

GeoReferencer 機能仕様書

1. プロジェクト概要

1.1 アプリケーションの目的

GeoReferencerは、PNG画像（ハイキングマップなど）を国土地理院の地理院地図上に精密にジオリファレンス（地理的位置合わせ）することを専門とするWebアプリケーションです。最小二乗法による6パラメータアフィン変換技術を用いて、高精度な画像位置合わせを実現します。

1.2 主要機能

- **GPS座標データの読み込み・表示**（Excel/GeoJSON形式）
- **Excel「区分」列フィルタリング**（「ポイント」のみ読み込み）
- **PNG画像ファイルの読み込み・オーバーレイ表示**
- **画像内座標データの読み込み・表示**（JSON形式、複数ファイル対応）
- **精密アフィン変換によるジオリファレンス**（最小二乗法）
- **変換済み座標データのGeoJSON出力**（仕様準拠、標高数値型対応）
- **自動ポイントマッチング機能**（IDベース）
- **リアルタイム座標同期機能**

1.3 技術的特徴

- **完全ES6モジュール構成**（13ファイル、モジュラーアーキテクチャ）
- **非同期初期化**（Promise-based確実な初期化）
- **精密座標変換**（Web Mercator ↔ WGS84）
- **標高データ数値型変換**（文字列→数値型変換処理）
- **File System Access API対応**（モダンブラウザ保存）

2. システム構成

2.1 アーキテクチャ概要

GeoReferencerは完全なES6モジュール構成を採用し、機能別に分離された13のコアモジュールからなるモジュラーアーキテクチャを採用しています。

2.2 モジュール構成と依存関係

```
GeoReferencerApp (app-main.js)
├── MapCore (map-core.js) [地図初期化・レイヤー管理]
├── ImageOverlay (image-overlay.js) [画像オーバーレイ処理]
├── GPSTData (gps-data.js) [GPS/GeoJSONデータ処理]
├── Georeferencing (georeferencing.js) [精密アフィン変換処理]
│   └── AffineTransformation (affine-transformation.js) [アフィン変換計算]
├── RouteSpotHandler (route-spot-handler.js) [ルート・スポットデータ管理]
├── CoordinateDisplay (coordinate-display.js) [座標表示・マーカー管理]
├── UIHandlers (ui-handlers.js) [UI操作ハンドラー]
└── FileHandler (file-handler.js) [ファイル読み込み・保存・Excel検証]
```

- └─ Utils (utils.js) [ログ・エラーハンドリング]
- └─ MathUtils (math-utils.js) [数学計算・座標変換]
- └─ Constants (constants.js) [定数定義]

2.3 外部依存関係

- **Leaflet.js v1.9.4:** 地図レンダリング (CDN経由)
- **SheetJS v0.18.5:** Excelファイル処理 (CDN経由)
- **国土地理院タイル:** 地図データソース

2.4 ファイル構成

```
GeoReferencer/  
├─ index.html           # メインHTML  
├─ styles.css           # 統合CSS (CSS変数使用)  
├─ CLAUDE.md            # プロジェクト仕様書  
├─ README.md            # プロジェクト概要  
├─ prompt.md            # 開発指示書  
├─ docs/  
│   ├── funcspec-202510.md # 機能仕様書 (本文書)  
│   ├── geojsonSpec-202510.md # GeoJSON入出力仕様  
│   └─ UsersGuide-202510.md # 利用者の手引  
├─ js/  
│   ├── app-main.js      # メインアプリケーション  
│   ├── map-core.js      # 地図コア機能・レイヤー管理  
│   ├── image-overlay.js  # 画像オーバーレイ処理  
│   ├── gps-data.js       # GPS/GeoJSONデータ処理  
│   ├── georeferencing.js # 精密アフィン変換処理  
│   ├── affine-transformation.js # アフィン変換計算専用  
│   ├── route-spot-handler.js # ルート・スポットデータ管理  
│   ├── coordinate-display.js # 座標表示・マーカー管理  
│   ├── ui-handlers.js    # UI操作ハンドラー  
│   ├── file-handler.js   # ファイル処理統合・Excel検証  
│   ├── math-utils.js     # 数学・座標変換統合  
│   ├── utils.js          # ユーティリティ・ログ機能  
│   └─ constants.js       # 定数定義
```

3. 機能詳細

3.1 メインアプリケーション (GeoReferencerApp)

ファイル: js/app-main.js **責任範囲:** アプリケーション全体の初期化・統合・イベント管理

主要メソッド:

