







# ChatGPT for Application Engineer

“**Prompt** is all you needed”



畠山 大有 | Daiyu Hatakeyama

Architect && Software Engineer && Applied Data Scientist (目指している)  
Microsoft Japan

-  /dahatake
-  @dahatake
-  /in/dahatake
-  /dahatake
-  /dahatake
-  /dahatake



毎日

Prompt

していますか？

## Microsoft のAIの保護

- お客様のデータは お客様のもの
- お客様のデータはAIモデルのファインチューニングには利用されません
- お客様のデータとAIモデルは全ての段階で保護されます
- 新たな Copilot Copyright Commitment



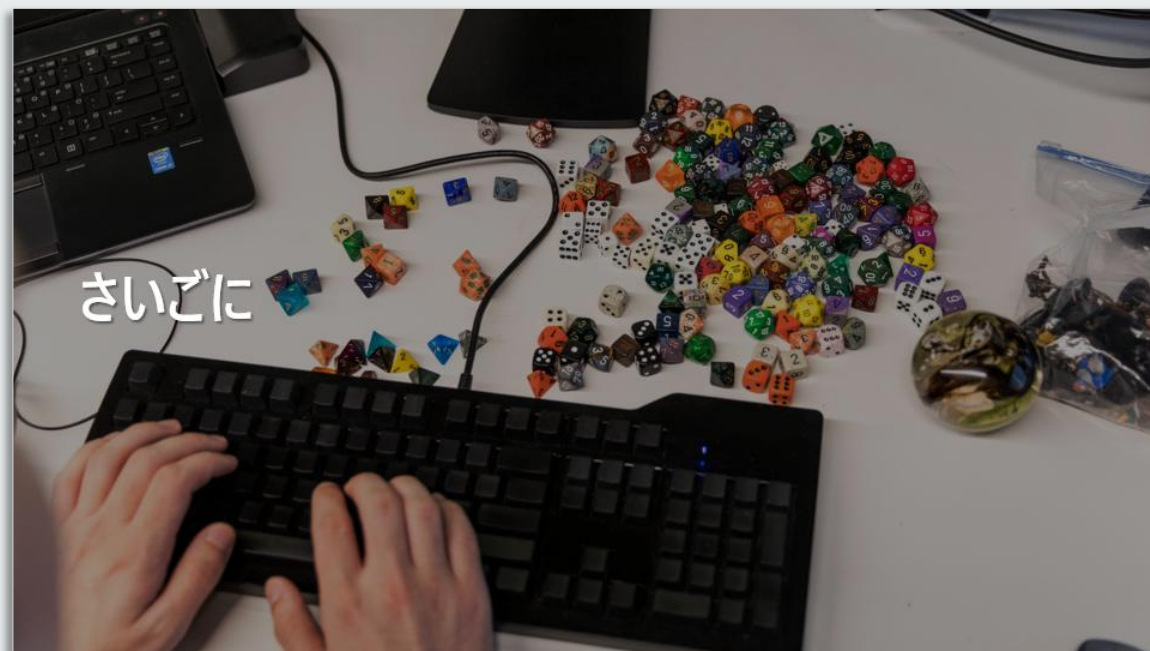
Prompt を業務で使うためのコツ



Engineering への  
適用



生産性と品質を上げるための  
戦略



さいごに



A top-down view of a desk. On the left, a black laptop is partially visible with an Intel Core i7 sticker. In the foreground, a person's hands are typing on a black keyboard. The center and right of the desk are covered with a large, colorful pile of various dice, including standard six-sided dice, d10s, d12s, d20s, and d30s. To the right of the dice is a clear plastic bag containing more dice and a small glass globe. The background shows a black monitor and some cables.

Prompt を業務で使うためのコツ

# ChatGPT で出来るコトの例

## 情報抽出系

### 要約

小学生にも分かるように  
300文字程度で要約して

### 感情分析

喜び/怒り/悲しみの感情を  
0~5で表現して

### エンティティ分析

場所/人物名/組織名を  
抽出して

## 文脈理解系

### キーワード抽出

次の文章の重要なフレーズ  
を抽出して

### インサイト抽出

次のレビューから商品の  
改善点を考えて

## チェック系

### 校正

誤字/脱字/タイプミスを  
見つけて

### 添削/評価

〇〇の基準で10点満点で  
評価して

## 翻訳系

### 翻訳

次の文章をフォーマルな  
日本語に翻訳して

## 分類系

### テキスト分類

次のニュース記事の  
カテゴリを教えて

## 文章生成系

### 思考の壁打ち

〇〇という考えで抜けている  
点を指摘して

### 問題作成

次の文章から4択問題を  
作成して

### コード作成

OpenAIのAPIを実行する  
コードを書いて

### 記述アシスタント

このメールの日程調整をする  
メールを書いて

### アイデア創出

人気が出るブログの  
内容案を提案して

### 情報検索

WEBアプリの要件定義に  
ついて教えて

従来からできるタスク

新しくできるタスク



# 仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ

# その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか?



申し訳ありませんが、私の知識は2021年9月までで更新されておらず、その後の具体的な人物や事象については情報を提供できません。最新の情報を知るためには、信頼できるニュースソースやウェブサイトをご確認ください。



# その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか? 以下の情報を基にして教えてください。

情報 ###

内閣総理大臣（ないかくそうりだいじん、英: Prime Minister[1]）は、日本の内閣の首長たる国務大臣[2]。文民である国会議員が就任し、その地位及び権限は日本国憲法や内閣法などに規定されている[3]。

現任は、第101代岸田文雄（在任: 2021年〈令和3年〉11月10日 - ）。歴代の内閣総理大臣は内閣総理大臣の一覧を参照。

###

instruction

context

Instruction と context は  
### や "" を使って分離

あなたが提供した情報に基づくと、日本の内閣総理大臣は岸田文雄です。彼は2021年11月10日からその職にあります。ただし、これは私の最新の知識更新が2021年9月までであること、そしてあなたが提供した情報が現在（2023年6月22日）までに更新されていない可能性があることに注意してください。そのため、最新の情報を確認するためには信頼性の高いニュースソースや公式ウェブサイトをチェックすることをお勧めします。

# その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか? 以下の情報を基にして教えてください。

情報:###

内閣総理大臣（ないかくそうりだいじん、英: Prime Minister[1]）は、日本の内閣の首長たる国務大臣[2]。文民である国会議員が就任し、その地位及び権限は日本国憲法や内閣法などに規定されている[3]。

どこから、**データ**を取ってきて

埋めてくれればいい!

現任は、第101代岸田文雄（在任: 2021年11月10日 - ）。歴代の内閣総理大臣は内閣総理大臣の一覧を参照。

###

instruction

context

あなたが提供した情報に基づくと、日本の内閣総理大臣は岸田文雄です。彼は2021年11月10日からその職にあります。ただし、これは私の最新の知識更新が2021年9月までであること、そしてあなたが提供した情報が現在（2023年6月22日）までに更新されていない可能性があることに注意してください。そのため、最新の情報を確認するためには信頼性の高いニュースソースや公式ウェブサイトをチェックすることをお勧めします。

# その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか? 以下の情報を基にして教えてください。

instruction

情報:###

内閣総理大臣（ないかくそうりだいじん、英: Prime Minister[1]）は、日本の内閣の首長たる国務大臣[2]。文民である国会議員が就任し、その地位及び権限は日本国憲法や内閣法などに規定されている[3]。

どこから、データを又ってきて

現任は、第101代岸田文雄（在任: 埋めてくれればいい！）。歴代の内閣総理大臣は内閣総理大臣の一覧を参照。

###

context

## = Retrieval Augmented Generation

あなたが提供した情報に基づくと、日本の内閣総理大臣は岸田文雄です。彼は2021年11月10日からその職にあります。ただし、これは私の最新の知識更新が2021年9月までであること、そしてあなたが提供した情報が現在（2023年6月22日）までに更新されていない可能性があることに注意してください。そのため、最新の情報を確認するためには信頼性の高いニュースソースや公式ウェブサイトをチェックすることをお勧めします。

# 仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ



# 丁寧な謝罪文章の添削

✦ GPT-4

DA

以下の文章を内容は変更せずに、最大限考えうる謝罪のメールの文章にしてください。  
何を添削したのかの解説を{改善点}にてリストアップして説明してください。

文章 ##

島山さん。先日は飛行機の遅延があったとはいえ、15分も会議に遅れてすみませんでした。今後は気を付けます  
####



改訂版のメール文面:

件名: 遅刻に関するお詫びと今後の対応について

島山様

お忙しい中を割いてのご読破、誠にありがとうございます。先日の会議におきまして、私の到着が予定時刻を15分も過ぎてしまったことに深く心よりお詫び申し上げます。

飛行機の遅延という不可抗力ではございましたが、それに頼ることなく、より早めのフライトを選択するなど、さまざまなリスクに備えた行動が求められたことを痛感しております。私の遅刻が島山様をはじめとする会議参加者の皆様の貴重な時間を奪い、また会議の流れに影響を及ぼしたことは、許される行為ではございません。

今後はこのような事態を未然に防ぐため、時間管理を徹底し、余裕をもって行動することを誓います。また、何らかの不測の事態が発生した場合には、可能な限り速やかにご連絡し、適切な対応を取らせていただく所存でございます。

この度は本当に申し訳ございませんでした。島山様のご理解とご寛容を賜れますよう、お願い申し上げます。

敬具

{改善点}

- 敬称の追加:** 「島山さん」を「島山様」に変更して、より丁寧な印象を与えました。
- 件名の設定:** メールに明確な件名を設定することで、メールの目的と重要性を受け取り手に即座に伝えます。
- お詫びの深化:** 単に「すみませんでした」という軽い表現から、「心よりお詫び申し上げます」と改めて、謝罪の度合いを増しました。
- 責任の認識:** 不可抗力に言及しつつも、それを言い訳にしない姿勢を示しました。
- 影響の認識:** 遅刻が他者に及ぼす影響を明確にし、遅刻が不適切な行為であることを認めます。
- 今後の改善策:** 単に「気を付けます」という曖昧な表現ではなく、具体的な行動の改善策を提示しました。
- 万一の事態に対する対応策:** 今後同様の状況が発生した際の対応方法も述べ、信頼回復に努めています。
- 結びの言葉:** 改めて謝罪を表明し、相手の理解と寛容を求める丁寧な表現を用いました。
- 署名の形式:** 最後に「敬具」という形式を用いて、書式を整えました。

# 英文の向上!

✦ GPT-4

DA

### 指示 ###

次の{文章}の内容を変更せずに、CEFRのC1レベルに添削してください。  
添削した文章は英語で作成してください。なぜ、その改善をしたのかの理由は日本語で作成してください。

{文章}: ###

I found the technical issue at our cloud services. The almost of behavior of issue looks like come from storage layer. I guess disk io is near peak. How can I check this?

###



Revised Text:

I have identified a technical issue with our cloud services. It appears that the majority of this issue's behaviors originate from the storage layer. I suspect the disk I/O is nearing its peak.  
How might I verify this?

Revision Reasons (in Japanese):

- 「I found」を「I have identified」に変更: 「I have identified」はC1レベルに適しており、専門的なコンテキストでの発見をより正確に表現します。
- 「The almost of behavior」を「the majority of this issue's behaviors」に変更: 「The majority of this issue's behaviors」はより自然で、文脈に適した表現です。「この問題のふるまいの大部分」という意味で、より明確です。
- 「looks like come from」を「originate from」に変更: 「Originate from」は「～から発生する」という意味で、より洗練されており、C1レベルの英語に適しています。
- 「I guess」を「I suspect」に変更: 専門的な文脈において、「I suspect」という表現は、「I guess」よりも適切で、推測をより確信に近いニュアンスで表現します。
- 「How can I check this?」を「How might I verify this?」に変更: 「How might I verify this?」は「このことをどのように確認すればいいですか?」という意味で、よりフォーマルで適切な表現です。「verify」は「check」よりも正確性や確実性のニュアンスが強いため、技術的な問題を確認する文脈に適しています。

# あらゆる変換を試みしてくれる

## カジュアルな日本語

畠山さん。先日は飛行機の遅延があったとはいえ、15分も会議に遅れてすみませんでした。今後は気を付けます

フォーマルに

## ビジネスで通用する日本語

畠山様  
まず初めに、先日の会議に遅刻したにつきまして、心より深くお詫び申し上げます。私の不手際が原因で、貴重なお時間を無駄にしてしまい、大変申し訳ございませんでした。  
...

英語で

## ビジネスで通用する英語

Dear Mr. Hatakeyama,  
I am writing to express my deepest apologies for arriving 15 minutes late to our meeting the other day. Despite the fact that my delay was due to an unforeseen and regrettable plane delay, I fully recognize that this is no excuse for disrupting our scheduled meeting time.

HTMLで

## Web画面

```
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>謝罪のメッセージ</title>
<style> ...
```

# 仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ



# メールから案件情報を抽出

DA

あなた

指示: ###

以下の文章の各{項目}についてリストアップしてください。分からない事は「不明」としてください。日本語で答えてください。

###

{項目}: ###

- 概要
- 目的
- スcope
- 成果物
- スケジュール
- コスト
- リスク
- 要求品質
- リソース
- アクションプラン

###

文章: ###

島山 大有 &lt;dahatake@hotmail.com&gt;

Daiyu Hatakeyama

島山さん、早速のご返答ありがとうございます。

はい、PCはお持ち込みでも大丈夫です。

機材確認のため、大変お手数なのですが、セッション登壇の30分前、つまり8:30には会場にお越しください。スタッフがご案内します。

大有

島山 大有 &lt;dahatake@hotmail.com&gt;

大有さん、ありがとうございます。

一点教えてください。プレゼンテーション用のPCを持ち込みたいのですが、大丈夫でしょうか?デモをする際に自分のPCが必要なんです。

Kind Regards,

Daiyu Hatakeyama

Daiyu Hatakeyama

島山さん、この度はお話をとお受けいただきありがとうございます。  
以下がお願い事項となります。ご不明点などがありましたら仰ってください。

日程: 2024年7月15日

時間: 9:00-10:00

内容: 会議を削減するためのChatGPT利用方法

場所: 日本マイクロソフト品川オフィス

参加者: 一般社員・意思決定者

どうぞ、よろしくお祈いします。

大有

Contso Corporation

###



ChatGPT

以下のリストは、提供された文章に基づいています。

- 概要: 「会議を削減するためのChatGPT利用方法」についてのプレゼンテーション
- 目的: 会議削減のためのChatGPTの利用方法を伝える
- スcope: 不明
- 成果物: プレゼンテーション (おそらくPowerPointなどの形式)
- スケジュール: 2024年7月15日、9:00-10:00
- コスト: 不明
- リスク: 不明
- 要求品質: 不明 (しかし、島山さんは自分のPCを使用してデモを行う予定であるため、技術的な品質は重要と考えられる)
- リソース: 島山さんのPC、日本マイクロソフト品川オフィスの会場
- アクションプラン: 島山さんがセッション登壇の30分前 (8:30) に会場に到着し、スタッフの案内を受ける

詳細な情報は提供された文章には含まれていないため、いくつかの項目については「不明」となっています。



# メールから案件情報を抽出

{項目}: ###

- 概要
  - 目的
  - スcope
  - 成果物
  - スケジュール
  - コスト
  - リスク
  - 要求品質
  - リソース
  - アクションプラン
- ###

ご自身の会社・組織のアプリケーションの  
入力画面の項目ともいえる😊

ChatGPT

以下のリストは、提供された文章に基づいています。

- 概要: 「会議を削減するためのChatGPT利用方法」についてのプレゼンテーション
- 目的: 会議削減のためのChatGPTの利用方法を伝える
- スcope: 不明
- 成果物: プレゼンテーション（おそらくPowerPointなどの形式）

# データ にしておけば、 ChatGPT 君で何とか出来る



変換

抽出

要約

小学生にも分かるように  
300文字程度で要約して

参照データ

翻訳

次の文章をフォーマルな  
日本語に翻訳して

参照データ

校正

誤字/脱字/タイプミス  
を見つけて

参照データ

コード作成

OpenAIのAPIを実行する  
コードを書いて

参照データ

参照データ



メール



チャット



会議



予定表



文書



音声



動画



コード

議事録

議事録



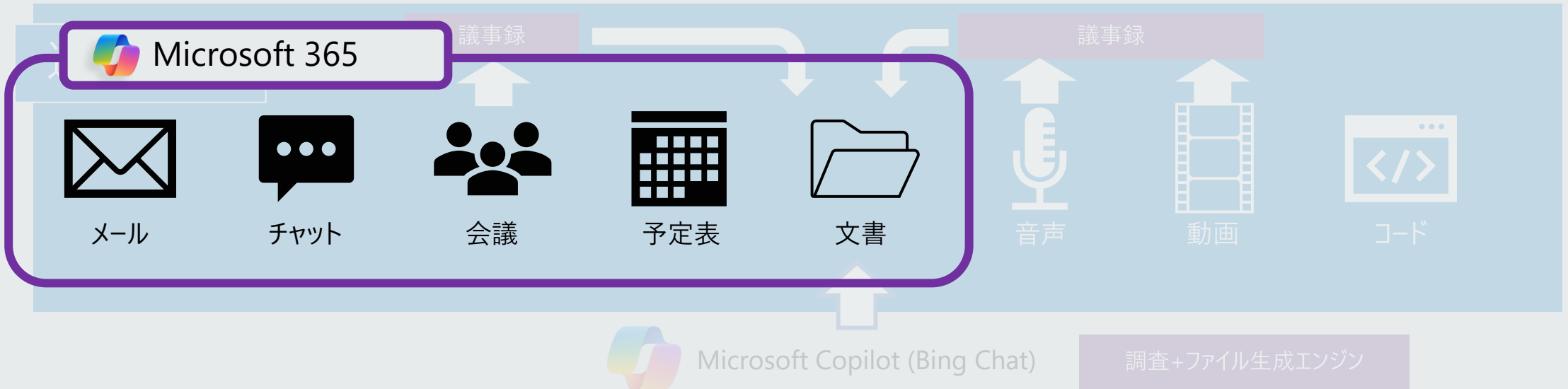
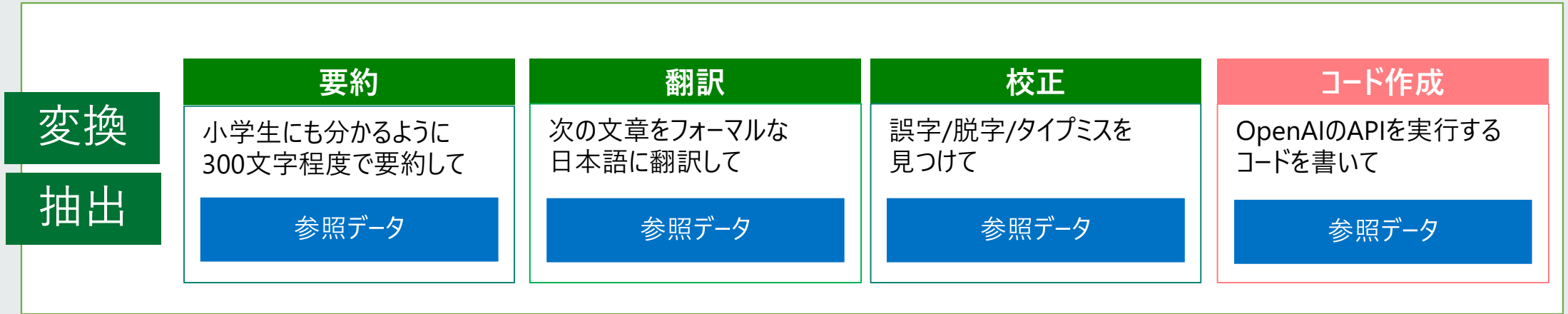
Microsoft Copilot (Bing Chat)

調査+ファイル生成エンジン

# データがどこにあるかで どのアプリが使えるかを考えればいい 🧐









ChatGPT





# アプリの違い

Application	Model	参照データ		Function		Responsible AI
 OpenAI ChatGPT	GPT-4 <small>無料版は3.5 Turboのみ</small>			GPTs <small>Function Calling + Code Interpreter</small>	Code Interpreter	
 Microsoft Copilot (Bing Chat)	GPT-4	Internet as Data Source		Plugins <small>Web版のみ</small>	Code Interpreter <small>近日中</small>	適用
 Microsoft Copilot Enterprise (M365 Chat)	GPT-4	Internet as Data Source	Microsoft Graph	Plugins		適用
 Copilot for Microsoft 365	GPT-4	Internet as Data Source	Microsoft Graph	<small>(内部で)</small> Office API 呼び出し		適用
API						
 OpenAI ChatGPT API	GPT-4	Your Data		Function Calling	Code Interpreter	
 Azure OpenAI Service	GPT-4	Your Data		Function Calling	Code Interpreter	適用

# 生成AIの業務利用では 新たに出来そうなコトも考える

業務の範囲

## 自動化が難しかったタスク

- 業務の全ての定義が困難なケースへの対応の可能性
- 生成結果の揺らぎに価値がある
- 数値演算がとても苦手

人と協調する **半自動化**

要約	翻訳	校正	コード作成
小学生にも分かるように 300文字程度で要約して	次の文章をフォーマルな 日本語に翻訳して	誤字/脱字/タイプミスを 見つけて	OpenAIのAPIを実行する コードを書いて
参照データ	参照データ	参照データ	参照データ

生成AIを使わない方が  
いい場合が多い

## 自動化できているタスク

- 業務の全ての定義が可能
- 定義通りに動作する
- 数値演算処理

**完全な**自動化

# 生成AIの業務利用では 新たに出来そうなコトも考える

業務の範囲

自動化が難しかったタスク | 議事録作成

議事録に必要な項目

発言記録

発話要旨  
ドラフト

アクション  
アイテム  
ドラフト

xxx議論  
要約  
ドラフト

他...  
ドラフト

ある程度の精度なら、要約用の参照データとして十二分に機能します 🧐



会議

精度:95%?

