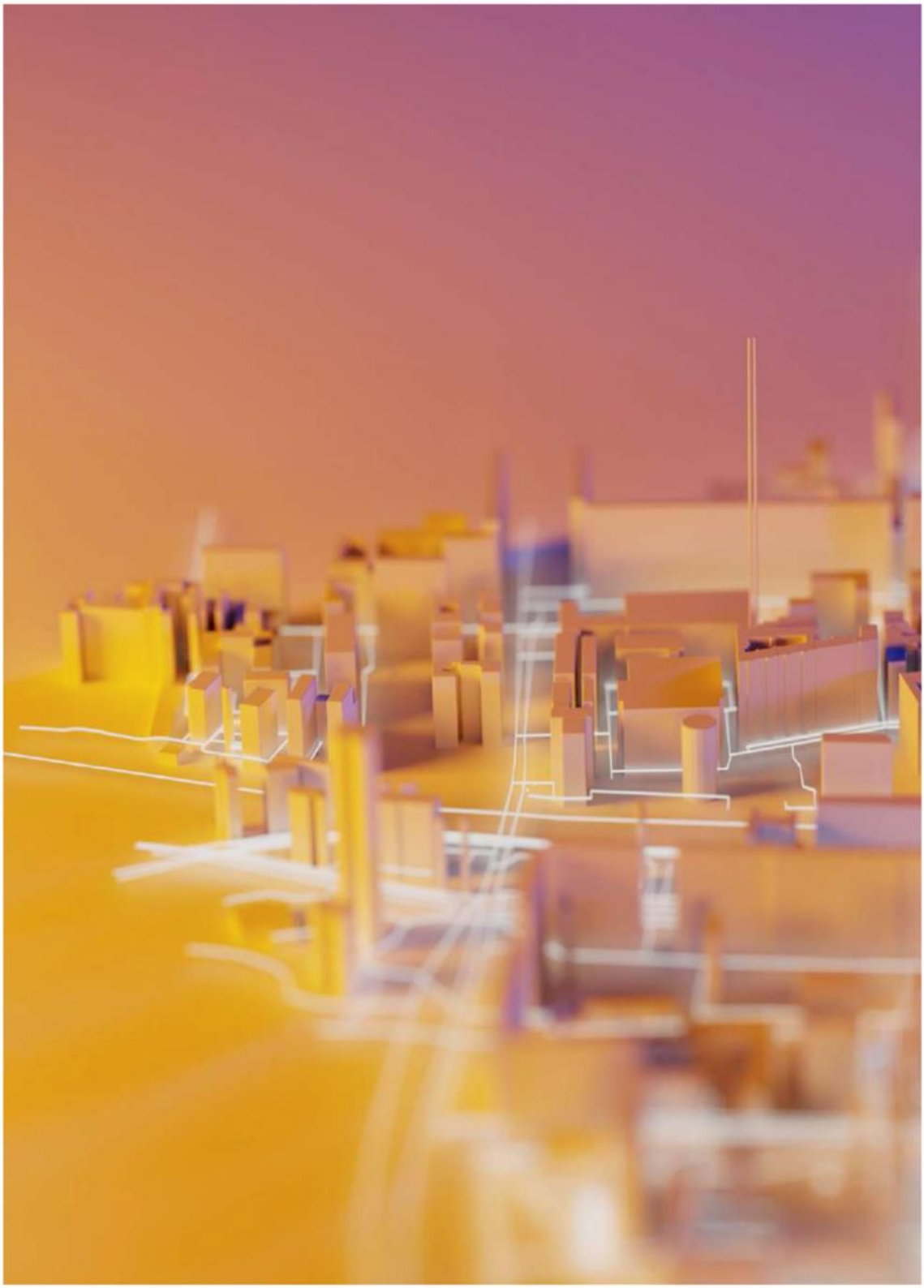
PLATEAU
by MLIT

Handbook of 3D City Models

3D都市モデル導入のためのガイドブック



PLATEAU VIEW 構築マニュアル

series No. 09

PLATEAU VIEW Setup Manual

はじめに

- Project PLATEAUでは、 2020年度に3D都市モデルのPLATEAU VIEW構築のための実証調査を実施した。本実証調査は、3D都市モデル及びこれを活用したユースケース開発のための可視化環境である「PLATEAU VIEW」を開発することで、3D都市モデルがもたらすソリューションの価値を検証することを目的としている。
- 2021年度には、クリッピング機能や日射シミュレーション機能など、Web上で実施可能な解析機能を追加「PLATEAU VIEW 1.1」をリリースした。また、「第3章　PLATEAU VIEWの構築手順」の内容を大幅に拡充し、PLATEAU VIEWを構築するためのチュートリアルを充実させるとともに、Project PLATEAU GitHub上でTerria.js用カタログ生成アプリとそのチュートリアルを公開した。
- 2022年度には、データセットの可視化機能に限定されていた「PLATEAU VIEW 1.1」を発展させ、データ登録・管理・配信機能及び機能追加を行った「PLATEAU VIEW 2.0」を開発し、GitHUB上でソーススクリプトを公開した。さらに、PLATEAU VIEWを利用するための技術チュートリアルを公式ウェブサイト上で公開した。
- 2023年度には、「PLATEAU VIEW 2.0」をUIUXの観点からの見直しと、3DCG技術を利用した描画品質の向上を図るとともに、PLATEAU CMSのデータ配信APIやコンテンツ管理機能の改善を行った「PLATEAU VIEW 3.0」を開発した。
- PLATEAU VIEWの構築を検討するに当たっては、Project PLATEAUのオープンデータの思想に基づき、できるだけオープンソースのフレームワークを利用することで、ベンダーロックを回避する形でシステム構成を行うことを心掛けた。また、オープンソースの利用には、自治体等が低コストで類似のシステムを構築できることや、技術者コミュニティとの連携によるシステムの持続的な発展が期待されること等のメリットもある。
- 本マニュアルは、PLATEAU VIEWの開発により得られた成果をもとに、その機能、システム環境、仕様、構築手法等を解説することで、自治体や民間企業、技術者コミュニティ等に所属する多様なプレイヤーが3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発への参画のすそ野を広げることを目的とするものである。
- 自治体や民間企業、技術者等の多くの方に本マニュアルを参照していただき、Project PLATEAUの技術コミュニティがさらに発展することを期待する。

