時系列・地理空間情報に関する
データ分析・可視化サンプルアプリケーション

システム構築ガイド

情報通信研究機構　総合テストベッド研究開発推進センター

改版履歴

版数	作成年月日	改定箇所	改定内容、理由
1.0	2023/5/20	全ページ	新規作成
1.1	2024/1/23	全ページ	評価報告書記載のシステム構築に関する指摘、改善案に従って、記載漏れ、不備を修正。
1.2	2024/2/19	全ページ	環境構築の順序に合わせて文書構成を修正。

＜　目 次　＞

1　はじめに

「時系列・地理空間情報に関するデータ分析・可視化サンプルアプリケーション」は、センサーデータ等、時系列・地理空間情報を２次元・３次元の地図上に可視化するツールであり、国立研究開発法人情報通信研究機構（NICT）総合テストベッド研究開発推進センターがデータを活用したサービス開発基盤（DCCS）の研究開発プロジェクトの一環として試作しました。本アプリケーションはオープンソースソフトウェアとして公開しており、本アプリケーションで用いるデータは原則オープン化されていますので、どなたでも無償で導入いただくことができます。本書は、当センターが本アプリケーションを構築した際の手順について記述したもので、システム開発者の方むけの内容です。各自の環境に導入する際の参考にしていただければ幸いです。一部、当機構のシステムに依存している内容がありますので、各自の環境に合わせてカスタマイズしてください。

2　システム概要

2.1.1　システム全体構成の概要

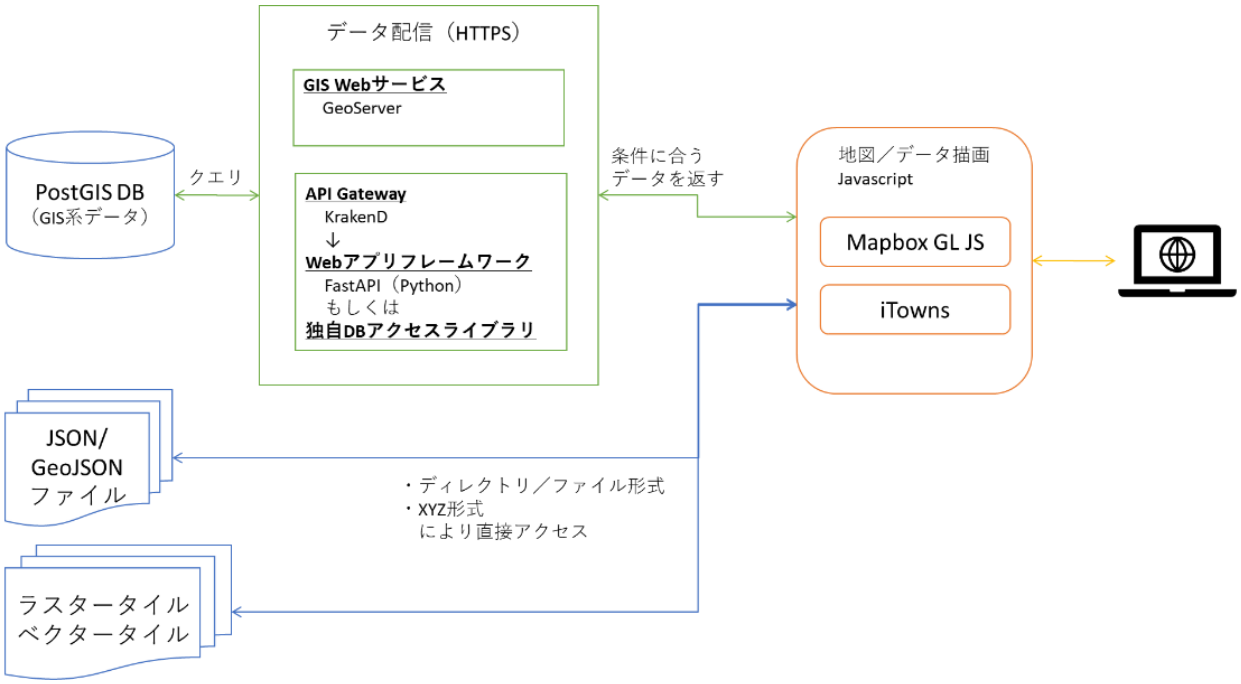


図 2-1　システムの全体構成

本システムは、主にサーバサイドでのデータ及びファイル管理機能、ユーザのデバイス上で動作する WebGIS アプリケーション、データ配信用の WebAPI を提供する Web サーバで構成されます。

2.1.2　動作環境

- 本システムが動作する Web サーバの OS は Ubuntu 20.04 LTS を使用します。

- Web クライアントの動作環境は次の通りとします。
  - Windows, Mac, Linux（Ubuntu と CentOS）
  - 特定のプラグイン (Java, Flash, .NET 等) のインストールを要さず、主要な Web ブラウザ（Chrome, Edge, Firefox, Safari）

2.1.3　制約事項

- 一部の Web 公開されているデータについては https により URL アクセスして参照します。

3　ソフトウェア機能

3.1　データ管理システム

本システムの対象データを管理するため、データの特性に応じて DBMS、GeoServer、Web サーバを使い分ける仕組みとします。

- DBMS

  PostgreSQL v12＋PostGIS v3.1.4 を利用します。

- GeoServer

  GeoServer のバージョンは 2.20.1 とします。WMTS による配信を行う GeoTIFF 形式データについて、GeoServer にデータを登録して配信します。また、DBMS で管理するデータについては GeoServer へ Data Store として登録の上、配信します。

- Web サーバ

  JSON、GeoJSON ファイル、XYZ 形式による配信を行うラスタータイル(標高タイルを含む)、ベクタータイルについて、ファイル/ディレクトリ形式で直接アクセス可能な形で配信します。

3.1.1　テーブル構成

DBMS で管理するデータについては地図レイヤとテーブルが 1 対 1 になる構成を標準とします。また、地理情報については PostGIS の提供する GIS オブジェクトで定義することを標準とします。

3.1.2　ファイル管理構成

ファイル/ディレクトリ形式で直接アクセス可能な形で配信するデータについて、XYZ タイル形式などのデファクトスタンダードに従い、使用性を考慮した構成で管理します。

XYZ タイル形式については以下を参照下さい。

国土交通省 国土地理院：https://maps.gsi.go.jp/development/siyou.html

3.1.3　データ取集仕様

本アプリで用いるデータは、https://github.com/nict-testbed-dalab ： Data_visualization_Component リポジトリ DataCatalog.pdf を参照してください。必要なデータは各自で取集し、提供元の利用規約に従ってください。

3.1.4　ユーザ管理

a　システム環境

Auth0 が提供するクラウドサーバ（バックグラウンドは AWS の日本国内サーバ）を使用します。

b　管理者アカウント

・アカウント作成時にメールアドレスが必要です
・二要素認証を実施します。

c　設定内容
・本システムの URL などを設定します。
・本システムを利用するユーザを登録します。Web アプリからは登録不可としています。
　ユーザ情報はメールアドレスとユーザ ID およびパスワードです。

3.2 データ取得 WebAPI

3.2.1　Web サーバ構成

Web サーバの構成について下図に示します。クライアントからは、Https のみのアクセスとし、WebAPI へのアクセスは Apache の ReverseProxy を利用して行います。また多様なサービス及び内部ストレージからデータを取得する必要があることから、小さな API を多数用意する必要があります。それを踏まえ、汎用的な設計を実現するため WebAPI ゲートウェイである krakenD を採用します。

データベースアクセスも API 化するため、FastAPI を採用し WebAPI として実装します。

一部の地図は外部サイトからの参照としますが、本システムで利用する際は WebAPI 経由でのアクセスとします。

API 利用ユーザ制御については 3.2.3 で記述しますが、クラウドサービスである Auth0 を利用してユーザ単位での API 制御を実現します。

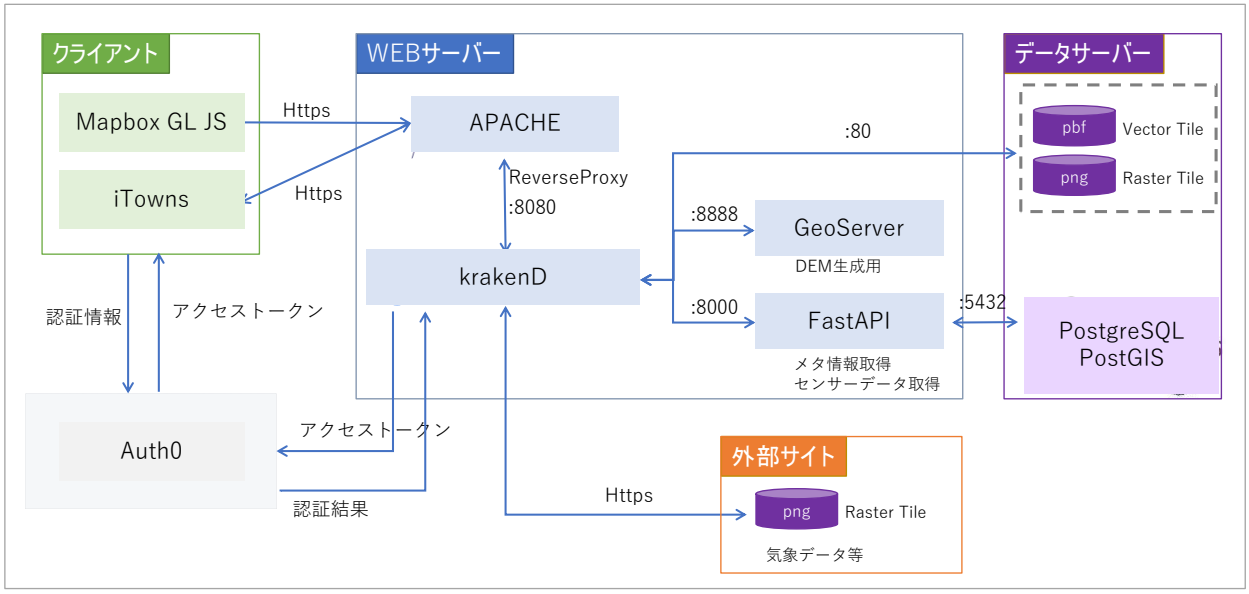


図 3-1 Web サーバ構成図

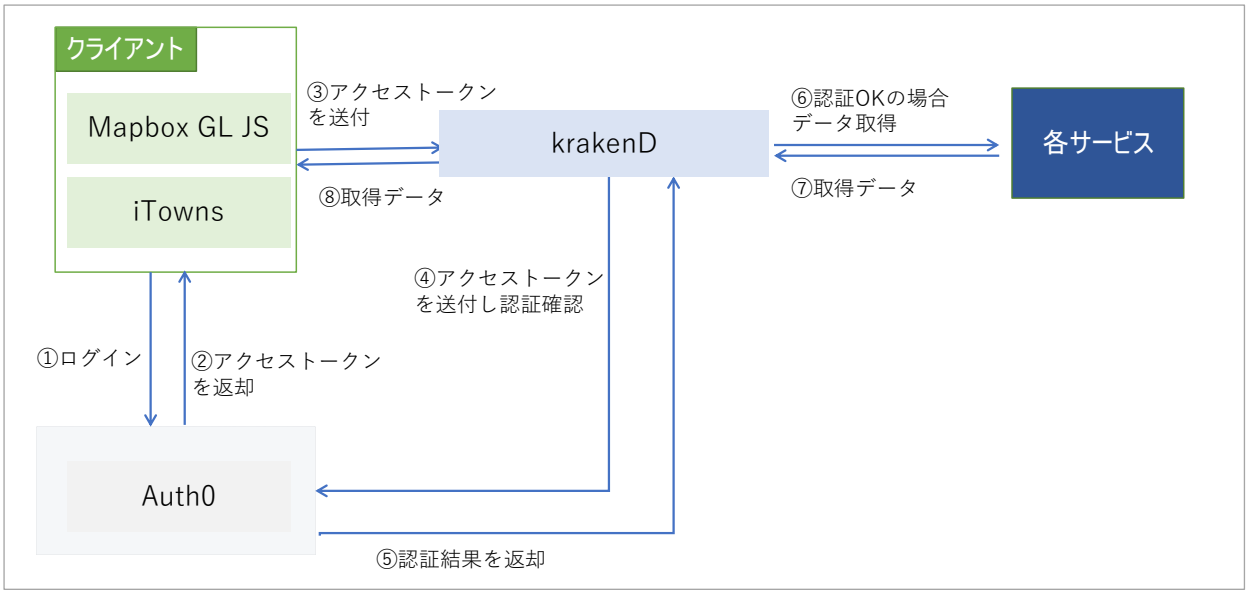
3.2.2　API 種別

API 種別としては、REST API を採用します。

3.2.3　API 利用ユーザ制御

API 利用ユーザ制御を行う際に、各 API で認証を実装するよりも、全 API を統括する krakenD 上で実施するのが理想的であるため、クラウドサービスとなるが krakenD との相性や利用実績から Auth0 でのユーザ管理を採用します。

Auth0 を使用した際の認証フローを下図に示します。実際には 3.2.1 で記述したとおり Apache による ReverseProxy にてアクセスしますが、説明上 Apache との通信部分は省略しています。



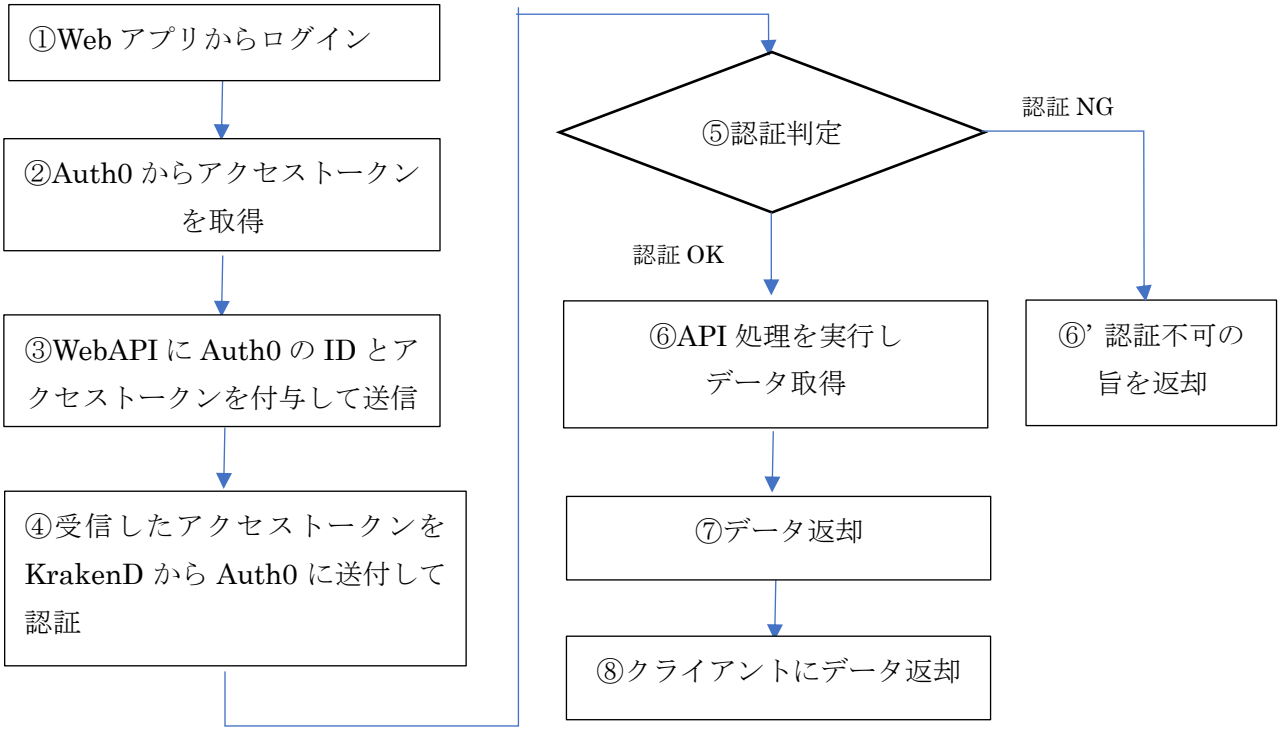


図 3-2 WebAPI 認証フロー

3.3 データ分析・可視化機能

3.3.1　2 次元及び 2.5 次元 WebGIS エンジン

クライアントアプリには MapboxGL を採用します。バージョンについては 2.0 からオープンソースでなくなったことから、オープンソース最終版である 1.13 を採用します。

サンプル Web アプリケーションを作成するにあたり必要な機能は下記の通りです。

- ログイン機能
  - ✓ データ取得 API にアクセスするための認証機能を作成します
- 地図操作機能
  - ✓ 拡大、縮小、移動などの基本地図操作機能は MapboxGL の標準機能を利用します
- 表示データ切り替え機能
  - ✓ 開発ライブラリであるデータ選択メニューを利用します
- グラフ表示機能
  - ✓ 開発ライブラリである立体グラフ表示ライブラリを利用します
- 時系列変化表示機能
  - ✓ 開発ライブラリであるタイムスライダー及び時空間同期機能を利用します
- URL 連動機能
  - ✓ 開発ライブラリである ViewURL を利用します
- 2.5 次元表示機能
  - ✓ MapboxGL の標準機能を利用します

3.3.2　3 次元 WebGIS エンジン

3 次元 Web エンジンとして、iTowns を用います。

バージョンは着手時点の最新版である 2.35.0(2021-09-16)を標準とし、iTowns の公式リポジトリ(https://github.com/iTowns/itowns)から取得するものとします。

- iTowns の利用方法(サーバ環境への導入)について

  iTowns の公式リポジトリ内(https://github.com/itowns/itowns#how-to-use)の記述に従い、npm（Node Package Manager）により導入する形を標準とします。npm について、サーバ環境へ Node.js をインストールし、同梱のものを利用するものとします。
- iTowns の依存ライブラリについて
  - https://github.com/iTowns/itowns/blob/v2.35.0/package.json
  - https://github.com/iTowns/itowns/blob/v2.35.0/package-lock.json

  を参照(バージョン部分は更新に合わせて参照ください)。

3.3.3　ライブラリ

データ分析・可視化機能を実現するため、各種ライブラリを実装しています。それぞれのライブラリは以下の機能を備えています。ライブラリのソースコードや組み込み方等はGitHub で公開しています。

https://github.com/nict-testbed-dalab

1. タイムスライダー(Timeline)
   - スライダーUI
     - ✓ 横方向の時間軸バーおよび摘みを描画します。
   - 現在時刻表示機能
     - ✓ 時間軸バーの摘み位置により、現在時刻を示す。またデジタル形式の時計でも現在時刻を表示します。
   - 現在時刻変更機能
     - ✓ 摘みの位置で Web アプリの現在時刻を示し、左右にドラッグすることで現在時刻を変更します。
   - 時間軸バー縮尺変更
     - ✓ 時間軸バーでホイールを操作すると時間軸バー目盛の縮尺を変更します。
   - 時間軸バー両端時刻変更機能
     - ✓ ボタン操作により時間軸バーの左右の端が示す日時を変更します。
   - 自動現在時刻変更機能
     - ✓ ボタン操作により現在時刻を指定の間隔で自動的に変更し、それに合わせ時間軸バーのつまみ、デジタル形式の時刻表示も変更します。

2. 時空間同期機能(STARScontroller)
   - 日時情報同期機能
     - ✓ 別ウィンドウで表示された 2 つの WebGIS アプリにおいて、操作中のアプリが保持する現在時刻及び、タイムスライダーの時間軸バーの両端の時刻を、他方のアプリへ同期します。
   - 空間情報同期機能
     - ✓ 別ウィンドウで表示された 2 つの WebGIS アプリにおいて、操作中のアプリの地図の中心位置、左上端位置、右上端位置、右下端位置、左下端位置及び、方位情報、拡縮情報を他方のアプリへ同期します。

3. 360 度画像保存(CreateSphereMap)
   - 360 度画像用サブビューの表示/非表示切り替え機能
     - ✓ バックグラウンドでキューブマップを作成しているサブビューを表示します。

- 360 度画像作成機能
  - ✓ 360 度画像の作成を開始します。作成が終了すると、自動的にダウンロードを行います。
- 360 度画像の解像度設定機能
  - ✓ 解像度を設定できます。単位は[px]で、最低 0[px]、最大解像度は 1024[px]
  - ✓ となります。
  - ✓ 最終的にできる全天球画像は、解像度を s とすると、4s×2s となります。
- 360 度画像用サブビュー作成の待ち時間設定機能
  - ✓ キューブマップ作成時、現在の視点での読み込み予定のタイル画像がサーバ上に存在しない場合の、画像取得のタイムアウト時間を設定できます。単位は[秒]です。

4. ViewURL
   - WebGIS アプリが表示している時空間情報や可視化されたデータの表示状況を文字情報に変換し、URL に付与することで一時記録する機能
   - ViewURL で変換された、時空間情報や可視化されたデータについても文字列を読み取り、WebGIS アプリの表示へ復元する機能

5. 連続画像キャプチャ(DisplayCapture)
   - WebGIS アプリの時間情報を一定間隔で変更し、時間変更ごとに画面キャプチャを 1 回取得する機能
   - 対応する Web ブラウザは Google Chrome

6. 点群表示ライブラリ(PointCloud)
   - 複数の点群データを WebGIS アプリ上に可視化表示する機能
     - ✓ 一部の値が欠損したデータをエラーなくも取り扱えるよう実装します。
     - ✓ 時刻、位置情報、1 種類以上の数値情報、または属性等のメタ情報を有する点群データを取り扱えるよう実装します。
     - ✓ Potree 等の階層型データ構造に対応します。
     - ✓ 可視化対象のデータ数は無制限に扱えるように実装します。
   - WebGIS アプリの現在時刻に合わせた点群データを可視化表示する機能
   - 点群データを DBMS から WebAPI にから取得する機能。
   - 点群データの数値や属性、及びデータの有無により、各面の表示・非表示を切り替える機能
   - 点群データの数値に応じて各点のテクスチャ色を変更する機能
   - 点群と他のポリゴン系データ（点、線、面）の相対的位置関係を判定する機