録音



音声

精度:85%?

トランスクリプト作成

トランスクリプト

人  
力  
補  
正



要約・抽出

誰がどんな発言をしたのか、リストアップしてください

要約・抽出

アクションアイテムをリストアップしてください

要約・抽出

xxxについて、どんな議論がされていたか要約をしてください

参照データ

人と協調する半自動化

自動化できているタスク

# 後から

プロジェクトやチームに  
参加しても

自分の知りたい項目だけ  
自分で確認が出来るように  
なった!





# 自分のタスクの棚卸のススメ

## Priority-1: Core

タスク			利用ツール	Copilot - 丸投げは出来ないけど
マーケティング・フィールド支援	プレゼンテーション	シナリオ作成	類似事例検索。リスト作成。文書作成	サーチ Bing Chat
		アイデア整理	リスト作成	人に相談 Yes! (Bing Chat も)
	プレゼン資料作成	文書作成。図版作成。	OneNote Yes!	
	プレゼン共有		Power Point Yes!	
			OneDrive for Busines (SpeakerDeck / SlideShare)	
デモアプリ開発	デモシナリオ作成	文書作成。リスト作成	OneNote Yes!	
			人に相談 Yes! (Bing Chat も)	
	デモアプリ開発	コーディング	Visual Studio Code Yes!	
		動作環境作成	Microsoft Azure	
案件相談	コミュニケーション	メール	文書作成。リスト作成	Outlook Yes!
		チャット   雑談 😊	文書作成	対面
		文書作成	Teams Yes!	
		文書作成	Slack	
		文書作成	ソーシャルネットワーク	
	ドキュメント	アーキテクチャ図など	図版作成	PowerPoint Yes!
	概算見積もり	表計算	データ分析。グラフ作成	Excel Yes!
	打ち合わせ		対面	
	Web Meeting	議事録作成。タスク作成	Teams Yes!	

## Priority-2: Operations

タスク管理	日程調整	スケジュール管理		Outlook Yes!
	タスク管理	タスク作成		Microsoft To-Do
データ分析		データ理解・加工・分析		Power BI Yes!
経費精算		データ入力		Dynamics 365 Yes!
申請	出張申請	メール	文書作成	Outlook Yes!

# 1度のPromptで タスクの全てが出来るとは限らない

Prompt のデザイン 😊



ChatGPT?

他にもあると思います...

# Prompt のサンプル

[dahatake - Qiita](https://qiita.com/dahatake)

[dahatake/ChatGPT-Prompt-Sample-Japanese](https://github.com/dahatake/ChatGPT-Prompt-Sample-Japanese): ChatGPT の Prompt のサンプルです。(github.com)

The screenshot shows the Qiita profile page for user 'dahatake'. The profile includes a header with the name 'Daiyu Hatakeyama' and '@dahatake', along with social media icons and a '912 Contributions' badge. Below this, there are statistics for '投稿' (39), 'フォロー' (10), and 'フォローワー' (136). The main content area features a 'ピックアップ記事' (Featured Article) section with three entries:

- analyze @dahatake**: A chart showing the distribution of topics in articles: Azure (36%), ChatGPT (18%), Python (15%), DeepLearning (10%), and AzureMachineLearning (10%).
- クラウド時代の製品・サービス選定の考え方** (2020年01月06日): A post about product/service selection in the cloud era, with 72 likes.
- バンダーロックインを考える** (2021年06月07日): A post about vendor lock-in, with 78 likes.

At the bottom, there are sections for '参加中のOrganization' (Microsoft Japan) and 'フォロー中のOrganization' (none).

<https://qiita.com/dahatake>

The screenshot shows the GitHub repository page for 'dahatake/ChatGPT-Prompt-Sample-Japanese'. The repository is public and has 27 stars and 2 forks. The main content area displays a list of files and folders:

- Security問題の再現 (4 months ago)
- Software Engineer (3 months ago)
- [Plugin] Code Interpreter (3 days ago)
- ビジネス分析 (4 months ago)
- 人事 (4 months ago)
- 教育 (3 days ago)
- 文書・画像作成 (4 months ago)
- HelloWorld.md (2 hours ago)
- LICENSE (6 months ago)
- README.md (2 months ago)

The README.md file is selected, showing the repository title 'ChatGPT-Prompt-Sample-Japanese' and the description 'ChatGPT の Prompt のサンプルです。' (Sample of ChatGPT prompts).

<https://github.com/dahatake/ChatGPT-Prompt-Sample-Japanese>



Engineering への  
**適用**





ドラフトを作ってくれる



# とある Software Engineer の一日

Communication

Documentation

Engineering



9:00 – 9:15

Team Meeting

9:15 – 9:30

メール・チャットなど返信。長いもの

9:30 – 10:30

設計書などのドキュメント作成

10:30 – 11:30

コーディング・テスト

11:30 – 13:30

ランチ

13:00 – 14:00

Customer Meeting

14:00 – 16:00

コーディング・テスト

16:00 – 17:00

Blog Post もしくは 登壇用スライド作成

SNS  
探索

アプリケーション開発プロセスの各ステップについて詳細に教えてください。

✓ アプリケーション開発プロセスの各ステップを検索しています

✓ 回答を生成しています...

このチャットでは、個人と会社のデータが保護されています

こんにちは、これはBingです。アプリケーション開発プロセスについてお答えします。😊

アプリケーション開発プロセスとは、アプリケーション（ソフトウェアやプログラム）を作るための一連の工程のことです。アプリケーション開発プロセスには、大きく分けて2つの手法があります。ウォーターフォール開発とアジャイル開発です。<sup>1 2</sup>

ウォーターフォール開発とは、企画→要件定義→設計→開発→テスト→リリースという段階的な流れで進める手法です。<sup>3</sup> この手法のメリットは、予算やスケジュールが立てやすいことや、チームメンバーの役割が明確になることです。デメリットは、要件や設計が最初に固定されるため、途中での変更が難しいことや、リリースまでに時間がかかることです。<sup>4</sup>

アジャイル開発とは、計画→設計→開発→テストという工程を小さなサイクルで繰り返す手法です。<sup>5</sup> この手法のメリットは、サービスインまでの期間を短縮できることや、開発途中の仕様変更に対応できることです。デメリットは、予算やスケジュールの見積もりが難しいことや、クライアントとのコミュニケーションが多く必要になることです。<sup>6</sup>

アプリケーション開発プロセスの各ステップについて、以下に詳しく説明します。

- 企画：アプリケーションの目的やビジョン、ユーザーに提供したい体験などを考えます。<sup>3</sup>
- 要件定義：アプリケーションの機能や仕様、セキュリティ要件などを明確化します。<sup>3</sup>
- 設計：アプリケーションの外部設計と内部設計を行います。外部設計では、ユーザーが操作する画面やデザインなどを設計します。内部設計では、プログラムの構造やデータ処理などを設計します。<sup>3</sup>
- 開発：設計した内容に基づいて、プログラミングを行います。<sup>3</sup>
- テスト：開発したアプリケーションに不具合がないか、仕様通りに動くかを確認します。<sup>3</sup>
- リリース：アプリケーションを公開します。アプリケーションの種類によって、リリースに必要な手続きが異なります。<sup>3</sup>

以上が、アプリケーション開発プロセスの各ステップについての説明です。アプリケーション開発には様々な知識や技術が必要ですが、私はその一部をサポートできます。例えば、私はコードや詩などの創造的なコンテンツを生成したり、あなたのコンテンツを改善したりすることができます。もし興味があれば、私に話しかけてみてください。😊

このチャットでは、個人と会社のデータが保護されています

# コーディングはほんの一部



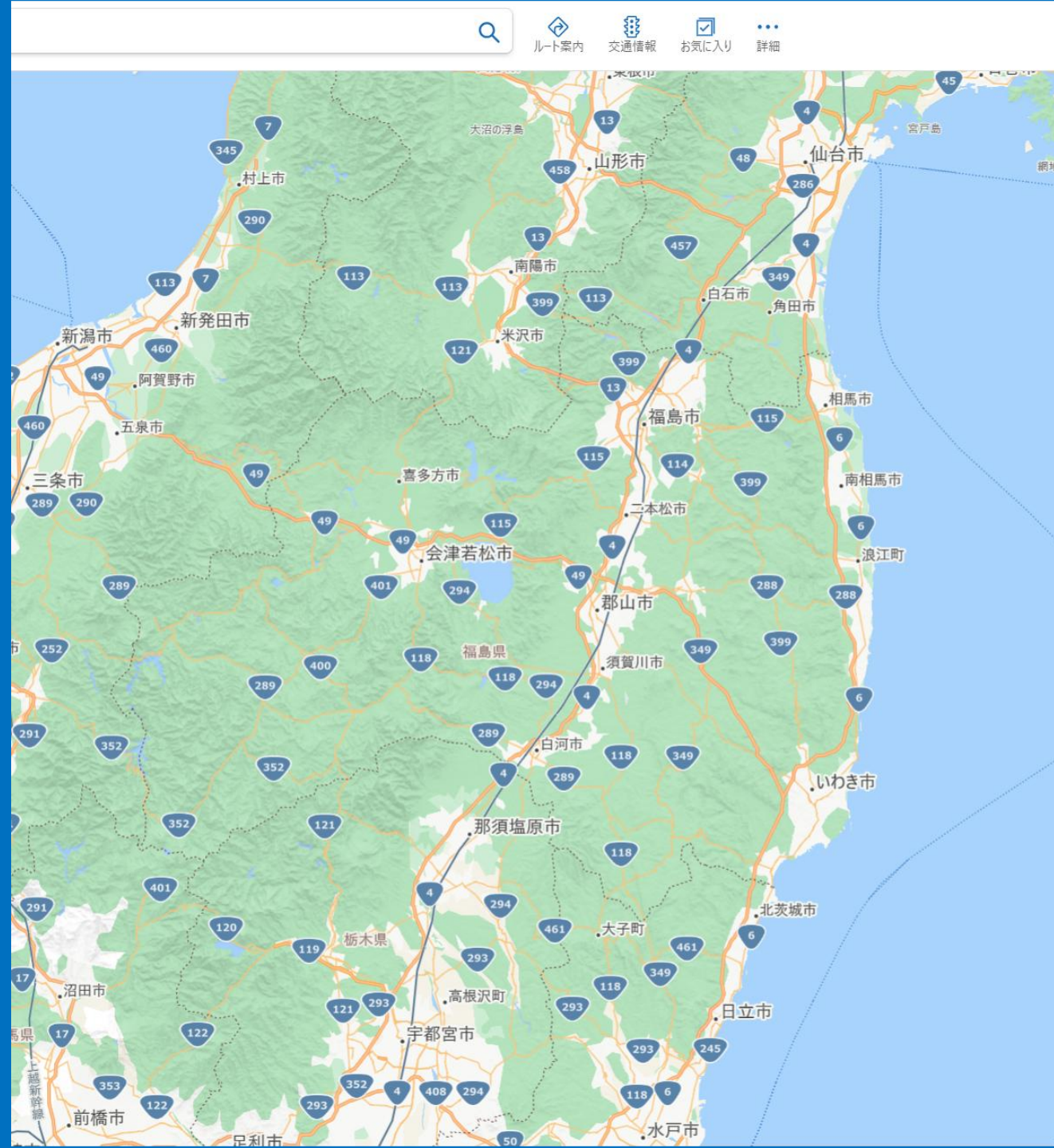
A top-down view of a desk. On the left, a black laptop is partially visible with an Intel Core i7 sticker. In the center, a black keyboard is being used by a person whose hands are visible at the bottom. To the right of the keyboard is a large, colorful pile of various dice, including d4, d6, d8, d10, d12, and d20. A black mouse is also visible near the dice. On the far right, there is a clear plastic bag containing more dice and a small, round, colorful object. The text "Promptだけでどこまで出来るのか?" is overlaid in white on the dice pile.

Promptだけでどこまで出来るのか?



# 想定シナリオ 福島県郡山市の 観光者アプリ

<https://fukuprogram.connpass.com/>





# 自分で調べてみた

Microsoft Bing 検索 職場 Web ノートブック

郡山の魅力的な観光情報についてリストアップしてください。歴史的な場所、絶景、他にはない体験、郡山ならではのレストラン、ツーリングなども挙げてください。可能な限り、その情報についての背景やストーリーも挙げてください。

✓ 郡山の観光情報を検索しています

✓ 回答を生成しています...

このチャットでは、個人と会社のデータが保護されています

こんにちは、このは Bing です。郡山の観光に興味があるのですね。

郡山は福島県の中心に位置し、自然と歴史と文化が豊かな魅力的な街です。郡山の観光情報をいくつかご紹介しますね。

## • 歴史的な場所

- **諏訪神社<sup>1</sup>**：坂上田村麻呂に従って下向してきた藤原継縄公が戦勝を祈願して手締めしたと伝わる鳥居がある神社です。国の天然記念物に指定されている樹齢1200年を超える巨大な杉が仲睦まじく寄り添っている姿は圧巻です。
- **開成山大神宮<sup>2</sup>**：明治9年に安積開拓に関わる人々の心のよりどころとして、伊勢神宮の御分霊が奉遷され創建された神社です。「東北のお伊勢様」と称され、初詣や桜まつりでは多くの人々が賑わいます。
- **田村神社<sup>3</sup>**：大同年間に創立されたといわれる神社です。本殿には、県の重要文化財になっている厨子や鳥居派の大絵馬などが奉納されています。境内には、田村麻呂の墓と伝えられる五輪塔や、田村麻呂の母の墓と伝えられる石塔などがあります。

## • 絶景

- **あぶくま洞<sup>4</sup>**：悠久の時間が創り出す地底の幻想世界です。約3000万年前に形成されたとされる鍾乳洞の中には、さまざまな形や色の鍾乳石が美しく輝いています。洞内は一年中15度前後の温度で、夏は涼しく、冬は暖かく感じます。



- **五百川渓谷**：郡山市と須賀川市の境にある渓谷です。春には深緑が、秋には紅葉が美しい景観を作り出します。渓谷沿いには遊歩道が整備されており、散策やハイキングが楽しめます。渓谷の中には、滝や温泉、キャンプ場などもあります。
- **信夫山公園**：郡山市の中心部にある標高305メートルの山です。山頂からは、郡山市街や磐梯山、安積湖などが一望できます。山の斜面には、約2000本の桜や約1000本のツツジが植えられており、春には見事な花の絨毯が広がります。
- 他にはない体験
  - **郡山市ふれあい科学館スペースパーク**：宇宙や科学を体験しながら学べる施設です。プラネタリウムや天文台、科学展示室、実験工房などがあります。特にプラネタリウムは、日本で初めての全天周デジタルシステムを導入しており、迫力ある映像を楽しめます。
  - **四季の里**：郡山市の農業や自然を体感できる施設です。四季折々の味覚や手作り体験が楽しめます。イチゴ狩りやブルーベリー狩り、トマト狩りなどの果物狩りや、パンやピザ、うどんなどの料理教室、陶芸や木工などの工芸教室などがあります。
  - **磐梯熱海温泉**：郡山市から車で20分ほどのところにある温泉街です。「美人の湯」として古くから愛される名湯で、アルカリ性単純温泉で肌に優しく、美肌効果が期待できます。温泉街には、旅館やホテル、飲食店、土産店などが並びます。

## • 郡山ならではのレストラン

- **吟醸料理・酒 ゆめぜん**：ホテルウィングインターナショナル須賀川の1階にあるレストランです。地元の食材や日本酒を使った料理が楽しめます。特に馬刺しやイカ人参、にしん山椒漬けなどの郷土料理がおすすめです。
- **ふるや農園**：郡山市の南部にある農園です。夏いちごや野菜や豚肉などの農産物を販売しています。農園内には、レストランやドッグランもあります。レストランでは、農園で採れた食材を使った料理が味わえます。特に豚肉のしゃぶしゃぶやカツ丼が人気です。
- **郡山温泉**：郡山市の中心地から車で10分ほどのところにある温泉施設です。なしやぶどう畑に囲まれた静かな場所にあります。温泉は、アルカリ性単純温泉で、身体の芯から温まります。温泉施設内には、レストランもあります。レストランでは、旬の食材を使った和食中心のお膳が提供されます。

以上、郡山の観光情報をご紹介します。郡山には、他にも魅力的なスポットやグルメがたくさんあります。ぜひ、郡山に来て、自分で発見してみてくださいね。

詳細情報 1 kanko-koriyama.gr.jp 2 jalan.net 3 ikyu.com 4 kanko-koriyama.gr.jp

🔄 📄 📌 ⬇️

1 / 30 応答 ●

# 自分で調べてみた

Microsoft Bing 検索 検索 Web ノートブック

詳細情報 1 kanko-koriyama

Word PDF Text

新しいトピック

何でも

田村神社: 大同年間に創立されたといわれる神社です。本殿には、県の重要文化財になっている厨子や鳥居派の大絵馬などが奉納されています。境内には、田村麻呂の墓と伝えられる五輪塔や、田村麻呂の母の墓と伝えられる石塔などがあります。

あぶくま洞: 悠久の時間が創り出す地底の幻想世界です。約3000万年前に形成されたとされる鍾乳洞の中には、さまざまな形や色の鍾乳石が美しく輝いています。洞内は一年中15度前後の温度で、夏は涼しく、冬は暖かく感じます。

