05

CHAPTER

Gemini API を組み込む

ゲームに AI を統合する — 本書で最も重要な章

本書のサンプル章

AGENDA

この章で学ぶこと

所要時間：約 2 時間

01 **Gemini API キーの取得**
Google AI Studio で発行 + 課金上限

02 **環境変数で安全に管理**
.env / python-dotenv / settings.py

03 **初めての API 呼び出し**
Django shell で動作確認 → ビュー実装

04 **プロンプトエンジニアリング**
ゲームマスター用プロンプト設計

05 **エラーハンドリング**
5 パターンのエラーに備える

06 **コスト管理 & レート制限対策**
指数バックオフ +4 秒間隔 + キャッシング

PART 1

準備 — API キーと環境変数

API は「身分証明 → 質問 → 回答」のシンプルな仕組み

CONCEPT

API の使い方は 3 ステップ

01

身分証明

API キーを取得

Google AI Studio で発行

02

質問する

リクエスト送信

プロンプトを送る

03

回答を受け取る

レスポンス受信

AI が生成したテキスト

API キー取得の流れ — Google AI Studio

STEP 1

AI Studio へ

`aistudio.google.com`

STEP 2

Google ログイン

規約同意

STEP 3

「 API キーを作成 」

プロジェクト指定

STEP 4

コピー & 保存

`.env` に貼る

⚠ *API キーは絶対に GitHub にコミットしない！。漏れたら即 Revoke 。*

PRICING

無料枠と有料プラン

FREE TIER

無料枠は厳しくなった

(2025 年後半以降)

- ✗ 学習・開発には不十分
- ✗ テストですぐに制限到達
- ✗ 連続呼び出しでブロック
- ✗ 実用的な開発には有料推奨

トークン ：日本語 1 文字 ≈ 2 〜3 トークン

BUDGET ALERT (必須)

課金上限を設定する

Google Cloud Console → 請求 → 予算とアラート

学習・開発	1,000 〜 3,000 円 / 月
小規模運用	5,000 〜 10,000 円 / 月
本書のゲーム	1,000 円 / 月で十分

⚠ *アラート閾値 : 50% / 90% / 100% で通知設定*

ENV

.env で安全に管理する 4 ステップ

01

INSTALL

ライブラリ追加

```
$ uv add python-dotenv
```

.env ファイルを読むためのライブラリ

02

CREATE

.env ファイル作成

```
GEMINI_API_KEY=AIzaSy…
DEBUG=True
SECRET_KEY=…
```

プロジェクトルート（manage.py と同じ場所）

03

IGNORE

.gitignore に追加

```
.env
.env.local
```

絶対に Git にコミットしない！

04

LOAD

settings.py で読込

```
load_dotenv(BASE_DIR / '.env')
GEMINI_API_KEY = os.getenv('GEMINI_API_KEY')
```

settings.py の冒頭で 1 度だけ

settings.py

環境変数を読み込む ＆ 動作確認

server/settings.py

```
import os
from pathlib import Path
from dotenv import load_dotenv

BASE_DIR = Path(__file__).resolve()\
    .parent.parent

# .env を読み込む
load_dotenv(BASE_DIR / '.env')

SECRET_KEY = os.getenv(
    'SECRET_KEY', 'fallback-dev')

DEBUG = os.getenv(
    'DEBUG', 'False') == 'True'

# Gemini API Key
GEMINI_API_KEY = os.getenv(
    'GEMINI_API_KEY')
```

ANIMATION CHECK

動作確認

```
$ uv run manage.py shell
```

```
>>> from django.conf \
...   import settings
>>> print(
...   settings.\
...   GEMINI_API_KEY[:10])
AIzaSyDb_D...
```

→ API キーが表示されれば OK

PART 2

API 呼び出しとプロンプト

実装を進め、 AI を呼び出す

FIRST CALL

Django shell で動作確認

STEP 1 — ライブラリ追加

```
$ uv add google-genai
```

STEP 2 — Python コード

```
from google import genai
from django.conf import settings

# クライアント作成
client = genai.Client(
    api_key=settings.GEMINI_API_KEY)

# 質問する
response = client.models.generate_content(
    model="gemini-2.5-flash",
    contents="日本で一番高い山は？")

print(response.text)
```

STEP 3 — 出力

```
日本で一番高い山は、
** 富士山（ふじさん）** です。

標高は **3,776 メートル **
です。

静岡県と山梨県にまたがり、
世界文化遺産にも登録
されています。
```

MODEL

Gemini モデルの選び方

RECOMMENDED

gemini-2.5-flash

推奨

速度	高速
コスト	低
品質	ゲームには十分
用途	チャット・対話・短文生成

本書ではこちらを使用

PREMIUM

gemini-2.5-pro

高品質モデル

速度	標準
コスト	高
品質	最高品質
用途	複雑な推論・長文生成

今回のゲームにはオーバースペック

views.py

ビュー関数で API を呼ぶ

game/views.py — test_gemini

```
client = genai.Client(
    api_key=settings.GEMINI_API_KEY)

def test_gemini(request):
    if request.method == 'POST':
        user_input = request.POST\
            .get('message', '')
        try:
            response = client.models\
                .generate_content(
                model="gemini-2.5-flash",
                contents=user_input)
            return JsonResponse({
                'user_message': user_input,
                'ai_response': response.text})
        except Exception as e:
            return JsonResponse(
                {'error': str(e)}, status=500)
```