能

7. 面群表示ライブラリ(PlaneGroup)
    - 複数の面群データ（エリア情報）を WebGIS アプリ上に可視化表示する機能。
        - ✓ 一部の値が欠損したデータをエラーなくも取り扱えるよう実装します。
        - ✓ 時刻、位置情報、1 種類以上の数値情報または属性等のメタ情報を有するデータを取り扱えるように実装します。
        - ✓ 可視化対象のデータ数は無制限に扱えるように実装します。
    - WebGIS アプリの現在時刻に合わせた面群データを可視化表示する機能
    - 面群データを DBMS から WebAPI により取得する機能
    - 面群データの数値や属性、及びデータの有無により、各面の表示・非表示を切り替える機能
    - 面群データの数値に応じて各面のテクスチャ色及び透明度を変更する機能
        - ✓ 複数の面群をオーバーレイ表示できる機能を実装します。
        - ✓ 面群を一つにまとめることで、対象エリア全体をヒートマップの様なグラデーションによる表現ができるように実装します。
        - ✓ 可視化対象のデータ数は無制限に扱えるように実装します。
    - 面群とポリゴン系データ（点、線、面）の相対的位置関係を判定できる機能
    - 面群は任意サイズの矩形や円（楕円を含む）等の標準的な形状で表示する機能
        - ✓ 緯度・経度、距離（㎞など）から選択できるように実装します。

8. 立体グラフ表示ライブラリ(3D_Graph)
    - 空間分布する時系列データを WebGIS アプリ上で立体的に表示する機能
    - 立体グラフの高さやテクスチャ色（濃淡を含む）がデータに応じて設定される機能
    - デザインをユーザが選択する機能
        - ✓ 高さや太さを選択できるように実装します。
        - ✓ デザインやテクスチャ色を選択する UI（メニュー）を実装します。

9. 3 次元オブジェクトのテクスチャ変更機能(3D_Object)
    - WebGIS アプリの 3 次元オブジェクトについてはテクスチャ色（濃淡を含む）をユーザが選択できる機能
        - ✓ テクスチャを選択するための UI を実装します。

3.4 テンプレート WebGIS アプリケーション

テンプレート WebGIS アプリケーション(以下「テンプレート WebGIS アプリ」)として、本システムの対象データを時系列並びに 2 次元（または 2.5 次元）及び 3 次元で可視化します。

3.4.1 機能項目

- テンプレート WebGIS アプリに実装する機能を以下に記載します。

1. WebGIS の標準機能
   - ✓ ラスタデータの可視化機能
   - ✓ ベクタデータの可視化機能
   - ✓ ベクタデータの属性表示機能
   - ✓ 表示位置及び縮尺の制御機能(マウスによるビュー操作及びスライダー等の GUI 操作)
   - ✓ 表示方向(2D では表示の方角のみ、2.5 次元及び 3 次元では視線のチルトを含めた回転)の制御機能(マウスによるビュー操作及びスライダー等の GUI 操作)
   - ✓ ランドマークへの表示位置移動機能

2. 時系列データの可視化機能
   - ✓ タイムスライダーライブラリを iTwons 及び Mapbox に組み込み、タイムスライダーの現在時刻によりテンプレート WebGIS アプリの現在時刻を定義する。対象データのうち時系列データについて、現在時刻に対応したデータを可視化する。
   - ✓ タイムスライダーが備える自動現在時刻変更機能と連動し、テンプレート WebGIS アプリで可視化する時系列データの時刻を変更する。

3. ユーザ操作による時系列データ表示時刻の指定機能
   - ✓ ユーザはタイムスライダーを操作することにより時系列データ表示時刻の指定できるものとする、ユーザが指定した時刻の設定に従い、テンプレート WebGIS アプリで時系列データを可視化する。

4. 複数データの地図上への表示／非表示の選択機能
   - ✓ テンプレート WebGIS アプリでは任意の対象データを可視化できるものとし、その一覧をユーザが操作可能なレイヤツリー又はレイヤリスト(以下「レイヤパネル」)の形でテンプレート WebGIS アプリ内に一覧表示する。
   - ✓ レイヤパネルに一覧表示された対象データはそれぞれに表示／非表示

をコントロールするための UI(チェックボックスまたは選択可能リスト)を装備する。

- ✓ テンプレート WebGIS アプリはレイヤパネルで表示設定された複数のレイヤを重畳表示する。

3.5 データ前処理機能

3.5.1 時間粒度変換

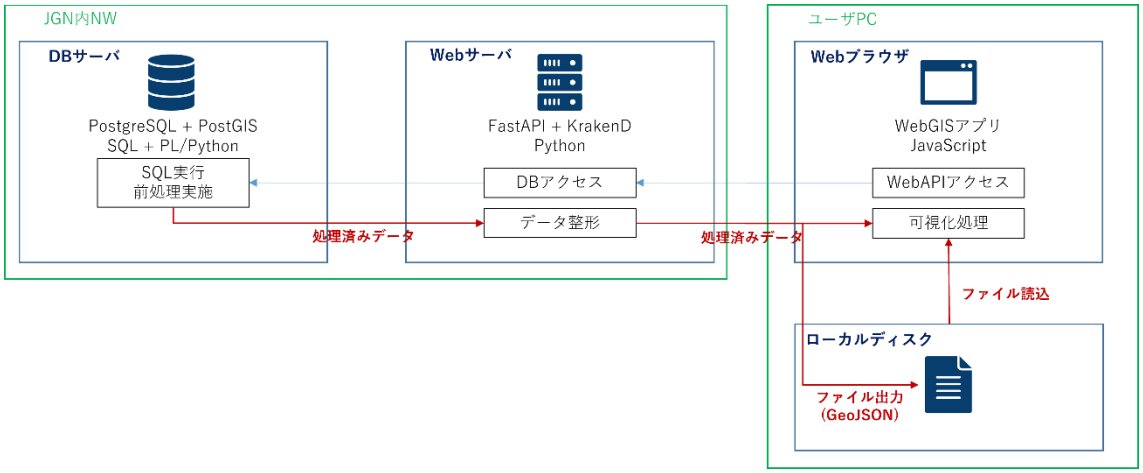
a 時系列データを秒、分、時、日、月、年単位等に変更できる機能。

b 秒、分、時、日、月、年単位にデータをリサンプルする手法として、平均値、最大値、最小値、中央値などの手法を選択する機能。

3.5.2 欠損値データの補完

a データ内に存在する欠損値データを、平均値、最大値、最小値、線形補完等の主要な方法で補完する機能。

データ前処理機能のデータフローは以下の通りです。



データ前処理機能として、データ分析・可視化システムの DB サーバ、Web サーバ、WebGIS アプリそれぞれに以下の機能を追加することで実装します。

・DB サーバ

－位置情報が固定されている測定データに対するデータ補完、集約処理

－時間で位置情報や値が変化するデータに対するデータ補完、集約処理

なお、それぞれの処理は PostgreSQL のユーザ定義関数(UDF: User-defined function) として、PL/Python を利用して実装します。

（参考）11.UDF の追加と削除

利用する RDBMS

PostgreSQL v12 + PostGIS v3.1.4

提供元(URL)

PostgreSQL：https://www.postgresql.org/

PostGIS：https://postgis.net/

・Web サーバ
－DB サーバ上で実装された、データ補完、集約処理を呼び出し、返却される前処理済みのデータを WebGIS アプリが読み込む形式へ整形
　なお、それぞれの処理は PostgreSQL のユーザ定義関数として、PL/Python を利用して実装します。

・利用する Web アプリフレームワーク
　FastAPI
　　提供元(URL)
　　https://fastapi.tiangolo.com/

・WebGIS アプリ
－データ補完、集約処理を呼び出す際の、時間粒度、処理手法を選択する UI
－前処理済みのデータのファイル出力機能、及び外部のファイル読み込み機能
－前処理済みのデータの可視化機能

返却される前処理済みのデータを WebGIS アプリが読み込む形式へ整形
　なお、それぞれの処理は PostgreSQL のユーザ定義関数として、PL/Python を利用して実装いたします。

・利用する 2.5 次元 GIS Web サービス
　Mapbox GL JS
　　提供元(URL)
　　https://www.mapbox.com/

・3 次元 GIS Web サービス
　iTowns
　　提供元(URL)
　　https://www.itowns-project.org/

4　サーバ環境構築

本章の構築をすると、Web サーバの構築ができます。その上で、8 章の WebGIS アプリケーションを参考に、各アプリを展開することで、背景地図の表示など最低限の機能を有したアプリを立ち上げることができます。

本システムは以下の構成で実装しています。構成例の一つとしてご参考ください。

仮想マシン名	用途
tb-gis-web	公開用 Web サーバ
tb-gis-proc	SSH フロントサーバ兼 データ収集処理サーバ
tb-gis-db	ストレージ兼 DB サーバ

4.1 OS インストール

本システムで使用する OS は Ubuntu 20.04.3 LTS とします。

4.1.1　Ubuntu のインストール手順

(1)　Ubuntu を以下からダウンロードする。
https://jp.ubuntu.com/download

(2)　インストールした Ubuntu を DVD などのメディアに書き込む。

(3)　作成したインストールディスクを DVD ドライブにセットしてコンピュータを再起動する。

(4)　インストーラーで使用する言語の確認画面が表示される。日本語未対応のため、[English]を選択する。

(5)　すでにインターネットに接続されている場合、[Update to the new installer]を選択する。
インターネットに接続されていない場合、[Continue without updating]を選択する。

(6)　キーボード選択画面が表示されるため[Layout]と[Variant]に[Japanese]を設定し、[Done]を選択する。

(7)　ネットワーク設定画面が表示される。DHCP で自動的に IP アドレスを取得する場合は[Done]を選択する。
固定で IP アドレスを設定する場合、「Tab」または「矢印」キーで設定を行いたい NIC を選択し、「Enter」キーを押下する。
ネットワーク設定画面が表示されるため、任意の設定値を入力する。

(8)　プロキシ設定画面が表示されるため、プロキシサーバの設定が必要な場合は[Proxy adress]の項目にプロキシサーバのアドレスを設定する。

設定した場合、あるいは特にプロキシの設定が必要ない場合は[Done]を選択する。

(9) ミラーサーバ設定画面が表示される。ミラーサーバは自動的に近くの物が選択されていることを確認し、[Done]を選択する。

(10) ストレージ設定画面が表示されるため、以下を設定し、[Done]を選択する。

(11) インストール確認画面が表示されるため、[Continue]を選択する。

(12) ユーザ登録画面が表示されるため、以下にサーバの設定を入力する。

Your server's name：サーバのホスト名

Pick a username：ログインユーザ名（デフォルトユーザは”ubuntu”）

Choose a password：ログインパスワード

Confirm your password：パスワードの再入力

(13) SSH サーバインストール確認画面が表示されるため、SSH 接続で接続する場合は[Install OpenSSH server]をチェックし[Done]を選択する。

(14) インストールパッケージ選択画面が表示される。インストールは不要なため[Done]を選択する。

(15) インストールとセキュリティアップデート完了後、[Reboot]を選択してサーバを再起動する。

4.2 OS 初期設定

各ホストの OS 初期設定（共通設定）は以下の通りです。適宜、各自の環境に合わせて設定してください。

```
【最新化】
# apt update
# apt upgrade
# apt autoremove
# reboot

【ホスト名設定】
# hostnamectl set-hostname ホスト名

【NTP 同期】
# vi /etc/systemd/timesyncd.conf
最終行に追記
NTP=ntp.nict.jp
# systemctl restart systemd-timesyncd
# timedatectl timesync-status

【ssh-server インストール・設定】
# apt install openssh-server
# vi /etc/ssh/sshd_config
PermitRootLogin no
```

```
PasswordAuthentication no
# systemctl restart ssh

【ユーザ追加】
# adduser ユーザ名
管理者権限付与
# usermod -G sudo ユーザ名
# passwd（パスワード入力）

【公開鍵登録】
# su – ユーザ名
$ mkdir .ssh
$ chmod 600 .ssh
$ vi .ssh/authorized_keys
$ chmod 700 .ssh/authorized_keys

【SSH の IP アドレス制限】
# vi /etc/hosts.deny
sshd: ALL
# vi /etc/hosts.allow
sshd: xxx.xxx.xxx.xxx

【アンチウィルスソフトウェア（Clam AntiVirus）】
# apt install clamav
# sed -i -e "s/^NotifyClamd/#NotifyClamd/g" /etc/clamav/freshclam.conf
# systemctl stop clamav-freshclam
# freshclam
# systemctl start clamav-freshclam

【拡張ストレージマウント】（tb-gis-db のみ）
# parted /dev/sdb
# mkfs.ext4 /dev/sdb1
# blkid /dev/sdb1
# vi /etc/fstab
UUID="89e901b8-23cb-40f1-978d-c633390432cb"   /storage   ext4   defaults   0   0
# mkdir /storage
# mount /storage

【NFS サーバ】（tb-gis-db のみ）
# apt install nfs-kernel-server
# vi /etc/exports
/storage      xxx.xxx.xxx.xxx (rw,no_root_squash)
# systemctl restart nfs-server

【NFS クライアント】（tb-gis-db 以外）
# apt install nfs-common autofs
```

```
# vi /etc/auto.master
/-      /etc/auto.mount
# vi /etc/auto.mount
(tb-gis-proc)
/storage     -fstype=nfs,rw    tb-gis-db:/storage
(tb-gis-web
/storage     -fstype=nfs,ro    tb-gis-db:/storage
# systemctl restart autofs
# ls /storage

【サーバファイアウォール】(tb-gis-web)
# ufw status
状態: 非アクティブ

# ufw allow ssh
ルールをアップデートしました
ルールをアップデートしました(v6)
# ufw allow from xxx.xxx.xxx.xxx to any app 'Apache Full'
# ufw allow from    xxx.xxx.xxx.xxx to any app 'Apache Full'
('Apache Full'は 80,443/tcp)
ルールをアップデートしました

# ufw enable
ファイアウォールはアクティブかつシステムの起動時に有効化されます。
# ufw reload

# ufw status numbered
状態: アクティブ
        To                          Action      From
        --                          ------      ----
[ 1] 22/tcp                       ALLOW IN    Anywhere
[ 2] Apache Full                  ALLOW IN    xxx.xxx.xxx.xxx
[ 3] Apache Full                  ALLOW IN    xxx.xxx.xxx.xxx
[ 4] 22/tcp (v6)                  ALLOW IN    Anywhere (v6)
```

4.3 Apache2

4.3.1　インストール

本マニュアルでは、Apache2 2.4 を使用した説明となります。

・下記コマンドを実行します。

```
sudo apt install apache2
```

・下記のような実行結果が表示されます。

```
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /
lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.
service → /lib/systemd/system/apache-htcacheclean.service.
ufw (0.36-6ubuntu1) のトリガを処理しています ...
systemd (245.4-4ubuntu3.13) のトリガを処理しています ...
man-db (2.9.1-1) のトリガを処理しています ...
libc-bin (2.31-0ubuntu9.2) のトリガを処理しています ...
webgis@tb-gis-web:~/web_mapbox$
```

・使用するモードを通常と SSL の双方にします

```
sudo ufw allow 'Apache Full'
```

・ステータスを確認します

```
sudo systemctl status apache2
```

下記のような実行結果が表示され、3 行目が「active(running)」と表示されていることを確認します。

・

```
webgis@tb-gis-web:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-01-24 12:57:20 JST; 1 weeks 1 days ago
     Docs: https://httpd.apache.org/docs/2.4/
  Process: 85237 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
 Main PID: 8187 (apache2)
    Tasks: 55 (limit: 19087)
   Memory: 26.1M
   CGroup: /system.slice/apache2.service
           ├─ 8187 /usr/sbin/apache2 -k start
           ├─89404 /usr/sbin/apache2 -k start
           └─89405 /usr/sbin/apache2 -k start

1月 28 00:00:16 webgis-tb-gis-web systemd[1]: Reloading The Apache HTTP Server.
1月 28 00:00:16 webgis-tb-gis-web systemd[1]: Reloaded The Apache HTTP Server.
1月 29 00:00:04 webgis-tb-gis-web systemd[1]: Reloading The Apache HTTP Server.
1月 29 00:00:04 webgis-tb-gis-web systemd[1]: Reloaded The Apache HTTP Server.
1月 30 00:00:08 webgis-tb-gis-web systemd[1]: Reloading The Apache HTTP Server.
1月 30 00:00:08 webgis-tb-gis-web systemd[1]: Reloaded The Apache HTTP Server.
1月 31 00:00:03 webgis-tb-gis-web systemd[1]: Reloading The Apache HTTP Server.
1月 31 00:00:03 webgis-tb-gis-web systemd[1]: Reloaded The Apache HTTP Server.
2月 01 00:00:14 webgis-tb-gis-web systemd[1]: Reloading The Apache HTTP Server.
2月 01 00:00:14 webgis-tb-gis-web systemd[1]: Reloaded The Apache HTTP Server.
webgis@tb-gis-web:~$
```

)

・ブラウザから確認します。
　ブラウザに WEB サーバの IP アドレスを入力し、下記のような Apache2 の初期画面が表示されていればインストールできています。

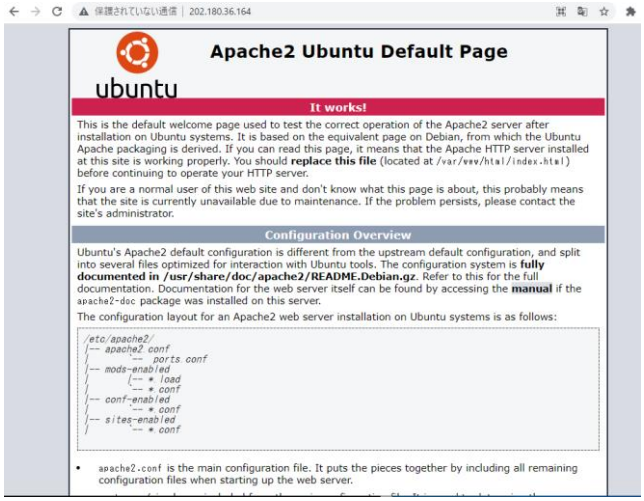


図 4-1 Apache2 の初期画面

4.3.2　各ミドルウェアへのリバースプロキシ設定

・下記コマンドを実行してプロキシを有効にします。

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod headers
```

・conf ファイルに追記します。

```
sudo vi /etc/apache2/sites-available/mapbox.conf
```

```
-----------------------------------------------------------------
<Proxy *>
    Require all granted
</Proxy>
ProxyRequests Off
ProxyPreserveHost On
ProxyPass /api http://localhost:8081/api
ProxyPassReverse /api http://localhost:8081/api
ProxyPass /geoserver http://localhost:8085/geoserver
ProxyPassReverse /geoserver http://localhost:8085/geoserver
-----------------------------------------------------------------
```

・Apache を再起動します。

```
sudo systemctl restart apache2
```

4.3.3 WEB サーバ証明書の設定

・Cerbot のインストール

```
sudo apt install certbot python3-certbot-apache
```

・mod_ssl を有効化

```
sudo a2enmod ssl
```

・SSL 証明書の取得

```
sudo certbot --apache
```

・conf ファイルのコピー

```
cd /etc/apache2/sites-available
sudo cp default-ssl mapbox.conf
```

・上記 conf を有効にする

```
sudo a2ensite mapbox.conf
```

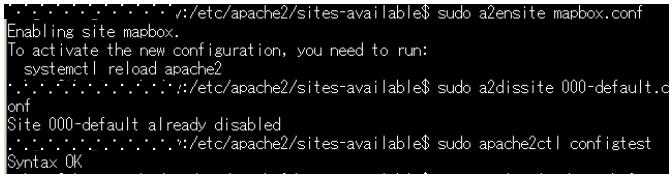
・他の有効になっている conf を無効にする

```
sudo a2dissite default-ssl.conf
sudo a2dissite 000-default.conf
```

・設定の確認

```
sudo apache2ctl configtest
```

下記のように「OK」が表示されることを確認します。



・apache2 の再起動

```
sudo systemctl restart apache2
```

・ブラウザの確認

https にて警告メッセージなく表示されることを確認します。

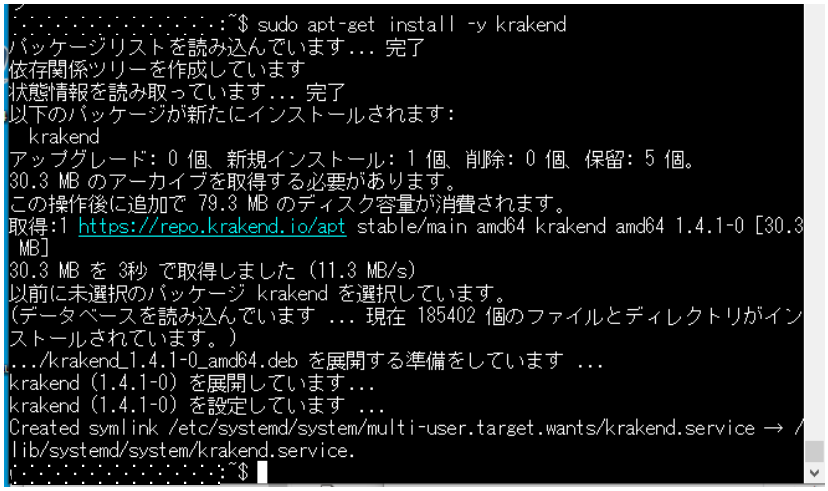
4.4 KrakenD

4.4.1　インストール

作業ディレクトリはログインユーザのディレクトリとします。
下記コマンドを実行します。
（1，２行目：krakend のキーの登録、３行目：パッケージリストの更新）

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 5DE6FD698AD6FDD2
sudo echo "deb https://repo.krakend.io/apt stable main" | sudo tee /etc/apt/sources.list.d/krakend.list
sudo apt-get update
sudo apt-get install -y krakend
```