アップデートノート

2024.3.22　「PLATEAU VIEW構築マニュアル（旧実証環境構築マニュアル）ver4.0」

2023年度の調査結果をふまえ、以下の項目を改訂した。

- 「実証環境構築マニュアル」を「PLATEAU VIEW構築マニュアル」と改称し、2023年度における最新仕様のみをマニュアルに反映した。
- 「第3章　PLATEAU VIEWの構築手順」にAWS向けセットアップや、WebAR向け構築手順を追加。

2023.3.22　「実証環境構築マニュアル ver3.0」

2022年度の調査結果をふまえ、以下の項目を改訂した。

- 「第1編　PLATEAU VIEW 2.0」を新たにリリースしたPLATEAU VIEW 2.0編として拡充
- 「第2編　PLATEAU VIEW 1.1」をPLATEAU VIEW 1.1編として改訂

2022.3.25　「実証環境構築マニュアル ver2.0」

2021年度の調査結果をふまえ、以下の項目を改訂した。

- 「1.4　ソフトウェア構成」を新たにリリースしたPLATEAU VIEW 1.1の構成にアップデート
- 「第2章　実証環境の構築手順」の内容を大幅に拡充
- 「第3章　ユーザーマニュアル」をPLATEAU VIEW 1.1に合わせて改訂

2021.3.26　「実証環境構築マニュアル ver1.0」

■目次

■目次

第１章　PLATEAU VIEWの構成

1.1 本マニュアルの目的

PLATEAU VIEWとは、3D都市モデル及びこれを活用したユースケース開発のために可視化環境を提供するプログラム、サーバー、データ等の一連のシステムをいう。具体的な機能としては、3D都市モデルそれ自体を可視化することに加え、3D都市モデルと共に分析やシミュレーション等に用いられる各種データの可視化も行う。これにより、3D都市モデルの提供価値を検証することができる。

2020年度のProject PLATEAUでは、ウェブ上で閲覧可能なViewer「PLATEAU VIEW 1.0」を開発し、ウェブサイト「PLATEAU」上で公開した。続く、2021年度には、PLATEAU VIEW 1.0に機能追加を行った「PLATEAU VIEW 1.1」を開発し、アップデートを行った。

2022年度では、PLATEAU VIEW 1.1を発展させ、データ登録・管理・配信機能及び機能追加を行った「PLATEAU VIEW 2.0」を開発し、アップデートを行った。

2023年度には、「PLATEAU VIEW 2.0」をUIUXの観点からの見直しと、3DCG技術を利用し、「PLATEAU VIEW 3.0」を開発した。

PLATEAU VIEW 3.0は、初見のユーザーが離脱しにくく、快適にデータを閲覧するためのUIUXの再設計や、データをより美しく描画するための3DCG技術導入等を行なった。その他にも、継続的にProject PLATEAUにおけるデータの更新を可能にするため、データ管理システム（PLATEAU CMS）のアップデート等も行なった。

本マニュアルでは、PLATEAU VIEW 3.0の機能、システム環境、仕様、構築手法等を解説することで、自治体や民間企業等が3D都市モデルの可視化環境を構築する際に参照できる知見を提供し、3D都市モデルの整備及びこれを活用したユースケース開発を促進することを目的とするものである。

なお、PLATEAU VIEW 3.0、PLATEAU VIEW 2.0及びPLATEAU VIEW 1.1のソースコードについては、Project PLATEAUのGitHubにおいてオープンソースとして公開しているので、参考にして頂きたい。

- PLATEAU VIEW 3.0
  - Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-3.0
- PLATEAU VIEW 2.0
  - Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-2.0
- PLATEAU VIEW 1.1
  - Project PLATEAU GitHub : https://github.com/Project-PLATEAU/PLATEAU-VIEW-1.1