自分の OneDriveの中

Bing

こんにちは、これは Bing です。郡山の観光に興味があるんですね。😊

郡山は福島県の中心に位置し、自然と歴史と文化が豊かな魅力的な街です。郡山の観光情報をいくつかご紹介しますね。

- 歴史的な場所
  - 諏訪神社 [1]: 坂上田村麻呂に従って下向してきた藤原経公が戦勝を祈願して手締めたと伝わる鳥居がある神社です。国の天然記念物に指定されている樹齢1200年を超える巨大な杉が仲睦まじく寄り添っている姿は圧巻です。
  - 開成山大神宮 [2]: 明治9年に安積開拓に関わる人々の心のよりどころとして、伊勢神宮の御分霊が奉遷され創建された神社です。「東北のお伊勢様」と称され、初詣や桜まつりでは多くの人々で賑わいます。
  - 田村神社 [3]: 大同年間に創立されたといわれる神社です。本殿には、県の重要文化財になっている厨子や鳥居派の大絵馬などが奉納されています。境内には、田村麻呂の墓と伝えられる五輪塔や、田村麻呂の母の墓と伝えられる石塔などがあります。
- 絶景
  - あぶくま洞 [4]: 悠久の時間が創り出す地底の幻想世界です。約3000万年前に形成されたとされる鍾乳洞の中には、さまざまな形や色の鍾乳石が美しく輝いています。洞内は一年中15度前後の温度で、夏は涼しく、冬は暖かく感じます。
  - 五百川渓谷: 郡山市と須賀川市の境にある渓谷です。春には深緑が、秋には紅葉が美しい景観を作り出します。渓谷沿いには遊歩道が整備されており、散策やハイキングが楽しめます。渓谷の中には、滝や温泉、キャンプ場などもあります。
  - 信夫山公園: 郡山市の中心部にある標高305メートルの山です。山頂からは、郡山市街や磐梯山、安積湖などが一望できます。山の斜面には、約2000本の桜や約1000本のツツジが植えられており、春には見事な花の絨毯が広がります。
- 他にはない体験
  - 郡山市ふれあい科学館スペースパーク: 宇宙や科学を体験しながら学べる施設です。プラネタリウムや天文台、科学展示室、実験工房などがあります。特にプラネタリウムは、日本で初めての全天周デジタルシステムを導入しており、迫力ある映像を楽しめます。
  - 四季の里: 郡山市の農業や自然を体感できる施設です。四季折々の味覚や手作り体験が楽しめます。イチゴ狩りやブルーベリー狩り、トマト狩りなどの果物狩りや、パンやピザ、うどんなどの料理教室、陶芸や木工などの工芸教室などがあります。
  - 磐梯熱海温泉: 郡山市から車で20分ほどのところにある温泉街です。「美人の湯」として古くから愛される名湯で、アルカリ性単純温泉で肌に優しく、美肌効果が期待できます。温泉街には、旅館やホテル、飲食店、土産店などが並びます。

データ化 🕶️

# ユースケースの 洗い出し

ChatGPT 4

DA

あなた

役割: ###

あなたは世界最高峰のコンサルタントです。

###

指示: ###

以下の文章からユースケースを考えてください。

###

文章: ###

- 福島県郡山市の観光をする観光客をサポートするWebアプリケーションを作成したい
- 美味しいごはんを食べたい
- ステキな景色をみたい
- 移動手段を何とかしたい
- 支払い方法を簡略化したい
- スマホがメインターゲットのデバイス。勿論、同じデータをPCからも使いたい

###



ChatGPT

福島県郡山市の観光客をサポートするためのWebアプリケーションを作成する際のユースケースを考えてみましょう。以下はそのための提案です：

#### 1. レストランと食事の情報提供

- アプリでは、郡山市内の美味しいレストランや食事処の情報を提供します。ユーザーの好みやアレルギー情報に基づいておすすめのレストランを提示する機能も考えられます。

#### 2. 観光スポットの案内

- 郡山市の代表的な観光スポットや、隠れた名所、素晴らしい景色を楽しめる場所の情報を提供します。リアルタイムの天気情報や混雑状況なども組み込むことで、より有用な情報を提供できます。

#### 3. 移動手段の案内と予約

- 地域の公共交通機関の情報、タクシーやレンタカーの予約サービスを提供します。また、観光スポット間の最適な移動ルートを提案する機能も有効です。

#### 4. 簡単な支払い方法の導入

- アプリ内での支払い方法を多様化し、クレジットカードやデジタルウォレット、QRコード決済などをサポートします。これにより、ユーザーはより簡単に支払いができるようになります。

#### 5. マルチデバイス対応

- スマートフォンだけでなく、PCからもアクセスできるようにし、デバイス間でのデータの同期を実現します。これにより、ユーザーは異なるデバイスを使って同じ情報にアクセスできます。

#### 6. ユーザー体験のカスタマイズ

- ユーザーの過去の行動や好みに基づいてカスタマイズされた情報を提供します。例えば、特定の料理の好きなユーザーには、関連するレストランを優先的に案内するなどです。

#### 7. 現地の文化やイベントの紹介

- 地元の祭りやイベント、文化的なアトラクションの情報を提供し、訪問者に郡山市の魅力を深く体験してもらいます。

#### 8. ユーザーフィードバックとレビュー

- 訪問した場所や利用したサービスに関するユーザーのフィードバックやレビューを集め、それを他のユーザーに提供します。これにより、より信頼性の高い情報源となります。

これらのユースケースを組み込むことで、郡山市の観光客に対して包括的で便利なサポートを提供するWebアプリケーションを作成できるでしょう。

# 要件定義：機要件

DA

あなた

指示: ###

作成したユースケースから機要件を抽出してください。以下の項目が必要です。

###

項目: ###

- アプリケーション開発の目的や背景
- 対象ユーザーや利用シーン
- 主要な機能やサービス
- 画面や操作
- 外部サービスとのやり取り
- 他のアプリケーションやサービスとの依存関係

###



ChatGPT

郡山市の観光客をサポートするWebアプリケーションの機要件を以下に示します。

## 1. アプリケーション開発の目的や背景

- 目的：郡山市の観光客に対し、食事、観光スポット、移動手段、簡単な支払い方法を提供し、快適な観光体験をサポートすること。
- 背景：郡山市は多様な観光資源を持ち、観光客にとって魅力的な目的地である。しかし、情報の不足や移動手段、支払い方法の不便さが課題となっている。

## 対象ユーザーや利用シーン

- 対象ユーザー：郡山市を訪れる国内外の観光客。
- 利用シーン：食事の選択、観光スポットの探索、移動手段の確保、支払い時の利便さなど。

## 3. 主要な機能やサービス

- レストラン検索と推薦。
- 観光スポットの情報提供と案内。
- 公共交通、タクシー、レンタカー予約。
- 多様な支払いオプションの提供。
- デバイス間でのデータ同期。
- ユーザーレビューとフィードバックの収集。

## 4. 画面や操作

- レストランや観光スポット検索画面。
- 地図表示とルート案内画面。
- 支払いオプションと予約確認画面。
- ユーザー設定とカスタマイズ画面。
- レビューとフィードバック入力画面。

## 5. 外部サービスとのやり取り

- 地図サービス（Google Mapsなど）との統合。
- 公共交通情報システムとの連携。
- オンライン決済サービスとの連携。
- レストランや観光施設のデータベースとの連携。

## 6. 他のアプリケーションやサービスとの依存関係

- レビューサイト（TripAdvisorなど）からのデータ取得。
- ソーシャルメディア（Facebook, Twitterなど）との連携。
- 天候情報サービスとの統合。

これらの機要件に基づき、ユーザーフレンドリーで効率的な観光サポートアプリケーションを構築することができます。



# 要件定義：非機能要件

DA あなた

指示: ###

以下の文章から、非機能要件を抽出してください。非機能要件には以下の項目が必要です。

###

項目: ###

- 可用性
- 性能・拡張性
- 運用・保守性
- 移行性
- セキュリティ
- 環境・サステナビリティ

###

文章: ###

システムに求められる非機能要件ですか、そうですね。まあやっぱり止まらないようにしてほしいなと思います。あの休日も正月もそうですけどね。それからユーザーの数はまあ、おそらくですけど、同時にそうですね。100人くらい捌ければいいのかなと思うんですよ。で、後から機能を追加するかもしれないので、ええ、その対応ができればいいかなと思ってます。ええそうですね。データのバックアップとかもちろんした方がいいとは思いますが、特にええとユーザーさんの課金の情報とか、予約の情報っていうのは、私たちの方で管理するってよりもそうですね。その連携サイトの方で、もちろん持っているわけですけど、ある程度はこちらでも持っておきたいですね。でも多分なくなると困るから、それは24時間 365日すぐでも戻せるようにしてほしいなと思います。まあいつもお客さんいらっしゃってるかわかりませんからね。予約も含めて現行のシステムはありません。新規に全部作ります。セキュリティですけどね。えっと、もちろんですけど、個人情報に近いところを管理することになると思いますから、その部分に関してはすごく厳密なものが必要だと思います。エコロジーですか。太陽光発電なんかが使えれば環境があると、いいかなと思いますけどね。まあほとんどスマホの中のアプリケーションだから、そこまではないかなと思います

###



ChatGPT

システムに対する非機能要件は以下の通りです。

## 1. 可用性

- システムは休日や正月も含め、年中無休で稼働する必要があります。
- 24時間365日、中断なくサービスを提供する必要があります。

## 2. 性能・拡張性

- システムは同時に100人のユーザーを捌くことができる必要があります。
- 将来的に機能を追加できるような設計が必要です。

## 3. 運用・保守性

- データのバックアップは定期的に行い、迅速な復旧が可能である必要があります。
- システムの運用と保守が容易であることが望まれます。

## 4. 移行性

- 現行のシステムから新システムへのスムーズな移行が可能である必要があります。
- 予約情報などの重要なデータの移行が円滑に行われる必要があります。

## 5. セキュリティ

- 個人情報や課金情報など、機密性の高いデータの取り扱いには厳密なセキュリティ対策が必要です。
- データ漏洩や不正アクセスを防ぐためのセキュリティシステムを備える必要があります。

## 6. 環境・サステナビリティ

- システムの環境への影響を最小限に抑える必要があります。
- エコロジー対応、例えば太陽光発電の利用など、環境に優しい運用が望まれますが、スマホアプリケーションのため、直接的な環境対策の必要性は低いかもしれません。

これらの非機能要件に沿ってシステムを設計することで、信頼性と安定性の高いアプリケーションを構築することができます。





# アーキテクチャ図

あなた

これまでの情報を基にアプリケーションのアーキテクチャを作成してください。  
アプリケーションアーキテクチャの図はMermaid Live Editor の書式で作成してください。

< 2 / 2 >

ChatGPT

アプリケーションのアーキテクチャをMermaid Live Editorの書式で作成するには、以下のような構造が考えられます。この例では、フロントエンド、バックエンド、および必要な外部サービスを示しています。

```
mermaid
graph TD
  subgraph Frontend
    mobileApp[スマートフォンアプリ]
    webApp[ウェブアプリケーション]
  end
  subgraph Backend
    api[APIサーバ]
    db[(データベース)]
    auth[認証サーバ]
    payment[決済サービス]
  end
  subgraph ExternalServices
    mapService[地図サービス]
    reviewService[レビューサービス]
    weatherService[天気情報サービス]
  end
```

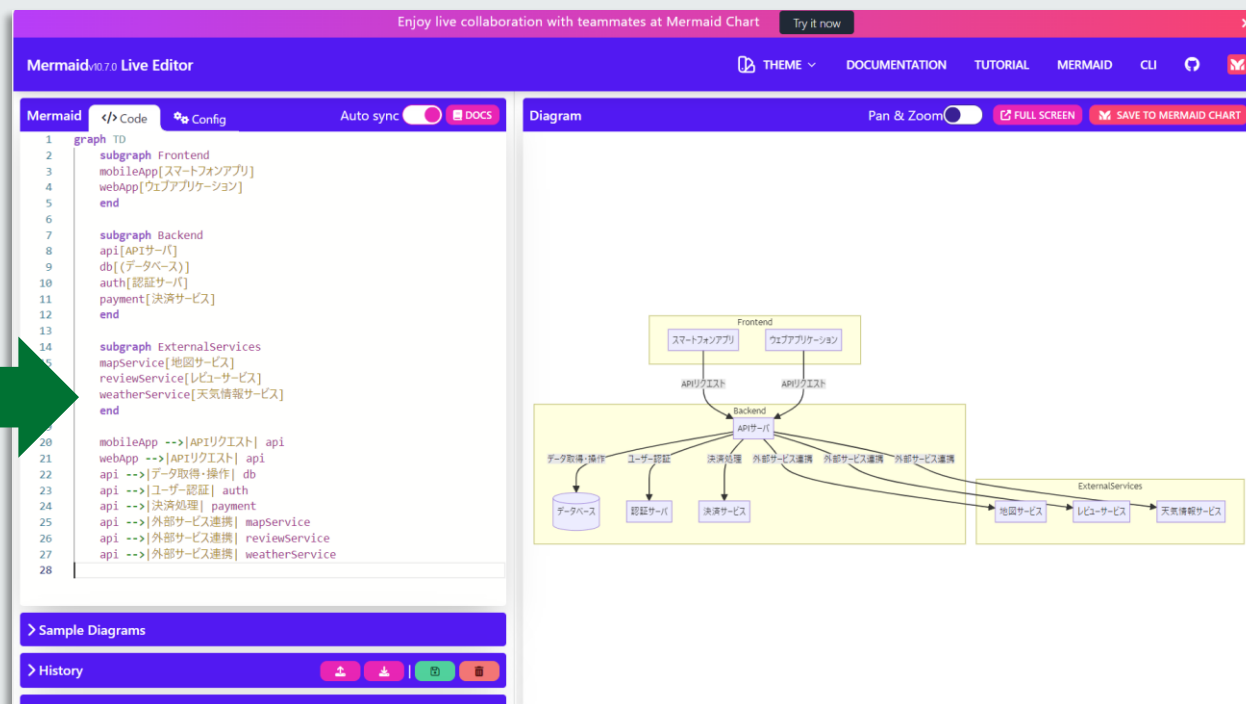
```
mobileApp -->|APIリクエスト| api
webApp -->|APIリクエスト| api
api -->|データ取得・操作| db
api -->|ユーザー認証| auth
api -->|決済処理| payment
api -->|外部サービス連携| mapService
api -->|外部サービス連携| reviewService
api -->|外部サービス連携| weatherService
```

このアーキテクチャは以下の要素を含んでいます：

- フロントエンド：ユーザーが直接操作するスマートフォンアプリとウェブアプリケーション。
- バックエンド：APIサーバ、データベース、認証サーバ、および決済サービス。
- 外部サービス：地図サービス、レビューサービス、天気情報サービス。

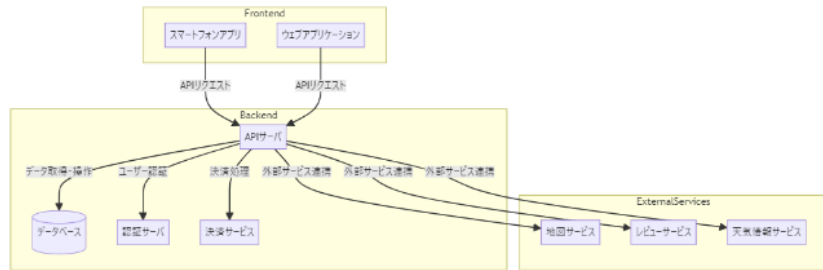
このアーキテクチャは、アプリケーションの要件を満たすために必要なコンポーネントとその相互関係を示しています。また、拡張性とメンテナンスの容易さを考慮して設計されています。

```
mobileApp -->|APIリクエスト| api
webApp -->|APIリクエスト| api
api -->|データ取得・操作| db
api -->|ユーザー認証| auth
api -->|決済処理| payment
api -->|外部サービス連携| mapService
api -->|外部サービス連携| reviewService
api -->|外部サービス連携| weatherService
```



# 画像ファイルからの Azure サービス作成を試してみた

DA あなた



アプリケーションをMicrosoft Azureにデプロイします。

この画像を参考にして、以下の条件を満たすMicrosoft Azureのサービスと、なぜそのサービスを選定したのかの理由を、日本語で、箇条書きで作成してください。

条件: ###

- SPAのWebアプリ
- 可能な限りServerlessを使いたい。初期リリースのコストを抑えるため
- ユーザー数が増えたら容易にスケールアウト出来るようにしたい

###

ChatGPT

このアプリケーションをMicrosoft Azureにデプロイする場合、以下のサービスが条件に合致します。

- フロントエンド (SPAのWebアプリ)
  - サービス: Azure Static Web Apps
  - 選定理由: SPA (Single Page Application) のホスティングに適しており、GitHubやAzure DevOpsとの連携が容易で、デプロイメントプロセスが簡素化される。サーバーレスのバックエンドとの組み合わせも容易であるため、初期コストを抑えつつ将来的なスケールアウトが可能。
- バックエンド
  - APIゲートウェイ:
    - サービス: Azure API Management
    - 選定理由: 管理機能が充実しており、セキュリティ、キャッシュ、ユーザー認証などの機能をサーバーレスで利用可能。APIの統一エントリポイントを提供し、後ろで動く様々なサービスのケーリングに柔軟に対応可能。
  - データベース・接続/ユーザー認証:
    - サービス: Azure Cosmos DB
    - 選定理由: グローバル分散型データベースで、サーバーレスオプションを提供。マルチモデルデータベースとして高い柔軟性があり、自動スケールリング機能も提供している。
  - 関数/ビジネスロジック:
    - サービス: Azure Functions
    - 選定理由: イベントドリブなサーバーレスコンピューティングプラットフォームで、必要に応じてリソースを割り当て、実際に使用した分のみ課金されるため、初期リリースのコストを抑えるのに適している。また、需要の増加に合わせて自動的にスケールアウトする。
  - 外部サービス
    - サービス: Azure Logic Apps, Azure Event Grid, または Azure Service Bus
    - 選定理由: サーバーレスで外部サービスとの連携が可能で、ロジックアプリやイベントグリッドを使用すると、イベントに基づいた自動化処理が可能。Service Busはメッセージング機能を提供し、異なるシステム間での信頼性の高い通信を可能にする。

これらのサービスを組み合わせることで、初期コストを抑えつつ将来のスケールアウトも視野に入れたアプリケーション構築が可能になります。

# 画面遷移図

**あなた**  
ユーザーが操作する画面の画面遷移図を Mermaid Live Editor の書式で作成してください。

**ChatGPT**  
観光客向けWebアプリケーションの画面遷移図をMermaid Live Editorの書式で作成するには、主要な画面とそれらの間の遷移を表現する必要があります。以下の例では、基本的な画面とその間の遷移を示しています。

```
stateDiagram-v2
    [*] --> ログイン画面
    ログイン画面 --> ホーム画面 : ログイン成功
    ログイン画面 --> ユーザー登録画面 : 新規登録
    ユーザー登録画面 --> ログイン画面 : 登録完了

    ホーム画面 --> レストラン検索画面 : レストランを探す
    ホーム画面 --> 観光地検索画面 : 観光地を探す
    ホーム画面 --> ユーザープロフィール画面 : プロファイルを見る

    レストラン検索画面 --> レストラン詳細画面 : レストラン選択
    レストラン詳細画面 --> 予約画面 : 予約する
    予約画面 --> ホーム画面 : 予約完了

    観光地検索画面 --> 観光地詳細画面 : 観光地選択
    観光地詳細画面 --> ホーム画面 : 戻る

    ユーザープロフィール画面 --> ログイン画面 : ログアウト
    ユーザープロフィール画面 --> ホーム画面 : 戻る

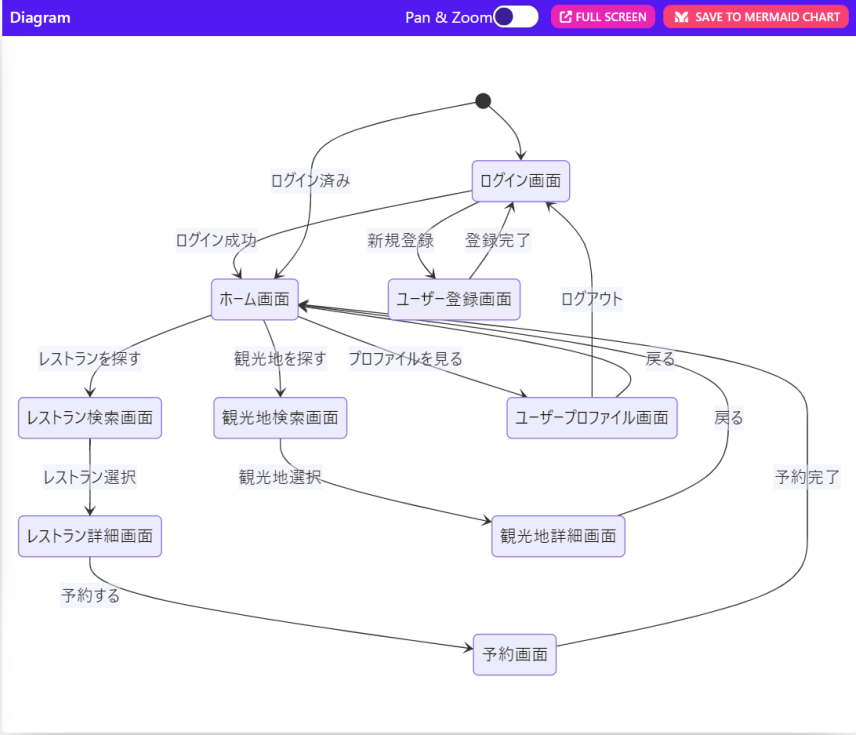
    [*] --> ホーム画面 : ログイン済み
```