- **init():** 非同期アプリケーション初期化
- **initializeModules():** 各モジュールの依存関係を考慮した初期化
- **setupEventHandlers():** 統合されたファイル読み込みボタンのイベント設定
- **handleGpsExcelLoad():** Excelファイル読み込み処理 (区分列フィルタリング対応)

- `handleMatchPoints()`: ジオリファレンス実行の統合処理
- `collectGeoreferencedData()`: 変換済み座標データの収集・GeoJSON生成（標高数値型対応）

データフロー: 統合されたファイル読み込み → Excel検証（区分フィルタ） → データ変換（標高数値化） → ジオリファレンス実行 → GeoJSON出力

3.2 地図コア機能（MapCore）

ファイル: `js/map-core.js` **責任範囲:** Leaflet地図の初期化・専用ペイン管理・スケールコントロール

主要機能:

- **非同期初期化:** Promise-baseの確実な地図初期化
- **専用ペイン管理:** z-index制御による5層レイヤー構造
 - `gpsMarkers` (z-index: 610): GPSポイント表示
 - `pointJsonMarkers` (z-index: 620): 画像座標ポイント表示
 - `wayPointMarkers` (z-index: 630): ルート中間点表示
 - `spotMarkers` (z-index: 630): スポット表示
 - `routeLines` (z-index: 600): ルート線表示
- **コントロール配置:** 右下にズーム・スケールコントロール配置

技術仕様: 国土地理院標準地図（2-18ズーム）、箕面大滝中心の初期表示

3.3 画像オーバーレイ処理（ImageOverlay）

ファイル: `js/image-overlay.js` **責任範囲:** PNG画像の読み込み・表示・境界計算・アフィン変換対応

主要機能:

- **画像読み込み:** PNG専用のFileReader処理
- **境界計算:** Mercator投影補正を考慮した精密境界計算
- **アフィン変換対応:** `setTransformedPosition()` による変換結果の反映
- **スケール管理:** 内部スケール値の管理・更新
- **コールバック機能:** 画像更新時の自動通知機能

計算式:

```
// メートル/ピクセル変換（Mercator投影補正）
const metersPerPixel = 156543.03392 * Math.cos(centerLat * Math.PI / 180) /
Math.pow(2, zoomLevel);

// 境界オフセット計算
const latOffset = (scaledImageHeightMeters / 2) / earthRadius * (180 / Math.PI);
const lngOffset = (scaledImageWidthMeters / 2) / (earthRadius * cosLat) * (180 /
Math.PI);
```

3.4 GPS/GeoJSONデータ処理（GPSData）

ファイル: `js/gps-data.js` **責任範囲:** GeoJSONファイルの読み込み・変換・地図表示

対応フォーマット:

- **GeoJSON**: FeatureCollection、単一Feature形式 (Point geometry)

データ変換機能:

- `loadGeoJsonFile()`: GeoJSONファイル読み込み (FileReader)
- `processGeoJsonData()`: GeoJSON構造解析・Point抽出
- `displayPointsOnMap()`: 地図上へのマーカー表示
- `setPointsFromExcelData()`: Excelデータから変換されたポイントデータ設定

マーカー表示: 緑色円形マーカー (半径16px)、専用GPSペイン使用

3.5 精密アフィン変換処理 (Georeferencing)

ファイル: `js/georeferencing.js` **責任範囲**: 最小二乗法による6パラメータアフィン変換・精度計算・座標同期

技術仕様:

- **変換方式**: 最小二乗法による6パラメータアフィン変換
- **最小制御点数**: 3点以上 (推奨: 4点以上)
- **座標系**: WGS84 ↔ Web Mercator変換
- **精度評価**: 平均誤差・最大誤差・最小誤差の計算表示 (メートル単位)

変換式:

$$\begin{aligned} X &= a \cdot x + b \cdot y + c \\ Y &= d \cdot x + e \cdot y + f \end{aligned}$$

主要メソッド:

- `executeGeoreferencing()`: ジオリファレンス実行
- `performAutomaticGeoreferencing()`: 自動変換処理
- `matchPointJsonWithGPS()`: IDベース自動マッチング
- `syncPointPositions()`: ポイント位置の自動同期
- `syncRouteSpotPositions()`: ルート・スポット位置の自動同期

3.6 アフィン変換計算 (AffineTransformation)

ファイル: `js/affine-transformation.js` **責任範囲**: 数学的アフィン変換計算の専用処理

計算手法:

- **正規方程式**: $(A^T \cdot A) \cdot x = A^T \cdot B$
- **ガウス・ジョーダン法**: 連立方程式の数値解法
- **スケール計算**: 制御点ベースの実際スケール算出

精度評価機能: Web Mercator座標系での誤差計算 (メートル単位)

3.7 ルート・スポットデータ管理 (RouteSpotHandler)

ファイル: `js/route-spot-handler.js` 責任範囲: JSON自動判定・ルート/スポット処理・マーカ表示

自動判定基準:

- ルート: `routeInfo+points+type="waypoint"`
- スポット: `spots`配列 or 単一`name`+座標
- ポイント: `points`配列で`type≠"waypoint"`

マーカ仕様:

- ルート中間点: オレンジ色ダイヤモンド型 (8×8px)
- スポット: 青色正方形 (12×12px)

重複判定: 座標・ID双方による重複回避機能

3.8 座標表示・マーカ管理 (CoordinateDisplay)

ファイル: `js/coordinate-display.js` 責任範囲: 画像座標の表示・境界ベース座標変換・マーカ管理

座標変換方式:

1. 境界ベース変換: 画像境界内での相対位置計算
2. フォールバック変換: 地図中心からの正規化オフセット

マーカ種別:

- ポイントJSON: 赤色円形 (半径6px)
- ジオリファレンスポイント: 制御点表示用

自動更新機能: 画像境界変更時のマーカ位置自動調整

3.9 UI操作ハンドラー (UIHandlers)

ファイル: `js/ui-handlers.js` 責任範囲: カウンター更新・マッチング結果表示

カウンター管理:

- GPS ポイント数: リアルタイム更新
- 画像内ポイント数: `type≠"waypoint"`をフィルタ
- ルート・スポット数: 自動判定結果の反映
- マッチング結果: 一致数・不一致ポイント表示

3.10 ファイル処理統合 (FileHandler)

ファイル: `js/file-handler.js` 責任範囲: ファイル読み込み・保存・Excel検証・標高数値型変換

対応ファイル形式:

- JSON: 自動構造判定・パース処理
- Excel (.xlsx): SheetJS使用・1000行制限・区分列フィルタリング
- PNG画像: MIME type検証・naturalWidth/Height取得

Excel検証機能（file-handler.js:312-406）：

- **必須列チェック**: 区分、ポイントID、名称、緯度、経度
- **区分列フィルタリング**: 「区分」列が「ポイント」の行のみ読み込み
- **標高数値型変換**: parseFloat()による文字列→数値変換
- **座標範囲検証**: 緯度（-90～90）、経度（-180～180）
- **データ型検証**: 数値・文字列の厳密チェック

標高処理（file-handler.js:384-391）：

```
// 標高を数値型に変換（文字列を数値に変換）
let elevation = null;
if (pointData['標高'] !== undefined && pointData['標高'] !== null && pointData['標高'] !== '') {
    const elevationValue = parseFloat(pointData['標高']);
    if (!isNaN(elevationValue)) {
        elevation = elevationValue;
    }
}
```

保存機能:

- **File System Access API**: ブラウザネイティブファイル保存
- **従来ダウンロード**: フォールバック対応
- **ディレクトリ記憶**: 前回ファイル位置の記録

ファイル名生成: {PNG名}-GPS.geojson 形式の自動生成

3.11 数学・座標変換統合（MathUtils）

ファイル: js/math-utils.js **責任範囲**: 座標変換・行列計算・マーカー作成の統合ユーティリティ

座標変換関数:

- **Web Mercator変換**: 経緯度 ↔ Web Mercator
- **距離計算**: Haversine公式による測地距離
- **メートル/ピクセル**: Mercator投影補正計算

行列計算機能:

- **転置・乗算**: 数値計算ライブラリ相当の機能
- **ガウス・ジョーダン法**: 部分ピボット法対応
- **連立方程式**: 最小二乗法用の数値解法

マーカー作成統合: 複数種類のカスタムマーカー生成機能

3.12 ユーティリティ・ログ機能（Utils）

ファイル: js/utils.js **責任範囲**: ログ管理・エラーハンドリング・バリデーション

ログ機能:

- **4レベルログ**: ERROR、WARN、INFO、DEBUG
- **LocalStorage保存**: 最大1000件の履歴保持
- **コンテキスト管理**: モジュール別ログ分類
- **デバッグモード**: URLパラメータ・LocalStorage制御

エラーハンドリング:

- **グローバルエラー**: 未処理エラー・Promise rejection捕捉
- **モーダル表示**: タイトル・メッセージ・タイプ別表示
- **自動クリア**: 成功・警告メッセージの5秒自動削除
- **ESCキー**: キーボードによる閉じる操作

バリデーション: ポイントID形式・全角半角変換・ファイル形式チェック

4. ユーザーインターフェース

4.1 UI構成

- **地図エリア**: 画面全体に表示される国土地理院地図
- **制御パネル**: 左上固定の320px幅操作パネル
- **コントロール**: 右下のズーム・スケールコントロール

4.2 統合ファイル読み込み機能

ラジオボタン選択式読み込み

```
<!-- 統合読み込みボタン -->
<button id="loadFileBtn">読み込み</button>

<!-- ファイル種類選択 -->
<input type="radio" name="fileType" value="gpsExcel" checked> ポイントGPS
<input type="radio" name="fileType" value="image"> PNG画像
```

動作仕様 (app-main.js:95-112) :

- ラジオボタンで選択されたファイル種別に応じて対応する入力要素をクリック
- **gpsExcel**: Excelファイル選択ダイアログ表示
- **image**: PNG画像ファイル選択ダイアログ表示

画像内座標読み込み

```
<button id="loadJsonBtn">画像内座標の読み込み</button>
```

- **複数ファイル対応**: JSON自動判定による一括読み込み
- **自動分類**: ポイント・ルート・スポットの自動判別

4.3 リアルタイムカウンター表示

- **GPSポイント数**: 読み込み済みGPS座標数（「区分」 = 「ポイント」のみ）
- **ポイント数**: 画像内ポイント座標数（waypointを除外）
- **ルート数**: 検出されたルート数
- **スポット数**: 検出されたスポット数
- **マッチング結果**: 一致ポイント数・不一致ポイント一覧

4.4 ジオリファレンス操作

```
<button id="matchPointsBtn">画像の重ね合わせ（ジオリファレンス） </button>
<button id="exportGeoJsonBtn">GPS出力(GeoJSON) </button>
```

4.5 マーカー表示仕様

- **GPSポイント**: 緑色円形（半径16px）
- **ポイントマーカー**: 赤色円形（半径6px）
- **ルート中間点**: オレンジ色ダイヤモンド型（8×8px）
- **スポットマーカー**: 青色正方形（12×12px）

5. ファイル処理

5.1 対応ファイル形式

入力ファイル

- **GPS座標（Excel .xlsx）**：
 - 必須列: 区分、ポイントID、名称、緯度、経度
 - オプション列: 標高、備考
 - **区分列フィルタリング**: 「区分」 = 「ポイント」 の行のみ読み込み
 - **標高数値型変換**: 文字列型の標高をparseFloat()で数値型に変換
- **GPS座標（GeoJSON）** : FeatureCollection/Feature Point geometry
- **画像**: PNG形式画像ファイル
- **画像内座標（JSON）** : ポイント・ルート・スポット座標データ

出力ファイル

- **GeoJSON**: 変換済み座標データ（docs/geojsonSpec-202510.md準拠）
 - 標高フィールドは数値型で出力

5.2 Excel読み込み仕様

必須列

列名	型	説明	例
区分	string	データ種別（「ポイント」のみ読み込み）	"ポイント"
ポイントID	string	識別子	"J-05"

列名	型	説明	例
名称	string	地点名	"東海道自然歩道"
緯度	number	緯度 (-90～90)	34.87202
経度	number	経度 (-180～180)	135.49331

オプション列

列名	型	説明	例
標高	number	標高（メートル、数値型に変換）	564.7
備考	string	備考情報	""

区分列フィルタリング仕様

- 「区分」列が必須になりました
- 「区分」 = 「ポイント」の行のみが読み込まれます
- 「分岐点」「交差点」などの行は自動的に除外されます

標高数値型変換仕様

- Excelから読み込まれた標高データ（文字列型の可能性あり）を`parseFloat()`で数値型に変換
- 変換できない値は`null`として扱う
- GeoJSON出力時に数値型として出力（"で囲まれない）

5.3 JSONファイル自動判定

ポイントデータ判定

```
{
  "points": [
    {
      "id": "A-01",
      "imageX": 150,
      "imageY": 200,
      "name": "ポイント名"
    }
  ]
}
```

ルートデータ判定

```
{
  "routeInfo": {
    "startPoint": "A-01",
```

```

    "endPoint": "A-05"
  },
  "points": [
    {
      "type": "waypoint",
      "imageX": 150,
      "imageY": 200
    }
  ]
}