インストールすると、下記のようなメッセージが表示されます。



4.4.2　ディレクトリ作成

```
mkdir krakend
```

4.4.3　設定ファイルの配置

下記の URL から設定ファイル krakend.json を上記ディレクトリに配置します。
https://github.com/nict-testbed-dalab/Data_visualization_Component/blob/main/configurationSample/ENVIRONMENT/krakend.json

krakend.json の中身は 10.1 を参照してください。

4.4.4　公開サーバの変更

設定ファイルに記載されている下記 URL は公開サーバの URL に変更してください。
https://tb-gis-web.jgn-x.jp

4.4.5　KrakenD のポート設定

WebAPI は 5000 番ポート、GeoServer は 8085 番ポートで起動していない場合は下記記述のポート番号を変更してください。

localhost:5000
localhost:8085

4.4.6　KrakenD の認証設定

認証を使用しない場合は上記記述を削除してください。（複数個所あります）

```
"extra_config": {
        "auth/validator": {
                "alg": "RS256",
                "audience": ["https://tb-gis-web.jp.auth0.com/api/v2/"],
                "jwk_url": "https://tb-gis-web.jp.auth0.com/.well-known/jwks.json"
        }
},
```

4.4.7　起動

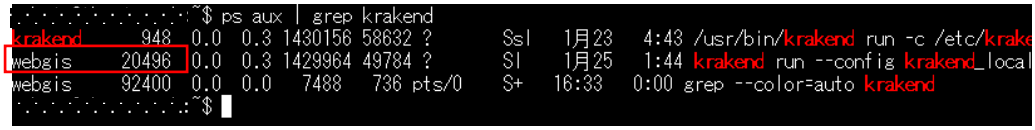
下記コマンドを実行し、バックグラウンドで実行します。

```
cd krakend
nohup krakend run --config krakend.json &
```

4.4.8　停止

下記コマンドを実行し、実行プロセスを表示します。

```
ps aux | grep krakend
```



次に、一覧の「krakend run」のプロセス ID を参照し、下記コマンドを実行します。
（プロセス ID は起動ごとに異なります。）

```
kill 20496
```

4.5 Node.js

Node.js は、iTowns アプリのインストールする際に、必要となります。

4.5.1 nodejs インストール実行

```
sudo apt install nodejs
```

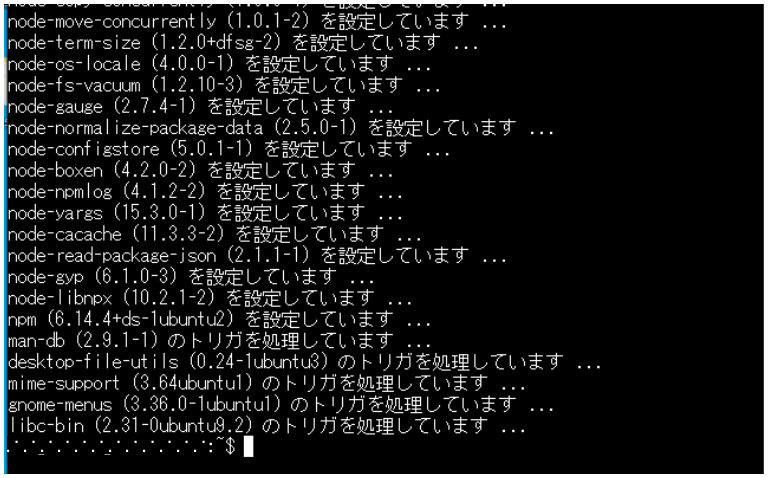
下記のようなメッセージが表示されます。

バージョン情報を表示し、インストールされたことを確認します。

```
node --version
```

4.5.2 npm インストール実行

```
sudo apt install npm
```



バージョン情報を表示し、インストールされたことを確認します。

```
npm --version
```

これで、Web サーバの構築は完了となります。すべての機能を利用したい方は、次の章へお進みください。もしくは、8 章の WebGIS アプリケーションをご参照することで、背景地図の表示など、最低限の機能のみで各アプリを実装できます。

5　GeoServer

5.1 GeoServer 及び依存パッケージについて

GeoServer は Web ブラウザなどが利用する地理情報を配信する Java アプリケーションです。ここでは、GeoServer インストールおよび併せて必要な OpenJDK、Tomcat のインストールについて説明します。

5.2 パッケージインデックスの更新

GeoServer 及び依存パッケージは Web サーバ(tb-gis-db.xxx.jp)へインストールします。次のコマンドを実行して下さい。

(1) パッケージインデックスを更新

```
sudo apt -y update
```

パッケージインデックスについて、GeoServer 以外の構築で直前に更新している場合は実行不要です。

5.3 OpenJDK のインストール

次のコマンドを実行して下さい。

(1) OpenJDK をインストール

```
sudo apt install default-jre
```

5.4 Tomcat のインストール

次のコマンドを実行して下さい。

(1) Tomcat をインストール

```
sudo apt install tomcat9 tomcat9-admin
```

(2) サービスを有効にします。

```
sudo systemctl enable tomcat9
```

5.5 Tomcat の設定変更

設定ファイルの編集は以下のコマンドで開始することができます。

(1) 設定ファイルを変更

```
sudo nano /etc/tomcat9/tomcat-users.xml
```

https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリの ConfigurationSample に含まれる tomcat-users.xml を参考に設定を変更して下さい。以下の内容を追記します。

```
<role rolename="admin-gui"/>
<role rolename="manager-gui"/>
<user username="tomcat" password="pass" roles="admin-gui,manager-gui"/>
```

5.6 Tomcat の開始

次のコマンドを実行して下さい。

(1) 起動

```
sudo systemctl restart tomcat9
```

5.7 GeoServer のインストール

以下の 4 つのコマンドを実行し、GeoServer のインストールを行います。
また、本マニュアルでは、GeoServer2.20.1 を使用した説明となります。

(1) GeoServer パッケージ(ZIP ファイル)の取得

```
sudo wget https://sourceforge.net/projects/geoserver/files/GeoServer/2.20.1/geoserver-2.20.1-war.zip
```

(2) GeoServer パッケージ(ZIP ファイル)の配置

```
sudo mv geoserver-2.20.1-war.zip /var/lib/tomcat9/webapps
```

(3) 配置ディレクトリへ移動

```
cd /var/lib/tomcat9/webapps
```

(4) GeoServer パッケージ(ZIP ファイル)の展開

```
sudo unzip geoserver-2.20.1-war.zip
```

unzip がインストールされていない場合は、7.3.1 をご参照ください。

5.8 GeoServer のポート設定(Apache の設定変更)

GeoServer がデフォルトで利用するポートは 8080 ですが、本システムでは 8085 を使用します。Apache の設定ファイル(/etc/apache2/sites-enabled/default.conf)を編集します。

設定ファイルの編集は以下のコマンドで開始することができます。

(1) 設定ファイルを変更

```
sudo nano /etc/apache2/sites-enabled/default.conf
```

https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリの ConfigurationSample に含まれる default.conf を参考に設定を変更して下さい。以下の内容を追記します。

```
ProxyPass /geoserver http://localhost:8085/geoserver
ProxyPassReverse /geoserver http://localhost:8085/geoserver
```

ここまでの設定を反映するため、一度 Tomcat 、Apache を再起動します。以下の 2 つのコマンドを実行して下さい。

(1) Tomcat の再起動

```
sudo systemctl restart tomcat9
```

(2) Apache の再起動

```
sudo systemctl restart apache2
```

5.9 Tomcat の SSL 対応(証明書の作成と設定)

SSL 対応(https 対応)を行うため、証明書の作成と設定を行います。以下の 2 つのコマンドを実行して下さい。

(1) Tomcat の設定ディレクトリへ移動

```
cd　/var/lib/tomcat9/conf
```

(2) 証明書の作成

```
sudo keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.keystore -validity 3650
```

次に、作成した証明書の情報を Tomcat 設定に追記します。設定ファイルの編集は以下のコマンドで開始することができます。

(1) 設定ファイルを変更

```
sudo nano /etc/tomcat9/server.xml
```

https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリの ConfigurationSample に含まれる server.xml を参考に設定を変更して下さい。以下の内容を追記します。既存の 8443 ポート設定がある場合はコメントアウトが必要です。

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
        maxThreads="150" SSLEnabled="true" >
<SSLHostConfig>
      <Certificate certificateKeystoreFile="conf/tomcat.keystore"
                   certificateKeystorePassword="P@ssw0rd"
                   certificateKeyAlias="tomcat"
                   certificateKeystoreProvider="SUN"
                   certificateKeystoreType="JKS"
                   type="RSA" />
</SSLHostConfig>
</Connector>
```

5.10　GeoServer の名前解決設定

GeoServer の名前解決を行うため、以下の情報を GeoServer の設定に追記します。設定ファイルの編集は以下のコマンドで開始することができます。

(1) 設定ファイルを変更

```
sudo nano /var/lib/tomcat9/webapps/geoserver/WEB-INF/web.xml
```

https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリの ConfigurationSample に含まれる web.xml を参考に設定を変更して下さい。以下の内容を追記します。

```
<context-param>
  <param-name>PROXY_BASE_URL</param-name>
  <param-value>https://xxx.jp/geoserver</param-value>
</context-param>

<context-param>
  <param-name>GEOSERVER_CSRF_WHITELIST</param-name>
  <param-value>xxx.jp</param-value>
</context-param>
```

ここまでの設定を反映するため、Tomcat を再起動します。以下のコマンドを実行して下さい。

(1) Tomcat の再起動

```
sudo systemctl restart tomcat9
```

以下のコマンドを実行して正常なレスポンスが返れば、構築は成功です。Web ブラウザから https://localhost:8443/ へアクセスして、GeoServer の管理を行うことができます。

Geoserver はデフォルトではユーザ名：admin、パスワード：geoserver となっているため、適切な強度を持ったものを設定して下さい。

(1) GeoServer 管理インターフェースへのアクセス

```
curl -k https://localhost:8443/
```

図 5-2 GeoServer の管理インターフェース(ログイン前)

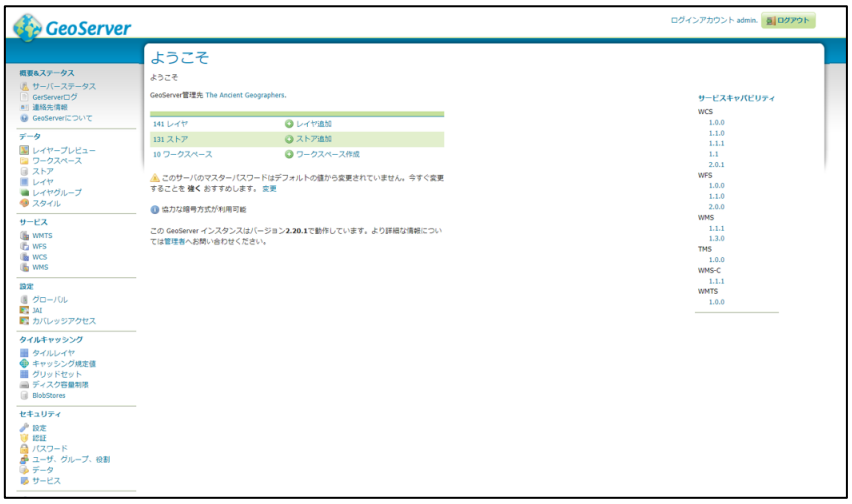


図 5-3 GeoServer の管理インターフェース(ログイン後)

6　WebAPI とデータベース

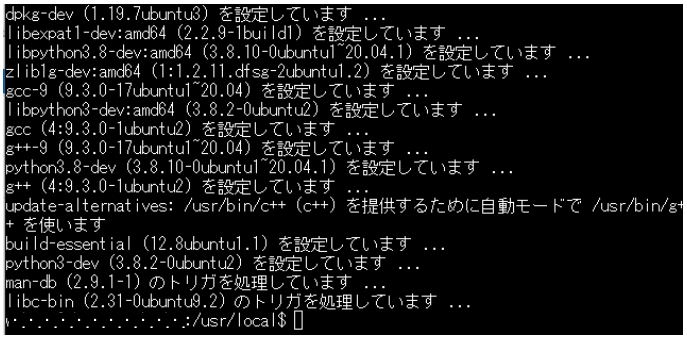
6.1 FastAPI

6.1.1　pip3（インストールされていない場合のみ）

下記コマンドを実行します。

```
sudo apt update
sudo apt install python3-pip
```

確認メッセージでは「Y」を入力します。
下記のような実行結果が表示されます。



6.1.2　各種フレームワーク

下記コマンドを実行し各種フレームワークをインストールします。

```
pip3 install fastapi
pip3 install db
pip3 install databases
pip3 install sqlalchemy
pip3 install starlette
pip3 install uvicorn
pip3 install python-dotenv
```

インストールされたことを確認します。

```
pip3 list
```

成功していれば、フレームワーク名が表示されます。

6.1.3　設定ファイルの設置

api ディレクトリ配下に下記コードを配置します。

https://github.com/nict-testbed-dalab/Data_api_Component

6.1.4　FastAPI のポート設定

.env ファイルのデータベース接続情報からポートなどの設定を変更できます。

DB_HOST=tb-gis-db　←ホスト名
DB_PORT=5432　←ポート
DB_NAME=tb ←データベース名
DB_USER=webgis　←ユーザ名
DB_PW=　←パスワード

6.1.5　起動

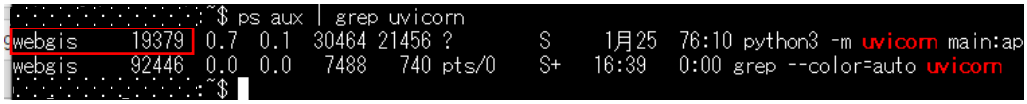
API のフォルダに移動し、uvicorn をバックグラウンドで実行します。
リロードモードでの実行のため、ソース修正時は自動的に反映されます。

```
cd /home/webgis/api
nohup python3 -m uvicorn main:app --reload --port 5000 &
```

6.1.6　停止

下記コマンドを実行し、実行プロセスを表示します。

```
ps aux | grep uvicorn
```



次に、一覧の「python3 -m uvcorn」のプロセス ID を参照し、下記コマンドを実行します。（プロセス ID は起動ごとに異なります。）

```
kill 19379
```

6.2 データベース

6.2.1　本システムで利用するデータベース

本システムでは、データベース(RDBMS)として PostgreSQL を利用し、PostgreSQL データベースで地理空間情報を扱うための拡張である PostGIS も併せて導入します。

本マニュアルでは、PostgreSQL12 を使用した説明となります。

また、DB へのデータ格納の具体例を 7.11 に記載しております。

6.2.2　PostgreSQL 12 のインストール

PostgreSQL は DB 兼ストレージサーバ(tb-gis-db.xxx.jp)へインストールします。次の 2 つのコマンドを実行して下さい。

(1) パッケージインデックスを更新

```
sudo apt -y update
```

パッケージインデックスについて、PostgreSQL 以外の構築で直前に更新している場合は実行不要です。

(2) PostgreSQL をインストール

```
sudo apt -y install postgresql-12 postgresql-client-12 postgresql-client-common
postgresql-common postgresql-contrib postgresql-12-postgis-3 postgresql-server-dev-12
postgresql-plpython3-12 libpq-dev
```

PostgreSQL 本体、及び必要な関連パッケージをインストールします。このコマンドで PostGIS(“… postgresql-12-postgis-3 …” と記述してある部分が該当箇所)も同時にインストールします。

6.2.3　パスワードの設定

PostgreSQL のインストール時に作成される postgres ユーザにパスワードを設定します。次のコマンドを入力し、パスワードを設定して下さい。以降ではここで設定するパスワードを” postgres”として記述します。実際のパスワードはセキュリティ確保のため、適切な強度を持ったものを設定して下さい。

(1) パスワードの設定

```
sudo passwd postgres
```

6.2.4　psql を使った PostgreSQL への接続

インストール、パスワード設定の確認として、次の 2 つのコマンドを実行して PostgreSQL への接続を確認して下さい。

(1) ユーザ切り替え

```
sudo -i -u postgres
```

(2) PostgreSQL への接続

```
psql
```

(3) パスワードの変更

```
ALTER ROLE postgres WITH PASSWORD 'postgres';
```

プロンプトの表示が”posgres=#”となっていれば接続は成功です。次のコマンドを実行して PostgreSQL から切断して下さい。

(4) PostgreSQL から切断

```
¥q
```

6.2.5　設定ファイルの適用とサービスの起動

設定ファイルの編集は以下のコマンドで開始することができます。
https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリの ConfigurationSample に含まれる pg_hba.conf を参考に設定を変更して下さい。
また、本マニュアルでは、nano エディタを使用し設定ファイルを編集しています。

(1) 設定ファイルを変更

```
sudo nano /etc/postgresql/12/main/pg_hba.conf
```

サービスの登録、起動、再起動、停止は以下のそれぞれのコマンドで行うことができます。上の設定ファイル変更を適用するため、サービスを再起動して下さい。再起動後、改めて 6.2.4 の手順で PostgreSQL への接続を確認できれば、データベースの構築は完了です。

(1) 登録

```
sudo systemctl enable postgresql
```

(2) 起動

sudo systemctl postgresql start

(3) 再起動
sudo systemctl restart postgresql

(4) 停止
sudo systemctl postgresql stop

6.2.6 PL/Python の導入

PostgreSQL 上で動作する Python 実装のユーザ独自定義関数を開発、実行するために、以下を実施し PL/Python を導入します。

本マニュアルでは、PL/Python を導入するにあたり、pip3 をインストール必要があります。pip3 がインストールされていない場合は、6.1.1 pip3（インストールされていない場合のみ）を参照してください。

(1) ユーザ切り替え
$ sudo -i -u postgres

(2) PostgreSQL への接続
$ psql

(3) plpython3u の有効化
postgres=# CREATE EXTENSION plpython3u;

(4) 導入の確認
postgres=# select lanname from pg_language;
実行結果に plpython3u が含まれていれば導入完了です。

