🏗️ ゆるVibe Pages システム構成概要図

> *インフラ・サービス・API の全体アーキテクチャ俯瞰*

システム構成概要

このシステム構成図は、ゆるVibe Pagesのインフラストラクチャ、外部サービス連携、内部アーキテクチャの全体像を表現しています。Vercelホスティング、Firebase サービス群、OpenAI API、フロントエンド・バックエンドの関係性を包括的に示しています。

全体システム構成図

```
graph TB
    %% ユーザー層
    U[ユーザー] --> B[ブラウザ]
    S[SNS ユーザー] --> X[Twitter/X]

    %% CDN・ホスティング層
    B --> V[Vercel CDN]
    V --> N[Next.js 15 App]

    %% フロントエンド層
    N --> H[ホームページ /]
    N --> P[詩表示ページ /view/[id]]
    N --> T[テストページ群]
    N --> D[デバッグページ /debug]
```

```
%% APIルーティング層
N --> A1[/api/generate-storage]
N --> A2[/api/generate-safe]
N --> A3[/api/generate-dummy]
N --> A4[/api/generate-simple]
N --> A5[/api/generate-direct]
N --> A6[/api/generate]

%% 外部AI サービス層
A1 --> O[OpenAI GPT-4o]
A1 --> G[OpenAI DALL-E 3]
A2 --> O
A2 --> G
A6 --> O
A6 --> G

%% Firebase サービス層
A1 --> FS[Firebase Storage]
A1 --> FD[Firebase Firestore]
A2 --> FD
A6 --> FS
A6 --> FD
P --> FD
D --> FS
D --> FD

%% データ永続化層
FD --> PD[(Poems Collection)]
FS --> IMG[(Generated Images)]

%% 外部画像サービス
A3 --> US[Unsplash API]

%% SNS 連携
P --> X
X --> SH[SNS 共有]

%% 監視・分析（将来拡張）
V --> AN[Vercel Analytics]
FD --> FB[Firebase Analytics]

%% スタイル定義
```

```
    classDef user fill:#e1f5fe
    classDef hosting fill:#e8f5e8
    classDef frontend fill:#f3e5f5
    classDef api fill:#fff3e0
    classDef external fill:#ffebee
    classDef database fill:#fce4ec
    classDef monitoring fill:#f1f8e9

    class U,S,B user
    class V,N hosting
    class H,P,T,D frontend
    class A1,A2,A3,A4,A5,A6 api
    class O,G,US,X external
    class FS,FD,PD,IMG database
    class AN,FB monitoring
```

インフラストラクチャ詳細

Vercel デプロイメント構成

```
graph LR
    A[Git Repository] --> B[Vercel Build]
    B --> C[Next.js Build]
    C --> D[Static Generation]
    C --> E[Server Functions]

    D --> F[Vercel Edge Network]
    E --> G[Vercel Serverless Functions]

    F --> H[Global CDN]
    G --> I[API Endpoints]

    H --> J[Static Assets]
    I --> K[Dynamic API Routes]

    J --> L[CSS/JS/Images]
    K --> M[/api/* Routes]
```

```
    %% 環境変数
    N[Environment Variables] --> G
    N --> O[OPENAI_API_KEY]
    N --> P[FIREBASE_CONFIG]

    %% スタイル
    classDef build fill:#e8f5e8
    classDef deploy fill:#f3e5f5
    classDef runtime fill:#fff3e0

    class A,B,C build
    class D,E,F,G deploy
    class H,I,J,K,L,M runtime
```

Firebase プロジェクト構成

```
graph TD
    A[Firebase Project] --> B[Firestore Database]
    A --> C[Firebase Storage]
    A --> D[Firebase Hosting - 未使用]
    A --> E[Firebase Analytics - 将来]

    B --> F[poems コレクション]
    F --> G[Document Structure]
    G --> H[id: string]
    G --> I[theme: string]
    G --> J[phrase: string]
    G --> K[imageUrl: string]
    G --> L[imagePrompt: string]
    G --> M[createdAt: Date]

    C --> N[generated-images/ バケット]
    N --> O[timestamp-nanoid.png]
    N --> P[Custom Metadata]
    P --> Q[theme]
    P --> R[prompt]
    P --> S[generatedAt]

    %% セキュリティ
    B --> T[Security Rules - 開発用]
```

```
    C --> U[CORS Configuration]

    %% SDK統合
    V[Firebase SDK] --> B
    V --> C
    W[Next.js App] --> V

    classDef firebase fill:#fff3e0
    classDef data fill:#e8f5e8
    classDef security fill:#ffebee

    class A,B,C,V firebase
    class F,G,H,I,J,K,L,M,N,O data
    class T,U security
```