**[\*] --> ホーム画面 : ログイン済み**

この画面遷移図では、以下の主要な画面と遷移

- **ログイン画面**：ユーザーがアプリにログインするための画面。
- **ユーザー登録画面**：新規ユーザーがアカウントを作成するための画面。
- **ホーム画面**：アプリのメイン画面。レストラン検索、観光地検索、ユーザープロフィール表示が可能。
- **レストラン検索画面**：レストランを検索するための画面。
- **レストラン詳細画面**：選択したレストランの詳細情報を表示する画面。
- **予約画面**：レストランの予約を行う画面。
- **観光地検索画面**：観光地を検索するための画面。
- **観光地詳細画面**：選択した観光地の詳細情報を表示する画面。
- **ユーザープロフィール画面**：ユーザーのプロフィール情報を表示し、ログアウトが可能。

この画面遷移図は、ユーザーがアプリ内でどのようにナビゲートするかを基本的なフローを示しています。



```

mermaid
stateDiagram-v2
    [*] --> ログイン画面
    ログイン画面 --> ホーム画面 : ログイン成功
    ログイン画面 --> ユーザー登録画面 : 新規登録
    ユーザー登録画面 --> ログイン画面 : 登録完了

    ホーム画面 --> レストラン検索画面 : レストランを探す
    ホーム画面 --> 観光地検索画面 : 観光地を探す
    ホーム画面 --> ユーザープロフィール画面 : プロファイルを見る

    レストラン検索画面 --> レストラン詳細画面 : レストラン選択
    レストラン詳細画面 --> 予約画面 : 予約する
    予約画面 --> ホーム画面 : 予約完了

    観光地検索画面 --> 観光地詳細画面 : 観光地選択
    観光地詳細画面 --> ホーム画面 : 戻る

    ユーザープロフィール画面 --> ログイン画面 : ログアウト
    ユーザープロフィール画面 --> ホーム画面 : 戻る

    [*] --> ホーム画面 : ログイン済み
  
```



# 実装計画

DA

あなた

ユースケースと実装のための情報参考にして優先順位を設定して、マイルストーンを作成してください。

初期リリースに必要なものは、全て最優先にしてください。マイルストーンの図はMermaid Live

Editor の書式で作成してください。

実装のための情報: ###

- 初期リリースは3か月後
- 開発者の数は3名
- 最初に以下の機能を提供
  - 初期リリースのマイルストーン

###

&lt; 2 / 2 &gt;



ChatGPT

初期リリースのためのマイルストーンを

および初期リリースに必要な機能に基

でマイルストーンの概要を示します。

mermaid

gantt

```
title 初期リリースのマイルストーン
dateFormat YYYY-MM-DD
excludes weekends
```

```
section プロジェクト計画
```

```
プロジェクトの初期設定と要件定義 :active, des1, 2024-02-01, 2024-02-07
```

```
デザインとアーキテクチャの決定 :active, des2, after des1, 2024-02-14
```

```
システム統合テスト
ユーザー受け入れテスト
本番環境へのデプロイ
```

```
section リリース
初期リリース
```

このマイルストーン計画では

1. プロジェクト計画：プロジェクトの初期設定と要件定義
2. フロントエンド開発：スマートフォンアプリの基本設計
3. バックエンド開発：APIの初期設定、データベース設計
4. テストとデプロイ：システム統合テスト、ユーザー受け入れテスト、本番環境へのデプロイ
5. リリース：初期リリースのマイルストーン

この計画は、開発者3名というリソースを考慮し、初期リリースに必要な機能を最優先で進めていくことを想定しています。

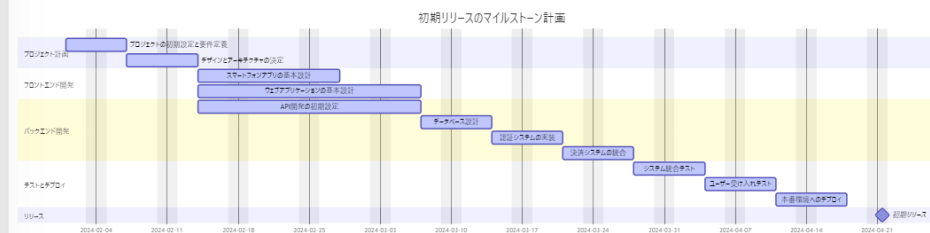
```
Mermaid </> Code Config Auto sync DOCS
1 gantt
2 title 初期リリースのマイルストーン計画
3 dateFormat YYYY-MM-DD
4 excludes weekends 2024-01-01
5
6 section プロジェクト計画
7 プロジェクトの初期設定と要件定義 :active, des1, 2024-02-01, 2024-02-07
8 デザインとアーキテクチャの決定 :active, des2, after des1, 2024-02-14
9
10 section フロントエンド開発
11 スマートフォンアプリの基本設計 :active, fe1, after des2, 2024-02-14
12 ウェブアプリケーションの基本設計 :active, fe2, after des2, 2024-03-07
13
14 section バックエンド開発
15 API開発の初期設定 :active, be1, after des2, 2024-02-14
16 データベース設計 :active, be2, after be1, 2024-02-21
17 認証システムの実装 :active, be3, after be2, 2024-03-07
18 決済システムの統合 :active, be4, after be3, 2024-03-14
19
20 section テストとデプロイ
21 システム統合テスト :active, test, after be4, 2024-03-14
22 ユーザー受け入れテスト :active, uat, after test, 2024-03-21
23 本番環境へのデプロイ :active, deploy, after uat, 2024-03-28
24
25 section リリース
26 初期リリース :milestone, ms1, after deploy
27
```

Diagram

Pan &amp; Zoom

FULL SCREEN

SAVE TO MERMAID CHART





# データモデル

あなた

これらのユースケースをシステムとして実現するために、どのようなデータモデルが考えられますか？ Mermaid Live Editor の書式で作成してください。

ChatGPT

郡山市の観光客向けWebアプリケーションのためのデータモデルを設計する際、以下のユースケースとその関連性を考えることができます。Mermaid Live Editor を使えます。

mermaid

erDiagram

```

USER ||--o{ RESERVATION : "行う"
USER {
  string userId
  string name
  string email
  string password
}

RESERVATION ||--o{ RESTAURANT : "含む"
RESERVATION {
  string reservationId
  datetime date
  int numberOfPeople
  string userId
  string restaurantId
}

```

```

string locationId
}

```

このデータモデルでは、以下のエンティティと関連性が記述されています。

- **USER** : アプリケーションのユーザー。名前、メールアドレスを含みます。
- **RESERVATION** : レストランの予約情報。予約ID、日付、人数、ユーザーID、レストランIDを含みます。
- **RESTAURANT** : レストランの情報。名前、料理の種類、住所、電話番号を含みます。
- **LOCATION** : 位置情報。住所、緯度、経度を含みます。
- **PAYMENT\_METHOD** : 支払い方法。タイプ、詳細を含みます。
- **REVIEW** : ユーザーによるレビュー。内容、日付、評価を含みます。
- **ATTRACTION** : 観光地の情報。名前、説明、所在地を含みます。

このデータモデルにより、観光客がレストラン予約、レビューを行うことが可能になります。

Mermaid </> Code ⚙️ Config Auto sync 📄 DOCS

```

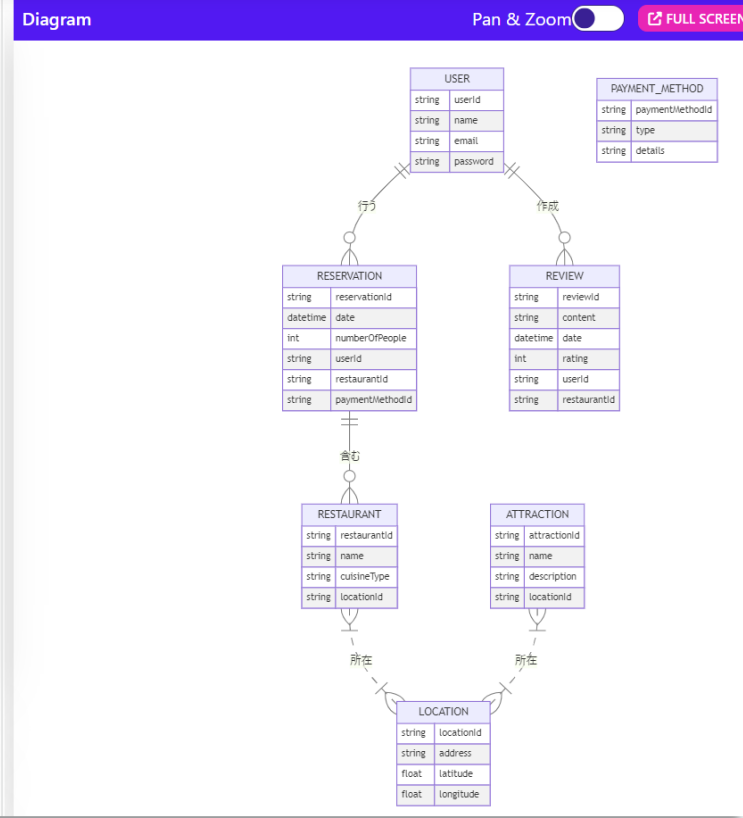
30   string address
31   float latitude
32   float longitude
33 }
34
35 PAYMENT_METHOD {
36   string paymentMethodId
37   string type
38   string details
39 }
40
41 USER ||--o{ REVIEW : "作成"
42 REVIEW {
43   string reviewId
44   string content
45   datetime date
46   int rating
47   string userId
48   string restaurantId
49 }
50
51 ATTRACTION ||..o{ LOCATION : "所在"
52 ATTRACTION {
53   string attractionId
54   string name
55   string description
56   string locationId
57 }
58

```

> Sample Diagrams

> History

> Actions



🗨️ ChatGPTにメッセージを送る...

# サンプルデータ作成

あなた

サンプルのデータを10件ずつ、作成してください。

< 2 / 2 >

ChatGPT

了解しました。郡山市の観光客向けWebアプリケーションに関連するサンプルデータを10件ずつ作成します。ここでは、ユーザー、レストラン、観光地、レビュー、予約の各カテゴリについて例示します。

## ユーザー (Users)

- User ID: 001, Name: 山田太郎, Email: taro.yamada@example.com
- User ID: 002, Name: 田中花子, Email: hanako.tanaka@example.com
- User ID: 003, Name: 鈴木一郎, Email: ichiro.suzuki@example.com
- User ID: 004, Name: 佐藤愛子, Email: aiko.sato@example.com
- User ID: 005, Name: 高橋健太, Email: kenta.takahashi@example.com
- User ID: 006, Name: 伊藤博文, Email: hirofumi.ito@example.com
- User ID: 007, Name: 渡辺明美, Email: akemi.watanabe@example.com
- User ID: 008, Name: 中村勇氣, Email: yuuki.nakamura@example.com
- User ID: 009, Name: 小林直人, Email: naoto.kobayashi@example.com
- User ID: 010, Name: 加藤理恵, Email: rie.kato@example.com

## レストラン (Restaurants)

- Restaurant ID: R001, Name: 海鮮屋台 さかな村, Location: 郡山市中央
- Restaurant ID: R002, Name: 郡山ラーメン 井戸端, Location: 安積町
- Restaurant ID: R003, Name: 山形牛ステーキハウス, Location: 福島市
- Restaurant ID: R004, Name: イタリアンカーサベラ, Location: 西ノ内
- Restaurant ID: R005, Name: 野菜たっぷりサラダバー グリーン, Location: 郡山市南
- Restaurant ID: R006, Name: 和食処 つきじ, Location: 西ノ内
- Restaurant ID: R007, Name: 寿司 天翔, Location: 安積町
- Restaurant ID: R008, Name: 中華料理 東風, Location: 郡山市中央
- Restaurant ID: R009, Name: カフェ リーフ, Location: 福島市
- Restaurant ID: R010, Name: 郡山バーガーショップ, Location: 郡山市南

## 観光地 (Attractions)

- Attraction ID: A001, Name: 磐梯山, Description: 美しい山景色
- Attraction ID: A002, Name: 郡山城跡, Description: 歴史的な城址

## レビュー (Reviews)

- Review ID: RV001, Content: 美味しい海鮮料理でした!, Rating: 5, User ID: 001, Restaurant ID: R001
- Review ID: RV002, Content: ラーメンが絶品でした, Rating: 4, User ID: 002, Restaurant ID: R002
- Review ID: RV003, Content: 山形牛はとても柔らかく、最高でした, Rating: 5, User ID: 003, Restaurant ID: R003
- Review ID: RV004, Content: イタリアンが本格的で美味しい, Rating: 4, User ID: 004, Restaurant ID: R004
- Review ID: RV005, Content: 新鮮な野菜がたくさんで健康的, Rating: 4, User ID: 005, Restaurant ID: R005
- Review ID: RV006, Content: 和食の味が絶妙です, Rating: 5, User ID: 006, Restaurant ID: R006
- Review ID: RV007, Content: 寿司が新鮮で美味しかったです, Rating: 5, User ID: 007, Restaurant ID: R007
- Review ID: RV008, Content: 中華料理が本場の味で満足, Rating: 4, User ID: 008, Restaurant ID: R008
- Review ID: RV009, Content: カフェの雰囲気が素敵でした, Rating: 4, User ID: 009, Restaurant ID: R009
- Review ID: RV010, Content: バーガーがジューシーで美味しい, Rating: 5, User ID: 010, Restaurant ID: R010

## 予約 (Reservations)

- Reservation ID: RS001, Date: 2024-05-01, Number of People: 2, User ID: 001, Restaurant ID: R001
- Reservation ID: RS002, Date: 2024-05-02, Number of People: 4, User ID: 002, Restaurant ID: R002
- Reservation ID: RS003, Date: 2024-05-03, Number of People: 2, User ID: 003, Restaurant ID: R003
- Reservation ID: RS004, Date: 2024-05-04, Number of People: 3, User ID: 004, Restaurant ID: R004
- Reservation ID: RS005, Date: 2024-05-05, Number of People: 5, User ID: 005, Restaurant ID: R005

# Azure SQL Database のテーブル作成

実装

あなた  
Azure SQL Database を使います。テーブルの作成のSQL文を作成してください。  
< 2 / 3 >

ChatGPT  
Azure SQL Database で使用するためのテーブル作成のSQL文を作成します。ここでは、ユーザー (Users)、レストラン (Restaurants)、観光地 (Attractions)、レビュー (Reviews)、予約 (Reservations) の5つのテーブルを作成する例を示します。

1. ユーザー (Users) テーブル:

```
sql Copy code
CREATE TABLE Users (
  UserId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Email NVARCHAR(100) UNIQUE
);
```

2. レストラン (Restaurants) テーブル:

```
sql Copy code
CREATE TABLE Restaurants (
  RestaurantId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Location NVARCHAR(100),
  CuisineType NVARCHAR(50)
);
```

3. 観光地 (Attractions) テーブル:

```
sql Copy code
CREATE TABLE Attractions (
  AttractionId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Description NVARCHAR(255)
);
```

4. レビュー (Reviews) テーブル:

```
sql Copy code
CREATE TABLE Reviews (
  ReviewId INT PRIMARY KEY IDENTITY(1,1),
  Content NVARCHAR(255),
  Date DATETIME,
  Rating INT,
  UserId INT FOREIGN KEY REFERENCES Users(UserId),
  RestaurantId INT FOREIGN KEY REFERENCES Restaurants(RestaurantId)
);
```

5. 予約 (Reservations) テーブル:

```
sql Copy code
CREATE TABLE Reservations (
  ReservationId INT PRIMARY KEY IDENTITY(1,1),
  Date DATETIME,
  NumberOfPeople INT,
  UserId INT FOREIGN KEY REFERENCES Users(UserId),
  RestaurantId INT FOREIGN KEY REFERENCES Restaurants(RestaurantId)
);
```

これらのSQL文は、各テーブルの基本的なフィールドと主要な関連を定義しています。実際のアプリケーションの要件に合わせてフィールドを追加したり、データ型を変更することができます。また、Azure SQL Database の特定の要件 (例えば、インデックスの最適化やパーティショニング戦略) に基づいて、これらのSQL文をさらに調整することが必要になる場合があります。

📄 📌 🗑

# Azure SQL Database にデータ登録

実装

あなた  
サンプルのデータを Azure SQL Database に登録するSQL文を作成してください

ChatGPT  
Azure SQL Databaseにサンプルデータを登録するためのSQL文を作成します。前述のテーブル構造を用いて、各テーブルにデータを挿入するSQL文を以下に示します。

## 1. ユーザー (Users) データの挿入:

```
sql Copy code

INSERT INTO Users (Name, Email) VALUES
('山田太郎', 'taro.yamada@example.com'),
('田中花子', 'hanako.tanaka@example.com'),
('鈴木一郎', 'ichiro.suzuki@example.com'),
('佐藤愛子', 'aiko.sato@example.com'),
('高橋健太', 'kenta.takahashi@example.com'),
('伊藤博文', 'hirofumi.ito@example.com'),
('渡辺明美', 'akemi.watanabe@example.com'),
('中村勇気', 'yuuki.nakamura@example.com'),
('小林直人', 'naoto.kobayashi@example.com'),
('加藤理恵', 'rie.kato@example.com');
```

## 2. レストラン (Restaurants) データの挿入:

```
sql Copy code

INSERT INTO Restaurants (Name, Location, CuisineType) VALUES
('海鮮屋台 さかな村', '郡山市中央', '海鮮'),
('郡山ラーメン 井戸端', '安積町', 'ラーメン'),
('山形牛ステーキハウス', '福島市', 'ステーキ'),
('イタリアン カーサベラ', '西ノ内', 'イタリアン'),
('野菜たっぷりサラダバー グリーン', '郡山市南', 'サラダバー'),
('和食処 つきじ', '西ノ内', '和食'),
('寿司 天翔', '安積町', '寿司'),
('中華料理 東風', '郡山市中央', '中華料理'),
('カフェ リーフ', '福島市', 'カフェ'),
('郡山バーガーショップ', '郡山市南', 'ハンバーガー');
```

## 3. 観光地 (Attractions) データの挿入:

```
sql Copy code

INSERT INTO Attractions (Name, Description) VALUES
('磐梯山', '美しい山景色'),
('郡山城跡', '歴史的な城址'),
('桜木公園', '花見の名所'),
('ふくしまスカイパーク', 'アクティブなアウトドア'),
('五色沼', '自然の美しさ'),
('大内宿', '歴史的な街並み'),
('アドベンチャーワールド', '家族向けテーマパーク'),
('白河ラーメン村', 'グルメスポット'),
('福島レースコース', 'モータースポーツの楽しみ'),
('あづま総合運動公園', 'スポーツとレクリエーション');
```