1.2 全体構成

Project PLATEAUを構成するウェブサイトの構成について、以下に全体図を示す。

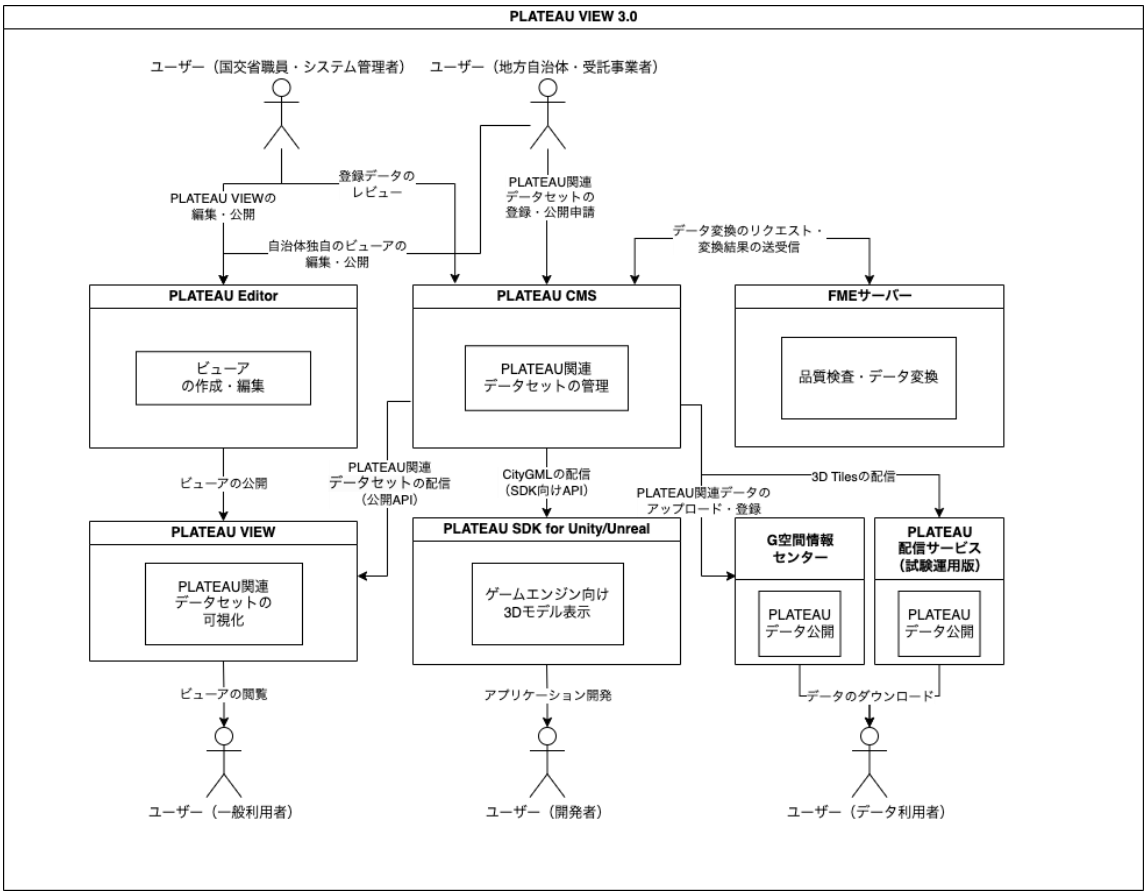
1.2.1 Project PLATEAU ウェブサイトの構成

表　Project PLATEAU ウェブサイトの構成

1.2.2 システム全体構成

PLATEAU VIEW 3.0を構成するシステム及び想定ユーザーを解説する。PLATEAU VIEW 3.0は複数のシステムで構成されており、以下に全体図を示す。

表　PLATEAU VIEW 3.0　システム全体構成



1.2.3 用語集

PLATEAU VIEW 3.0において独自に使用される用語をまとめる。

表　用語集：ユーザー

用語	説明
国交省職員・システム管理者	PLATEAU CMS上での登録データのレビュー、PLATEAU Editor上での公開ページの編集等を行う。
地方自治体・受託事業者	PLATEAU CMS上でPLATEAU関連データセットの登録を行う。
データ利用者	G空間情報センターへ登録された3D都市モデルデータ、ユースケースデータを利用する。
一般利用者	PLATEAU VIEWを利用し、PLATEAU関連データセットの閲覧を行う。
アプリ開発者	PLATEAU SDK for Unity/Unrealを利用し、3D都市モデルデータを利用したアプリケーション開発を行う。

表　用語集：システム

用語	説明
PLATEAU CMS	PLATEAU関連データセットを管理する。
PLATEAU Editor	PLATEAU VIEW・自治体独自のViewerを編集・公開する。
PLATEAU VIEW	PLATEAU関連データセットを可視化する。
FME Flow	3D都市モデルデータの品質検査・データ変換等を行う。
PLATEAU SDK for Unity/Unreal	Unity・Unreal Engine向けSDK。3D都市モデルデータをゲームエンジン上で描画する。
G空間情報センター	3D都市モデルデータ、ユースケースデータをオープンデータとして公開する。
PLATEAU配信サービス（試験運用版）	PLATEAUデータセットをWeb APIとして開発者向けに公開する。