API アーキテクチャ詳細

API エンドポイント戦略

```
graph TD
    A[クライアントリクエスト] --> B{エンドポイント選択}

    B -->|本番| C[/api/generate-storage]
    B -->|テスト| D[/api/generate-safe]
    B -->|開発| E[/api/generate-dummy]
    B -->|高速| F[/api/generate-simple]
    B -->|直接| G[/api/generate-direct]
    B -->|基本| H[/api/generate]

    %% Storage版（メイン）
    C --> I[GPT-4o + DALL-E 並列]
    I --> J[Firebase Storage 保存]
    J --> K{Storage成功？}
    K -->|成功| L[Storage URL使用]
    K -->|失敗| M[DALL-E URL フォールバック]
    L --> N[Firestore保存]
    M --> N
```

```
    %% Safe版
    D --> O[GPT-4o + Safe DALL-E]
    O --> P[基本Firestore保存]

    %% Dummy版
    E --> Q[DummyData Service]
    Q --> R[Unsplash画像]
    R --> S[Dummy Firestore保存]

    %% Simple版
    F --> T[GPT-4o + DALL-E]
    T --> U[Storage回避]
    U --> V[DALL-E URL直接保存]

    %% Direct版
    G --> W[DALL-E URL直接使用]
    W --> X[即座Firestore保存]

    %% Basic版
    H --> Y[標準GPT-4o + DALL-E]
    Y --> Z[標準Firestore保存]

    classDef mainEndpoint fill:#e8f5e8
    classDef testEndpoint fill:#fff3e0
    classDef process fill:#f3e5f5
    classDef storage fill:#fce4ec

    class C mainEndpoint
    class D,E,F,G,H testEndpoint
    class I,O,Q,T,W,Y process
    class J,K,L,M,N,P,S,V,X,Z storage
```

データフロー アーキテクチャ

```
sequenceDiagram
    participant U as ユーザー
    participant V as Vercel CDN
    participant N as Next.js App
    participant A as API Endpoint
    participant O as OpenAI API
```

```
    participant D as DALL-E API
    participant S as Firebase Storage
    participant F as Firestore
    participant P as 詩ページ

    U->>V: ページリクエスト
    V->>N: アプリ配信
    N->>U: ホームページ表示

    U->>N: テーマ入力・送信
    N->>A: POST /api/generate-storage

    par 並列AI処理
        A->>O: 詩生成リクエスト
        O-->>A: 生成詩
    and
        A->>D: 画像生成リクエスト
        D-->>A: 画像URL
    end

    A->>S: 画像アップロード
    alt Storage成功
        S-->>A: Storage URL
    else Storage失敗
        Note over A: DALL-E URL使用
    end

    A->>F: 詩データ保存
    F-->>A: 保存完了

    A-->>N: 生成結果
    N-->>U: 詩ページリダイレクト

    U->>P: 詩ページアクセス
    P->>F: 詩データ取得
    F-->>P: 詩データ

    P->>S: 画像読み込み
    alt getBlob() 成功
        S-->>P: Blob データ
        Note over P: Object URL作成
    else getBlob() 失敗
        P->>S: getDownloadURL()
```

```
        S-->>P: 直接URL
    end

    P-->>U: 完全な詩ページ表示
```

セキュリティ・認証構成

現在のセキュリティ実装

```
graph TD
    A[セキュリティレイヤー] --> B[環境変数管理]
    A --> C[API Key保護]
    A --> D[CORS対応]
    A --> E[入力サニタイズ]

    B --> F[Vercel Environment Variables]
    F --> G[OPENAI_API_KEY]
    F --> H[FIREBASE_CONFIG]

    C --> I[サーバーサイド限定]
    C --> J[クライアント露出なし]

    D --> K[Firebase SDK getBlob()]
    D --> L[CORS設定回避]

    E --> M[React標準サニタイズ]
    E --> N[XSS対策]

    %% 将来の強化予定
    O[将来のセキュリティ強化] --> P[Firebase Auth]
    O --> Q[Firestore Security Rules]
    O --> R[API Rate Limiting]
    O --> S[Content Moderation]

    classDef current fill:#e8f5e8
    classDef future fill:#fff3e0
    classDef protection fill:#ffebee
```

```
    class A,B,C,D,E,F,G,H,I,J,K,L,M,N current
    class O,P,Q,R,S future
    class G,H,I,J protection
```