```
   lanname
------------
 plpython3u
```

(5) python ライブラリの追加
ユーザ独自定義関数内で Python ライブラリを利用する場合は、以下でインストールします。

$ sudo pip3 install [利用ライブラリ]

7　データ取得変換ツール

7.1 前提条件および概要

データ取得変換ツールでは、収集したデータを必要に応じてデータ分析・可視化システムで利用可能な形に変換します。変換したファイルを Web サーバに配置する、又は直接 Web サーバを出力先として変換配置することで WebGIS から参照可能となり、本システムにデータを追加できます。本システムからのデータ削除は、配置又は変換配置したファイルの削除により可能です。ただし、収集データうち GeoServer へ登録して配信するものについては、GeoServer への登録/削除により本システムへの追加/削除を行います。

7.1.1　データ取得変換ツールの実行場所

データ取得変換ツールは、予めデータ取得変換ツールの依存パッケージを導入した上で、任意のディレクトリ（以下「対象ディレクトリ」）にツールを設置し、実行します。取得または変換データは対象ディレクトリ下に保存されます。以下の説明に従ってデータ取得変換ツールを使用する場合は、初めに対象ディレクトリに移動して下さい。

7.2 データ取得変換ツールの構成について

データ取得変換ツールは、表 1 の 2 つの個別ツールで構成されます。

表 1　個別ツール一覧

個別ツール名	用途
地理空間情報データ変換ツール	国土数値情報(シェープファイル)のバイナリベクトルタイル形式への変換
3 次元建物データ変換ツール	国土交通省 PLATEAU による 3 次元建物データのバイナリベクトルタイル形式への変換

7.3 データ取得変換ツール依存パッケージのインストール

7.3.1　unzip のインストール手順

データ取得変換ツールに必要な unzip のインストールについて記述します。
次のコマンドを実行して下さい。

(1) unzip インストール
sudo apt install unzip

7.3.2　gdal のインストール手順

データ取得変換ツールに必要な gdal のインストールについて記述します。
本マニュアルでは、gdal 3.4.0 を使用した説明となります。
次のコマンドを実行して下さい。

(1) gdal インストール
sudo apt install gdal-bin

(2) gdal のバージョン確認
ogr2ogr --version

「GDAL 3.0.4, released 2020/01/28」が表示されていれば成功です。

(3) gdal 更新用のリポジトリ追加
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable

画面の指示に従ってエンターキーを押し、追加を完了します。

(4) パッケージの更新
sudo apt upgrade

(5) gdal のバージョン確認
ogr2ogr --version

「GDAL 3.4.0, released 2021/11/04」以降が表示されていれば成功です。

7.3.3　make のインストール手順

7.3.1 のインストール手順で必要な make 及び関連パッケージのインストールについて記述します。

次のコマンドを実行して下さい。

(1) make 及び関連パッケージのインストール
sudo apt install sqlite3 libsqlite3-dev

7.3.4　tippecanoe のインストール手順

地理空間情報データ変換ツール及び 3 次元建物データ変換ツールの動作には tippecanoe のインストールが必要です。ここでは、tippecanoe のインストールについて記述します。

次の 3 つのコマンドを実行して下さい。

(1) tippecanoe を取得
git clone https://github.com/mapbox/tippecanoe.git

(2) tippecanoe のビルド場所へ移動
cd tippecanoe

(3) tippecanoe のビルド
make

(4) tippecanoe のインストール
sudo make install

7.4 データ取得変換ツールのインストール

https://github.com/nict-testbed-dalab
Data_visuallization_Componet リポジトリ
DataPreparationTool の以下ファイルを対象ディレクトリに配置して下さい。

対象ディレクトリに、
data_preparation_tool_mvt.sh
data_preparation_tool_3d_mvt.sh
が配置されていれば構築は成功です。

7.5 運用

本システムでは、標準で DataCatalog.pdf（https://github.com/nict-testbed-dalab：Data_visualization_Component リポジトリ）の対象データを運用できるようになっています。また、形式が同一のデータであれば利用者が独自にデータを追加して運用することが可能です。

7.6 データ収集

DataCatalog.pdf に記載されているデータ取得元から、本システムで運用したデータを収集して下さい。対象となるデータには年度更新(数年に一度の更新も含む)のあるものもあり、更新があった場合には最新のデータを追加して本システムを運用することが可能です。

7.6.1　国土地理院が提供するダウンロードツールによる地理院タイルの収集

国土地理院は、国土地理院が公開する地理院タイルについてダウンロードツールを公開しています。

地理院タイルダウンロードツール
https://github.com/gsi-cyberjapan/tdlmn

導入手順書及び使用マニュアル
https://github.com/gsi-cyberjapan/tdlmn/blob/main/docs/導入手順書及び使用マニュアル.pdf

この導入手順に従って、ツールで推奨されている Window10+Ruby 環境で

ruby tdlmn.rb -mn -dt -merge

を実行することにより、タイルのダウンロードが可能です。

本システムに標準で収集、公開されているデータは、表 2 のとおりです。program.ini の TILE_ID、ZOOM_LEVEL、TILE_FOLDER を都度変更してコマンドを実行し、タイルをダウンロードして下さい。

表 2　使用タイル一覧

タイル名	TILE_ID	ZOOM_LEVEL	TILE_FOLDER について
航空写真	seamlessphoto	2,3,4,5,6,7,8,9,10,11,12,13,14,15,16	任意の作業ディレクトリを指定して実行し、ファイルを保存
淡色地図	pale	5,6,7,8,9,10,11,12,13,14,15,16,17,18	任意の作業ディレクトリを指定して実行し、ファイルを保存
標準地図	std	5,6,7,8,9,10,11,12,13,14,15,16	任意の作業ディレクトリを指定して実行し、ファイルを保存
地理院地図 Vector	experimental_bvmap	5,6,7,8,9,10,11,12,13,14,15,16	任意の作ディレクトリを指定して実行し、ファイルを保存

※ZOOM_LEVEL 指定は改行なしで指定

国土地理院の推奨環境は Windows10＋Ruby3.0.0 ですが、Linux（Ubuntu）でも実行可能です（未確認）。

7.6.2　3D 都市モデル（Project PLATEAU）の収集

本システムでは、国土交通省の Project PLATEAU

https://www.mlit.go.jp/plateau/

(3D 都市モデル（Project PLATEAU）ポータルサイト)
https://www.geospatial.jp/ckan/dataset/plateau

により整備、公開されている MVT 形式の 3D 都市モデルを表示することが可能です。また、CityGML としてしか公開されていないものについても、収集時に QGIS で GeoJSON ファイルに変換して保存することにより、データ取得変換ツールで GeoJSON から MVT 形式に変換して利用することができます。以降では、この方法について記述します。CityGML が zip ファイルに圧縮されている場合は、7.6.1 の手順で展開して下さい。

(1) QGIS のバージョン確認

QGIS は 3.16 以降、かつ QGIS に同梱の GDAL のバージョンは 3.4 以降で利用して下さい。JVN 上(CUI)での QGIS 操作となるため、QGIS が利用できる Python コンソールで作業します。以下の説明で、日本語込みの[](例：[レイヤ名])は Python 配列ではなく、コマンド実行時に置き換える箇所を示します。

(参考)

対応する QGIS がインストールされていない場合は、

```
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt upgrade
sudo apt install qgis
```

でインストールが可能です。インストールするタイミングにより公開場所が変わっている可能性もあるため、確認の上導入して下さい。

(2) QGIS (Python コンソールから使用)の起動

次のコマンドを順に実行して、QGIS を起動して下さい。はじめに、Processing を利用できるよう設定します。

```
export QT_QPA_PLATFORM=offscreen
python3
import sys
from qgis.core import (QgsApplication, QgsVectorLayer)

QgsApplication.setPrefixPath('/usr', True)
qgs = QgsApplication([], False)
qgs.initQgis()

sys.path.append('/usr/share/qgis/python/plugins')

import processing
from processing.core.Processing import Processing
Processing.initialize()
```

(3) CityGML の読み込み

次のコマンドを実行して、CityGML を QGIS のレイヤとして読み込んで下さい。

[レイヤ] = QgsVectorLayer([CityGML ファイルのパス], [レイヤ名], 'ogr')

(例) samplelayer = QgsVectorLayer('sample.gml', 'sample', 'ogr')

(4) GeoJSON の保存設定

次のコマンドを実行して、CityGML を QGIS のレイヤとして読み込んで下さい。

```
params = {
    'LAYERS' : [レイヤ],
    'CRS' : 'EPSG:4326',
    'OUTPUT' : [保存する GeoJSON ファイルのパス]
}
```

(例)

```
params = {
    'LAYERS' : samplelayer,
    'CRS' : 'EPSG:4326',
    'OUTPUT' : ' sample.geojson'
}
```

(5) GeoJSON の保存

次のコマンドを実行して、GeoJSON ファイルを保存して下さい。

```
processing.run('native:mergevectorlayers', params)
```

[保存する GeoJSON ファイルのパス]で指定したディレクトリに GeoJSON ファイルが保存されていれば、GeoJSON 形式での保存は完了です。

(6) QGIS (Python コンソールから使用)の終了

次のコマンドを実行して、QGIS (Python コンソールから使用)を終了して下さい。

```
exit()
```

同等の操作は、Ubuntu をデスクトップモードで起動した場合や Windows10 にインストールした QGIS で行う場合には GUI 操作で行うことが可能です。また、QGIS の GUI に用意された専用の Python コンソールから行うことも可能です。

7.7 地理空間情報データ変換ツールの利用

7.7.1　gml から GeoJSON への変換

(1) 収集した zip ファイル(GML)を展開します。

unzip -d [展開先の指定フォルダへのパス] [展開したい zip フォルダへのパス]

(2) 展開したフォルダにはシェープファイルが格納されています。このファイルを GeoJSON へ変換します。

ogr2ogr -f GeoJSON [出力先のフォルダ] [変換したい shp 形式のファイルへのパス]

7.7.2　GeoJSON から MVT への変換

data_preparation_tool_mvt.sh を実行して、GeoJSON 形式のデータを MVT に変換します。

利用方法は

data_preparation_tool_3d_mvt.sh [MVT を出力したいフォルダ名のパス] [レイヤ名] [MVT ファイルに変更したい GeoJSON データのパス]

です。

(参考) 直接 tippecanoe を実行する場合
tippecanoe で GeoJSON 形式のデータを MVT に変換します。

tippecanoe -e [出力先のフォルダ] -pC -z13 -aN [GeoJSON 形式のファイルへのパス]

7.8 3 次元建物データ変換ツールの利用

7.8.1　GeoJSON から MVT への変換

data_preparation_tool_3d_mvt.sh を実行して、GeoJSON 形式のデータを MVT に変換します。

利用方法は

data_preparation_tool_3d_mvt.sh [MVT を出力したいフォルダ名のパス] [レイヤ名] [MVT ファイルに変更したい GeoJSON データのパス]

です。

(参考) 直接 tippecanoe を実行する場合
tippecanoe で GeoJSON 形式のデータを MVT に変換します。

tippecanoe -pC -ad -an -ps -z18 -e [MVT を出力したいフォルダ名のパス] -l [レイヤ名] -ai [MVT ファイルに変更したい GeoJSON データのパス]

※オプション ps をつけないと、建物が合成され、形が崩れて表示される箇所が発生するため、このオプションは必ず指定して下さい。

＜オプション一覧＞
・-e：出力フォルダの指定
・--no-tile-compression：ファイル圧縮の抑制
・-z：出力するズームレベルの指定
・-l：出力に使用するレイヤ名の指定
・-ai：ID の自動付番
・-ad：各ズームレベルから機能の一部を動的にドロップして、大きなタイルを 500K のサイズ制限未満に保ちます。
・-an：各ズームレベルから最小のフィーチャ（物理的に最小：最短の線または最小のポリゴン）を動的にドロップして、大きなタイルを 500K のサイズ制限未満に保ちます。
・-ps：線と多角形を単純化しないでください

オプションは下記のサイトを参考にして下さい。

https://github.com/mapbox/tippecanoe#line-and-polygon-simplification

7.9 GeoServer への GeoTiff の登録

7.9.1　ワークスペースの準備

本システムで利用するワークスペースが未作成の場合は、予めワークスペースの作成を行います(図 7-1)。

図 7-1　ワークスペース作成

サイドメニューの「ワークスペース」から、「新規ワークスペース追加」でも同様にワークスペースを作成できます。

Name とネームスペース URI を設定し、送信をクリックします。NAME とネームスペース URI は自由に設定してください(図 7-2)。

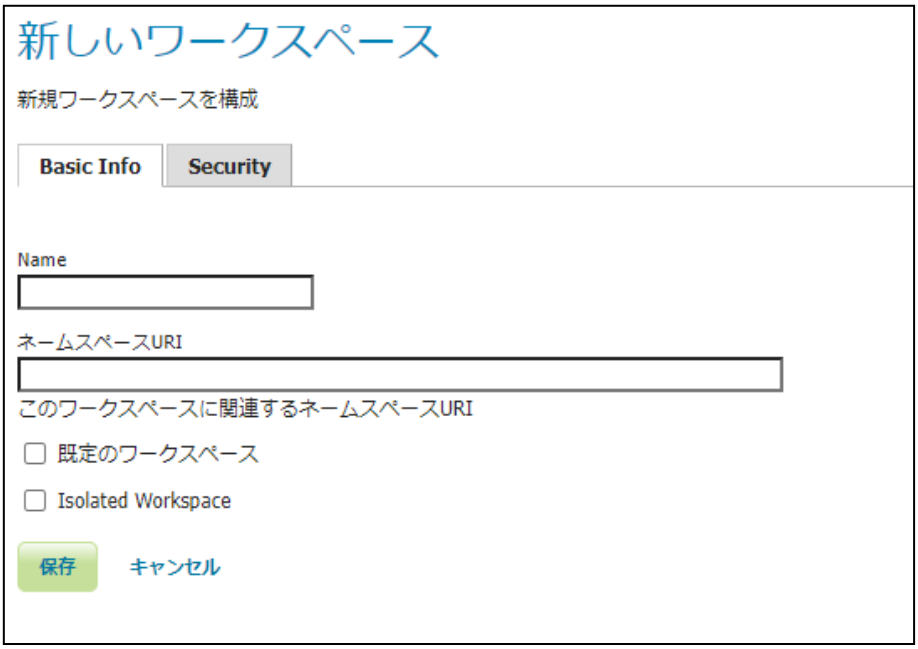


図 7-2　ワークスペース作成(入力)

サイドメニューの「ワークスペース」をクリックすると、ワークスペースの一覧が表示されるので、先ほど作成したワークスペースをクリックしてください。

ワークスペース編集のページに遷移するので、「WMTS」にチェックをつけて保存してください(図 7-3)。

図 7-3　ワークスペース作成(設定)

7.9.2　GeoTiff データの登録

サイドメニューの「ストア」をクリックし、「ストア新規追加」をクリックしてください。

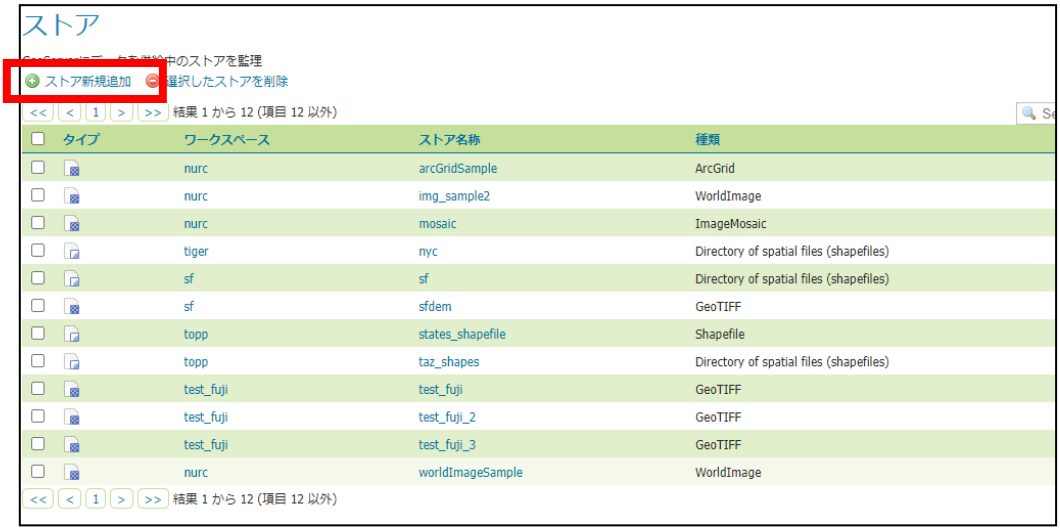


図 7-4　ワークスペース作成(設定)

新規データソースのページに遷移するので、ラスターデータソースに内にある「GeoTIFF」をクリックします。

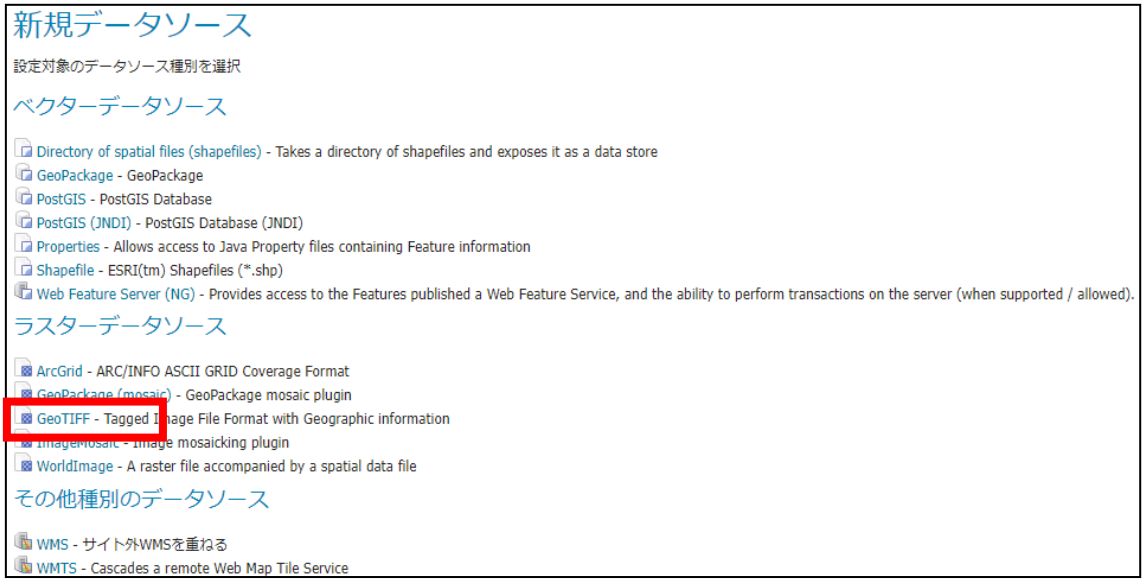


図 7-5　ワークスペース作成(設定)

「GeoTIFF」をクリックすると、ラスターデータの追加のページに遷移します。「ワークスペース」を事前に作成したものを選択し、「データソース名」は任意に設定してください。「URL」には、tif データが置いてあるディレクトリを指定し、保存してください。

例）file:/ファイルパス/~.tif

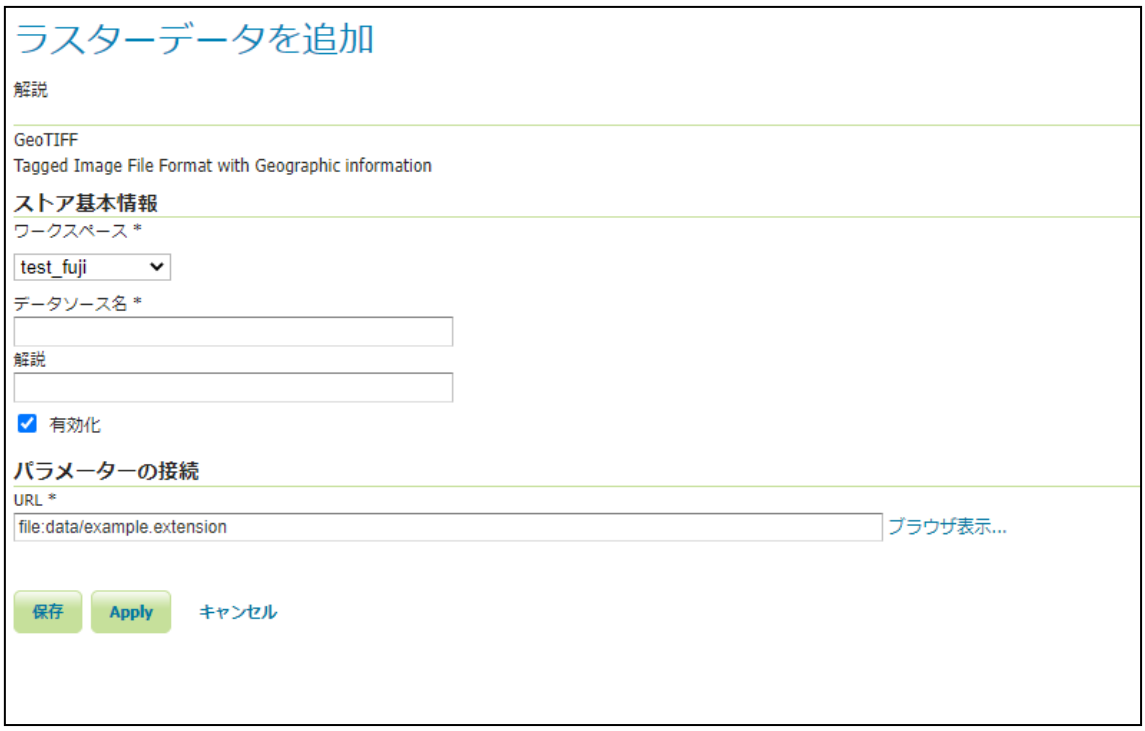


図 7-6　ワークスペース作成(設定)

保存をクリックすると、新規レイヤの画面に遷移するので、「公開」をクリックしてください。

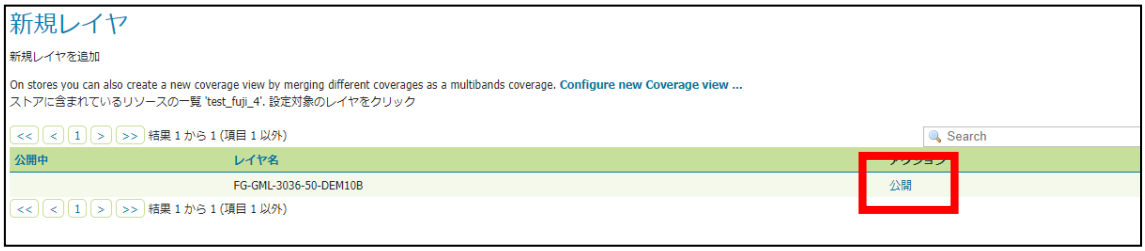


図 7-7　ワークスペース作成(設定)

「公開」をクリックすると、レイヤ編集ページに移動します。設定を変更せずに画面下の「保存」をクリックしてください。

図 7-8　ワークスペース作成(設定)

サイドメニューの「レイヤ」に移動すると、先ほど作成したレイヤが追加されています。

サイドメニューの「レイヤプレビュー」から作成されたレイヤの「OpenLayers」をクリックすると、以下のように追加したレイヤが WMTS で配信されているのが確認できます。

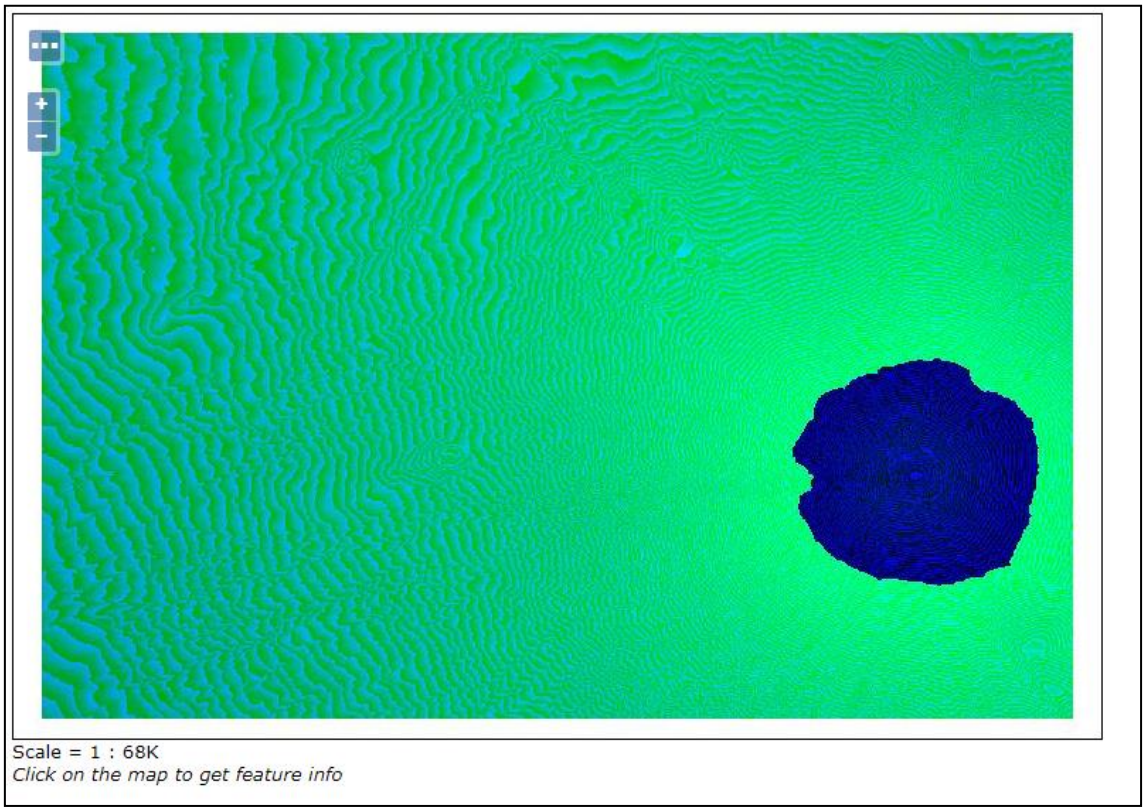


図 7-9　ワークスペース作成(設定)

7.9.3　レイヤのグループ化

複数のレイヤをひとつにまとめる場合は、サイドメニューの「レイヤグループ」をクリックし、「レイヤグループの新規追加」を選択してください。

図 7-10　ワークスペース作成(設定)

「レイヤグループの新規追加」をクリックすると、レイヤグループの画面に遷移するので、「ユーザ名」と「タイトル名」を任意で設定し、「ワークスペース」を先ほど作成したものを選択します。

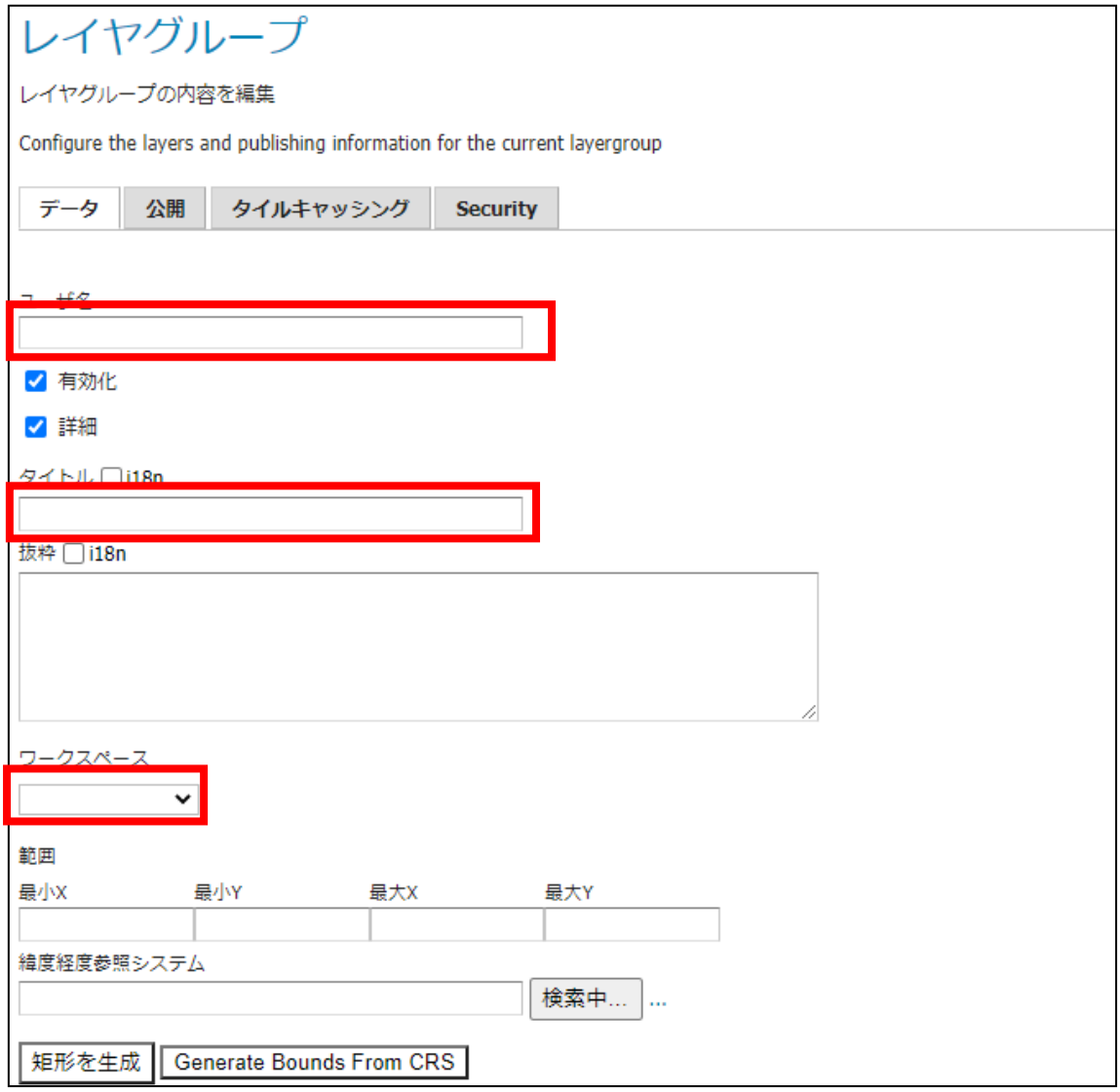


図 7-11　ワークスペース作成(設定)

「レイヤ追加」をクリックすると、ポップアップが表示されるので、追加したレイヤを全て選択してください。

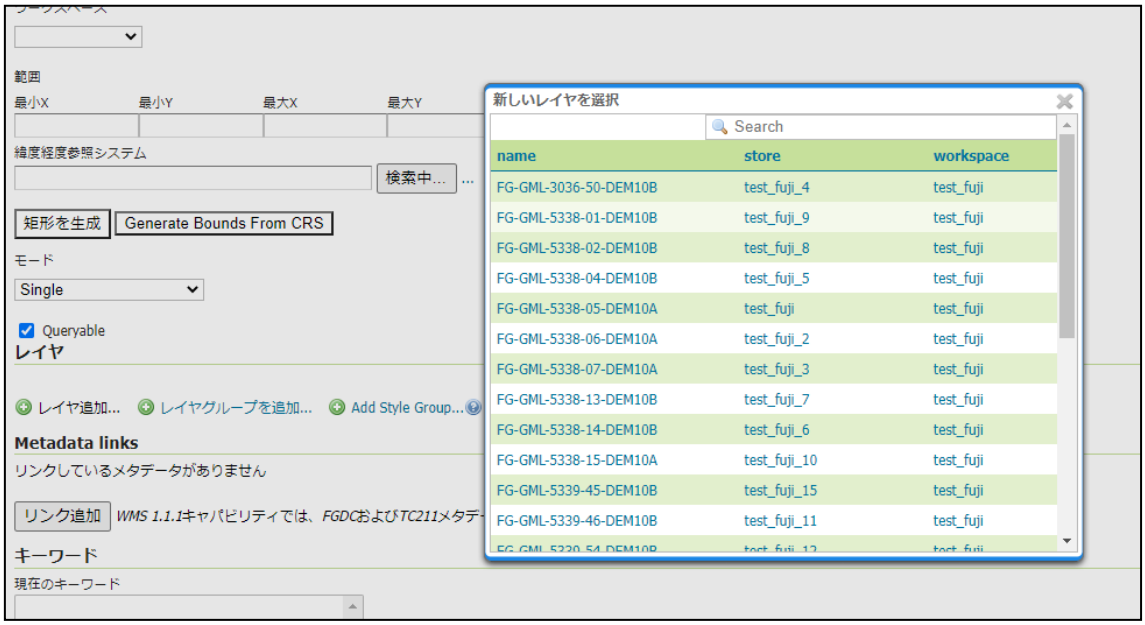


図 7-12　ワークスペース作成(設定)

「短径を生成」をクリックすると、追加されたレイヤの範囲を自動で読み取り、最小 X などの項目にバウンディングボックスが生成されるので、一番下の保存をクリックし、設定を保存します。

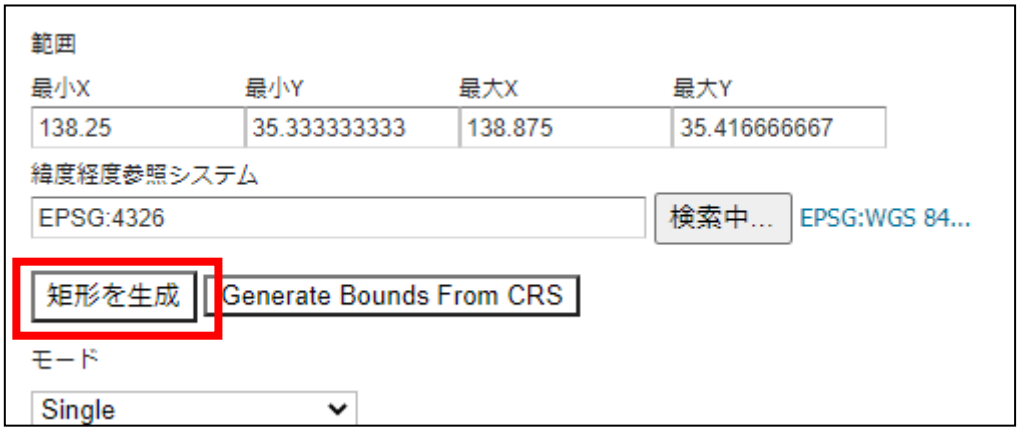


図 7-13　ワークスペース作成(設定)

設定が完了したら、サイドメニューの「レイヤグループ」のレイヤグループ一覧に先ほどグループ化したレイヤがあるか確認します。

確認後、サイドメニューの「レイヤプレビュー」からグループ化したレイヤの「OpenLayers」をクリックすると、以下のように追加したレイヤが WMTS で配信されているのが確認できます。

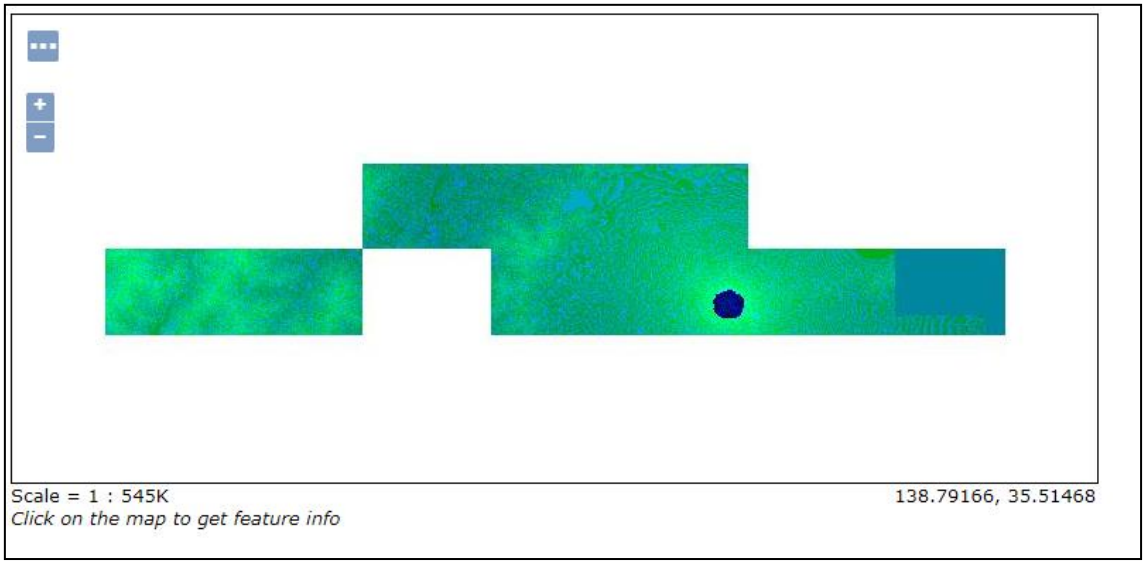


図 7-14　ワークスペース作成(設定)

7.10　style.json の記述によるベクトルタイル地図のデザイン

タイルに対して style.json を記述して、ベクトルタイル地図のデザインを行います。サンプル記述や既存のベクトルタイルを参考にして、style.json を作成して下さい。また詳細については

https://docs.mapbox.com/mapbox-gl-js/style-spec/

を参考にして下さい。

7.10.1 style.json サンプル

