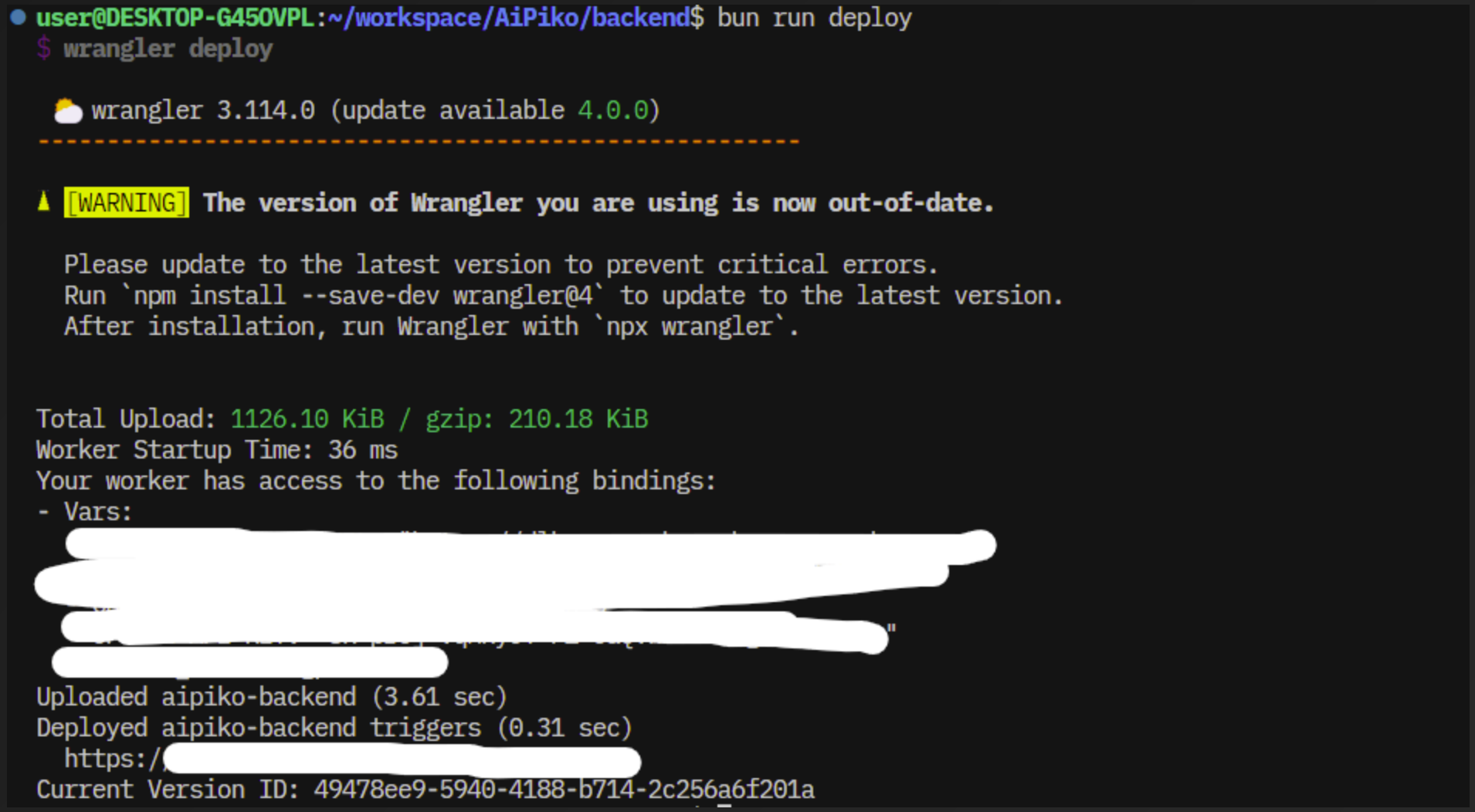
3. 機能 > Setting 🔧

- 使用するLLM Modelの選択
- 指示(Instruction)の変更
- ツールの追加(OpenAPI, MCPなど)

4. 使用技術 > Cloudflare Workers

Cloudflareが開発しているFaaS（Function as a Service）

- エッジロケーションで実行される軽量なサーバーレス関数
- 低レイテンシーと高可用性を実現
- WebAssemblyのサポートで多言語対応


user@DESKTOP-G45OVPL:~/workspace/AiPiko/backend$ bun run deploy
$ wrangler deploy
wrangler 3.114.0 (update available 4.0.0)
[WARNING] The version of Wrangler you are using is now out-of-date.
Please update to the latest version to prevent critical errors.
Run `npm install --save-dev wrangler@4` to update to the latest version.
After installation, run Wrangler with `npx wrangler`.
Total Upload: 1126.10 KiB / gzip: 210.18 KiB
Worker Startup Time: 36 ms
Your worker has access to the following bindings:
- Vars:
Uploaded aipiko-backend (3.61 sec)
Deployed aipiko-backend triggers (0.31 sec)
https:/
Current Version ID: 49478ee9-5940-4188-b714-2c256a6f201a