パフォーマンス・監視構成

パフォーマンス最適化

```
graph LR
    A[パフォーマンス最適化] --> B[フロントエンド]
    A --> C[API最適化]
    A --> D[データベース最適化]
    A --> E[画像最適化]

    B --> F[Next.js 15最適化]
    B --> G[Tailwind CSS v4]
    B --> H[Canvas 2D API]
    B --> I[コンポーネント最適化]

    C --> J[並列API処理]
    C --> K[Promise.allSettled]
    C --> L[エラーハンドリング]
    C --> M[タイムアウト設定]

    D --> N[Firestore インデックス]
    D --> O[適切なクエリ]
    D --> P[データ構造最適化]

    E --> Q[16:9 アスペクト比]
    E --> R[1792x1024 解像度]
    E --> S[Object URL管理]
    E --> T[自動クリーンアップ]

    classDef optimization fill:#e8f5e8
    classDef technique fill:#f3e5f5

    class A optimization
```

```
    class B,C,D,E optimization
    class F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T technique
```

監視・ログ戦略

```
graph TD
    A[監視システム] --> B[アプリケーション監視]
    A --> C[インフラ監視]
    A --> D[エラー監視]
    A --> E[パフォーマンス監視]

    B --> F[コンソールログ]
    B --> G[構造化ログ]
    B --> H[API レスポンス時間]

    C --> I[Vercel Functions]
    C --> J[Firebase 使用量]
    C --> K[OpenAI API 使用量]

    D --> L[API エラー率]
    D --> M[Firebase エラー]
    D --> N[OpenAI エラー]

    E --> O[ページ読み込み時間]
    E --> P[画像読み込み時間]
    E --> Q[API処理時間]

    %% 将来の拡張
    R[将来の監視拡張] --> S[Sentry統合]
    R --> T[Google Analytics]
    R --> U[カスタム監視]

    classDef current fill:#e8f5e8
    classDef future fill:#fff3e0

    class A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q current
    class R,S,T,U future
```

スケーラビリティ設計

水平スケーリング対応

```
graph TB
    A[スケーラビリティ戦略] --> B[Vercel Auto Scaling]
    A --> C[Firebase Scaling]
    A --> D[OpenAI API Scaling]
    A --> E[コード設計]

    B --> F[Serverless Functions]
    B --> G[Edge Caching]
    B --> H[Global CDN]

    C --> I[Firestore Auto Scaling]
    C --> J[Storage Auto Scaling]
    C --> K[Concurrent Connections]

    D --> L[API Rate Limits対応]
    D --> M[Fallback策]
    D --> N[エラーハンドリング]

    E --> O[ステートレス設計]
    E --> P[コンポーネント分離]
    E --> Q[ライブラリ関数独立性]

    classDef scaling fill:#e8f5e8
    classDef infrastructure fill:#f3e5f5

    class A scaling
    class B,C,D,E scaling
    class F,G,H,I,J,K,L,M,N,O,P,Q infrastructure
```

将来拡張アーキテクチャ

Phase 3-5 拡張予定

```
graph TD
    A[現在のアーキテクチャ] --> B[Phase 3: セキュリティ強化]
    B --> C[Phase 4: パフォーマンス最適化]
    C --> D[Phase 5: 高度な機能]

    B --> E[Firebase Auth導入]
    B --> F[セキュリティルール実装]
    B --> G[API Rate Limiting]

    C --> H[Redis キャッシュ]
    C --> I[CDN最適化]
    C --> J[Database Query最適化]

    D --> K[リアルタイム機能]
    D --> L[AI学習データ活用]
    D --> M[高度なアナリティクス]

    %% 新しいサービス統合
    N[新サービス統合] --> O[WebSocket Server]
    N --> P[Machine Learning Pipeline]
    N --> Q[Advanced Analytics]

    classDef current fill:#e8f5e8
    classDef phase3 fill:#fff3e0
    classDef phase4 fill:#f3e5f5
    classDef phase5 fill:#fce4ec

    class A current
    class B,E,F,G phase3
    class C,H,I,J phase4
    class D,K,L,M,N,O,P,Q phase5
```

> *「システムの構成は宇宙の調和のように。美しい秩序で心に響く体験を紡ぐ、にゃ〜」* ✨