URL ルーティング

```
# game/urls.py
urlpatterns = [
    path('', views.home),
    path('test-gemini/',
         views.test_gemini),
]
```

PROMPT

良いプロンプトの基本構造

良いプロンプト ＝ 良い応答

ROLE

【役割】

あなたは〇〇です

ゲームマスター / 講師 / 翻訳者など

CONTEXT

【設定】

舞台は△△です

状況・世界観・前提条件

RULES

【ルール】

以下のルールに従って

200 文字以内 / 番号付きリストで など

INPUT

【入力】

ユーザーの入力： XXX

プレイヤーの行動 / 質問 / データ

4 つの工夫

1 役割を明確に

「ゲームマスター」

2 制約を設ける

「200 文字以内」

3 具体的な指示

「2-3 個の選択肢」

4 文脈を含める

過去 5 件の履歴

PROMPT DESIGN

ゲームマスタープロンプト

game/prompts.py — GAME_SYSTEM_PROMPT

```
あなたは「廃墟の研究所からの脱出」という
テキストアドベンチャーゲームの GM です。

【設定】
- 舞台： 放棄された地下研究施設
- 雰囲気： ミステリアス、怖すぎない

【 GM としてのルール】
1. 200 文字以内で状況を描写
2. 次の行動を 2-3 個提示
3. 危険な状況も作る
4. 謎解き要素を含める
5. 詰まったらヒントを出す

【現在のゲーム状況】
{current_context}

【プレイヤーの行動】
{user_action}
```

なぜ 200 文字？

読みやすさ

長すぎるとプレイヤーが疲れる

トークン節約

200 文字 ≈ 500 トークン

テンポ

短い応答で会話が進む

※ 100 〜 300 文字が目安

+ 直近 5 件の会話履歴を文脈として渡す

PART 3

エラーハンドリング & コスト管理

API は常に成功するとは限らない

ERRORS

よくあるエラー　TOP 5

401

認証エラー

```
API key not valid
```

→ API キーを確認 / 再発行

リトライ不可

429

レート制限

```
Resource exhausted
```

→ 4 秒待つ / 指数バックオフ

リトライ可

Timeout

応答が遅い

```
Request timed out
```

→ タイムアウト延長

リトライ可

400

不正リクエスト

```
Invalid prompt
```

→ プロンプト短縮 / サニタイズ

リトライ不可

5xx

サーバーエラー

```
Internal Server Error
```

→ 少し待って再試行

リトライ可

RESILIENCE

リトライ ＋ 指数バックオフ

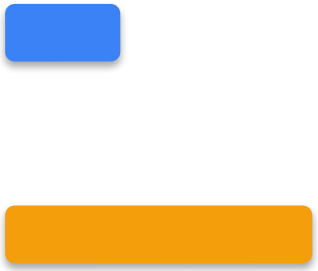
EXPONENTIAL BACKOFF

失敗するたび待機時間を倍に

1 試行 1
失敗 ✗
1 秒待つ
2^0

2 試行 2
失敗 ✗
2 秒待つ
2^1

3 試行 3
成功 ✓
—
—



→ サーバー負荷を抑えながら確実に再試行

リトライロジック

```
for attempt in range(
    max_retries):
    try:
        return client\
            .models\
            .generate_content(
              ...).text
    except APIError as e:
        if e.status_code\
           in [429,500,503]:
            wait = 2 ** attempt
            time.sleep(wait)
            continue
        return error_msg
```

COST

レート制限とコスト管理

15 RPM

Gemini 無料枠の上限

1 分あたり 15 リクエスト

4 秒

推奨インターバル

60 秒 ÷ 15 = 4 秒

3 回

最大リトライ

1 秒→ 2 秒→ 4 秒で待機

MODEL

安価モデルを選ぶ

gemini-2.5-flash を使用
(pro より安価)

OUTPUT

出力長を制限

200 文字程度に
max_output_tokens=300

CACHE

プロンプトを最適化

システムプロンプト固定
変化部分だけ送信

DONE

AI を組み込んだ。次は UI で命を吹き込む。

この章で学んだこと

- ✓ Gemini API キーの取得
- ✓ 環境変数（.env）で安全に管理
- ✓ 初めての API 呼び出し
- ✓ プロンプトエンジニアリング
- ✓ エラーハンドリング（5 パターン）
- ✓ リトライと指数バックオフ
- ✓ レート制限とコスト管理

NEXT

第 6 章

チャット UI の実装

- htmx で非同期通信
- Bootstrap で美しい UI
- リアルタイムなゲーム体験

→ generate_game_response はそのまま再利用

ゲームの心臓部（LLM）ができた。次は美しく表示する UI を作る。