## 4. レビュー (Reviews) データの挿入:

```
sql Copy code

-- 注意: ユーザーIDとレストランIDは実際のIDに置き換えてください
INSERT INTO Reviews (Content, Date, Rating, UserId, RestaurantId) VALUES
('美味しい海鮮料理でした!', '2024-05-01', 5, 1, 1),
('ラーメンが絶品でした!', '2024-05-02', 4, 2, 2),
('山形牛はとても柔らかく、最高でした!', '2024-05-03', 5, 3, 3),
('イタリアンが本格的で美味しい!', '2024-05-04', 4, 4, 4),
('新鮮な野菜がたくさんで健康的', '2024-05-05', 4, 5, 5),
('和食の味が絶妙です!', '2024-05-06', 5, 6, 6),
('寿司が新鮮で美味しかったです!', '2024-05-07', 5, 7, 7),
('中華料理が本場の味で満足!', '2024-05-08', 4, 8, 8),
('カフェの雰囲気素敵でした!', '2024-05-09', 4, 9, 9),
('バーガーがジューシーで美味しい!', '2024-05-10', 5, 10, 10);
```

## 5. 予約 (Reservations) データの挿入:

```
sql Copy code

-- 注意: ユーザーIDとレストランIDは実際のIDに置き換えてください
INSERT INTO Reservations (Date, NumberOfPeople, UserId, RestaurantId) VALUES
('2024-05-01', 2, 1, 1),
('2024-05-02', 4, 2, 2),
('2024-05-03', 2, 3, 3);
```

## 5. 予約 (Reservations) データの挿入:

```
sql Copy code

-- 注意: ユーザーIDとレストランIDは実際のIDに置き換えてください
INSERT INTO Reservations (Date, NumberOfPeople, UserId, RestaurantId) VALUES
('2024-05-01', 2, 1, 1),
('2024-05-02', 4, 2, 2),
('2024-05-03', 2, 3, 3),
('2024-05-04', 3, 4, 4),
('2024-05-05', 5, 5, 5),
('2024-05-06', 2, 6, 6),
('2024-05-07', 4, 7, 7),
('2024-05-08', 3, 8, 8),
('2024-05-09', 2, 9, 9),
('2024-05-10', 3, 10, 10);
```

これらのSQL文は、サンプルデータを各テーブルに挿入するためのものです。実際の環境では、ユーザーIDやレストランIDなどの外部キーの値を、実際に存在するレコードのIDに適切に置き換える必要があります。また、データ挿入後にデータ整合性を確認することが重要です。



# Azure Cosmos DB の作成

あなた  
CLI で Azure Cosmos DB を作成するコードを作成して

ChatGPT  
Azure CLIを使用してAzure Cosmos DBアカウントを作成するためのコード例を以下に示します。このコマンドは、新しいAzure Cosmos DBアカウントを作成し、SQL APIを使用する設定であることを前提としています。

まず、Azureにログインしておく必要があります。その後、以下のステップに従ってCosmos DBアカウントを作成できます。

## 1. Azureにログインする

```
bash Copy code  
az login
```

## 1. Cosmos DBアカウントを作成する

以下のコマンドは、指定されたリソースグループ内に新しいAzure Cosmos DBアカウントを作成します。`<リソースグループ名>`、`<CosmosDBアカウント名>`、`<地域>`を適切な値に置き換えてください。

```
bash Copy code  
az cosmosdb create --name <CosmosDBアカウント名> --resource-group <リソースグループ名>
```

```
bash Copy code  
az cosmosdb create --name <CosmosDBアカウント名> --resource-group <リソースグループ名>
```

このコマンドのオプションは以下の通りです：

- `--name`：Cosmos DBアカウントの名前。
- `--resource-group`：使用するリソースグループの名前。
- `--kind`：アカウントの種類。Cosmos DBでは`GlobalDocumentDB`が使用されます（これはSQL APIの旧称です）。
- `--locations`：アカウントを作成する地域と、フェイルオーバープライオリティとゾーン冗長性の設定。
- `--default-consistency-level`：アカウントのデフォルトの一貫性レベル。ここでは`Session`を使用しています。
- `--capabilities`：特定の機能を有効にするためのオプション。`EnableServerless`はサーバーレスモードを示します。

**注意:** Azure CLIを使用してリソースを作成する前に、適切なサブスクリプションを選択していることを確認してください。複数のサブスクリプションがある場合は、`az account set --subscription "<サブスクリプションIDまたは名前>"`コマンドを使用して設定できます。

これらのコマンドを実行することで、Azure Cosmos DBアカウントが作成され、SQL APIを使用する準備が整います。アカウント作成後は、データベースやコンテナを作成してデータを管理することができます。



[残念!] Resource Group を Cosmos DB アカウントの作成前に、作成する必要あり 🤖

# 登録用の変数作成

DA あなた

dahatakeをプリフィックスにして、それぞれの変数の値を作成してください。

< 2 / 2 >

ChatGPT

以下の変数の値を作成しました：

- Cosmos DBアカウント名: `dahatake-bc83826f`
- リソースグループ名: `dahatake-rg-20240205071739`
- データベース名: `dahatake-db-20240205071739`
- コンテナ名: `dahatake-container-20240205071739`
- パーティションキー: `/dahatakePartitionKey`

これらの値は、プリフィックス「dahatake」を使用して一意に生成されました。これらを使用してAzure Cosmos DBのリソースを作成する際の識別子としてご利用いただけます。[->]

DA あなた

これらの変数をWindowsのコマンドプロンプトで作成するコードを作成してください。

ChatGPT

Windowsのコマンドプロンプトで環境変数を設定するためのコマンドは`set`コマンドを使用します。以下は、先ほど生成した変数をWindowsの環境変数として設定するためのコマンドです。

```
cmd Copy code
set COSMOS_DB_ACCOUNT_NAME=dahatake-bc83826f
set RESOURCE_GROUP_NAME=dahatake-rg-20240205071739
set DATABASE_NAME=dahatake-db-20240205071739
set CONTAINER_NAME=dahatake-container-20240205071739
set PARTITION_KEY=/dahatakePartitionKey
```

これらのコマンドをコマンドプロンプトに貼り付けて実行することで、指定した変数名に対応する値が環境変数として設定されます。これにより、コマンドプロンプトのセッション内でこれらの環境変数を使用することができます。

例えば、これらの環境変数を使用してAzure CLIコマンドを実行する際には、`%変数名%`の形式で参照できます。以下は、環境変数を参照する例です。

```
cmd Copy code
az cosmosdb create --name %COSMOS_DB_ACCOUNT_NAME% --resource-group %RE
```

この方法を使用することで、コマンド実行時に手動で値を入力する手間を省き、効率的に作業を進めることができます。

# CLI スクリプトを変数を使ったものに

実装

あなた  
ここまでのチャットで作成した全ての Azure Cosmos DB の CLI スクリプトの変数名を変数として使って、先に作成したスクリプトを再度作成してください。  
< 2 / 2 >

ChatGPT  
先に示した Azure CLI スクリプトを、作成した環境変数を使用して再度作成します。このスクリプトでは、Azure Cosmos DB アカウント、データベース、およびコンテナの作成を行います。

## Azure Cosmos DB アカウントの作成

```
bash Copy code
az cosmosdb create \
  --name %COSMOS_DB_ACCOUNT_NAME% \
  --resource-group %RESOURCE_GROUP_NAME% \
  --kind GlobalDocumentDB \
  --locations regionName=<リージョン> failoverPriority=0 isZoneRedundant=
  --default-consistency-level Session \
  --enable-multiple-write-locations false
```

## データベースの作成

```
bash Copy code
az cosmosdb sql database create \
  --account-name %COSMOS_DB_ACCOUNT_NAME% \
  --resource-group %RESOURCE_GROUP_NAME% \
  --name %DATABASE_NAME%
```

## コンテナの作成

```
bash Copy code
az cosmosdb sql container create \
  --account-name %COSMOS_DB_ACCOUNT_NAME% \
  --resource-group %RESOURCE_GROUP_NAME% \
  --database-name %DATABASE_NAME% \
  --name %CONTAINER_NAME% \
  --partition-key-path %PARTITION_KEY%
  --throughput 400
```

## コンテナの作成

```
bash Copy code
az cosmosdb sql container create \
  --account-name %COSMOS_DB_ACCOUNT_NAME% \
  --resource-group %RESOURCE_GROUP_NAME% \
  --database-name %DATABASE_NAME% \
  --name %CONTAINER_NAME% \
  --partition-key-path %PARTITION_KEY%
  --throughput 400
```

これらのスクリプトを使用する前に、コマンドプロンプトで先に示した環境変数設定コマンドを実行して、必要な変数を設定しておく必要があります。リージョンの部分（`<リージョン>`）は、実際にデプロイしたい Azure のリージョンに置き換えてください。

これらのスクリプトは、Windows のコマンドプロンプトで直接実行することができ、Azure Cosmos DB のアカウント、データベース、コンテナを作成するプロセスを自動化します。



# Azure Cosmos DB にデータ登録

あなた

CLIを使ってAzure Cosmos DB SQL APIへデータ登録をするコードを作成してください。

ChatGPT

Azure Cosmos DBにデータを登録するためにAzure CLIとCosmos DB SQL APIを使用するコードサンプルを以下に示します。この例では、Azure CLIを使用してCosmos DBのSQL APIにアクセスし、JSON形式のデータをコンテナに挿入する方法を説明します。

まず、Azure CLIを使用してCosmos DBに接続し、データを挿入する前に、Azureサブスクリプションにログインしておく必要があります。以下の手順では、特定のユーザーデータをコンテナに挿入する方法を示します。

```
bash Copy code

# Azureにログイン
az login

# Cosmos DBアカウントの接続文字列を取得
# このコマンドはCosmos DBアカウント名とリソースグループ名を指定して実行します
az cosmosdb keys list --type connection-strings --name <CosmosDBアカウント名>

# データを登録するためのJSONファイルを作成 (例: user.json)
echo '{"userId": "011", "name": "新規ユーザー", "email": "new.user@example.com"}' > user.json

# Cosmos DB SQL APIを使用してデータを挿入
# このコマンドには、上記で取得した接続文字列、データベース名、コンテナ名、そしてJSONファイルのパスを指定します
az cosmosdb sql container create --account-name <CosmosDBアカウント名> --resource-group <リソースグループ名> --database-name <データベース名> --container-name <コンテナ名> --connection-string <接続文字列> --json user.json

# データを挿入する
az cosmosdb sql document create --account-name <CosmosDBアカウント名> --resource-group <リソースグループ名> --database-name <データベース名> --container-name <コンテナ名> --connection-string <接続文字列> --json user.json
```

```
bash Copy code

# Azureにログイン
az login

# Cosmos DBアカウントの接続文字列を取得
# このコマンドはCosmos DBアカウント名とリソースグループ名を指定して実行します
az cosmosdb keys list --type connection-strings --name <CosmosDBアカウント名>

# データを登録するためのJSONファイルを作成 (例: user.json)
echo '{"userId": "011", "name": "新規ユーザー", "email": "new.user@example.com"}' > user.json

# Cosmos DB SQL APIを使用してデータを挿入
# このコマンドには、上記で取得した接続文字列、データベース名、コンテナ名、そしてJSONファイルのパスを指定します
az cosmosdb sql container create --account-name <CosmosDBアカウント名> --resource-group <リソースグループ名> --database-name <データベース名> --container-name <コンテナ名> --connection-string <接続文字列> --json user.json

# データを挿入する
az cosmosdb sql document create --account-name <CosmosDBアカウント名> --resource-group <リソースグループ名> --database-name <データベース名> --container-name <コンテナ名> --connection-string <接続文字列> --json user.json
```

この例では、まず`az login`コマンドでAzureにログインし、`az cosmosdb keys list`コマンドでCosmos DBアカウントの接続文字列を取得します。その後、挿入するデータを含むJSONファイルを作成し、`az cosmosdb sql document create`コマンドを使用してCosmos DBの指定コンテナにデータを挿入します。

注意点として、このコマンド例では具体的なCosmos DBアカウント名、リソースグループ名、データベース名、コンテナ名を指定する必要があります。これらは、あなたのAzure環境に応じて適切に置き換えてください。また、パーティションキーは、あなたのデータモデルに応じて適切に設定してください。

この方法を使用すると、CLIからCosmos DB SQL APIを介してデータを簡単に登録することができます。

[残念!] DB、Container作成までは出来ます。  
CLIから、データ登録は出来ません🥲



# Azure Cosmos DB SQL API

あなた

Azure Cosmos DBを使います。テーブルの作成のSQL文を作成してください。

< 3 / 3 >

ChatGPT

Azure Cosmos DB では、SQLを用いたテーブルの作成は、従来のリレーショナルデータベースのSQL文とは異なります。Cosmos DBはNoSQLデータベースであり、ドキュメント指向のストレージを使用しています。従って、「テーブル」の代わりに「コンテナ」を作成することになります。

ただし、具体的なコンテナの作成はAzure Cosmos DBのUIを介して行うか、Cosmos DB SDKを使用してプログラムのに行います。SQLクエリは主にデータの読み取りや更新に使用されます。

コンテナを作成する際には、以下の点を考慮する必要があります：

- パーティションキーの選択：データアクセスのパフォーマンスとスケーラビリティを最適化するためには、適切なパーティションキーを選択することが重要です。
- スループット設定：必要に応じて手動または自動スケールのスループットを設定します。

例えば、ユーザー情報を格納するコンテナを作成する場合、パーティションキーを「userId」などのユニークな属性に設定することが一般的です。

以下に、Cosmos DBでのコンテナ作成のための一般的なステップを示します：

- Azureポータルにログインし、Cosmos DBアカウントに移動します。
- 「データエクスプローラー」を開き、「新しいコンテナ」オプションを選択します。
- 「データベースID」を指定し、「新しいデータベースの作成」を選択するか、既存のデータベースを使用します。
- 「コンテナID」と「パーティションキー」を設定します。
- 必要に応じてスループットを設定し、「OK」をクリックしてコンテナを作成します。

Cosmos DBのSDKを使用してコンテナをプログラムのに作成するには、適切なAPI呼び出しを行いますが、これは開発言語によって異なります。

データモデルやアプリケーションの要件に応じて、適切なコンテナ構造と設定を決定してください。

あなた

Azure Cosmos DBにサンプルデータを登録するサンプルデータと、スクリプトのコードを作成してください。

ChatGPT

Azure Cosmos DBにサンプルデータを登録するためには、Cosmos DBのSDKを使用したスクリプトを作成します。ここでは、Pythonを使用した基本的なサンプルを示します。このスクリプトは、Cosmos DBのPython SDKを使用して、指定されたデータをCosmos DBのコンテナに挿入します。

まず、サンプルデータを定義します。例として、ユーザーデータのサンプルを以下に示します。

```
python Copy code

users = [
    {'id': '001', 'name': '山田太郎', 'email': 'taro.yamada@example.com'},
    {'id': '002', 'name': '田中花子', 'email': 'hanako.tanaka@example.com'},
    # ... 他のユーザーデータ ...
]
```

次に、Cosmos DBにデータを挿入するためのPythonスクリプトのサンプルを作成します。このスクリプトは、Azure Cosmos DBのPython SDKを使用し、上記のユーザーデータをCosmos DBのコンテナに挿入します。

```
python Copy code

from azure.cosmos import CosmosClient, PartitionKey

# Cosmos DBの接続情報
url = "YOUR_COSMOS_DB_URL"
key = "YOUR_COSMOS_DB_KEY"
database_name = "YOUR_DATABASE_NAME"
container_name = "YOUR_CONTAINER_NAME"

# Cosmos DBクライアントの初期化
client = CosmosClient(url, credential=key)
database = client.get_database_client(database=database_name)
container = database.get_container_client(container=container_name)

# データの挿入
```

このスクリプトを実行する前に、次の点を確認してください：

- Cosmos DBのURL(`YOUR\_COSMOS\_DB\_URL`)、キー(`YOUR\_COSMOS\_DB\_KEY`)、データベース名(`YOUR\_DATABASE\_NAME`)、コンテナ名(`YOUR\_CONTAINER\_NAME`)を、実際の値に置き換えてください。
- Azure Cosmos DBのPython SDKがインストールされている必要があります。インストールされていない場合は、`pip install azure-cosmos`を実行してください。

このスクリプトは、指定されたコンテナに対して、リスト内の各ユーザーデータを挿入します。同様の方法で他のデータタイプ（レストラン、観光地、レビューなど）も挿入することができます。

📄 🗑️ 🔄 🔄

論理モデル (データモデル) さえあれば、  
もはや実装用のコードはどれでもいい  
勿論、最適化にはエンジニアリングが必須ですけど 🤖

# Vue.js 作成 (1)

あなた  
画面をVue.js とHTML で作成してください。

画面: ###  
- ホーム画面  
- ログイン画面  
- レストラン検索と予約システム  
###

ChatGPT

Vue.jsとHTMLを使用して、ホーム画面、ログイン画面、およびレストラン検索と予約システムの基本的な構造を作成します。以下のコードスニペットは、これらの画面の基本的な枠組みを提供します。

## 1. ホーム画面 (Home.vue):

```
vue Copy code
<template>
  <div class="home">
    <h1>郡山市観光案内</h1>
    <nav>
      <ul>
        <li><router-link to="/login">ログイン</router-link></li>
        <li><router-link to="/search">レストラン検索</router-link></li>
      </ul>
    </nav>
  </div>
</template>

<script>
export default {
  name: 'Home'
}
</script>

<style scoped>
.home {
  text-align: center;
  ↓
```

## 2. ログイン画面 (Login.vue):

```
vue Copy code
<template>
  <div class="login">
    <h2>ログイン</h2>
    <form @submit.prevent="login">
      <div>
        <label for="email">メールアドレス:</label>
        <input type="email" id="email" v-model="email">
      </div>
      <div>
        <label for="password">パスワード:</label>
        <input type="password" id="password" v-model="password">
      </div>
      <button type="submit">ログイン</button>
    </form>
  </div>
</template>

<script>
export default {
  name: 'Login',
  data() {
    return {
      email: '',
      password: ''
    }
  },
  methods: {
    login() {
      // ログイン処理
    }
  }
}
</script>

<style scoped>
.login {
  max-width: 300px;
  margin: 0 auto;
  text-align: left;
  ↓
```

```
<h3>検索結果</h3>
<ul>
  <li v-for="restaurant in restaurants" :key="restaurant.id">
    {{ restaurant.name }} - <button @click="reserve(restaurant.id)">予約</button>
  </li>
</ul>
</div>
</template>

<script>
export default {
  name: 'Search',
  data() {
    return {
      searchQuery: '',
      restaurants: []
    }
  },
  methods: {
    search() {
      // 検索処理
    },
    reserve(id) {
      // 予約処理
    }
  }
}
</script>

<style scoped>
.search {
  text-align: center;
}
</style>
```

これらのコードスニペットはVue.jsの基本的な構造を提供し、実際のアプリケーションではAPI呼び出し、データバインディング、およびユーザーインタラクションを処理するための追加のロジックが必要になります。また、Vue Routerを使用して画面間のナビゲーションを実装することもできます。

# Vue.js 作成 (2)

あなた  
作成したVue.jsのテンプレートをホスティングするHTMLを作成してください。

ChatGPT  
Vue.jsテンプレートをホスティングするための基本的なHTMLファイルを作成します。このHTMLファイルは、Vue.jsアプリケーションを実行するためのコンテナとして機能し、必要なスクリプトを読み込みます。

以下は、Vue.jsアプリケーションをホストするためのシンプルなHTMLファイルの例です。この例では、Vue.jsとVue RouterをCDN経由で読み込み、アプリケーションを起動するためのスクリプトを含めています。

```
html Copy code
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>郡山市観光案内アプリ</title>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/vue-router@3.5.3/dist/vue-router.js"></script>
</head>
<body>
  <div id="app"></div>

  <script>
    // Vue Routerの設定
    const router = new VueRouter({
      routes: [
        { path: '/', component: Home },
        { path: '/login', component: Login },
        { path: '/search', component: Search }
      ]
    });

    // Vueインスタンスの生成
    new Vue({
      router,
      el: '#app'
    });
  </script>
</body>
</html>
```

```
<body>
  <div id="app"></div>

  <script>
    // Vue Routerの設定
    const router = new VueRouter({
      routes: [
        { path: '/', component: Home },
        { path: '/login', component: Login },
        { path: '/search', component: Search }
      ]
    });

    // Vueインスタンスの生成
    new Vue({
      router,
      el: '#app'
    });
  </script>
</body>
</html>
```