```

スポットデータ判定

```

{
  "spots": [
    {
      "name": "展望台",
      "imageX": 300,
      "imageY": 400
    }
  ]
}

```

5.4 GeoJSON出力仕様

[docs/geojsonSpec-202510.md](#)に準拠した3つのFeatureタイプを出力：

1. ポイントGPS

```

{
  "type": "Feature",
  "properties": {
    "id": "J-05",
    "name": "東海道自然歩道",
    "type": "ポイントGPS",
    "source": "GPS_Excel",
    "description": "ポイント（Excel管理GPS値）",
    "notes": ""
  },
  "geometry": {
    "type": "Point",
    "coordinates": [135.49331, 34.87202, 564.7]
  }
}

```

- **description**: "ポイント（Excel管理GPS値）"（固定値）
- **source**: "GPS_Excel"（固定値）

- `coordinates[2]`: 標高は**数値型**で出力

2. ルート中間点

```
{
  "type": "Feature",
  "properties": {
    "id": "route_C-03_to_J-01_waypoint_06",
    "name": "waypoint_06",
    "type": "route_waypoint",
    "source": "image_transformed",
    "route_id": "route_C-03_to_J-01",
    "description": "ルート中間点"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [135.49353, 34.86449]
  }
}
```

- `id`: {route_id}_{waypoint_XX} 形式
- `route_id`: route_{開始ポイント}_to_{終了ポイント} 形式
- `name`: waypoint_XX 形式（2桁ゼロパディング）

3. スポット

```
{
  "type": "Feature",
  "properties": {
    "id": "spot08_薬師堂",
    "name": "薬師堂",
    "type": "spot",
    "source": "image_transformed",
    "description": "スポット"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [135.49052, 34.86557]
  }
}
```

- `id`: spot{連番2桁}_{name} 形式
- `name`: スポット名（meta.spotIdから取得）

5.5 ファイル保存機能

- **File System Access API**: モダンブラウザでのネイティブ保存
- **従来ダウンロード**: フォールバック対応

- **ディレクトリ記憶**: 前回使用フォルダの記録
- **ファイル名自動生成**: {PNG名}-GPS.geojson

6. ジオリファレンス機能

6.1 精密アフィン変換詳細

変換パラメータ

6パラメータアフィン変換：

$$\begin{aligned} X &= a \cdot x + b \cdot y + c \\ Y &= d \cdot x + e \cdot y + f \end{aligned}$$

- **a, b, c**: X座標変換係数（回転・スケール・平行移動）
- **d, e, f**: Y座標変換係数（回転・スケール・平行移動）

最小二乗法による計算

正規方程式： $(A^T \cdot A) \cdot x = A^T \cdot B$

- **A**: 係数行列（2n×6）
- **B**: 定数ベクトル（Web Mercator座標）
- **x**: 求める変換パラメータ

精度評価

```
const accuracy = {
  meanError: number,    // 平均誤差（メートル）
  maxError: number,     // 最大誤差（メートル）
  minError: number,     // 最小誤差（メートル）
  errors: Array<number> // 各点の誤差配列
}
```

6.2 座標系対応

- **入力座標系**: WGS84（緯度経度）
- **内部計算**: Web Mercator投影
- **出力座標系**: WGS84（GeoJSON準拠）

6.3 自動位置同期機能

ポイント同期

- **制御点**: ジオリファレンス変換適用
- **GPSポイント**: 元座標維持
- **リアルタイム更新**: 画像変更時の自動調整

ルート・スポット同期

- **画像由来**: アフィン変換適用
- **GPS由来**: 元座標維持
- **メタデータ**: origin情報による自動判別

6.4 IDマッチング機能

- **自動マッチング**: GPS座標のポイントIDと画像内座標のIDによる自動対応付け
- **マッチング結果表示**: 一致数・不一致ポイント一覧をUI表示
- **不一致処理**: マッチしないポイントの警告表示

7. 技術仕様

7.1 使用技術・ライブラリ

- **フロントエンド**: ES6モジュール、Vanilla JavaScript
- **地図エンジン**: Leaflet.js v1.9.4
- **ファイル処理**: SheetJS v0.18.5、FileReader API
- **スタイル**: CSS変数、Flexbox、CSS Grid
- **数値計算**: 自実装（行列計算、アフィン変換）