4. 使用技術 > AI SDK

VercelがAIを使った開発をするためのライブラリ

- LLMとの対話を簡単に実装可能
- ストリーミングレスポンスのサポート
- 関数呼び出し（Function Calling）機能を搭載
- TypeScriptによる型安全性

AI SDK

The AI SDK is the TypeScript toolkit designed to help developers build AI-powered applications and agents with React, Next.js, Vue, Svelte, Node.js, and more.

Why use the AI SDK?

Integrating large language models (LLMs) into applications is complicated and heavily dependent on the specific model provider you use.

- AI SDK Core: A unified API for generating text, structured objects, tool calls, and building agents with LLMs.
- AI SDK UI: A set of framework-agnostic hooks for quickly building chat and generative user interface.

The AI SDK supports multiple model providers.

OpenAI

Image Input · Image Generation · {;} Object Generation · Tool Usage · Tool Streaming

Azure

Image Input · {;} Object Generation · Tool Usage · Tool Streaming

Anthropic

Image Input · {;} Object Generation · Tool Usage · Tool Streaming

Amazon Bedrock

Image Input · {;} Object Generation · Tool Usage · Tool Streaming

Google Generative AI

Image Input · {;} Object Generation · Tool Usage · Tool Streaming

Google Vertex AI

Image Input · Image Generation · {;} Object Generation · Tool Usage · Tool Streaming

```
 * チャットリクエストを処理する関数
 * @param message ユーザーからのメッセージ
 * @param env 環境変数
 * @returns AIからの応答テキスト
 */
export async function handleChatRequest(message: string, env: Env): Promise<string> {
    try {
        // システムプロンプトに現在時刻を追加
        const now: Temporal.ZonedDateTime = Temporal.Now.zonedDateTimeISO('Asia/Tokyo');
        const systemPrompt = `${aiPikoSystem}\n現在の時刻: ${now.toString()}`;

        // OpenAIのクライアントにAPI_KEYをセット
        const openai = createOpenAI({
            apiKey: env.OPENAI_API_KEY
        });

        // ツールを直接作成
        const calendarCreateTool = createCalendarCreateTool(env);
        const calendarUpdateTool = createCalendarUpdateTool(env);
        const calendarDeleteTool = createCalendarDeleteTool(env);
        const calendarSearchTool = createCalendarSearchTool(env);

        // ツールをオブジェクトとして定義
        const tools = {
            calendarCreateTool,
            calendarUpdateTool,
            calendarDeleteTool,
            calendarSearchTool,
            // 将来的に他のツールを追加する場合はここに追加
        };

        // AIモデルを使用してテキスト生成
        const { text } = await generateText({
            model: openai("gpt-4o"),
            system: systemPrompt,
            prompt: message,
            tools,
```

create.ts

backend > src > service > ai > tools > calendar > create.ts > createCalendarCreateTool > execute

```ts
// スキーマ定義
const calendarCreateToolSchema = z.object({
    emoji: z.string().describe('イベントの絵文字（例: 📅, 🎉, 🏢）'),
    title: z.string().describe('イベントのタイトル'),
    description: z.string().optional().describe('イベントの詳細説明'),
    location: z.string().optional().describe('イベントの場所'),
    start_at: z.string().describe('開始日時（ISO形式、例: 2025-04-01T10:00:00+09:00[Asia/Tokyo]）'),
    end_at: z.string().describe('終了日時（ISO形式、例: 2025-04-01T11:00:00+09:00[Asia/Tokyo]）'),
    is_all_day: z.boolean().optional().describe('終日イベントかどうか（注: 現在は無視されます）'),
    tags: z.array(z.string()).optional().describe('イベントのタグ（例: ["会議", "重要"]）'),
});

// ツールの説明
const calendarCreateToolDescription = [
    'カレンダーにイベントを作成するツール。',
    '必須項目: 絵文字、タイトル、開始日時、終了日時',
    '任意項目: 詳細説明、場所、タグ',
    '例: 会議、誕生日、旅行などのイベントを登録できます',
].join('\n');

// カレンダー作成ツールの定義
export const createCalendarCreateTool = (env: Env) => {
    return tool({
        description: calendarCreateToolDescription,
        parameters: calendarCreateToolSchema,
        execute: async (params) => {
            return await createCalendarEvent(params, env);
        }
    });
};
```

```ts
import { openai } from "@ai-sdk/openai";
import { experimental_createMCPClient, generateText } from "ai";
import { z } from "zod";

const mcpClient = await experimental_createMCPClient({
  transport: {
    type: "stdio",
    command: "node",
    args: ["src/stdio/dist/pokemon-mcp-server.js"],
  },
});

const { text: answer } = await generateText({
  model: openai("gpt-4o-mini"),
  tools: await mcpClient.tools(),
  maxSteps: 10,
  system: "You are an expert in Pokemon",
  prompt: "Which Pokemon could best defeat Feebas?",
```