```
{
    "version": 8,
        "type": "overlay",
        "format": "pbf",
    "glyphs":"https://maps.gsi.go.jp/xyz/noto-jp/{fontstack}/{range}.pbf",
    "sources": {
```

```
        "mvt": {
            "type": "vector",
            "tiles": [
                "https://x.jp/storage/data/vectortile/1_a_3_2_tochi/style.json"
            ]
        }
    },

"layers": [
    {
        "id": "Lineex",
        "source": "mvt",
        "source-layer": "1_a_4_1_hinansisetu",
        "minzoom": "0",
        "maxzoom": "14",
        "type": "line",
        "paint": {
          "line-color": "#000000",
            "line-width": 1,
            "line-opacity": 1.0
        }
    },
        {
            "id": "line-1",
            "source": "mvt",
            "source-layer":"kaigansen",
            "filter": ["==","C23_003","1"],
            "type":"symbol",
            "layout":
                {
                    "symbol-placement": "point",
                    "text-allow-overlap": false,
                    "text-field":["literal","都道府県知事"],
                    "text-font":["NotoSansCJKjp-Regular"],
                    "text-anchor": "top",
                    "text-rotation-alignment":"auto",
```

```
                "text-pitch-alignment": "auto"
            },
        "paint":
            {
                "text-color":"rgba(0,0,0,1)",
                "text-halo-color":"rgba(255,255,255,1)",
                "text-halo-width":1
            }
    },
}
```

(サンプルの記述について)
「tiles」に style.json の場所を設定します
※サンプルにはラインとラベルの表示設定があります。

「layers」に表示用の設定を記述します
id・・・任意の名称
source-layer・・・geojson のファイル名（拡張子抜き）
type・・・主に line,polygon,circle など種類があります。元データの形式に合わせます。
filter・・・項目を追加すると、一定の条件の地物にのみスタイルが適用されます。サンプルの場合だと、地物の属性で、C23_005 の値が 1 の地物にのみスタイルを適用することができます。

・ラベル
symbol-placement・・・地物に対してラベルが表示される位置。
text-allow-over-lap・・・他のラベルに重ねて表示させるかどうか。
text-field・・・ラベルとして表示させる文字。
[“literal”, 愛知]・・愛知が表示される。
[“get”, C23_003]・・C23_003 の値が表示される。
text-rotation-alignment・・・ラベルの表示向き

作成した style.json は pbf フォルダと metadata.json のある場所に保存しておきます。

7.11 PostgreSQL への GTFS 形式データの収集、登録

検証用データとして、公共交通オープンデータの内、バスデータを収集、変換し、PostgresSQL のテーブルへ登録しています。データを利用する場合は、事前にユーザ登録し、アクセストークンを取得してください。

データ公開元：公共交通オープンデータセンター

https://developer-dc.odpt.org/

公開されているデータ一覧と各配信用 API も同サイトから検索できます。

7.11.1 データ収集

収集しているバスデータは、GTFS リアルタイムで配信されており、データの構造は以下になります。

https://developers.google.com/transit/gtfs-realtime/reference?hl=ja

バスデータをパースするために、以下の Python3 のライブラリが提供されており、以下でインストールできます。

```
pip install --upgrade gtfs-realtime-bindings
```

実際に配信されるバスデータから、各車両の ID、車両 ID、時刻、緯度経度をパースする Python3 の実装が以下になります。

```
from google.transit import gtfs_realtime_pb2
import requests

feed = gtfs_realtime_pb2.FeedMessage()
response = requests.get( [データ配信 API] )
feed.ParseFromString(response.content)

for entity in feed.entity:
    id = entity.id
    vid = entity.vehicle.vehicle.id
    dt = formatTimestamp(entity.vehicle.timestamp)
    lon = str(entity.vehicle.position.longitude)
    lat = str(entity.vehicle.position.latitude)
```

例えば、経度情報であれば

FeedMessage - FeedEntity - VehiclePosition - Position - longitude

に格納されており、各階層の具体的な変数名は以下で規定されています。

https://developers.google.com/transit/gtfs-realtime/gtfs-realtime.proto

そのため、Python から経度情報を取り出す場合は、上記の実装で取得できます。

7.11.2 データ格納

PostgresSQL へバスデータを格納する前に、事前にテーブルを準備しておきます。

ターミナルから PostgreSQL にログインし、以下でバスデータを格納するテーブルを作成します。

```
postgres =#CREATE TABLE ［テーブル名］（
  ［カラム名 1］［データ型 1］,
  ［カラム名 2］［データ型 2］,
     ・・・
)
```

また、テーブルを作成したユーザとは、別のユーザがそのテーブルのデータを利用する場合、権限の付与が必要な場合があります。例として検索権限を付与する SQL は以下になります。

```
postgres =# GRANT SELECT on ［テーブル名］ to ［利用するユーザ名］;
```

Python でパースされたバスデータを、PostgreSQL へ登録するには以下のインストールが必要です。

```
  $ sudo pip install psycopg2
```

PostgreSQL へ接続し、実際にバスの ID、時刻、緯度、経度をテーブルへ追加する実装は以下になります。

```
from google.transit import gtfs_realtime_pb2
import requests
import psycopg2

con = psycopg2.connect(
    host= ［PostgreSQL を導入したサーバ IP］,
    port= ［PostgreSQL のポート番号］,
    database= ［接続する DB 名］,
    user= ［接続するユーザ名］,
    password= ［接続するユーザのパスワード］ )
cur = con.cursor()

feed = gtfs_realtime_pb2.FeedMessage()
response = requests.get( ［データ配信 API］ )
feed.ParseFromString(response.content)

for entity in feed.entity:
    id = entity.id
    dt = formatTimestamp(entity.vehicle.timestamp)
    lon = str(entity.vehicle.position.longitude)
    lat = str(entity.vehicle.position.latitude)

    location = 'ST_GeomFromText(¥'POINT(' + lon + ' ' + lat + ')¥',4326)')
```

```
    cur.execute('INSERT INTO ［テーブル名］（[IDカラム], [時刻カラム], [座標カラム]) VALUES (' + id + ',' + dt + ',' + location + ')')