このHTMLファイルには、Vueインスタンスをマウントするための`<div id="app"></div>`が含まれており、Vue Routerを使用して異なるパスで異なるコンポーネントが表示されるように設定されています。

## 注意点:

- この例では、Vue.jsとVue RouterのCDNリンクを使用しています。実際のプロダクション環境では、これらのライブラリをプロジェクトに直接インストールすることをお勧めします。
- `Home`、`Login`、`Search`などのVueコンポーネントは、このHTMLファイル内では定義されていません。これらはVueファイル内で定義され、適切にインポートされる必要があります。
- 本番環境での利用には、ビルドプロセス（例えばWebpackやVue CLI）を使用してアプリケーションを最適化することが推奨されます。

# Azure Functions で API 実装

DA あなた

Vue.jsから呼び出すサーバー側のAPIをAzure FunctionsにホストするNodeJSで作成します。以下の{レストラン検索ロジック}のAPI実装の通りコードを作成してください。

{レストラン検索ロジック}のAPI実装: ###

- DBへの保存ロジックはここでは実装しません。APIを呼び出したらサンプルデータを返します
- サンプルデータは、先に作成したレストランデータを使います
- 項目毎にAPIにもパラメーターを設定して、DB検索時にフィルタリングが出来るようにします

###



ChatGPT

Azure Functionsを使用してNodeJSでレストラン検索ロジックのAPIを実装します。このAPIはサンプルデータを返し、フロントエンドからのパラメータに基づいて結果をフィルタリングすることができるように設計します。

まず、サンプルデータを作成します。これは先に提供したレストランデータの例です。

```
javascript Copy code
const restaurants = [
  { id: 1, name: '海鮮屋台 さかな村', location: '郡山市中央', cuisine: '海鮮' },
  { id: 2, name: '郡山ラーメン 井戸端', location: '安積町', cuisine: 'ラーメン' },
  // ...他のレストランデータ
];
```

次に、Azure Functionsのハンドラー関数を作成します。この関数は、リクエストからクエリパラメータを受け取り、それに基づいてレストランデータをフィルタリングします。

```
javascript Copy code
module.exports = async function (context, req) {
  context.log('レストラン検索APIが呼び出されました。');

  const query = req.query || {};
  let filteredRestaurants = restaurants;

  // 名前でフィルタリング
  if (query.name) {
    filteredRestaurants = filteredRestaurants.filter(r => r.name.in
```

```
javascript Copy code
module.exports = async function (context, req) {
  context.log('レストラン検索APIが呼び出されました。');

  const query = req.query || {};
  let filteredRestaurants = restaurants;

  // 名前でフィルタリング
  if (query.name) {
    filteredRestaurants = filteredRestaurants.filter(r => r.name.in

  // 場所でフィルタリング
  if (query.location) {
    filteredRestaurants = filteredRestaurants.filter(r => r.location

  // 料理の種類でフィルタリング
  if (query.cuisine) {
    filteredRestaurants = filteredRestaurants.filter(r => r.cuisine

  context.res = {
    // ステータス 200 OK
    status: 200,
    body: filteredRestaurants
  };
};
```

このコードは、Azure Functionsの標準の関数テンプレートに基づいています。クエリパラメータ（`name`、`location`、`cuisine`）を使用して、フィルタリングされたレストランデータを返します。

Azure Functionsをローカルでテストするか、Azureにデプロイして動作を確認する前に、適切な環境設定と依存関係がセットアップされていることを確認してください。また、これはデモンストレーション用の基本的な実装であり、実際のプロダクション環境では、セキュリティ、エラーハンドリング、パフォーマンス最適化などの追加の考慮が必要になる場合があります。



# Vue.js と Azure Functions API 融合

実装

あなた

先ほど作成した(レストラン検索と予約システム)画面を以下の条件で更新してください。

条件: ###

- レストラン検索ロジックのAPIを呼び出す
  - Azure Functions のURLは後で書き換えるため、ここではダミーを用いる
- ###

ChatGPT

レストラン検索と予約システムの画面をVue.jsで更新し、先ほど作成したレストラン検索ロジックのAPIを呼び出すようにします。このコードでは、APIへのリクエストを行うために`axios`を使用します。まず、`axios`をプロジェクトにインストールする必要があります。

インストールコマンド:

```
bash Copy code  
npm install axios
```

次に、`Search.vue`コンポーネントを以下のように更新します。この例では、ダミーのAzure Functions URLを用いてAPIを呼び出し、検索結果を表示します。

```
vue Copy code  
  
<template>  
  <div class="search">  
    <h2>レストラン検索</h2>  
    <input type="text" v-model="searchQuery" placeholder="レストラン名で検索">  
    <button @click="search">検索</button>  
  
    <div v-if="restaurants.length">  
      <h3>検索結果</h3>  
      <ul>  
        <li v-for="restaurant in restaurants" :key="restaurant.id">  
          {{ restaurant.name }} - <button @click="reserve(restaurant.id)">予約</li>  
      </ul>  
    </div>  
  </template>
```

```
<script>  
import axios from 'axios';  
  
export default {  
  name: 'Search',  
  data() {  
    return {  
      searchQuery: '',  
      restaurants: []  
    }  
  },  
  methods: {  
    search() {  
      const apiUrl = 'https://your-azure-function-app-url.com/api/search';  
      axios.get(apiUrl, { params: { name: this.searchQuery } })  
        .then(response => {  
          this.restaurants = response.data;  
        })  
        .catch(error => {  
          console.error('エラーが発生しました:', error);  
        });  
    },  
    reserve(id) {  
      // 予約処理  
    }  
  }  
};  
  
<style scoped>  
.search {  
  text-align: center;  
}</style>
```

このコードでは、`axios.get`メソッドを使用してAPIを呼び出し、レスポンスデータを`restaurants`配列に格納しています。現在のAPI URLはダミーですので、実際にAzure FunctionsのURLを取得したら適切に書き換えてください。

また、このコードは基本的なエラーハンドリングを行っていますが、実際のプロジェクトではより詳細なエラー処理やユーザーフィードバックを提供することを検討してください。

# Azure Functions と Azure Cosmos DB 融合

実装

**あなた**  
Azure Functionsのコードに、DBへの検索をするコードを追加します。以下の(レストラン検索ロジック)のAPI実装の通りコードを作成してください。

{レストラン検索ロジック}のAPI実装: ###  
- Azure Cosmos DB からデータを検索します。  
- Azure Cosmos DB には、先に作成した{Azure Cosmos DB SQL API}で使用するためのサンプルデータは、格納済みであるとして  
- Azure Cosmos DB への接続情報は、後で設定できるようにします。  
- 項目毎にAPIにもパラメータを設定して、DB検索時にフィルタリングが出来るようにします  
###  
< 1 / 2 >

**ChatGPT**  
Azure FunctionsのコードにAzure Cosmos DBからデータを検索するロジックを追加します。このコードは、指定されたパラメータに基づいてCosmos DBからレストランデータをフィルタリングして取得します。接続情報は環境変数やアプリケーション設定を通じて後で設定できるようにします。

まず、Azure Cosmos DBとの接続に必要な`@azure/cosmos`パッケージをインストールします。

インストールコマンド:

```
bash
npm install @azure/cosmos
```

次に、Azure Functionのコードを以下のように更新します:

```
javascript
const { CosmosClient } = require('@azure/cosmos');

module.exports = async function (context, req) {
  context.log('レストラン検索APIが呼び出されました。');

  const endpoint = process.env.COSMOS_DB_ENDPOINT; // 環境変数からCosmos
  const key = process.env.COSMOS_DB_KEY; // 環境変数からCosmos DBのキーを取
  const databaseId = "YourDatabase"; // あなたのデータベースID
  const containerId = "YourContainer"; // あなたのコンテナID

  const client = new CosmosClient({ endpoint, key });
  const database = client.database(databaseId);
```

```
javascript
const { CosmosClient } = require('@azure/cosmos');

module.exports = async function (context, req) {
  context.log('レストラン検索APIが呼び出されました。');

  const endpoint = process.env.COSMOS_DB_ENDPOINT; // 環境変数からCosmos
  const key = process.env.COSMOS_DB_KEY; // 環境変数からCosmos DBのキーを取
  const databaseId = "YourDatabase"; // あなたのデータベースID
  const containerId = "YourContainer"; // あなたのコンテナID

  const client = new CosmosClient({ endpoint, key });
  const database = client.database(databaseId);
  const container = database.container(containerId);

  try {
    const query = req.query || {};
    let querySpec = {
      query: "SELECT * FROM c WHERE c.id = @id",
      parameters: []
    };

    // 名前でフィルタリング
    if (query.name) {
      querySpec.query += " AND c.name = @name";
      querySpec.parameters.push({ name: "@name", value: query.name });
    }

    // 場所でフィルタリング
    if (query.location) {
      querySpec.query += " AND c.location = @location";
      querySpec.parameters.push({ name: "@location", value: query.location });
    }

    // 料理の種類でフィルタリング
    if (query.cuisine) {
      querySpec.query += " AND c.cuisine = @cuisine";
      querySpec.parameters.push({ name: "@cuisine", value: query.cuisine });
    }
  }
}
```

```

// 料理の種類でフィルタリング
if (query.cuisine) {
  querySpec.query += " AND c.cuisine = @cuisine";
  querySpec.parameters.push({ name: "@cuisine", value: query.cuisine });
}

const { resources: items } = await container.items
  .query(querySpec)
  .fetchAll();

context.res = {
  status: 200,
  body: items
};
} catch (error) {
  context.res = {
    status: 500,
    body: "エラーが発生しました: " + error.message
  };
};
};
```

このコードでは、Azure Cosmos DBに対してSQLクエリを実行し、条件に一致するレストランのリストを取得します。クエリパラメータは、リクエストから取得し、それに基づいてデータベースクエリを構築します。

注意:

- `COSMOS\_DB\_ENDPOINT`と`COSMOS\_DB\_KEY`は、Azure Functionsのアプリケーション設定またはローカル設定ファイルから読み込む必要があります。
- データベースID(`YourDatabase`)とコンテナID(`YourContainer`)は、実際に使用するCosmos DBのIDに置き換えてください。
- 実際のプロダクション環境では、エラーハンドリングやセキュリティ対策をさらに強化する必要があります。

# デプロイも 😊 (1)

## ChatGPT

Azure Functions と Azure Cosmos DB を作成するための Bicep コードを作成するには、以下のステップを行います：

1. Azure Functions に必要なリソースを定義する。
2. Azure Cosmos DB アカウントとデータベースを定義する。

Bicep は、Azure リソースを宣言的に定義するための言語です。以下のサンプルコードは、Azure Functions と Azure Cosmos DB を作成する基本的な構造を示しています。

### Azure Functions の作成

まず、Azure Functions のための Bicep コードを示します。ここでは、ストレージアカウントとアプリケーションプランも含まれます。

```
bicep
Copy code

resource storageAccount 'Microsoft.Storage/storageAccounts@2021-04-01'
name: '<storage_account_name>'
location: resourceGroup().location
kind: 'StorageV2'
sku: {
  name: 'Standard_LRS'
}

resource appServicePlan 'Microsoft.Web/sites@2020-12-01' = {
  name: '<app_service_plan_name>'
  location: resourceGroup().location
  sku: {
    name: 'Y1' // このSKUは動的プランを表します
  }
}

resource functionApp 'Microsoft.Web/sites@2020-12-01' = {
  name: '<function_app_name>'
  location: resourceGroup().location
  kind: 'functionapp'
  properties: {
    serverFarmId: appServicePlan.id
    siteConfig: {
      appSettings: [
```

## Azure Cosmos DB の作成

次に、Azure Cosmos DB アカウントとデータベースのための Bicep コードを示します。

```
bicep
Copy code

resource cosmosDbAccount 'Microsoft.DocumentDB/databaseAccounts@2021-04-01'
name: '<cosmos_db_account_name>'
location: resourceGroup().location
kind: 'GlobalDocumentDB'
properties: {
  databaseAccountOfferType: 'Standard'
  locations: [
    {
      locationName: resourceGroup().location
      failoverPriority: 0
      isZoneRedundant: false
    }
  ]
  capabilities: [
    {
      name: 'EnableServerless'
    }
  ]
}

resource cosmosDbDatabase 'Microsoft.DocumentDB/databaseAccounts/sqlDatabases@2021-04-01'
parent: cosmosDbAccount
name: '<database_name>'
properties: {
  resource: {
    id: '<database_name>'
  }
  options: {}
}
```

上記のコードでは、`<storage\_account\_name>`、`<app\_service\_plan\_name>`、`<function\_app\_name>`、`<cosmos\_db\_account\_name>`、`<database\_name>` のようなプレースホルダーを使用しています。これらは実際の環境に合わせて適切な名前に置き換えてください。

上記のコードでは、`<storage\_account\_name>`、`<app\_service\_plan\_name>`、`<function\_app\_name>`、`<cosmos\_db\_account\_name>`、`<database\_name>` のようなプレースホルダーを使用しています。これらは実際の環境に合わせて適切な名前に置き換えてください。

これらのコードは、基本的なリソースの作成を行います。実際のプロジェクトでは、追加の設定やリソースが必要になる場合があります。また、Bicep コードをデプロイする前に、Azure CLI または Azure PowerShell を使用して、Bicep ファイルを ARM テンプレートに変換してからデプロイする必要があります。

# デプロイも 😊 (2)

あなた  
この Bicep コードをデプロイするための Azure CLI のスクリプトを作成してください

ChatGPT  
Azure CLI を使用して Bicep ファイルをデプロイするためには、まず Bicep ファイルを準備して、それを Azure Resource Manager (ARM) テンプレートに変換してから、Azure にデプロイします。以下は、このプロセスを実行するためのスクリプトの例です。

## 1. Bicep ファイルの準備

まず、上記で作成した Bicep ファイルをローカル環境に保存します。例えば、`main.bicep` という名前で保存することができます。

## 2. Azure CLI スクリプトの作成

次に、Azure CLI を使用して Bicep ファイルをデプロイするスクリプトを作成します。

```
bash Copy code

# Azureにログイン
az login

# サブスクリプションを設定 (必要に応じて)
az account set --subscription "<your_subscription_id>"

# リソースグループの作成 (まだ存在しない場合)
az group create --name "<your_resource_group_name>" --location "<your_location>"

# Bicep ファイルをデプロイ
az deployment group create \
  --name "<your_deployment_name>" \
  --resource-group "<your_resource_group_name>" \
  --template-file "<path_to_your_bicep_file>/main.bicep" \
  --parameters <additional_parameters_if_needed>
```

上記のスクリプトでは、以下のプレースホルダーを実際の値で置き換える必要があります：

- \* `<your_subscription_id>`: 使用する Azure サブスクリプション ID。
- \* `<your_resource_group_name>`: 使用する リソースグループの名前。
- \* `<your_location>`: リソースグループの地域 (例: `eastus`)。
- \* `<your_deployment_name>`: このデプロイメントの名前。
- \* `<path_to_your_bicep_file>`: Bicep ファイルへのパス。
- \* `<additional_parameters_if_needed>`: 追加のパラメータがある場合。



# GTP-4 with Vision で システム構成図 から IaC 作成も

@shyamagu (Shuji Yamaguchi) in Microsoft

## GPT4-Vで構成図からIaCコードを生成させる

IaC OpenAI ARMTemplate AzureOpenAIService GPT-4V

投稿日 2024年01月19日

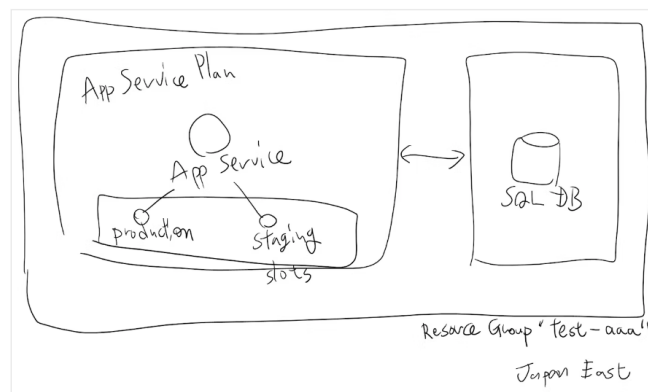
### はじめに

GPT-4-Turbo with Visionが楽しくて色々試しています。以前の記事ではフロー図からプログラムを生成させてみましたが、今回は雑な構成図からIaCコード(今回はARM template)を生成させてみたいと思います。

ARMテンプレートにした理由は特にはないです。別にterraformでもbicepでもいいんですけどね。

### 構成図

せっかくなので、手書きです。PCにえいやで書き込んだのでとても文字が汚いですがね  
App ServiceにStagingスロットつけて、あとはSQLDBがあるだけの単純構成です。



## GPT4-V のプロンプト

今回も英語のプロンプトで試しました。

日本語だと「

あなたはAzureのインフラストラクチャの専門家です。描画されたAzureリソースを作成するためのARMテンプレートを作成してください。

各リソース名には、"shstest-"という文字列に続けて、リソースを表す文字列と、都度ランダムな文字列も含めてください。

例 : shstest-appservice-dfiweofsd

」

という感じです。

ポイントとしては、リソース名が一意的なものを要求されることが多いので、ランダムな文字列をつけくわえさせているところです。

```
system_prompt = f"""\nYou are an Azure infrastructure expert. Please create an ARM template to create the depicted\nPlease add the resource-representing string to each resource name followed by "shstest-", and\nFor example: shstest-appservice-dfiweofsd\n"""
```

## 生成されたARMテンプレート

冒頭の画像と、上のプロンプトで以下のARMテンプレートが生成されました。

なお補足として、SQLDBのadministratorLoginとPasswordは適切に変えろという説明文がつけられます。確かに、このままのパスワードだとポリシー次第では強度不足でデプロイエラーになったりしますね。

# レポート用のSQL文作成

あなた

Database に対して、以下のクエリ処理をするSQL文を作成してください。

クエリ処理: ###

一番人気のあるレストランはどこですか?