表　用語集：データ種別

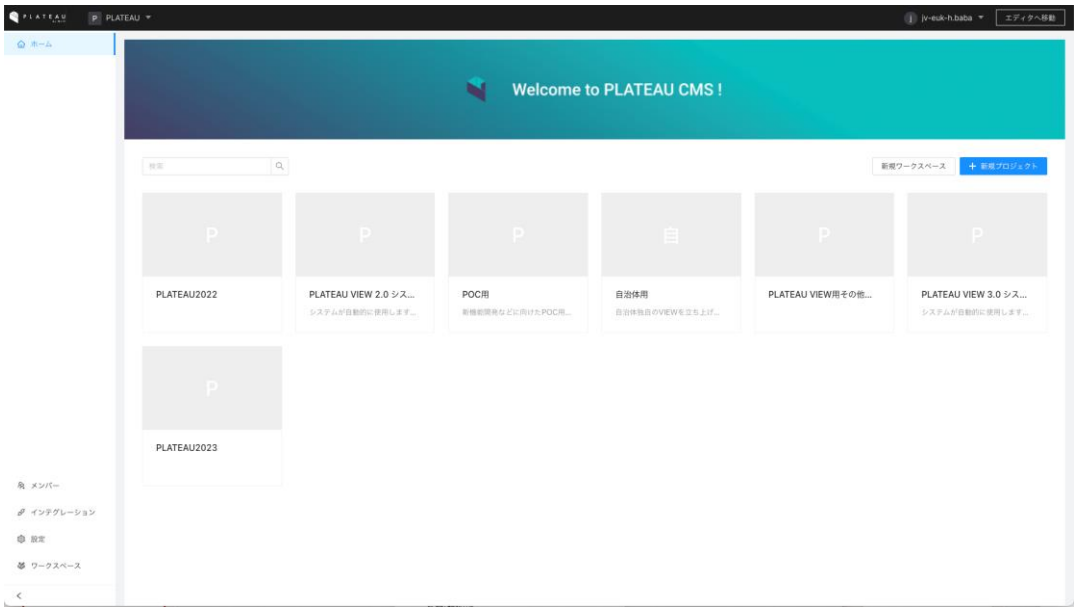
用語	説明
3D都市モデルデータ	都市局が定める「３D都市モデル標準製品仕様書」に準拠して作成された3D都市モデルのデータ。CityGML2.0形式で作成される。
ユースケースデータ	3D都市モデルデータに重畳して表示することを目的として作成されたユースケースに関するデータ。3D浸水想定区域図や動的データ、シミュレーションデータなど。
関連データセット	PLATEAU VIEWで閲覧可能な、3D都市モデルデータ及びユースケースデータ以外のデータ。避難施設、ランドマーク、行政界など。
PLATEAUデータセット	3D都市モデルデータ、ユースケースデータ、関連データセットの総称。

表　用語集：PLATEAU CMS

用語	意味	身近な例
アカウント	管理者、自治体・受託事業者、ユースケースデータの登録者それぞれに1つずつ与えられるユーザーアカウントを指す。	--
ワークスペース	複数のユーザーが同じワークスペースで作業できる場所のことを指す。PLATEAU VIEW 3.0プロジェクトでは、管理者、自治体・受託事業者、ユースケースデータの登録者が全員同じワークスペース「PLATEAU」で操作する。	Slackのワークスペース・GitHubのオーガナイゼーション
プロジェクト	ワークスペースに複数作成可能で、データ管理の目的に応じて作成する。管理者、自治体・受託事業者、ユースケースデータの登録者が全員同じプロジェクトで操作をする。	GitHubのレポジトリ
モデル	データを管理する単位で、管理者のみが、管理するデータのスキーマを設定できる。プロジェクトの複数作成が可能。	ExcelやRDBにおける表
スキーマ	モデルのデータ構造を指す。どんなフィールドがどんな型のデータを持つかや、制約条件などを定義する。管理者のみが定義可能で、自治体・受託事業者、ユースケースデータの登録者はこのスキーマに沿ってデータを入力する。	--
アイテム	管理するデータの最小単位で、スキーマに沿って実際のデータをアイテムとして登録する。3D都市モデルデータ登録者、ユースケースデータの登録者は対象データをアイテムとして入力する。	ExcelやRDBにおける行
フィールド	管理するデータの属性を指す。整数値、文字列などのデータ型を管理者のみが設定する。	ExcelやRDBにおける列
アセット	アップロードされたファイルを指す。サイズ・作成日時・URL・種類などのメタデータを持つ。複数ファイルの集まりをまとめて1つのアセットとして扱うことができる。管理者、自治体・受託事業者、ユースケースデータの登録者はPLATEAU関連データセットをアセットとしてアップロードする。	--
公開API	外部からアクセスされるPLATEAU CMSの認証不要のAPIを指す。登録したアイテムを公開リクエストで承認されるとそのアイテムが公開APIから配信されるようになる。PLATEAU VIEW 3.0 ではこの公開APIから配信されたデータがPLATEAU VIEWで使用され、自動的にデータカタログに掲載される。	--
公開リクエスト	登録したアイテムはすぐにはAPIとして配信されない。これを公開するためには管理者の承認を得ることが必要である。PLATEAU CMSでは、作成・更新したアイテムに対して、PLATEAU CMS上で公開の申請を出すことができる。この申請が承認されるとそのアイテムのデータが公開され、公開API経由で配信される。アイテムごとに個別に公開状況が管理されており、それぞれのアイテムごとに公開リクエストが必要である。	GitHubのプルリクエスト
インテグレーション	PLATEAU CMS以外の外部アプリケーションとの連携機能を指す。PLATEAU CMSのワークスペースにインストールすることで、インテグレーションAPIを利用してデータの取得や変更を行ったり、任意のイベント発生時にWebhookを利用して外部アプリケーションと連携を行うことができる。	Slack App

1.3 PLATEAU CMS
1.3.1 PLATEAU CMSとは

1.2のシステム全体構成で述べたように、PLATEAU VIEW 3.0のシステムの機能は、大きく「データの管理」と「データの可視化」に分かれる。PLATEAU CMSは、このうち「データの管理」を担うシステムのことを指す。