con.commit()
cur.close()
con.close()
```

なお、テーブルの形式は以下を想定しています。

```
postgres =#CREATE TABLE ［テーブル名］（
  [IDカラム] text,
  [時刻カラム] timestamp,
  [座標カラム] geometory
)
```

8　WebGIS アプリケーション

8.1 Mapbox

8.1.1　テンプレートアプリケーションの設置

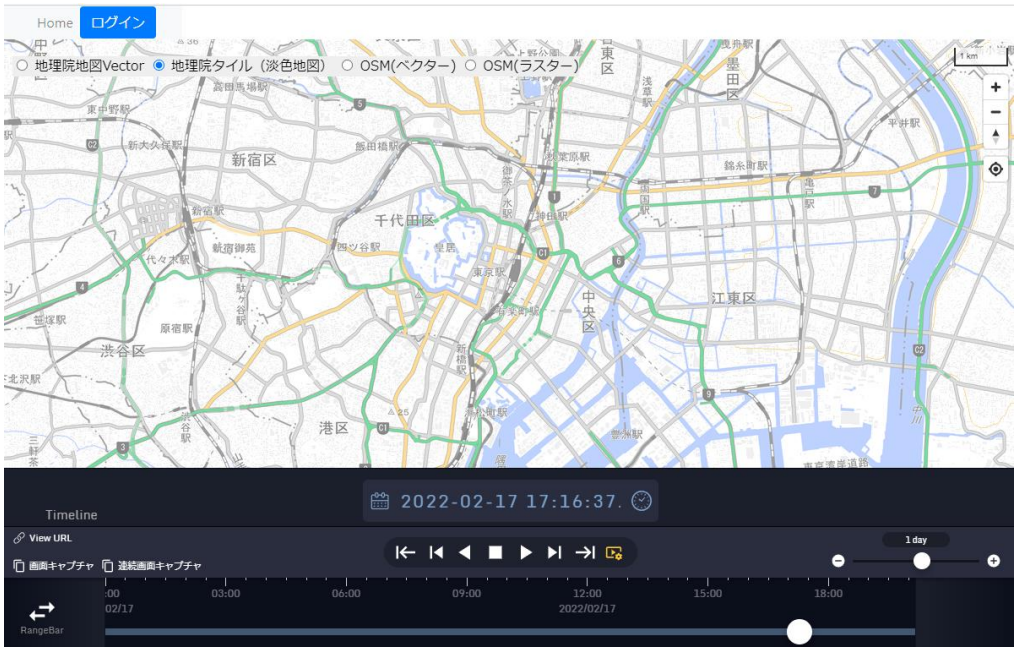
https://github.com/nict-testbed-dalab：TemplateWebGIS_Mapbox から Mapbox_map、css、img、js、timeline、auth_config.json、index.html を取得し、/var/www/html/mapbox_template/ へ配置して下さい。

その後、対応 Web ブラウザにより

https://tb-gis-web.xxx.jp/mapbox_template

へアクセスして、下図のように地理情報が表示できれば設置は成功です。表示が成功しない場合は、環境構築の状況を確認して下さい。

もしくは、/js/の中にある各レイヤを設定している箇所を適宜修正してください。



テンプレートアプリケーションの初期画面(Mapbox)

8.1.2　サンプルアプリケーションの設置

成果物一式に格納されている mapbox_sample.zip を展開し、WEB サーバの下記ディレクトリに配置します。

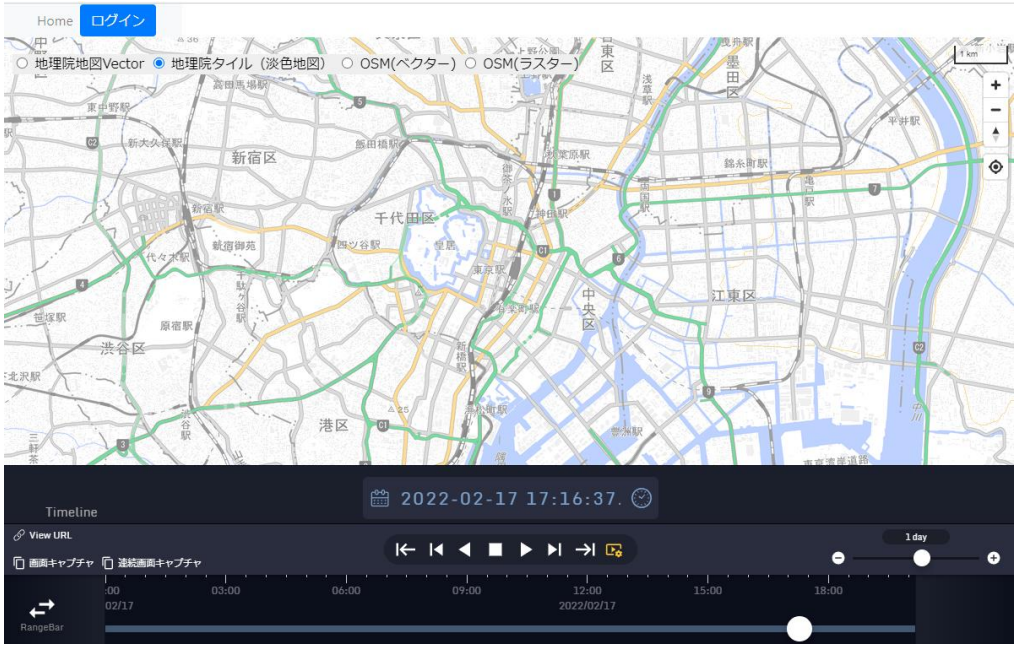
/var/www/html/mapbox_sample/

その後、対応 Web ブラウザにより

https://tb-gis-web.jgn-x.jp/mapbox_sample

へアクセスして、下記のように地理情報が表示できれば設置は成功です。表示が成功しない場合は、環境構築の状況を確認して下さい。

もしくは、/js/の中にある各レイヤを設定している箇所を適宜修正してください。



8.2 iTowns

8.2.1　テンプレートアプリケーションの設置

https://github.com/nict-testbed-dalab：TemplateWebGIS_iTowns から JSONLayers、data、font、jquery-k2go-timeline、projectNict、src、auth_config.json、index.html、package-lock.json、package.json、webpack.config.js を取得し、

/var/www/html/itowns_template

へ配置して下さい。その後、下記コマンドを実行します。

npm install
npm audit fix　（*1）
npm run build

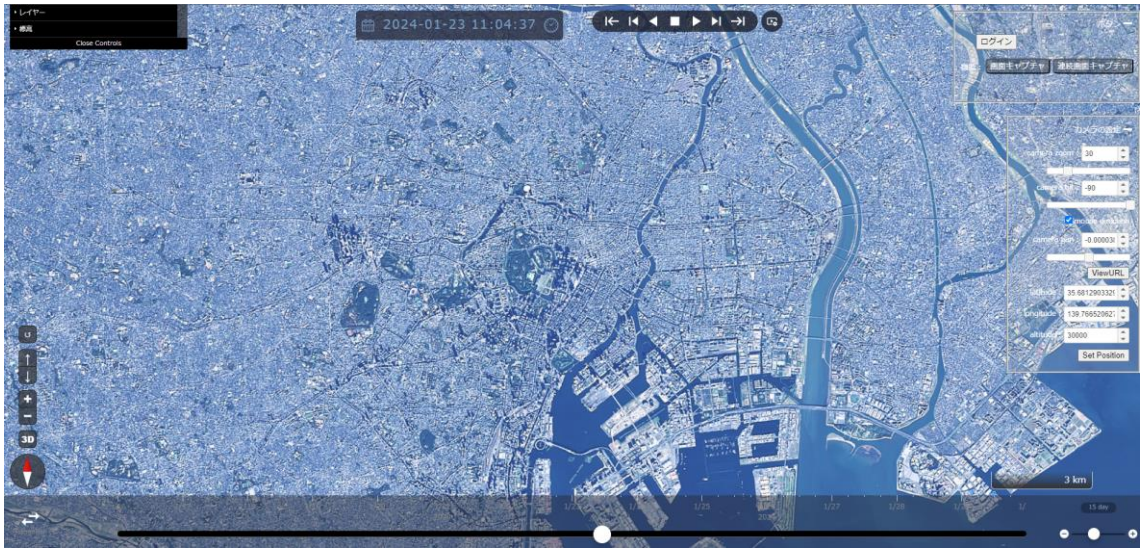
*1 インストールするモジュールのバージョンによっては、セキュリティに関する警告が表示される場合がありますので、こちらのコマンドを実行してください。

その後、対応 Web ブラウザにより

https://tb-gis-web.xxx.jp/itowns_template/

へアクセスして、下図のように地理情報が表示できれば設置は成功です。表示が成功しない場合は、環境構築の状況を確認して下さい。

もしくは、/JSONLayers/layers/の中にある各レイヤ設定ファイル（json ファイル）の中にあるデータの参照先を適宜修正してください。



テンプレートアプリケーションの初期画面(iTowns)

8.2.2　サンプルアプリケーションの設置

成果物一式に格納されている「itwons_sample.zip」を展開し、

/var/www/html/itowns_sample

へ配置して下さい。この状態で、対応 Web ブラウザにより

https://tb-gis-web.jgn-x.jp/itwons_sample/

へアクセスして、図 8-1 のように地理情報が表示できれば設置は成功です。表示が成功しない場合は、環境構築の状況を確認して下さい。

もしくは、/test_itw/app.js の中にある各レイヤを設定している箇所を適宜修正してください。

図 8-1 サンプルアプリケーションの初期画面

8.2.3　(参考)iTowns のビルド環境も併せて構築する場合

テンプレートアプリケーションやサンプルアプリケーションを参考に独自のアプリケーションを開発する際、ベースとなっている iTowns のビルド環境も併せて構築すると、iTowns の公式サンプルによる学習を行いながら開発を進めることができます。

iTowns のビルド環境を構築する場合は下の URL を参考にして下さい。

iTowns リポジトリ
https://github.com/iTowns/itowns

iTowns コーディングガイド
https://github.com/iTowns/itowns/blob/master/CODING.md

また、iTowns のバグ情報や開発の状況を知りたい場合は、次の URL を参考にして下さい。

iTowns issues
https://github.com/iTowns/itowns/issues

iTowns pull requests
https://github.com/iTowns/itowns/pulls

8.3 MapLibre

8.3.1　テンプレートアプリケーションの設置

https://github.com/nict-testbed-dalab：TemplateWebGIS_MapLibre から Maplibre_map、css、img、js、timeline、auth_config.json、index.html を取得し、/var/www/html/maplibre_template/ へ配置して下さい。

以降の手順は、MapBox と同様の手順となります。

9　認証

・前提

本システムにおいて、ユーザ認証はクラウドサービス「Auth0」を用います。
本書では管理者アカウントの登録と、すでに作成済みの設定内容について記載します。

9.1 Auth0 管理者アカウント作成

9.1.1　　アカウント作成ページにアクセス

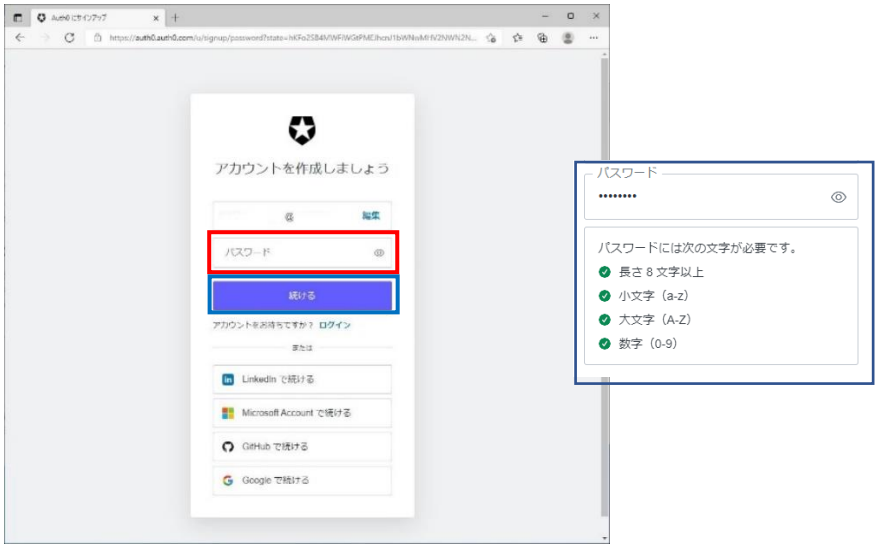
ブラウザから URL「https://auth0.com/jp/signup」を表示します。 以降は、メールアドレスによる作成方法を説明します。

9.1.2　　メールアドレス入力

下記のような画面が表示され、「メール」に登録する管理者のメールアドレスを入力し、サービス規約を確認後、「登録」ボタンを押下します。

9.1.3　　パスワード設定

次に、下記のような画面が表示され、入力したメールアドレスであることを確認後、パスワードを入力し、「続ける」ボタンを押下します。パスワードは右記のようにすべてチェックがされていることを奨励します。

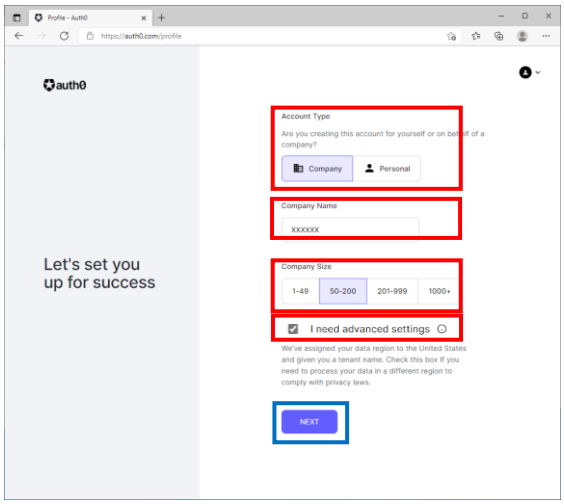


9.1.4　　アカウントタイプ記入

本件では Account Type は「Company（会社）」を選択します。

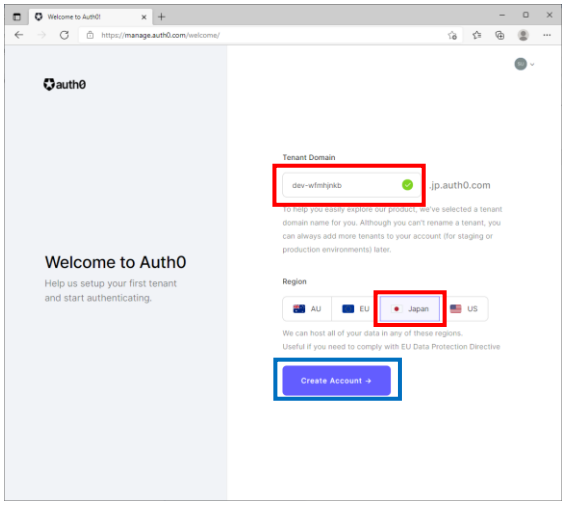
次に、Company Name（会社名）, Company Size（社員数）は該当する内容を入力します。

最後のチェックボックスは、説明の後半を直訳すると「プライバシー法に準拠するために（アメリカ合衆国以外の）別の地域でデータを処理する必要がある場合は、このチェックボックスをオンにしてください。」とあるため、<u>チェックします。</u>



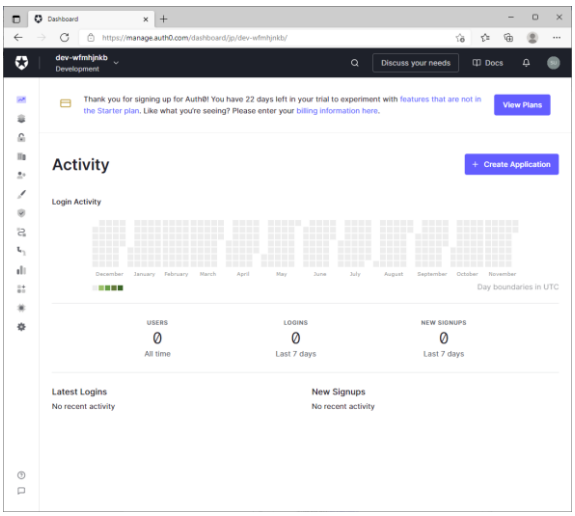
9.1.5　　テナント作成

下記のように、テナント ドメイン名(ここでは tb-gis-web)を記入し、地域(Region)は「**Japan**」を選択し、「Create Account」ボタンを作成します。



9.1.6　　作成が完了

下記のような管理画面トップ（ダッシュボード）が表示されます。

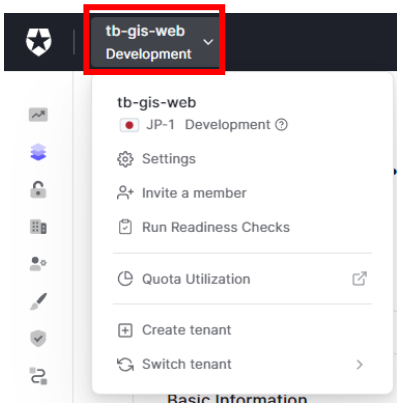


9.2 テナント設定（各種設定画面）

Applications と API について説明します。

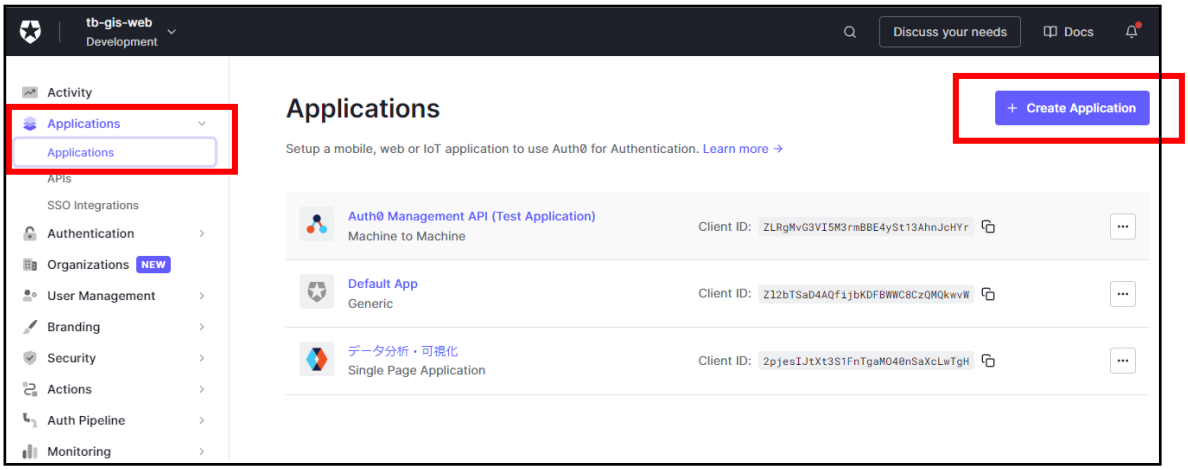
9.2.1 ログインおよびテナント確認

ブラウザから https://auth0.com/ にアクセスしログインします。
（もしくは上記リンクを初回クリック後に表示されます。）
メニューバーに、本件のテナント名「tb-gis-web」が表示されていることを確認します。

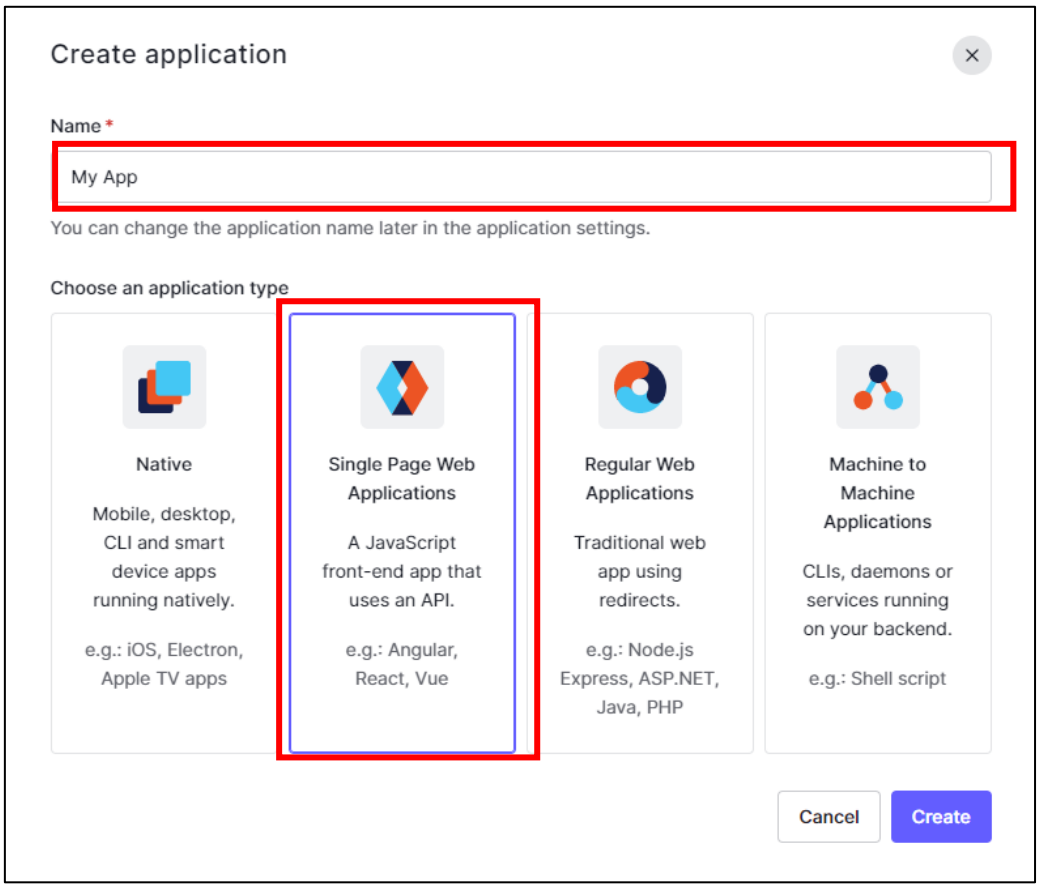


9.2.2 Application の新規作成

メニューから「Applications -> Application」をクリックします。



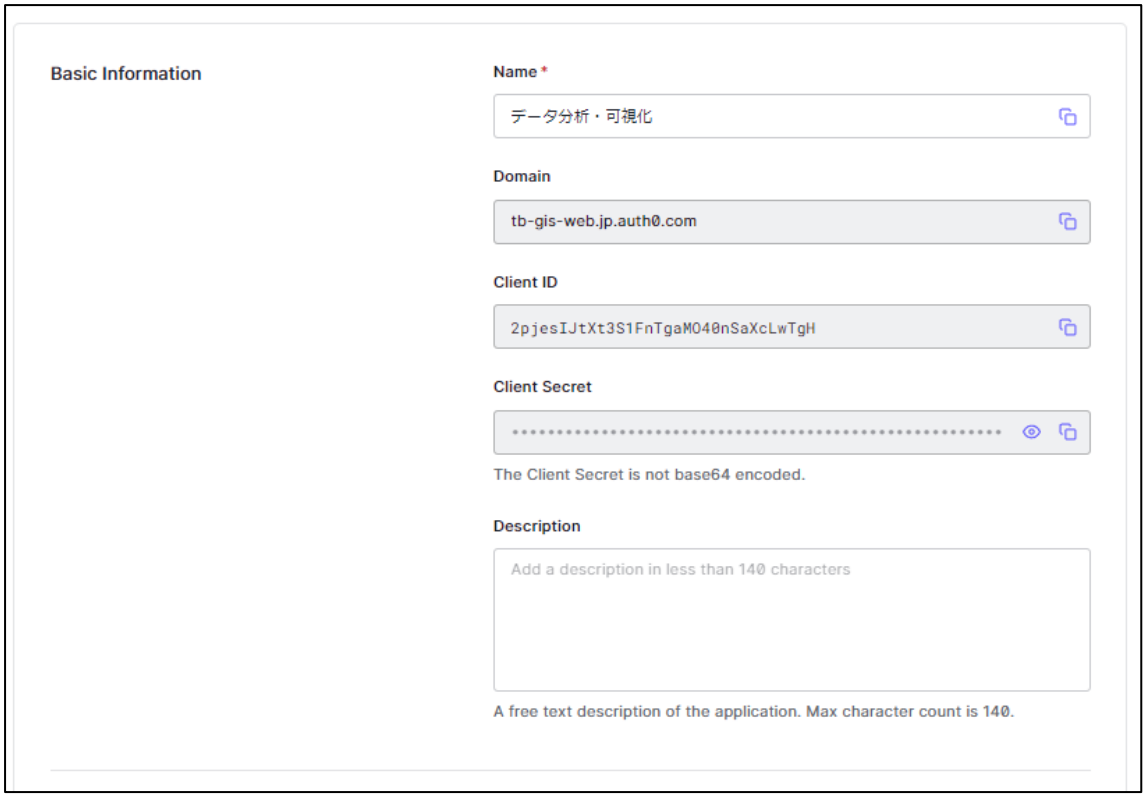
続いて「Create Application」をクリックして、アプリケーション名（本システムでは「データ分析・可視化」）を入力し、application type（本システムでは「Single Page Web Applications」）を選択します。



①　　基本設定(Setting)の内容確認

・Basic Information：名称や URL（ドメイン）などの基本情報です。

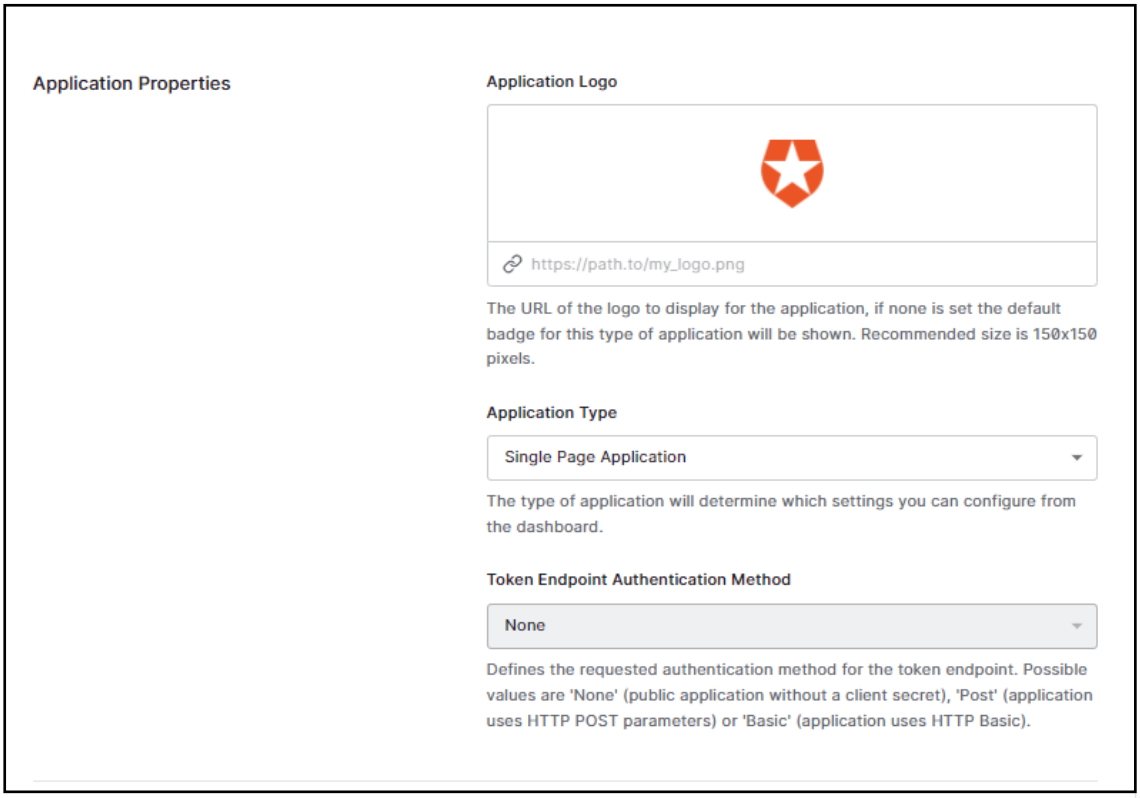
＊背景が灰色のものは変更できません。



内容説明

項目	内容
Name	アプリケーション名です。 Web アプリのログイン画面にも表示されます。
Domain	テナント単位のアクセス URL です。 Web アプリケーションや KrakenD に記入する項目です。
Client ID	本アプリケーションの ID です。 Web アプリケーションに記入する項目です。
Client Secret	本アプリケーションの暗号化キーです。 （本件では使用しません。）
Description	本アプリケーションの説明文です。 （本件では使用しません。）

・Application Properties　：　プロパティ項目です。



内容説明

項目	内容
Application Logo	管理画面におけるアプリケーションのロゴ（アイコン画像）です。特に変更する必要はありません。
Application Type	アプリケーションタイプです。
Token Endpoint Authentication Method	エンドポイント認証のトークンです。

・Application URIs　：　アプリケーションに関連する URI, URL です。

Application URIs

Application Login URI

https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. Learn more

Allowed Callback URLs

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://` . You can use Organization URL parameters in these URLs.

Allowed Logout URLs

A set of URLs that are valid to redirect to after logout from Auth0. After a user logs out from Auth0 you can redirect them with the `returnTo` query parameter. The URL that you use in `returnTo` must be listed here. You can specify multiple valid URLs by comma-separating them. You can use the star symbol as a wildcard for subdomains (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. Read more about this at https://auth0.com/docs/authenticate/login/logout

Allowed Web Origins

Comma-separated list of allowed origins for use with Cross-Origin Authentication, Device Flow, and web message response mode, in the form of `<scheme> "://" <host> [ ":" <port> ]` , such as `https://login.mydomain.com` or `http://localhost:3000` . You can use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Query strings and hash information are not taken into account when validating these URLs.

Allowed Origins (CORS)

Allowed Origins are URLs that will be allowed to make requests from JavaScript to Auth0 API (typically used with CORS). By default, all your callback URLs will be allowed. This field allows you to enter other origins if you need to. You can specify multiple valid URLs by comma-separating them or one by line, and also use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Query strings and hash information are not taken into account when validating these URLs.. You can use Organization URL placeholders in these URLs.

内容説明（適宜、各自の環境に合わせて各項目を設定してください）

項目	作業内容
Application Login URI	Web アプリ側で別途ログイン画面がある場合の URI です。 本件では特にないため空欄のままです。
Allowed Callback URLs	ログイン後の再表示を許可する URL です。 本件の全 Web アプリの URL を指定しています。 https://tb-gis-web.xxx.jp/mapbox/, https://tb-gis-web.xxx.jp/mapbox_template/, https://tb-gis-web.xxx.jp/mapbox_sample/, ・・・
Allowed Logout URLs	ログアウト後の再表示を許可する URL です。 本件の全 Web アプリの URL を指定しています。 https://tb-gis-web.xxx.jp/mapbox/, https://tb-gis-web.xxx.jp/mapbox_template/, https://tb-gis-web.xxx.jp/mapbox_sample/, ・・・
Allowed Web Origins	当テナントの呼び出し元を許可する Web の URL です。 ログインが、他の Web アプリ（サイト）からは呼び出せないように指定しています。 本件のルート URL に*を記載したものを指定しています。 https://tb-gis-web.xxx.jp/*
Allowed Origins (CORS)	当テナントの API へのリクエストを許可する URL です。 API が他の Web アプリ（サイト）からは呼び出せないように指定しています。 本件のルート URL に*を記載したものを指定しています。 https://tb-gis-web.xxx.jp/*

・その他

ID Token

ID Token Expiration

36000 seconds

This setting allows you to set the lifetime of the `id_token` (in seconds)

Refresh Token Rotation

Rotation

When enabled, as a result of exchanging a refresh token, a new refresh token will be issued and the existing token will be invalidated. This allows for automatic detection of token reuse if the token is leaked. In addition, an absolute expiration lifetime must be set. Learn more

Reuse Interval

0 seconds

The allowable leeway time that the same `refresh_token` can be used to request an `access_token` without triggering automatic reuse detection.

Refresh Token Expiration

Absolute Expiration

When enabled, a `refresh_token` will expire based on an absolute lifetime, after which the token can no longer be used. If rotation is enabled, an expiration lifetime must be set. Learn More

expiration lifetime must be set. Learn More

Absolute Lifetime

2592000 seconds

Sets the absolute lifetime of a `refresh_token` (in seconds).

Inactivity Expiration

When enabled, a `refresh_token` will expire based on a specified inactivity lifetime, after which the token can no longer be used.

Inactivity Lifetime

1296000 seconds

Sets the inactivity lifetime of a `refresh_token` (in seconds).

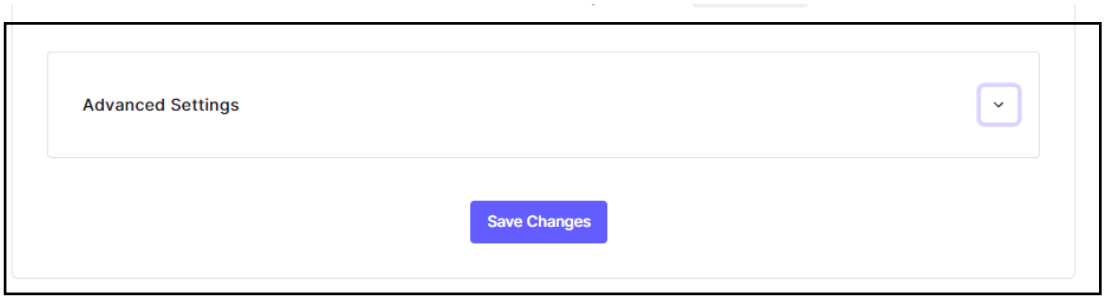
Advanced Settings

Save Changes

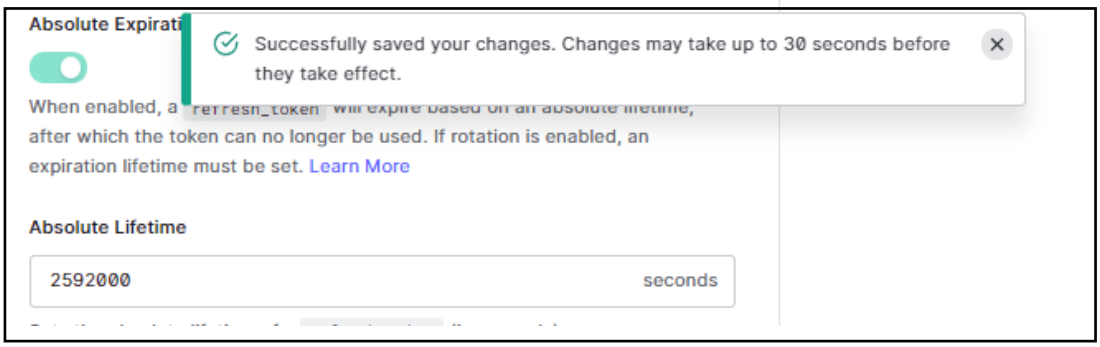
内容説明

項目	作業内容
ID Token	Expiration：発行された ID トークンの有効期限です。 ※API のアクセストークンではありません。 本件では特に使用しません。
Refresh Token Rotation	Web アプリにログイン後、API にアクセスする際のトークンを、自動的に更新可能にします。 （Web アプリで地図を表示した状態のまま、API へのアクセスをできるようにするためです。） 本件においては、デフォルトのままです。
Refresh Token Expiration	上記の拡張内容です。 本件においては、デフォルトのままです。
Advanced Settings	高度な設定内容です。 本件においては、デフォルトのままです。

※変更する場合は、入力後「Sava changes」ボタンを押し、設定内容を保存します。

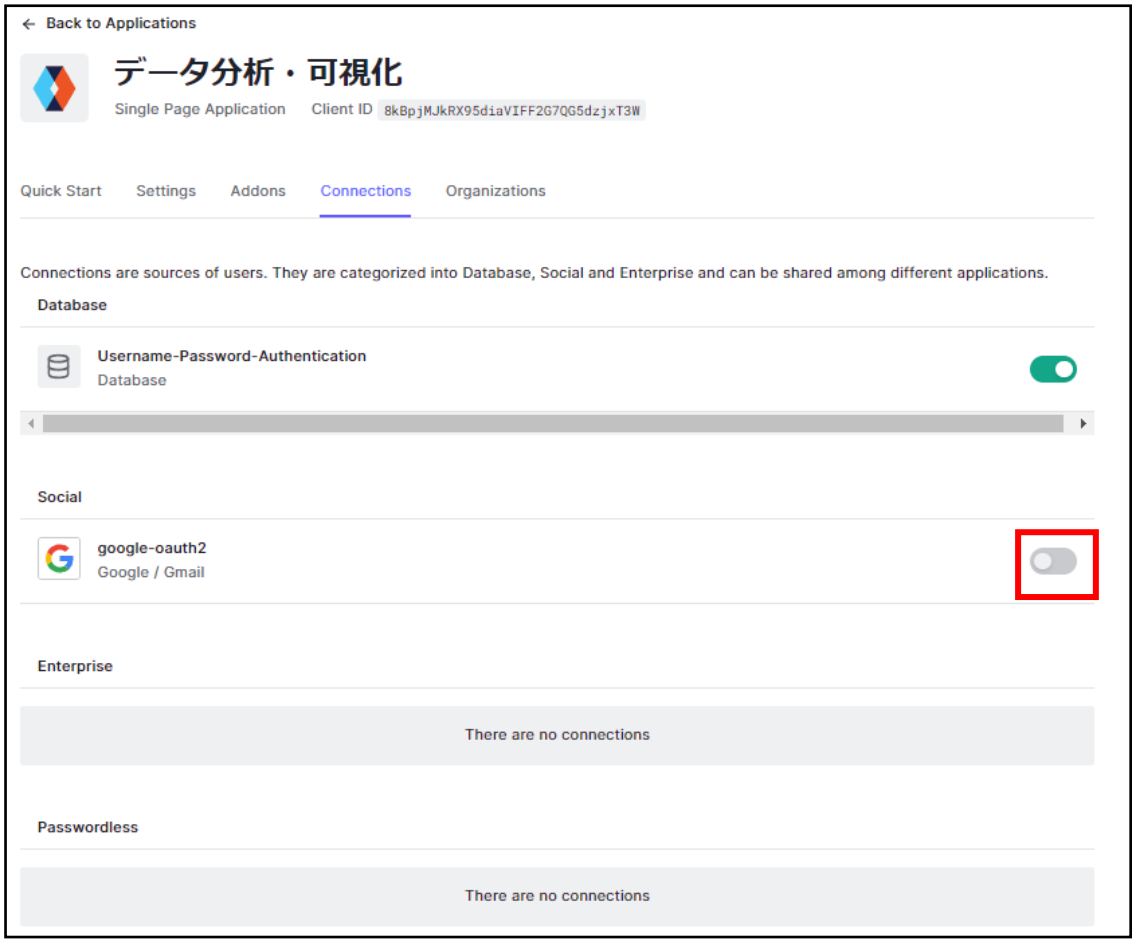


変更が成功すると、画面右上に下記のようなメッセージが表示されます。



②ユーザ接続方法の選択の確認

アプリケーションのタブ「Connection」をクリックします。



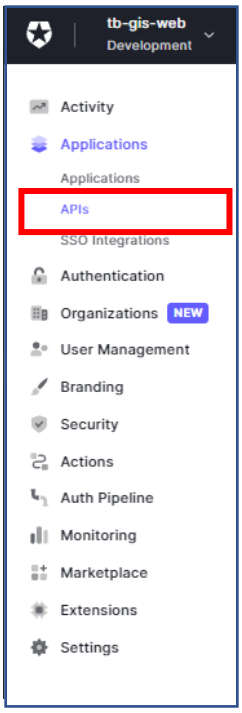
内容説明

項目	作業内容
Database： Username-Password-Authentication	ユーザ情報の管理が Auht0 内のデータベースであることを示します。 本件では、オン（有効：デフォルト）です。
Social： google-oauth2	Google アカウント（連携）を利用することを有効/無効にします。（有効にしても、Google アカウントのパスワードなどの機密情報を保持するものではありません。） 本件においてはメールアドレスのみのため、上記のように、オフ（無効）に変更しています。

9.2.3　　APIs の設定確認

①表示

メニューから「APIs」をクリックします。



API 一覧が表示されます。

②設定内容

上記アプリケーション（Auth0 Management API）をクリックします。

下記のように設定画面が表示されます。

← Back to APIs

Auth0 Management API

System API Identifier `https://tb-gis-web.jp.auth0.com/api/v2/`

Quick Start | Settings | Permissions | Machine to Machine Applications | Test | API Explorer

⚠ This API represents an Auth0 entity and cannot be modified or deleted. You can still authorize applications to consume this API.

General Settings

Id

61e79f0f34ec85003ffa1944

The API id on our system. Useful if you prefer to work directly with Auth0's Management API instead.

Name *

Auth0 Management API

A friendly name for the API. The following characters are not allowed `< >`

Identifier

https://tb-gis-web.jp.auth0.com/api/v2/

Unique identifier for the API. This value will be used as the `audience` parameter on authorization calls.

Token Settings

Token Expiration (Seconds) *

86400

Expiration value (in seconds) for `access tokens` issued for this API from the Token Endpoint.

Token Expiration For Browser Flows (Seconds) *

7200

Expiration value (in seconds) for `access tokens` issued for this API via Implicit or Hybrid Flows. Cannot be greater than the Token Lifetime value.

Signing Algorithm

RS256

Algorithm to be used when signing the `access tokens` for this API. You can find more information in this document.

Access Settings

Allow Skipping User Consent

If this setting is enabled, this API will skip user consent for applications flagged as First Party.

Save

内容説明

種類	項目	作業内容
General Settings	Id	本 API の Auth0 における一意番号です。 特に使用しません。
	Name	本 API の Auth0 における名称です。 特に使用しません。
	Identifier	アクセストークンを認証するために呼び出す URL です。 本件では、KrakenD（APIGateway）にて使用します。
Token Settings	Token Expiration (Seconds)	API のアクセストークンの有効期限（秒）です。 本件では、デフォルトの 1 日(86400 秒)とします。
	Token Expiration For Browser Flows (Seconds)	ハイブリッドフロー（Web サーバのバックエンドなど）における、アクセストークンの有効期限（秒）です。 変更できません。
	Signing Algorithm	アクセストークン署名のアルゴリズムです。 変更できません。
Access Settings	Allow Skipping User Consent	有効にすると、Web アプリにおける初回ログイン時のユーザ同意処理をスキップさせます。 本件では、同意処理（画面）は必要とし、オフ（無効）のままです。

10 API の追加と削除（データ取得 API）

データを取得する API を作成するときは、下記手順に従って追加・削除します。本システムで作成した各処理ファイルは https://github.com/nict-testbed-dalab：Data_api_Component リポジトリを参照してください。

10.1　本体の作成

・api 配下のディレクトリ「routers」に分類ごとのディレクトリを作成
　（amedas、people_flow など）
・処理ファイル「controllers.py」を作成
・処理ファイルは下記のように①に API 名を指定し、本体を実装します。