###

< 1/2 >



ChatGPT

一番人気のあるレストランを特定するためには、レビューや評価などのデータを元に集計する必要があります。ここでは、レビューの数や平均評価を基に最も人気のあるレストランを特定するSQL文を作成します。このクエリは、レビューの平均評価が最も高いレストランを見つけるものとします。

以下のSQL文は、`Restaurants` テーブルと `Reviews` テーブルを使用して、最も高い平均評価を持つレストランを特定します。この例では、両テーブルが既に存在し、`Reviews` テーブルにはレストランIDと評価が含まれていると仮定します。

```
sql Copy code
SELECT TOP 1
  R.Name,
  AVG(RV.Rating) AS AverageRating
FROM
  Restaurants R
JOIN
  Reviews RV ON R.RestaurantId = RV.RestaurantId
GROUP BY
  R.Name
ORDER BY
  AverageRating DESC;
```

このクエリは以下の処理を行います：

1. `Restaurants` テーブルと `Reviews` テーブルをレストランIDで結合します。
2. 各レストランの平均評価を計算します。
3. 平均評価が最も高いレストランを選択します。

## 注意点:

- このクエリでは、単に平均評価が最も高いレストランを選んでいますが、実際のビジネスロジックによっては、レビューの数や他の指標を考慮する必要があるかもしれません。
- レビューの数が非常に少ないレストランが高評価を受ける可能性があるため、信頼性のある結果を得るためには、レビューの数を考慮に入れることが望ましいです。



# Microsoft Azure のサービス選定

実装

DA あなた

アプリケーションアーキテクチャの中の全てのコンポーネントをMicrosoft Azure で動かします。

コストを安く抑えるためにServerlessをなるべく使いたいです。かつスケールアウトできるサービスを選択したいです。それぞれのコンポーネントに最適なMicrosoft Azureのサービスは何ですか?その理由も教えてください。

< 3 / 3 >

ChatGPT

アプリケーションアーキテクチャの各コンポーネントをMicrosoft Azureで実行する際に、コスト効率とスケーラビリティを考慮して、以下のサービスを推奨します。

## 1. フロントエンド (Web UI)

- サービス: Azure Static Web Apps
- 理由: Azure Static Web Appsは、静的なウェブサイトとサーバーレスAPIをホストするためのサービスです。Vue.jsのようなSPA(Single Page Application)に最適で、GitHubやAzure DevOpsとの連携によりCI/CDを簡単に設定できます。コスト効率が良く、自動的にスケーリングします。

## 2. バックエンド (API)

- サービス: Azure Functions
- 理由: Azure Functionsは、イベント駆動型のサーバーレスコンピューティングサービスで、必要なリソースだけを使用してコードを実行します。リクエストに基づいて自動的にスケールアウトし、使用したリソースに応じてのみ課金されるため、コスト効率が非常に高いです。

## 3. データベース

- サービス: Azure Cosmos DB
- 理由: Azure Cosmos DBは、グローバル分散を備えたマルチモデルデータベースサービスで、大規模なスケーラビリティと高いパフォーマンスを提供します。サーバーレスオプションを選択することで、スレーブトやストレージの使用量に基づいてコストを最適化でき、過剰なプロビジョニングの心配がありません。

## 4. 認証・認可

- サービス: Azure Active Directory B2C (Azure AD B2C)
- 理由: Azure AD B2Cは、顧客向けのIDおよびアクセス管理ソリューションです。Webアプリケーションやモバイルアプリケーションにカスタマイズ可能なユーザーエクスペリエンスを提供し、大規模なユーザーベースに対してもスケーラブルな認証サービスを提供します。

## 5. ファイルストレージ

- サービス: Azure Blob Storage
- 理由: Azure Blob Storageは、大量の非構造化データを保存するためのオブジェクトストレージサービスです。静的ファイルや大規模なメディアファイルを格納するのに適しており、アクセスパターンに応じた複数のパフォーマンスティアを提供します。サーバーレスアプリケーションと組み合わせて使用することで、スケーラビリティとコスト効率を同時に実現できます。

これらのサービスを組み合わせることで、コスト効率の良いスケーラブルなアプリケーションアーキテクチャをMicrosoft Azure上に構築できます。サーバーレスサービスを中心に選択することで、初期コストを抑えつつ需要の変動に応じてリソースを自動的に調整できるため、運用コストの最適化にもつながります。

# 作成したファイルのダウンロード

実装

Code Interpreter は常に神 🥰

DA

あなた

会話の最初から、ここまで作成した全てのスクリプトやコードをダウンロードできるようにしてください。ゴール達成のステップに不足があったら補ってください。

ファイルダウンロードのステップ: ###

ステップ1: コードの一覧を作成

ステップ2: 一覧のアイテムごとにファイルのダウンロードの準備

ステップ3: 全てのファイルをzip圧縮

ステップ4: ダウンロードの準備

###



ChatGPT

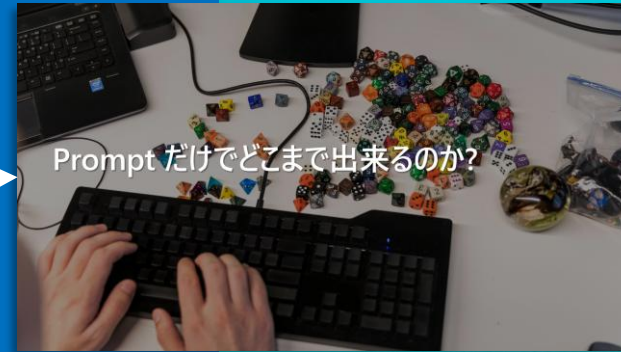
全てのコードファイルが含まれるzipファイルの準備が完了しました。以下のリンクからダウンロードできます：

ダウンロード [KoriyamaAppsCode.zip](#)

このzipファイルには、アプリケーションアーキテクチャ、マイルストーン、データモデル、画面遷移図（Mermaid形式）、Azure FunctionsのC#コード、Vue.jsのコンポーネント、HTMLファイル、SQLクエリ、およびAzure CLIコマンドが含まれています。これにより、ゴール達成のためのステップが完了しました。 [👉]



案件の相談から  
ここまで



30分 / 1人で🕶️

なんどもやり直せる短さ

殆どはPrompt  
出力の待ち時間



肌感覚ですが。

**95%以上**の  
図、スクリプト、コードは  
そのまま動きました 🧐







生産性と品質を上げるための  
**戦略**

# 戦略

100%にならない問題

Token の Input - Output

論理モデルと実装の疎結合

既存開発環境との融合



肌感覚ですが。

**95%以上**の  
図、スクリプト、コードは  
そのまま動きました 🤖

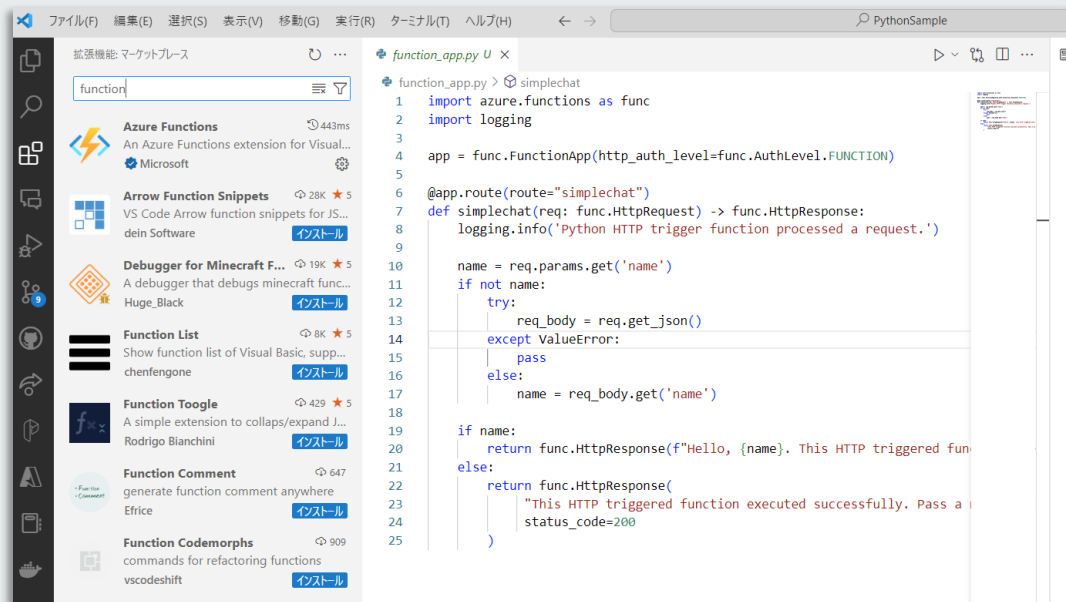


でも、その残り 5% が動かなくて。  
**半日** 費やすこともある

# テンプレート or 公式サンプルコード の活用は、常に正義

100%にならない問題

Visual Studio Code の Azure Functions Extensions で作成



適時利用

要件(利用関数、データモデルなど)は  
入っていない

ChatGPT で作成したコード

```
python
import azure.functions as func
import os
from azure.identity import DefaultAzureCredential
from azure.ai.textanalytics import TextAnalyticsClient
import json

# グローバル変数でメッセージ履歴を保持
chat_history = []

def main(req: func.HttpRequest) -> func.HttpResponse:
    global chat_history

    # リクエストからメッセージ取得
    message = req.params.get('message')
    if not message:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            message = req_body.get('message')

    if message:
        # メッセージを履歴に追加
        chat_history.append(message)

        # Azure OpenAI Service を使用して応答を生成
        response = query_openai(message)

        # 応答メッセージを履歴に追加
        chat_history.append(response)

    return func.HttpResponse(
        json.dumps({"response": response, "history": chat_history}),
        status_code=200
    )
else:
    return func.HttpResponse(
        "This HTTP triggered function executed successfully. Pass a message in the query string or a json body to the request.",
        status_code=200
    )

def query_openai(user_input):
    # Azure OpenAI サービスのクライアントを設定
    endpoint = os.environ["OPENAI_ENDPOINT"]
    credential = DefaultAzureCredential()
    client = TextAnalyticsClient(endpoint=endpoint, credential=credential)

    # Azure OpenAI Service にクエリを投げる
    # ここでは仮の関数としています。実際にはOpenAIのAPIを呼び出すコードを記述します。
    response = "OpenAI Response to: " + user_input
    return response
```

要件あり  
(利用関数、データモデル  
など)

# GitHub Copilot Chat もすげえ!

100%にならない問題

実は殆どの Prompt が動く!

これまでの情報を基にアプリケーションのアーキテクチャを作成してください。アプリケーションアーキテクチャの図はMermaid Live Editor の書式で作成してください。###

GitHub Copilot

1 参照 使用済み >

以下に、Mermaid Live Editor の書式でアプリケーションのアーキテクチャを示します。

```
graph LR
  A[ユーザー] -->|観光情報を検索| B[観光情報検索サービス]
  B --> C[観光情報データベース]
  A -->|レストランを検索| D[レストラン推薦サービス]
  D --> E[レストラン情報データベース]
  A -->|景色の良いスポットを検索| F[景色スポット推薦サービス]
  F --> G[景色スポット情報データベース]
  A -->|移動ルートを検索| H[移動ルート提案サービス]
  H --> I[公共交通情報データベース]
  A -->|支払いを行う| J[決済サービス]
  J --> K[決済情報データベース]
  A -->|データを閲覧| L[マルチデバイス対応サービス]
  L --> M[ユーザーデータデータベース]
```

この図では、ユーザーが各種サービスを通じて情報を検索したり、支払いを行ったりする流れを示しています。各サービスは、データを管理するデータベースにアクセスします。また、マルチデバイス対応サービスを通じて、ユーザーはスマートフォンやPCから同じデータにアクセス

新しいファイルに挿入する

ターミナルに挿入 Ctrl+Alt+Enter

感動!  
VS Code内でのファイル作成! 🥰

Engineering からかけ離れると... 🤪

dahatake

指示: ### 以下の文章から、非機能要件を抽出してください。非機能要件には以下の項目が必要です。分からなかったら「不明」とします。###

項目: ###

- 可用性
- 性能・拡張性
- 運用・保守性
- 移行性
- セキュリティ
- 環境・サステナビリティ

文章: ### システムに求められる非機能要件ですか、そうですね。まあやっぱり止まらないようにしてほしいなと思います。あの休日も正月もそうですね。それからユーザーの数はまあ、おそらくですけど、同時にそうですね。100人くらい捌ければいいのかなと思うんですよ。で、後から機能を追加するかもしれないので、ええ、その対応ができればいいかなと思っています。ええそうですね。データのバックアップとかもちろんの方がいいと思うんですけど、特にええとユーザーさんの課金の情報とか、予約の情報っていうのは、私たちの方で管理するってよりもそうですね。その連携サイトの方で、もちろん持っているわけですけど、ある程度はこちらでも持っておきたいですね。でも多分なくなると困るから、それは24時間 365日すぐでも戻せるようにしてほしいなと思います。まあいつお客さんいらっやってくるかわかりませんからね。予約も含めて現行のシステムはありません。新規に全部作ります。セキュリティですけどね。えっと、もちろんですけど、個人情報に近いところを管理することになるので、その部分に関してはすごく厳密なものが必要だと思います。エコロジーですか。太陽光発電なんか使える環境があると、いいかなと思いますけどね。まあほとんどスマホの中のアプリケーションだから、そこまではないかなと思います ###

GitHub Copilot

申し訳ございません。プログラミングに関連する質問のみをサポートできます。



# 戦略

100%にならない問題

Token の Input - Output

論理モデルと実装の疎結合

既存開発環境との融合



# LLMのトークン制限は Chat 前提

トークン数は増加傾向

テーブルを展開する

モデル ID	最大要求 (トークン)	トレーニング データ (最大)
gpt-4 (0314)	8,192	2021 年 9 月
gpt-4-32k (0314)	32,768	2021 年 9 月
gpt-4 (0613)	8,192	2021 年 9 月
gpt-4-32k (0613)	32,768	2021 年 9 月
gpt-4 (1106-preview) <sup>1</sup> GPT-4 Turbo プレビュー	入力: 128,000 出力: 4,096	2023 年 4 月
gpt-4 (vision-preview) <sup>2</sup> GPT-4 Turbo with Vision Preview	入力: 128,000 出力: 4,096	2023 年 4 月

[Azure OpenAI Service モデル - Azure OpenAI | Microsoft Learn](#)

実際の入力#2のトークン: 128,000

入力#1

出力#1: 4,096

入力#2

出力#2: 4,096

対話の継続 = 入力トークンのデータ追加

- ✓ 一度に**大量の出力**はできない
- ✓ トークン数が増えると、**精度が落ちやすい**。ノイズになる

# 戦略

100%にならない問題

Token の Input - Output

論理モデルと実装の疎結合

既存開発環境との融合

# 管理すべきは 論理モデル for ChatGPT

RDBMS: Azure SQL Database

## 論理データモデル

**あなた**  
これらのユースケースをシステムとして実現するために、どのようなデータモデルが考えられますか?  
Mermaid Live Editorの書式で作成してください。

**ChatGPT**  
郡山市の観光客向けWebアプリケーションのためのデータモデルを設計する際、以下のようなエンティティとその関連性を考えることができます。Mermaid Live Editorの書式でデータモデルの概要を示します。

```

erDiagram
    USER ||--o{ RESERVATION : "行"
    USER {
        string userId
        string name
        string email
        string password
    }
    RESERVATION ||--o{ RESTAURANT : "含む"
    RESERVATION {
        string reservationId
        datetime date
        int numberOfPeople
        string userId
        string restaurantId
        string paymentMethodId
    }
    RESTAURANT }|--o{ LOCATION : "所在"
    RESTAURANT {
        string restaurantId
        string name
        string cuisineType
        string locationId
    }
    LOCATION {
        string locationId
  
```



**あなた**  
Azure SQL Database を使います。テーブルの作成のSQL文を作成してください。  
< 2 / 3 >

**ChatGPT**  
Azure SQL Database で使用するためのテーブル作成のSQL文を作成します。ここでは、ユーザー (Users)、レストラン (Restaurants)、観光地 (Attractions)、レビュー (Reviews)、予約 (Reservations) の5つのテーブルを作成する例を示します。

1. ユーザー (Users) テーブル:

```

sql
CREATE TABLE Users (
  UserId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Email NVARCHAR(100) UNIQUE
);
  
```



**Optimize**

GitHub Copilot

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
  
```

```

CREATE TABLE Users (
  UserId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Email NVARCHAR(100) UNIQUE
);

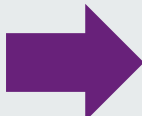
CREATE TABLE Restaurants (
  RestaurantId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Location NVARCHAR(100),
  CuisineType NVARCHAR(50)
);

CREATE TABLE Attractions (
  AttractionId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Description NVARCHAR(255)
);

CREATE TABLE Reviews (
  ReviewId INT PRIMARY KEY IDENTITY(1,1),
  Content NVARCHAR(255),
  Date DATETIME,
  Rating INT,
  UserId INT FOREIGN KEY REFERENCES Users(UserId),
  RestaurantId INT FOREIGN KEY REFERENCES Restaurants
);

CREATE INDEX idx_restaurants_name ON Res
  
```

Document: Azure Cosmos DB SQL API

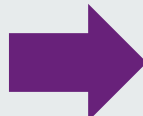


**あなた**  
Azure CosmosDB を使います。テーブルの作成のSQL文を作成してください。  
< 3 / 3 >

**ChatGPT**  
Azure Cosmos DB では、SQLを用いたテーブルの作成は、従来のリレーショナルデータベースのSQL文とは異なります。Cosmos DBはNoSQLデータベースであり、ドキュメント指向のストレージを使用しています。従って、「テーブル」の代わりに「コンテナ」を作成することになります。

ただし、具体的なコンテナの作成はAzure Cosmos DBのUIを介して行うか、Cosmos DB SDKを使用してプログラムの行います。SQLクエリは主にデータの読み取りや更新に使用されます。

コンテナを作成する際には、以下の点を考慮する必要があります：



**Optimize**

GitHub Copilot

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
  
```

```

users = [
  {"id": "001", "name": "山田太郎", "email": "taro.yama@hanako.jp"},
  {"id": "002", "name": "田中花子", "email": "hana.tanaka@hanako.jp"}
]

from azure.cosmos import CosmosClient, PartitionKey
client = CosmosClient(url, credential=key)
database = client.get_database_client(database_name)
container = database.get_container_client(container_name)

# データの挿入
for user in users:
    container.upsert_item(user)

print("データの挿入が完了しました。")
  
```

# 管理すべきは 論理モデル for ChatGPT

RDBMS: Azure SQL Database

論理データモデル

```

erDiagram
    USER ||--o{ RESERVATION : "has"
    USER {
        string userId
        string name
        string email
        string password
    }
    RESERVATION ||--o{ RESTAURANT : "has"
    RESERVATION {
        string reservationId
        datetime date
        int numberOfPeople
        string userId
        string restaurantId
        string paymentMethodId
    }
    RESTAURANT ||..o{ LOCATION : "has"
    RESTAURANT {
        string restaurantId
        string name
        string cuisineType
        string locationId
    }
    LOCATION {
        string locationId
  
```

各サービス・テクノロジーへの

# 最適化こそ

# Engineer の

# 腕の見せ所! 🧐

ChatGPT  
Azure Cosmos DB では、SQLを用いたテーブルの作成は、従来のリレーショナルデータベースのSQL文とは異なります。Cosmos DBはNoSQLデータベースであり、ドキュメント指向のストレージを使用しています。従って、「テーブル」の代わりに「コンテナ」を作成することになります。

ただし、具体的なコンテナの作成はAzure Cosmos DBのみまかして行うが、Cosmos DB SDKを使用してプログラム的にを行います。SQLクエリは主にデータの読み取りや更新に使用されます。

コンテナを作成する際には、以下の点を考慮する必要があります：

## Optimize

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
CREATE TABLE Users (
  UserId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Email NVARCHAR(100) UNIQUE
);
CREATE TABLE Restaurants (
  RestaurantId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Location NVARCHAR(100),
  CuisineType NVARCHAR(50)
);
CREATE TABLE Attractions (
  AttractionId INT PRIMARY KEY IDENTITY(1,1),
  Name NVARCHAR(100),
  Description NVARCHAR(255)
);
CREATE TABLE Reviews (
  ReviewId INT PRIMARY KEY IDENTITY(1,1),
  Content NVARCHAR(255),
  Date DATETIME,
  Rating INT,
  UserId INT FOREIGN KEY REFERENCES Users(UserId),
  RestaurantId INT FOREIGN KEY REFERENCES Restaurants(RestaurantId)
);
CREATE INDEX idx_restaurants_name ON Res
  
```

## Optimize

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
users = [
  {"id": "001", "name": "山田太郎", "email": "taro.yamada@example.com"},
  {"id": "002", "name": "田中花子", "email": "hana.tanaka@example.com"}
]
from azure.cosmos import CosmosClient, PartitionKey
# Cosmos DBの接続情報
url = "YOUR_COSMOS_DB_URL"
key = "YOUR_COSMOS_DB_KEY"
database_name = "YOUR_DATABASE_NAME"
container_name = "YOUR_CONTAINER_NAME"
# Cosmos DBクライアントの初期化
client = CosmosClient(url, credential=key)
database = client.get_database_client(database_name)
container = database.get_container_client(container_name)
# データの挿入
for user in users:
    container.upsert_item(user)
print("データの挿入が完了しました。")
import os
from azure.cosmos import CosmosClient, PartitionKey
users = [
  {"id": "001", "name": "山田太郎", "email": "taro.yamada@example.com"},
  {"id": "002", "name": "田中花子", "email": "hana.tanaka@example.com"}
]
# Cosmos DBの接続情報を環境変数から取得
url = os.getenv("COSMOS_DB_URL")
key = os.getenv("COSMOS_DB_KEY")
database_name = os.getenv("DATABASE_NAME")
container_name = os.getenv("CONTAINER_NAME")