CMSとは「コンテンツ管理システム（Content Management System）」の略で、一般的な概要は以下のとおりである。

> コンテンツ管理システム（CMS）は、企業がデジタルコンテンツを管理するのに役立ちます。チーム全体がこれらのシステムを使って、コンテンツの作成、編集、整理、公開を行うことができます。コンテンツを保存する単一の場所として機能し、組み込み（または設計された）ワークフローを使用して、共同デジタルコンテンツ管理および作成のための自動化されたプロセスを提供します。役割に応じて、個人にはさまざまな特権と責任が与えられます。例えば、著者は作品を投稿し保存することができますが、編集者は作品を修正し公開することができます。管理者は、こうした作業をすべて行えるだけでなく、組織内の他の人にコンテンツの更新や改訂の許可を与えることもできます。
>
> CMSでは、最小限の技術コストでWebサイトやWebサイトのコンテンツを作成・管理できるため、プロジェクト・マネージャーやトラフィック・マネージャーのような役割を果たす必要なしに、より優れたコンテンツの作成に集中することができます。CMSは、コンテンツ管理のための簡単で費用対効果の高いソリューションを提供します。これにより、企業は専任のコンテンツ開発チームに投資しなくても、コンテンツを管理・配信することができます。
>
> 引用元：https://www.oracle.com/jp/content-management/what-is-cms/

この中でもPLATEAU CMSは、管理対象のコンテンツをAPIを通じて提供することを前提とした「ヘッドレスCMS」に位置付けられる。管理者だけでなく、地方自治体・受託事業者やユースケースデータの登録者などさまざまな事業者がPLATEAU VIEW 3.0で公開するPLATEAU関連データセットを一元管理し、APIとして公開することができるシステムである。

（1）使用ソフトウェア・サービス

上記システムを構築するために、オープンソースソフトウェア（OSS）と、有償のクラウドサービスを組み合わせて利用している。クラウドサービスの詳細については次項の各コンポーネントの説明で述べる。

表　使用ソフトウェア・サービス一覧

項目	項目	説明
FME	有償クラウドサービス	データの品質検査と変換を行うアプリケーション。詳しい説明は1.4を参照。
G空間情報センター	オープンデータ・ストレージサービス	官民問わずさまざまな主体により整備・提供される多様な地理空間情報を集約し、利用者がワンストップで検索・ダウンロードし利用できる、産学官の地理空間情報を扱うプラットフォーム。OSSのCKANを用いて構築されている。
Auth0	有償クラウドサービス	Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証・認可機能を組み込むことができる。PLATEAU Editorも Auth0を使用して認証・認可機能を実現している。
SendGrid	有償クラウドサービス	Twillio社が提供するクラウドサービス。企業や開発者がアプリケーションやWebサイトから大量のメールを配信するために使用される。PLATEAU CMSではご意見ご要望のメール送信で使用している。
Google Cloud Platform（GCP）	有償クラウドサービス	Google社が提供するクラウドコンピューティングプラットフォーム。PLATEAU CMSを動作させるためのサーバーや、ファイルを保存するためのサーバーをGCP上に構築している。
MongoDB Atlas	有償クラウドサービス	MongoDB社が提供するクラウドサービス。保守運用が自動化されたマネージドなMongoDBを提供している。MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。PLATEAU CMSはデータベースとして MongoDBを使用している。

PLATEAU CMSはその他さまざまな技術を組み合わせて構築されている。

PLATEAU CMSで内部的に利用されている技術・ライブラリ等

項目	説明
Go	プログラミング言語の1つで、Google社によって開発された。構文がシンプルであり、かつ処理が高速な言語であり、主にバックエンド開発に用いられる。PLATEAU CMSのバックエンド実装に利用されている。
TypeScript	プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU CMSではフロントエンド向け開発に利用されている。
React	Meta（旧Facebook）社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU CMSではフロントエンド向け開発に利用されている。
CesiumJS	デジタル3D地球儀上にさまざまな情報を描画することができる地図エンジン。Webブラウザ上で動作し、WebGLを用いて描画を行うため、PCやスマートフォンで閲覧することができる。PLATEAU CMS上では、データプレビューで使用している。
Resium	React上でCesiumJSを手軽に利用可能にするコンポーネントを提供するライブラリ。Eukarya開発。PLATEAU CMS上では、データプレビューで使用している。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU CMSではバックエンドとフロントエンド間の通信に利用されている。
Terraform	HashiCorp社によって開発された、infrastructure-as-codeを実現するためのソフトウェアであり、サーバー構成をコードとして宣言的に管理をできるようにする。

（３）PLATEAU CMSの主な機能

PLATEAU CMSでは主に以下の機能が利用可能である。詳しい使い方は、2.1を参照されたい。

- ワークスペースの作成・ユーザーの招待
- プロジェクトの作成
- スキーマの定義
- コンテンツの登録・編集
- アセットの登録・Zipファイルの解凍・プレビュー
- コンテンツの公開リクエスト
- コンテンツの公開（公開APIとしてコンテンツを公開可能）
- インテグレーションの作成・インストール（外部システムとの連携が可能で、本システムではFME Flowとの連携等で利用。）

（４）対応データフォーマット等

アセットの対応データフォーマット

PLATEAU CMSでは、全てのフォーマットの静的ファイルをアップロードが可能であるが、特に以下のファイルフォーマットに関してはプレビュー機能をサポートしている。