```
from fastapi import APIRouter
from starlette.requests import Request        ③

@router.get("/{API 名}")        ①
def 関数名(request:Request, {リクエストパラメータ})        ②
    本体処理
```

No	場所	説明
①	API 名	API(FastAPI 内部)における名称 ※外部から呼び出される API 名は KrakenD で記載します。
②	リクエストパラメータ	URL に記載されるリクエストパラメータを定義します。
③	import	上記は API を作成するための必要最小限のものです。 上記以外で、DB アクセスなど、必要に応じて追加します。

10.2　main.py に追加

下記のように作成した controllers をインポートし、app.include_router を追加します。

```
from routers.{作成したディレクトリ名}.controllers import router as {作成したディレクトリ名}_routers

＜中略＞
# router を登録する。
app.include_router({上記追加した router の別名})
```

※API を公開する手順は、下記「10. API 認証」を参照ください。

10.3　削除

削除する場合は、main.py から追加した行を削除します。
API 本体も削除します。（削除しなくても main.py から削除されれば呼び出されません）

11 API 認証

11.1 WEB サーバにおける API 認証付与手順（API 公開含む）

API のゲートウェイである KrakenD の設定ファイル(krakenD.json)に下記のような内容を記入します。先頭の「extra_config 」は 1 つだけですが、API 1 つ当たり 1 つの「endpoints（下記②～④）」のセットを記入します。

```
"extra_config": {
①     "github_com/devopsfaith/krakend-cors": {
            "allow_origins": ["*"],
            "allow_methods": [],
            "allow_headers":["Authorization"]
        }
},
"endpoints": [
  {
    "endpoint": "/api/XXXX",  ②
④   "extra_config": {
            "github.com/devopsfaith/krakend-jose/validator": {
                "alg": "RS256",
                "audience": ["https://tb-gis-web.jp.auth0.com/api/v2/ "],
                "jwk-url": "https://tb-gis-web.jp.auth0.com/.well-known/jwks.json"
                }
    },
    "backend": [
      {
③       "url_pattern": "XXXXXXX",
        "host": [
          "localhost:5000"
        ]
      }
    ]
  }
],
```

No	場所	説明
①	extra_config	github_com/devopsfaith/krakend-cors を記入します。 リクエストヘッダーからの認証パラメータ（Authorization）を取得するようにします。 記入内容は上記サンプルの通りです。
②	endpoints -> endpoint	API の表示名を記載します。 （ブラウザなどから参照する API 名）
③	endpoints -> extra_config	Auth0 の情報で、API ごとに記載します。（記載内容は同一） ---------- alg ：暗号化のアルゴリズム名です。（デフォルト） audience： API 用の Auth0 識別名です。 jwk-url ： 公開鍵の JSON ファイルパスです。
④	endpoints -> backend	url_pattern：FastAPI の API 名（相対パス）を記入します。 host：FastAPI のサーバ名（同一サーバであれば localhost）とポート番号を記入します。

・記入完了後、KrakenD を再起動します。

11.1.1 記載例１（ローカルディレクトリとして参照）