7.2 ブラウザ要件

- **ES6モジュール対応**: Chrome 61+、Firefox 60+、Safari 10.1+
- **File System Access API**: Chrome 86+（オプション）
- **CORS対応**: ローカルサーバー必須

7.3 パフォーマンス特性

- **初期化時間**: 地図読み込み完了まで約1-3秒
- **ファイル読み込み**: Excel 1000行まで、PNG 10MB程度まで対応
- **ジオリファレンス**: 3-100制御点で実用的な処理速度
- **メモリ効率**: Leafletネイティブ機能の活用

7.4 セキュリティ

- **ファイル検証**: MIME type・拡張子の厳密チェック
- **入力サニタイズ**: HTML エスケープ・XSS対策
- **エラーハンドリング**: グローバルエラー捕捉
- **CSP対応**: Content Security Policy適用可能

8. 設定・制限事項

8.1 設定可能項目（constants.js）

```
export const CONFIG = {  
  // 地図設定  
  MAP_INITIALIZATION_TIMEOUT: 5000,
```

```
// ファイル制限
MAX_EXCEL_ROWS: 1000,
ACCEPTED_IMAGE_TYPES: ['image/png'],

// アフィン変換設定
AFFINE_TRANSFORMATION_MODE: 'auto',

// UI設定
MESSAGE_BOX_Z_INDEX: 10000
};

export const DEFAULTS = {
  // 地図初期位置
  MAP_CENTER: [34.853667, 135.472041], // 箕面大滝
  MAP_ZOOM: 15,

  // 画像設定
  IMAGE_OVERLAY_DEFAULT_SCALE: 0.8,
  IMAGE_OVERLAY_DEFAULT_OPACITY: 50
};
```

8.2 制限事項

- **ファイル形式**: PNG画像のみ対応（JPEG、SVG等は未対応）
- **Excel制限**: 最大1000行、.xlsx形式のみ、「区分」列必須
- **座標系**: WGS84のみ（JGD2011等は未対応）
- **ブラウザ制限**: ES6モジュール必須、CORS制限あり
- **制御点**: 最低3点、推奨4点以上

8.3 エラーハンドリング

自動エラー処理

- **ファイル形式不正**: 自動検出・ユーザー通知
- **区分列不一致**: 「ポイント」以外の行を自動除外
- **標高変換失敗**: 数値変換できない場合はnullとして扱う
- **座標範囲外**: 自動除外・警告表示
- **変換失敗**: フォールバック処理・エラー詳細表示
- **メモリ不足**: 制限値による予防・段階的処理

ログシステム

- **4レベル**: ERROR、WARN、INFO、DEBUG
- **永続化**: LocalStorage 1000件履歴
- **デバッグモード**: URL/LocalStorage制御
- **エクスポート**: ログデータ出力機能

9. 開発・運用情報

9.1 開発環境

```
# ローカルサーバー起動（CORS回避）
python -m http.server 8000
# または
npx serve .

# ブラウザアクセス
http://localhost:8000
```

9.2 プロジェクト特徴

- **完全モジュラー**: 13の独立したES6モジュール
- **非同期初期化**: Promise-based確実な初期化
- **型安全性**: 厳密なデータ検証・エラーハンドリング
- **標高数値型変換**: Excel読み込み時の自動型変換
- **拡張性**: 新機能追加に配慮した設計
- **保守性**: 単一責任原則に基づく明確な分離

9.3 最新アップデート（v1.3）

- **Excel「区分」列フィルタリング**: 「ポイント」のみ読み込み対応
- **標高数値型変換**: parseFloat()による文字列→数値変換
- **読み込みボタン修正**: ラジオボタン値の不一致解消（gpsExcel対応）
- **GeoJSON仕様準拠**: description、スポットID形式の修正

9.4 今後の拡張可能性

- **座標系追加**: JGD2011、UTM等への対応
- **ファイル形式**: JPEG、GeoTIFF等への対応
- **変換方式**: 多項式変換、TIN変換等の追加
- **API連携**: 外部座標変換サービスとの統合
- **PWA化**: Service Worker、オフライン機能の追加

改訂履歴

- **v1.0** (2025-09-12): 初版リリース
- **v1.1** (2025-09-20): リファクタリング版対応、モジュール統合・分離、不要コード削除
- **v1.2** (2025-09-28): GeoJSON出力仕様準拠、データソース分類実装、命名規則統一
- **v1.3** (2025-10-26): Excel「区分」列フィルタリング、標高数値型変換、読み込みボタン修正