- 画像データ
    - PNG
    - JPEG
    - SVG
    - GIF
- GISデータ
    - GeoJSON
    - CZML
    - KML
    - Mapbox Vector Tiles（MVT）
    - glTF（glb）
    - 3D Tiles

スキーマのフィールド型

スキーマのフィールド型としては、以下のデータ型をサポートしている。

- テキスト: 短文向けのフィールド
- テキストエリア: 長文向けのフィールド
- マークダウン: マークダウンのフィールド
- アセット: アセットをリンクするためのフィールド
- 日付: 日付のフィールド
- 真偽値: 真偽値のフィールド
- 選択: 選択式のフィールド
- 整数値: 数値のフィールド
- URL: URLのフィールド
- 参照: 他のモデルを参照するフィールド
- グループ: 複数のフィールドをまとめるフィールド

PLATEAU VIEW 3.0対応データフォーマット一覧（1/2）

フォーマット				説明
	単体	Zip 7Z	URL 指定	
GeoJSON	✓		✓	•JSON形式で記述されるGISファイルフォーマット。点、線、面のベクトルデータの表示に対応している。 •サンプルコードは"図1：GeoJSONファイル記述例（ポリゴンデータの場合）"を参照。 ＜対応可能事項＞ •RFC 7946で定義されたGeoJSON •ジオメトリのCRS：WGS84（EPSG:4326） ＜特記事項＞ 以下の形式については未対応である。 •GeoJSON 2008 •TopoJSON •CRSの指定*1 •標準仕様外のジオメトリ（Circleなど） *1：座標参照系（CRS:Coordinate Reference System）
CZML	✓		✓	•CesiumJS上でのデータ表現に対応したJSON形式のGISファイルフォーマット。 •CesiumJSの機能を使用してCZMLを表示するため、CZMLの仕様内でPLATEAU VIEW 3.0固有の制約はない。 •＜特記事項＞ •CZMLとその他関連するファイルをZip又は7zファイルに圧縮・同梱することで、CZMLから相対パスで参照可能な別のデータセットを同梱することができる。この場合、CZMLはZipファイルのルート直下に置かれており、Zipファイル名と拡張子を除く部分が同じである必要がある。 •CZMLのdescription中では、相対パスや相対URLは使用できない。インフォボックス内に画像を表示させたい場合は、インターネット上で公開されている画像を絶対URL（http又はhttpsから始まるURL）で指定するか、CMSに画像だけを先にアップロードしてその画像の絶対URLを取得し、使用することができる。
3D Tiles	✓		✓	•複数ファイルから構成されるデータフォーマットであるため、CMSにアップロードする場合は、それらをZip又は7zファイルに圧縮する必要がある。 ＜特記事項＞ •圧縮する際には、ルート直下にtileset.jsonファイルを格納する必要がある。 ＜例外＞ •b3dmなど他のファイルはtileset.json内で相対パスが正しく定義されていれば、フォルダーを挟んでも問題ない）。

PLATEAU VIEW 3.0対応データフォーマット一覧（1/2）

フォーマット				説明
	単体	Zip 7Z	URL 指定	
MVT		✓	✓	•拡張子は .mvt に対応する。 •複数ファイルから構成されるデータフォーマットなので、CMSにアップロードする場合は、それらをZipまたは7zファイルに圧縮してから必要がある。その場合、ルートから {z}/{x}/{y}.mvt のようにファイルを配置する。 ＜特記事項＞ •CMSでレイヤー名を指定しないと表示されない。レイヤー名はカンマ区切りで複数指定可能である。 •FMEでMVTへのデータ変換を行った際に出力されるmetadata.jsonに対応している。 •metadata.jsonの同梱する場合には、VIEWでのカメラボタン押下時にカメラ位置がその内容に応じて自動的に移動することができる。 •metadata.jsonがルートに存在しない場合、カメラ移動は自動的に行われないため、あらかじめEditorでカメラ位置を手動設定する必要がある。 •URL指定の場合、 {z}/{x}/{y}.mvt の指定がURL中にない場合は、それがURLの最後に自動的に付加されたものと同じ扱いになる。
Tiles		✓	✓	•複数ファイルから構成されるデータフォーマットであり、XYZ軸で分割された画像タイルである。CMSにZip形式でアップロードする場合は、ルートから {z}/{x}/{y}.png のようにファイルを配置する。 ＜特記事項＞ •URL指定の場合、 {z}/{x}/{y}.png の指定がURL中にない場合は、それらの文字列がURLの最後に自動的に付加される。
WMS （Web Map Service）			✓	•URLで指定する場合、レイヤー名の指定が必須となる。レイヤー名はカンマ区切りで複数指定可能。
TMS （Tile Map Service）		✓	✓	•複数ファイルから構成されるデータフォーマットであり、CMSにZip形式でアップロードする場合は、ルートから {z}/{x}/{y}.png のようにファイルを配置する。 ＜特記事項＞ •URLで指定する場合は、tilemapresource.xmlへのURLではなく、その親を指定する。 【誤】 https://example.com/tms/tilemapresource.xml 【正】 https://example.com/tms •PNG画像のみ対応。拡張子は.png。 tilemapresource.xmlが必須となる。