```
{
  "endpoint": "/api/data/{type}/{id}/{year}/{z}/{x}/{y}",
  "output_encoding": "no-op",
  "extra_config": {
        "github.com/devopsfaith/krakend-jose/validator": {
              "alg": "RS256",
              "audience": ["https://tb-gis-web.jp.auth0.com/api/v2/"],
              "jwk-url":          "https://tb-gis-web.jp.auth0.com/.well-known/jwks.json"
              }
  },
  "backend": [
    {
      "url_pattern": "/storage/data/{type}/{id}/{year}/{z}/{x}/{y}",
      "encoding": "no-op",
      "host": [
        "https://tb-gis-web.xxx.jp"
      ]
```

11.1.2 記載例２（外部参照）

```
{
  "endpoint": "/api/std/experimental_bvmap/{z}/{x}/{y}",
  "method" :"GET",
  "output_encoding": "no-op",
  "backend": [
    {
      "url_pattern": "/xyz/experimental_bvmap/{z}/{x}/{y}",
      "encoding": "no-op",
      "host": [
        "https://cyberjapandata.gsi.go.jp"
      ]
    }
  ]
```

11.2 Web アプリケーション側の認証設定手順

Auth0 のサンプルとしてある認証処理 app.js を利用します。
（URL : auth0-javascript-samples/app.js at master · auth0-samples/auth0-javascript-samples · GitHub）
※本サンプルアプリケーションでは、fetchAuthConfig におけるパスを変更しています。

11.2.1 Auth0 情報記載

auth_config.json に下記内容を記載します。

項目名	内容
domain	Auth0 のコンテナの URL です。
audience	API の識別 ID です。
clientId	Auth0 の使用するアプリケーションの ID です。

11.2.2 アクセストークン取得

認証処理 app.js で auth0 のインスタンスを生成し、下記のように getTokenSilently()を呼び出してアクセストークンを取得しており、実際にリクエストする際に使用します。

```
accessToken = await auth0.getTokenSilently();
```

11.2.3 データまたは JSON 取得時の認証（タイル地図以外）

API を呼び出してデータや JSON を取得する場合、リクエストヘッダーの Authorization に、取得したアクセストークンと接頭辞"Bearer "を付与して fetch します。

```
const response = await fetch(
API_URL,
{
   method: 'GET',
   mode: 'cors',
   headers:{
      "Authorization": "Bearer " + accessToken,
   }
}
);
```

項目または変数名	説明
API_URL	API の URL
method: 'GET'	リクエストメソッド名
mode: 'cors'	WEB アプリケーションと API の URL が異なる場合でも通信許可する。

11.2.4 WEB サーバを経由したタイル地図アクセスの認証

使用する地図ライブラリ(MapboxGL, iTowns)に従い、リクエストヘッダーにアクセストークンを組み込ませます。

例１）MapboxGL の場合

下記のように、mapboxgl.Map の transformRequest を用いて、リクエストヘッダーにアクセストークンを組み込ませます。

```
var map = new mapboxgl.Map({
  container: 'map',
  style: './XXXX.json',      // tiles に Web サーバの URL が記載された json
  transformRequest: (url, resourceType) => {
      if (resourceType === 'Tile') {
          return {
                  url: url,
                  headers: {
                      'Authorization': 'Bearer ' + accessToken
                  }
          }
      }
  }
```

例２）iTowns の場合

Source プロパティの networkOptions に

headers: { 'Authorization': 'Bearer: (アクセストークン)' }

を追加します。

参考 URL：http://www.itowns-project.org/itowns/docs/#api/Source/Source

11.2.5 WEB アプリケーションのログイン・ログアウト

・app.js の login,logout を呼び出します。

12 UDF の追加と削除

PostgreSQL のユーザ定義の関数（UDF）についての追加、削除などの手順を記載します。以下の手順はすべて、管理権限のあるユーザで PostgreSQL へ接続して、実施してください。

12.1 PostgreSQL への接続

1. postgres ユーザで PostgreSQL へ接続する。

環境は Ubuntu 20.04.3 LTS に PostgreSQL 12、python3 がインストールされている環境を想定しています。

```
$ sudo -i -u postgres
$ psql
```

2. plpython3u を有効化する。確認結果に plpython3u が含まれていることを確認する。

```
-- 有効化
postgres=# CREATE EXTENSION plpython3u;

-- 以下で plpython3u の有効／無効の確認
postgres=# select lanname from pg_language;

   lanname
------------
 internal
 c
 sql
 plpgsql
 plpython3u
```

3. ユーザ定義関数内で利用する Python のライブラリを追加する。

```
$ sudo -i -u postgres
$ sudo pip3 install [利用ライブラリ]
```

なお、本リポジトリで準備している関数は以下のライブラリを使用しています。
・pandas
・SciPy

12.2 UDF の追加

(1) UDF の追加

以下のクエリ文で python 実装を PostgreSQL へ登録します。

ただし、UDF として PostGIS の関数も登録されているため、関数名のフォーマット（接頭句など）を決めておくと、確認がしやすくなります。

```
postgres =# CREATE FUNCTION 関数名(引数 1 データ型 1, 引数 2 データ型 2,…)
postgres =# RETURNS 返却値のデータ型
postgres =# AS $$
postgres $# 以下 python の実装
postgres $# ・・・
postgres $# return 返却値
postgres $# $$ LANGUAGE plpython3u;
```

SQL ファイルを実行する場合は以下。

```
$ sudo -i -u postgres
$ psql -f [SQL ファイル]
```

また、UDF を追加したユーザと別のユーザが利用する場合、権限の付与が必要な場合があり、その際は以下で実行権限が付与されます。

```
postgres =#grant EXECUTE on FUNCTION [追加した UDF] to [利用するユーザ名];
```

(2) UDF の確認

以下のクエリ文で、これまでに追加した UDF を表示します。

```
postgres =# SELECT proname, prosrc FROM pg_proc WHERE proname = '関数名';
```

また作成した関数を一覧で確認する場合、pg_proc には PostGIS の関数も含まれるため、接頭句などが必要になります。

```
postgres =# SELECT proname,prosrc FROM pg_proc WHERE proname like '関数の接頭句%';
```

12.3 UDF の削除

以下のクエリ文で、これまでに追加した UDF を表示します。

```
postgres =# DROP FUNCTION '関数名';
```

もし、関数をオーバーライドしていた場合は、関数名だけでなく引数まで指定します。

本アプリで実装した「補間・集約ユーザ定義関数」は、
https://github.com/nict-testbed-dalab/ Data_convert_Component リポジトリを参照してください。

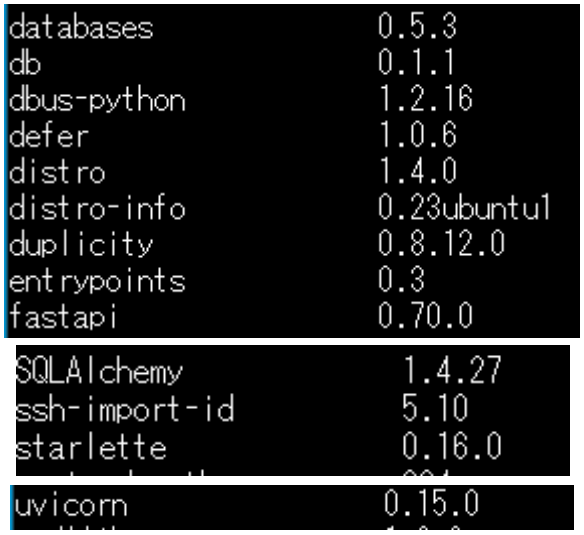
13 パッケージ更新

13.1 FastAPI

(1) 現バージョンを確認します。

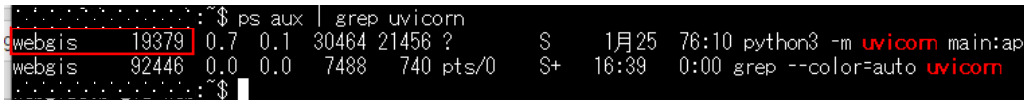
```
pip3 list
```

リスト結果例）



(2) FastAPI をいったん停止します。
下記コマンドを実行し、実行プロセスを表示します。

```
ps aux | grep uvicorn
```



次に、一覧の「python3 -m uvcorn」のプロセス ID を参照し、下記コマンドを実行します。（プロセス ID は起動ごとに異なります。）

```
kill 19379
```

(3) パッケージの更新（アップグレード）を実行します。

・fastapi、uvicorn

（db、databases、SQLAlchemy と starlette は、fastapi と同時に必要なバージョンに更新されます）

```
python3 -m pip install -U fastapi
python3 -m pip install -U uvicorn
```

(4)　更新後のバージョンを確認します。

```
pip3 list
```

最新のバージョン(0.89.1)であることを確認します。

```
databases          0.7.0
db                 0.1.1
dbus-python        1.2.16
defer              1.0.6
distro             1.4.0
distro-info        0.23ubuntu1
duplicity          0.8.12.0
entrypoints        0.3
fastapi            0.89.1
SQLAlchemy         1.4.46
ssh-import-id      5.10
starlette          0.22.0
uvicorn            0.20.0
```

(5)　起動

API のフォルダに移動し、uvicorn をバックグラウンドで実行します。

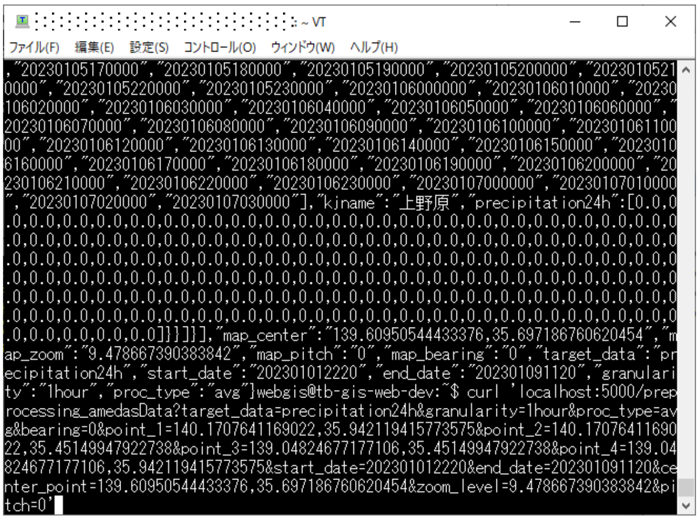
```
cd /home/webgis/api
nohup python3 -m uvicorn main:app --reload --port 5000 &
```

(6)　確認

サーバーコンソールから、下記 curl を実行し、動作することを確認します。

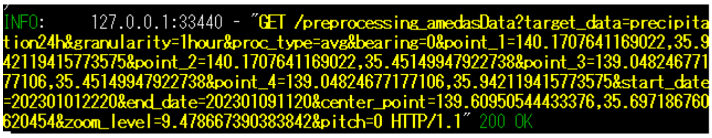
```
curl
'localhost:5000/preprocessing_amedasData?target_data=precipit
ation24h&granularity=1hour&proc_type=avg&bearing=0&point_1=14
0.1707641169022,35.942119415773575&point_2=140.1707641169022,
35.45149947922738&point_3=139.04824677177106,35.4514994792273
8&point_4=139.04824677177106,35.942119415773575&start_date=20
2301012220&end_date=202301091120&center_point=139.60950544433
376,35.697186760620454&zoom_level=9.478667390383842&pitch=0'
```

レスポンスが取得されたことと、API のログに下記のように OK が出力されていることを確認します。



```
,"20230105170000","20230105180000","20230105190000","20230105200000","2023010521
0000","20230105220000","20230105230000","20230106000000","20230106010000","20230
106020000","20230106030000","20230106040000","20230106050000","20230106060000","
20230106070000","20230106080000","20230106090000","20230106100000","202301061100
00","20230106120000","20230106130000","20230106140000","20230106150000","2023010
6160000","20230106170000","20230106180000","20230106190000","20230106200000","20
230106210000","20230106220000","20230106230000","20230107000000","20230107010000
","20230107020000","20230107030000"],"kjname":"上野原","precipitation24h":[0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0]}}]}],"map_center":"139.60950544433376,35.697186760620454","m
ap_zoom":"9.478667390383842","map_pitch":"0","map_bearing":"0","target_data":"pr
ecipitation24h","start_date":"202301012220","end_date":"202301091120","granulari
ty":"1hour","proc_type":"avg"}webgis@tb-gis-web-dev:~$ curl 'localhost:5000/prep
rocessing_amedasData?target_data=precipitation24h&granularity=1hour&proc_type=av
g&bearing=0&point_1=140.1707641169022,35.942119415773575&point_2=140.17076411690
22,35.45149947922738&point_3=139.04824677177106,35.45149947922738&point_4=139.04
824677177106,35.942119415773575&start_date=202301012220&end_date=202301091120&ce
nter_point=139.60950544433376,35.697186760620454&zoom_level=9.478667390383842&pi
tch=0'
```

API のログ



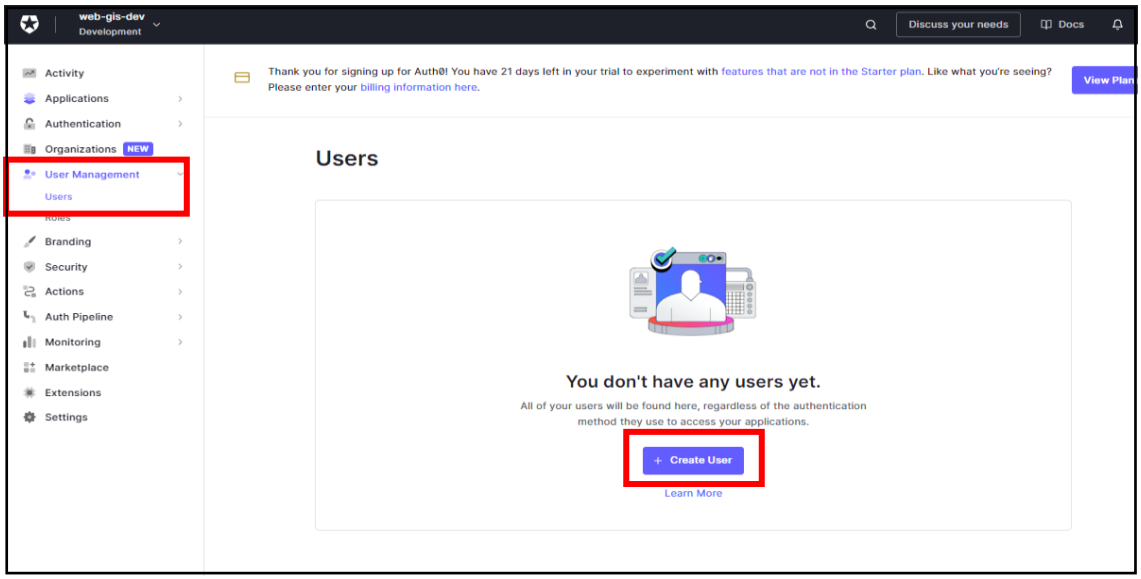
```
INFO:     127.0.0.1:33440 - "GET /preprocessing_amedasData?target_data=precipita
tion24h&granularity=1hour&proc_type=avg&bearing=0&point_1=140.1707641169022,35.9
42119415773575&point_2=140.1707641169022,35.45149947922738&point_3=139.048246771
77106,35.45149947922738&point_4=139.04824677177106,35.942119415773575&start_date
=202301012220&end_date=202301091120&center_point=139.60950544433376,35.697186760
620454&zoom_level=9.478667390383842&pitch=0 HTTP/1.1" 200 OK
```

14 ユーザ管理

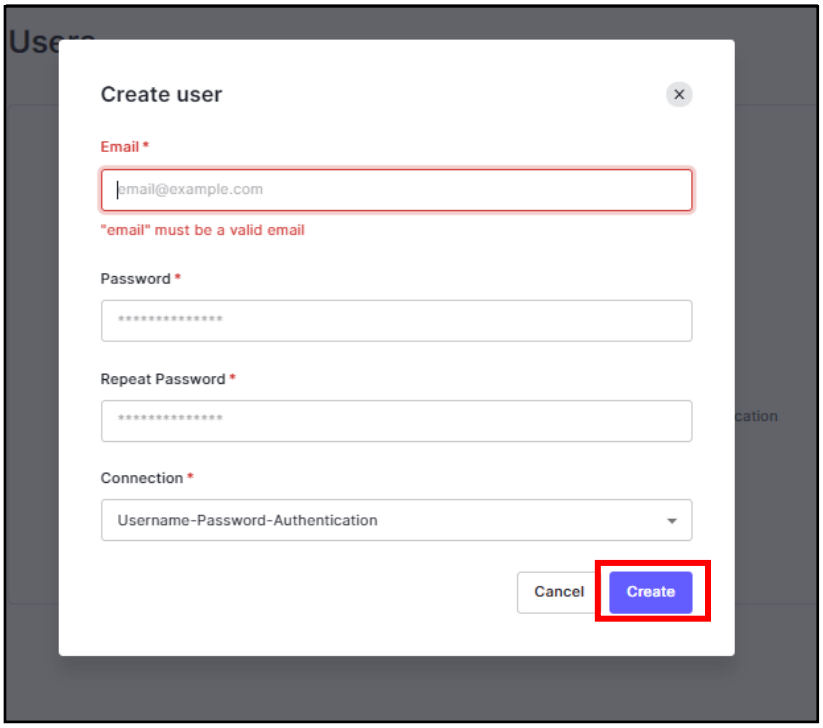
本システムを利用するユーザの登録などをします。

14.1 ユーザ登録

Auth0 管理画面のメニューから「User Management-> Users」をクリックします。
下記のように Users 画面が表示されます。



上記「+ Create User」ボタンを押します。
下記のようなダイアログ画面が表示されます。



内容説明

項目	入力内容
Email	メールアドレスを入力します。
Password	パスワードを入力します。 ※該当ユーザに通知する必要があります。
Repeat Password	上記を再入力します。
Connection	ユーザ情報の管理方法リストです。 （設定画面で選択した１つのみです。）

上記で「Create」ボタンを押し、登録します。

※登録すると、ユーザのメールアドレスに、確認メールが自動送信されます。

14.2 Web アプリケーションにおけるユーザ初回ログイン

ユーザが本 Web アプリケーションを表示し、「ログイン」ボタンを押すと、下記のようなログインダイアログ画面が表示されます。

本システムに登録したメールアドレスとパスワードを入力し、「続ける」ボタンを押します。

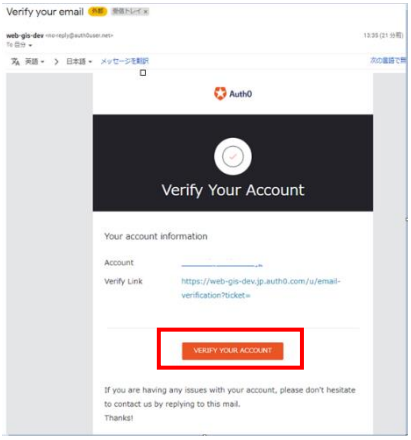
初めてログインすると、下記のような認証許可ダイアログ画面が表示されます。

内容を確認し「許可」ボタンを押します。（誤って「拒否」ボタンを押した場合は、再度ログインします。）

14.3 メールアドレス確認

登録されたメールアドレスが正しいことをユーザ自身で確認します。

・Auth0 管理者がユーザ登録した際に、ユーザに下記のようなアカウント確認依頼メール送信されます。



・送信されたメール本文が HTML 形式で表示されていれば「VERIFY YOUR ACCOUNT」ボタンを押します。

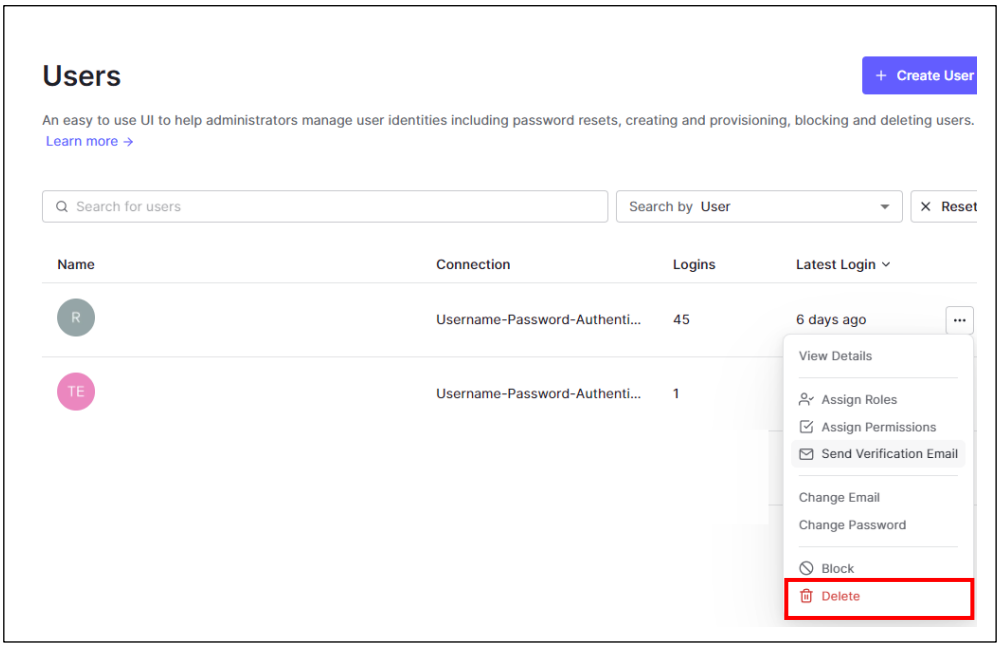
（テキストの場合は Verify Link をコピーしてブラウザに表示します。）

・すると下記のようなメール確認済みダイアログが表示されます。（表示のみ）

・表示後、ブラウザを閉じます。

14.4 ユーザ削除

Users 画面で、削除するユーザの右側のボタンを押します。
下記のような項目が表示されるので、「Delete」を押します。



項目に表示されているように、メールアドレス変更、パスワード変更なども可能です。