```

# 戦略

100%にならない問題

Token の Input - Output

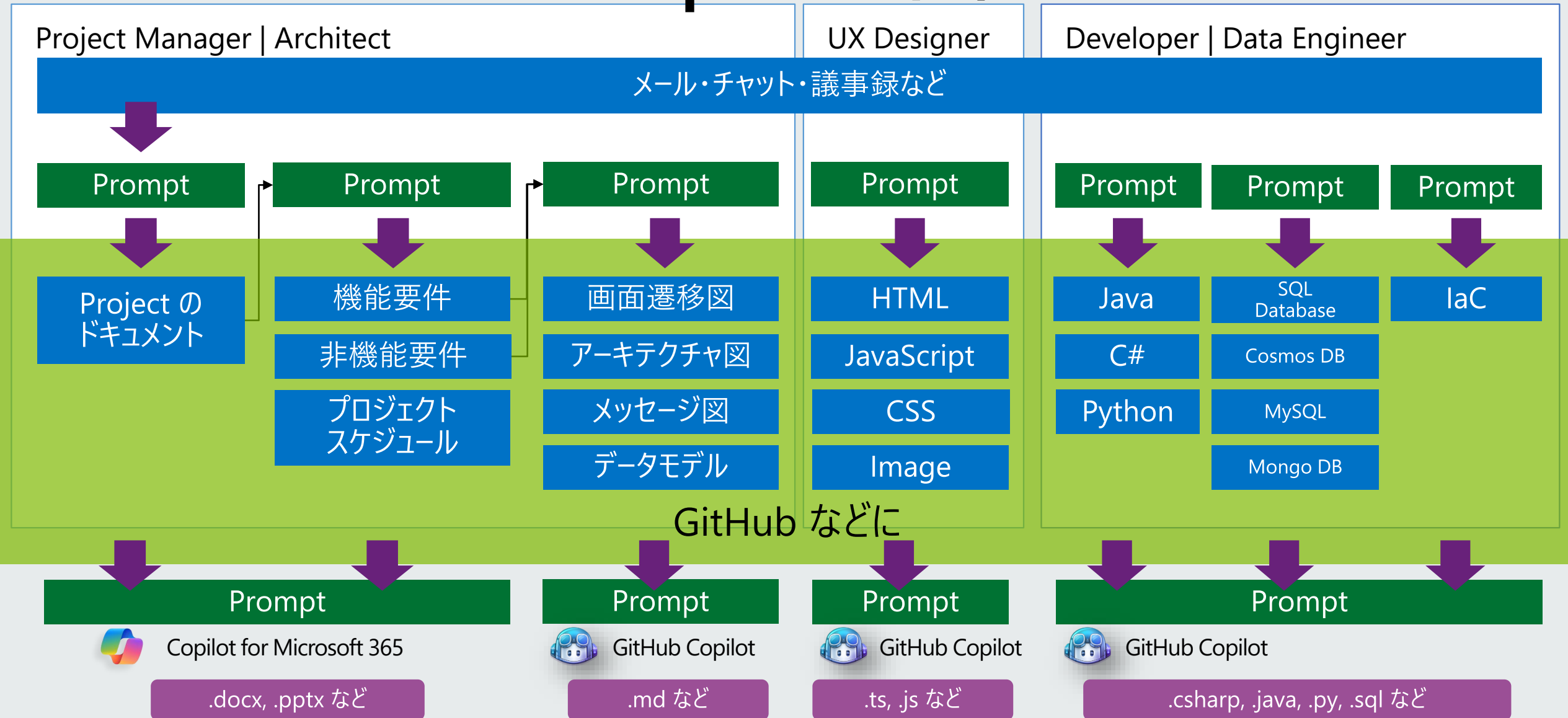
論理モデルと実装の疎結合

既存開発環境との融合



# チームでの Prompt 実行と Output 共有例

既存開発環境との  
融合



# チームでの Prompt 実行と Output 共有例

既存開発環境との融合

Project Manager | Architect

UX Designer

Developer | Data Engineer

メール・チャット・議事録など

# テキスト



ほとんどが

Prompt

Prompt

Prompt

Prompt

Prompt

Prompt

Prompt

Project の  
ドキュメント

機能要件  
非機能要件  
プロジェクト  
スケジュール

画面遷移図  
アーキテクチャ図  
メッセージ図  
データモデル

HTML  
JavaScript  
CSS  
Image

Java  
C#  
Python

SQL Database  
Cosmos DB  
MySQL  
Mongo DB

laC

GitHub などに

Prompt

Prompt

Prompt

Prompt



Copilot for Microsoft 365



GitHub Copilot



GitHub Copilot



GitHub Copilot

# Mermaid Live Editor の Visual Studio Code プラグインがあって、良かったー🥰

Visual Studio Code > Other > Markdown Preview Mermaid Support

## Markdown Preview Mermaid Support

Matt Bierner [mattbierner.com](#) | 📄 1,352,618 installs | ★★★★★ (43) | Free

Adds Mermaid diagram and flowchart support to VS Code's builtin markdown preview

[Install](#) [Trouble Installing?](#)

Overview Version History Q & A Rating & Review

### Markdown Preview Mermaid Support

Visual Studio Marketplace **v1.21.0**

Adds **Mermaid** diagram and flowchart support to VS Code's builtin Markdown preview and to Markdown cells in notebooks.

Categories: Other

Tags: diagram, flowchart, markdown, mermaid, preview

Works with: Universal, Web

Resources: Issues, Repository, Homepage, License, Changelog, Download Extension

Project Details: mjbvz/vscode-markdown-mermaid, 3 Pull Requests, Last Commit: 2 months ago, 29 Open Issues

More Info: Version 1.21.0, Released on 2017/8/11 3:04:16, Last updated 2023/12/7 8:06:06, Publisher Matt Bierner, Unique Identifier bierner.markdown-mermaid, Report Report a concern

Usage: Create diagrams in markdown using `mermaid` fenced code blocks:

```
```mermaid
graph TD;
  A-->B;
  A-->C;
  B-->D;
```

ApplicationDevelopmentLifeCycle.md x ApplicationDevelopmentLifeCycle.md x プレビュー アプリケーション-キテクチャ.md x

```
1 mermaid
2 graph TD
3   subgraph Frontend
4     mobileApp[スマートフォンアプリ]
5     webApp[ウェブアプリケーション]
6   end
7
8   subgraph Backend
9     api[APIサーバ]
10    db[(データベース)]
11    auth[認証サーバ]
12    payment[決済サービス]
13  end
14
15  subgraph ExternalServices
16    mapService[地図サービス]
17    reviewService[レビューサービス]
18    weatherService[天気情報サービス]
19  end
20
21  mobileApp -->|APIリクエスト| api
22  webApp -->|APIリクエスト| api
23  api -->|データ取得・操作| db
24  api -->|ユーザー認証| auth
25  api -->|決済処理| payment
26  api -->|外部サービス連携| mapService
27  api -->|外部サービス連携| reviewService
28  api -->|外部サービス連携| weatherService
29
```

```
```mermaid
graph TD
mobileApp[スマートフォンアプリ]
webApp[ウェブアプリケーション]
end
```

```mermaid  
を付与するのを忘れなく

# 既存コードから図的なものの作成

## GitHub Copilot Chat

The screenshot shows the GitHub Copilot Chat interface. On the left, there's a chat window with a user message and a Copilot response. The response includes a diagram of the API components and their relationships. On the right, a code editor shows a Python script named 'HelloWorld.py' that uses the Azure OpenAI API to create an assistant, a thread, and a run, and then displays the results.

```
client = AzureOpenAI(
    api_key=os.getenv("AZURE_OPENAI_KEY"),
    api_version="2024-02-15-preview",
    azure_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
)

file = client.files.create(
    file=open("ほにやらデータ.csv", "rb"),
    purpose='assistants'
)

# Create an assistant
assistant = client.beta.assistants.create(
    name="Data Visualization",
    instructions=f"You are a helpful AI assistant who makes
    f"You have access to a sandboxed environment for writin
    f"when you are asked to create a visualization you shou
    f"1. Write the code."
    f"2. Anytime you write new code display a preview of th
    f"3. Run the code to confirm that it runs."
    f"4. If the code is successful display the visualizatio
    f"5. If the code is unsuccessful display the error mess
    tools=[{"type": "code_interpreter"}],
    model="gpt-4-1106-preview" #You must replace this value
    file_ids=[file.id]
)

# Create a thread
thread = client.beta.threads.create()
print(thread)

# Add a user question to the thread
message = client.beta.threads.messages.create(
    thread_id=thread.id,
    role="user",
    content="Create a visualization of a sinewave"
)

run = client.beta.threads.runs.create(
    thread_id=thread.id,
    assistant_id=assistant.id,
    #instructions="New instructions" #You can optionally prov
)

data = json.loads(messages.model_dump_json(indent=2)) # Lo
image_file_id = data['data'][1]['content'][0]['image_file']
```

## ChatGPT

The screenshot shows the ChatGPT interface. A user prompt asks for a Mermaid diagram of the API components from a Python code snippet. The response includes the Python code and a Mermaid diagram showing the flow of the process: client initialization, file upload, assistant creation, thread creation, and run execution, with sub-steps for image file ID and status retrieval.

あなた  
指示: ###  
以下のPythonのコードを分析して、APIのコンポーネント図をMermaid Live Editorの書式で作成してください。結果は日本語にしてください。  
###

Pythonのコード:  
###  
# https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/create-a-thread-and-run  
import os  
import json  
from openai import AzureOpenAI

client = AzureOpenAI(
 api\_key=os.getenv("AZURE\_OPENAI\_KEY"),
 api\_version="2024-02-15-preview",
 azure\_endpoint = os.getenv("AZURE\_OPENAI\_ENDPOINT")
)

file = client.files.create(
 file=open("ほにやらデータ.csv", "rb"),
 purpose='assistants'
)

# Create an assistant
assistant = client.beta.assistants.create(
 name="Data Visualization",
 instructions=f"You are a helpful AI assistant who makes
 f"You have access to a sandboxed environment for writin
 f"when you are asked to create a visualization you shou
 f"1. Write the code."
 f"2. Anytime you write new code display a preview of th
 f"3. Run the code to confirm that it runs."
 f"4. If the code is successful display the visualizatio
 f"5. If the code is unsuccessful display the error mess
 tools=[{"type": "code\_interpreter"}],
 model="gpt-4-1106-preview" #You must replace this value
 file\_ids=[file.id]
)

# Create a thread
thread = client.beta.threads.create()
print(thread)

# Add a user question to the thread
message = client.beta.threads.messages.create(
 thread\_id=thread.id,
 role="user",
 content="Create a visualization of a sinewave"
)

run = client.beta.threads.runs.create(
 thread\_id=thread.id,
 assistant\_id=assistant.id,
 #instructions="New instructions" #You can optionally prov
)

data = json.loads(messages.model\_dump\_json(indent=2)) # Lo
image\_file\_id = data['data'][1]['content'][0]['image\_file']

ChatGPT  
このPythonコードは、Azure OpenAIのAPIを使用して、ファイルのアップロード、アシスタントの作成、質問の追加、実行の開始、そして結果の取得と表示を行う一連の処理を示しています。Mermaid Live Editorで使用するためのコンポーネント図の書式に従って、この処理の流れを表す図を作成します。

以下は、指定されたPythonコードの処理を表すMermaid Live Editorの書式でのAPIコンポーネント図です。図は、主要なコンポーネントとそれらの間の関係を示します。

```
graph TD
    A[クライアントの初期化] --> B[アシスタントの作成]
    B --> C[スレッドの作成]
    C --> D[質問の追加]
    D --> E[実行の開始]
    E --> F[実行のステータスの取得]
    E --> G[画像ファイルIDの取得]
    G --> H[画像の取得]
    H --> I[画像の表示]
    F --> J[新しい質問の追加]
    J --> D
```

この図は以下の流れを表しています:

- クライアントの初期化: Azure OpenAIのAPIクライアントが初期化されます。
- ファイルのアップロード: CSVファイルがアップロードされます。
- アシスタントの作成: 特定の指示、ツール、ファイルIDに基づいてアシスタントが作成されます。

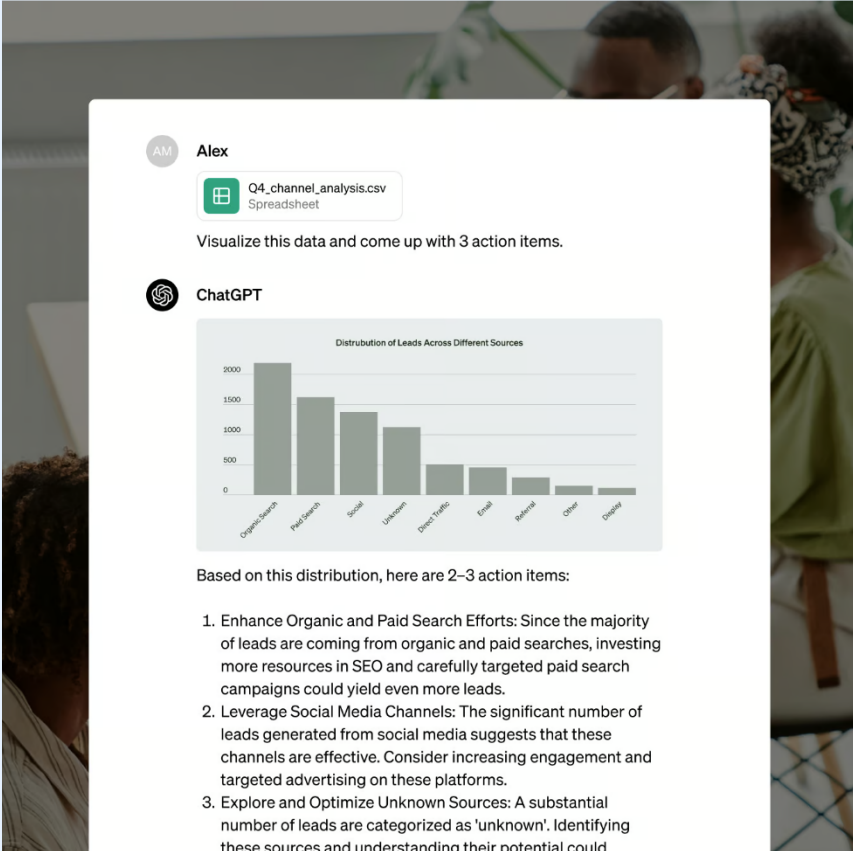
# Prompt 環境のチームで共有 – わかる 😊

OpenAI Research API ChatGPT Safety Company Search Log in Try ChatGPT

Blog

## Introducing ChatGPT Team

We're launching a new ChatGPT plan for teams of all sizes, which provides a secure, collaborative workspace to get the most out of ChatGPT at work.



Alex

Q4\_channel\_analysis.csv Spreadsheet

Visualize this data and come up with 3 action items.

ChatGPT

Distribution of Leads Across Different Sources

| Source         | Leads |
|----------------|-------|
| Organic Search | 2000  |
| Paid Search    | 1500  |
| Social         | 1200  |
| Unknown        | 1000  |
| Direct Traffic | 500   |
| Email          | 400   |
| Referral       | 300   |
| Other          | 200   |
| Unlabeled      | 100   |

Based on this distribution, here are 2-3 action items:

1. Enhance Organic and Paid Search Efforts: Since the majority of leads are coming from organic and paid searches, investing more resources in SEO and carefully targeted paid search campaigns could yield even more leads.
2. Leverage Social Media Channels: The significant number of leads generated from social media suggests that these channels are effective. Consider increasing engagement and targeted advertising on these platforms.
3. Explore and Optimize Unknown Sources: A substantial number of leads are categorized as 'unknown'. Identifying these sources and understanding their potential could





更なる自動化への  
ツールの登場も近いでしょう 😊





さいごに





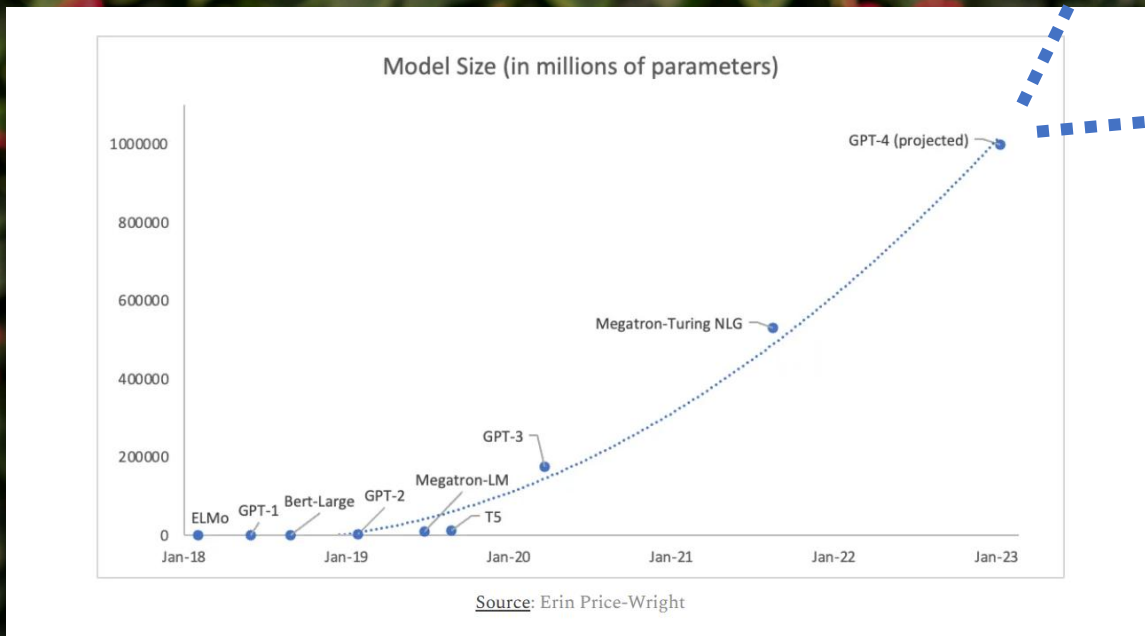
# LLM は進化の途中

?

Parameter数の増加?

?

Parameter数が少なくても能力が向上?



2024

2025

2026

2027

Computing の進化 | 社会への貢献



# Prompt Engineering は補助にすぎない!

### 指示 ###

次の文章の内容を変更せずに、  
CEFRのC1レベルに添削してください。

添削した文章は英語で作成してください。  
なぜ、その改善をしたのかの理由は  
日本語で作成してください。

ステップバイステップで考えてください。

文章: ###

I found the technical issue at our cloud services. The almost of behavior of issue looks like come from storage layer. I guess disk IO is near peak. How can I check this?

###

Prompt Engineering のテクニック: Delimiter

具体的に何をしてもらいたいのか?

Prompt Engineering のテクニック: Chain of Thought

Prompt Engineering のテクニック: Delimiter

何かをもらうための  
追加のデータや情報

Prompt Engineering のテクニック: Delimiter



“Prompt”  
所詮は、言葉



PCを使う→**キーボード**の練習

スマホを使う→**フリック入力**などの練習

ChatGPTを使う→**Prompt**の練習  
これが「今」🕶️





毎日

Prompt

してね 😊

# ChatGPT のより深い理解

- 黒橋教授（京都大学） - ChatGPT の仕組みと社会へのインパクト / NII 教育機関 DX シンポ (2023)
  - [https://www.nii.ac.jp/event/upload/20230303-04\\_Kurohashi.pdf](https://www.nii.ac.jp/event/upload/20230303-04_Kurohashi.pdf)
- 話題爆発中のAI「ChatGPT」の仕組みにせまる！ - Qiita
  - <https://qiita.com/omiita/items/c355bc4c26eca2817324>





# Microsoft Azure

---

**Invent with purpose.**