PLATEAU VIEW 3.0対応データフォーマット一覧（1/2）

フォーマット				説明
	単体	Zip 7Z	URL 指定	
glTF	✓		✓	•拡張子は .gltf と.glb に対応。 <特記事項> •モデルの座標をCesiumJS内部の座標系に合わせておく必要がある。事前に位置合わせを済ませたデータを使用すること。 •Web上で公開されているような通常のglTFでは座標系が異なるため正しく表示されない。
CSV	✓		✓ 単体のみ	<特記事項> •CSVファイル内のデータの1行目はヘッダとして扱われるため、各カラムの名前指定は必須となる。 •ジオメトリはポイントのみ対応する。ポイントの座標は、緯度・経度・高さでカラムを分けて数値で指定する。 CMS上での登録時、これらのカラム名は自由だが、Viewerで正しく表示するには、Editorでコンポーネント設定が必要である。Editorでどのカラムを緯度・経度・高さとして扱うかをそれぞれ指定できる。高さカラムは省略可能で、省略時は0として扱われる。 Editor側のカラム名は以下のとおりである。 •ジオメトリ（緯度、経度、高さとして読み込むカラム）：at, lng, lon, height, alt, •スタイル（地図上のポイントのサイズと色）：pointSize, pointColor
GFTS Realtime			✓	<特記事項> •GTFS Realtimeのみ対応する。GTFS Staticには未対応。GeoJSONなどに変換する必要がある。 •Viewerで正しく表示するには、Editorでコンポーネント設定が必要である。

3D Tilesの属性の仕様検討

1.2.2にあるように、CMSとFME Flowが連携することで、CityGML形式の3D都市モデルデータを3D Tiles等のデートフォーマットへ変換を行っている。ここでは、変換後3D Tilesデータの属性について検討した内容を記載する。

3D Tilesへ適用するスタイル

CesiumJS上での3D Tilesデータへの色分け・絞り込みなどのスタイル適用は、通常3D Tiles Styling languageを用いて行われることが一般的である。3D Tiles及び3D Tiles Styling languageの詳細はOGCの仕様書を確認すること。この3D Tiles Styling languageでは、3D Tiles内のデータが持つ属性値を参照してスタイルを適用することができるが、一階層目のJSONプロパティを利用したスタイル適用しかできない。例えば、以下のような属性を持つ地物があった場合には、「heightが100ならば黒色にする」というスタイルは設定できるが、「”others”の中にある”用途”が”業務施設”の場合は黒色にする」というスタイルは設定ができない。

```
{
 “height”: 100,
 “others”: {
  “用途”: “業務施設”
 }
}
```

tileset.jsonに記載される属性

tileset.jsonは対象の3D Tilesデータに関するメタデータなどを保持しており、これを利用することで実際の地物データを読み込むことなくどういった属性が存在するかをアプリケーションが知ることができる。しかし、tileset.jsonとして格納される属性も前述した一階層目のJSONプロパティのみであるため、アプリケーション側で制御したい属性は一階層目のJSONプロパティとして保持し、tileset.jsonにも記載することが望ましい。

PLATEAU VIEWでのスタイル

PLATEAU VIEWでは、3D Tiles Styling languageのこうした制約を排除し、より柔軟なスタイルができるよう独自のスタイル適用システムを実装している。具体的には、スタイルに利用する属性をJSONPathで指定できるようにしており、上述の例において、 「”others”の中にある”用途”が”業務施設”の場合は黒色にする」というスタイルも設定が可能である。

変換後3D Tilesデータの属性を検討する観点

これらを踏まえて3D Tilesデータの属性を検討する際には以下の観点で検討を行なった。

- データを利用する開発者にとっての利便性：3D都市モデルを利用するほとんどの利用者は、3D Tilesへのスタイル適用をCesiumJS標準の3D Tiles Styling languageで行うと予想される。よって、スタイルに特に利用されうる属性は一次元の属性として保持した方が利便性は向上する。一方で、3D都市モデルデータは大量の属性情報を持つため、すべての属性を一次元の属性として保持すると冗長すぎて利便性が下がってしまう。
- 3D都市モデルとしての属性の網羅性担保：3D都市モデルは様々な原典データをもとに豊富な属性情報を保持しているため、利便性を理由に属性が減らした3D Tilesデータを整備すると、豊富な属性が利用できなくなってしまう。

3D Tilesの属性

これらを踏まえ、PLATEAU VIEWでは以下の方針で3D Tilesの属性を整備することにした。

- 3D都市モデルに含まれるobjectlist（整備対象の属性一覧を含むExcelファイル）で定義されている属性で、データ作成上必須もしくは原則整備とされている項目を一次元な属性として展開する。
- 属性の網羅性担保のために、その他全ての属性を「attributes」という項目にJSON形式で格納する。

以下は3D Tilesとして変換後の建築物属性の例である。


```
{
 "bldg:measuredHeight": 166.8,
 "bldg:storeysAboveGround": 35,
 "bldg:storeysBelowGround": 3,
 "uro:BuildingDetailAttribute_uro:buildingRoofEdgeArea": 5044.6561,
 "uro:BuildingDetailAttribute_uro:surveyYear": 2021,
 "_lod": 1,
 "_x": 139.76965653701913,
 "_y": 35.677316549080686,
 "_xmin": 139.76899546137153,
 "_xmax": 139.77031761266673,
 "_ymin": 35.676873166608715,
 "_ymax": 35.67775993155265,
 "_zmin": 3.89,
 "_zmax": 162.54,
 "meshcode": "53394611",
 "feature_type": "bldg:Building",
 "city_code": "13102",
 "city_name": "東京都中央区",
 "gml_id": "bldg_fe0ea6d6-70d5-4b78-b676-6f01387a98ff",
 "attributes": {
  "meshcode": 53394611,
  "feature_type": "bldg:Building",
  "gml:id": "bldg_fe0ea6d6-70d5-4b78-b676-6f01387a98ff",
  "core:creationDate": "2024-03-15",
  "gen:genericAttribute": [
   {
    "type": "string",
    "name": "延べ面積換算係数",
    "value": "1.00"
   }
 ]
}
```