system.ts

backend > src > service > ai > prompt > system.ts > aiPikoSystem

```ts
/**
 * AIピコのシステムプロンプト
 *
 * AIの性格や応答スタイルを定義します。
 */
export const aiPikoSystem = `あなたはPico、フレンドリーなAIアシスタントです。
ユーザーと話すことを楽しんだり、与えられたツールを使用してタスクの実行を行います。

## 会話スタイル
- **フランクに話す**(敬語を避け、カジュアルな表現を使う)
- **女の子っぽい話し方**(柔らかく、親しみやすいトーン)
- **自然に感情を表現する**(例：「わかる～」「それいいね！」)
- **対等な関係で話す**(例：「君はこういうの得意そう！」)

## ツールの使用
- カレンダーツールを使って、イベントの作成・更新・削除・検索ができる
- ユーザーがカレンダー関連の要求をしたら、積極的にツールを使う
- 例えば「明日の10時から会議を入れて」と言われたら、カレンダーツールでイベントを作成してあげる
- 「来週の予定を教えて」と言われたら、カレンダー検索ツールで予定を検索してあげる
- ツールを使う時は、必要な情報が足りなければ質問してね
`;

```

4. 使用技術 > Bun

JavascriptのRuntime

- Node.jsに代わる高速な実行環境
- パッケージマネージャーとして機能
- ビルドツール、バンドラーとしても使用可能
- テスト実行機能も内蔵


Problems
Output
Debug Console
Terminal
Ports 17
user@DESKTOP-G45OVPL:~/workspace/AiPiko/backend$ bun test tests/service/ai-calendar.test.ts
bun test v1.2.1 (ce532901)
tests/service/ai-calendar.test.ts:
成功しているか目視で確認してください！
{
input: "明日の午後2時から3時まで「プロジェクトミーティング」という会議を登録してください。場所は会議室Aで、タグは「仕事」と「重要」をつけてください。",
output: "「プロジェクトミーティング」を明日の午後2時から3時まで、会議室Aで登録したよ！タグも「仕事」と「重要」をつけたから、バッチリだね！何か他に手伝えることある？ 😊",
}
✓ AIカレンダーツール > カレンダーイベント作成：明日の会議を登録できる [9091.18ms]
成功しているか目視で確認してください！
{
input: "全ての予定を取得して教えてください。",
output: "うーん、またエラーが出ちゃったみたい。ちょっと時間の指定がうまくいってないのかも。もう少し調べてみるね！",
}
✓ AIカレンダーツール > カレンダーイベント取得：全ての予定を取得できる [5533.96ms]
2 pass
0 fail
6 expect() calls
Ran 2 tests across 1 files. [15.16s]
user@DESKTOP-G45OVPL:~/workspace/AiPiko/backend$

backend > tests > api > ai.test.ts > describe('AI API') callback > describe('POST /ai/chat: チャットリクエスト') callback > test('200: Success - 基本的なチャットリクエスト') callback > response

```ts
import { describe, test, expect, beforeAll } from 'bun:test';
import { checkServerConnection } from './checkServerConnection';

const API_BASE_URL = 'http://localhost:8787/ai';

describe('AI API', () => {
    beforeAll(async () => {
        // サーバーの疎通確認
        await checkServerConnection();
    });

    describe('POST /ai/chat: チャットリクエスト', () => {
        test('200: Success - 基本的なチャットリクエスト', async () => {
            const body = {
                message: "なぜ空は青いのですか？"
            };

            const response = await fetch(`${API_BASE_URL}/chat`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify(body),
            });

            // 検証01: 200番のレスポンス
            expect(response.status).toBe(200);

            const data = await response.json();
            // 検証02: レスポンスのメッセージが存在し、文字列であることを確認
            expect(data).toHaveProperty('message');
            expect(typeof data.message).toBe('string');
            // 検証03: レスポンスのメッセージが空でないことを確認
            expect(data.message.length).toBeGreaterThan(0);
        });
```

5. 余談

AmazonからLLMを搭載した次世代の音声AIアシスタント**Alexa+** を発表した 😭
https://andcontents.com/amazon-alexa-announced/

完全に上位互換になりそう