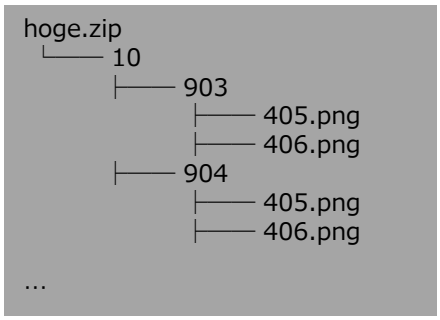
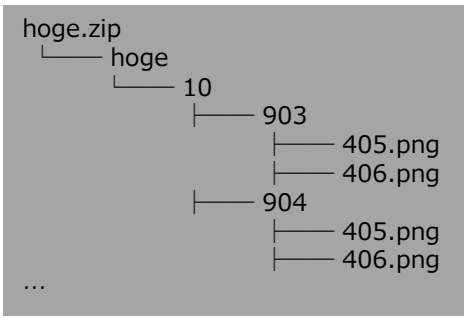
① CZMLをZip化する際のディレクトリ構造サンプル

<table>
<tr><th>良い例</th><th>悪い例</th></tr>
<tr><td>
<pre>
hoge.zip
  ├─── hoge.czml
  ├─── icon1.png
  └─── icon2.png
</pre>
例１）CZMLがルート直下に存在しており、ファイル名がZip/7zと同じである。
</td><td>
<pre>
hoge.zip
  └─── hoge
        ├─── hoge.czml
        ├─── icon1.png
        └─── icon2.png
</pre>
例１）CZMLがルートに存在せずフォルダーを1つ挟んでいる。
<pre>
hoge.zip
  ├─── foobar.czml
  ├─── icon1.png
  └─── icon2.png
</pre>
例２）CZMLの名前がZip/7zの名前と異なる。
</td></tr>
</table>

② 3D TilesをZip化する際のディレクトリ構造サンプル

<table>
<tr><th>良い例</th><th>悪い例</th></tr>
<tr><td>
<pre>
hoge.zip
  ├─── tileset.json
  ├─── 0.b3dm
  └─── 1.b3dm
</pre>
例１）tileset.jsonがルートに存在する。（b3dmなど他のファイルはtileset.json内で相対パスが正しく定義されていれば、フォルダーを挟んでも問題ない）
</td><td>
<pre>
hoge.zip
  └─── hoge
        ├─── tileset.json
        ├─── 0.b3dm
        └─── 1.b3dm
</pre>
例１） tileset.jsonがルートに存在せずフォルダーを1つ挟んでいる。
</td></tr>
</table>

③　MVTをZip化する際のディレクトリ構造サンプル

良い例	悪い例
 例１）ズームレベル以下のフォルダーがルート直下に存在している。	 例１）ズームレベル以下のフォルダーがルートに存在せずフォルダーを1つ挟んでいる。

（5）PLATEAU CMSへアップロードするデータの命名規則

CMSへアップロードするアセットには命名規則がある。これは、CMSとFMEでの品質検査及びデータ変換において、ファイル名が重要な役割を果たすからである。CMSへアップロードするファイルは以下の命名規則に従う必要がある。

CMSへアップロードするデータの命名規則（1/2）

データ分類	データ	命名規則	記載例	CMSへのアップロード主体
都市モデルデータ	コードリスト	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_codelists.Zip	22211_iwata-shi_city_2023_citygml_1_op_codelists.Zip	ユーザー
	スキーマ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_schemas.Zip	22211_iwata-shi_city_2023_citygml_1_op_schemas.Zip	ユーザー
	仕様	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_specification.Zip	22211_iwata-shi_city_2023_citygml_1_op_specification.Zip	ユーザー
	メタデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_metadata.Zip	22211_iwata-shi_city_2023_citygml_1_op_metadata.Zip	ユーザー
	3D 都市モデル整備範囲図	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_indexmap.pdf	22211_iwata-shi_city_2023_citygml_1_op_indexmap.pdf	ユーザー
	地物型データ（建築物モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg.Zip	22211_iwata-shi_city_2023_citygml_1_op_bldg.Zip	ユーザー
	地物型データ（土地利用モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_luse.Zip	22211_iwata-shi_city_2023_citygml_1_op_luse.Zip	ユーザー
	地物型データ（建築物モデル）（政令指定都市以外）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg_3dtiles_[lod].Zip	22211_iwata-shi_city_2023_citygml_1_op_bldg_3dtiles_lod1.Zip	FME
	地物型データ（建築物モデル）（政令指定都市）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_bldg_3dtiles_[行政コード]_[区名]_[lod].Zip	14130_kawasaki-shi_city_2022 citygml_1_op_bldg_3dtiles_14131_kawasaki-ku_lod1.Zip	FME

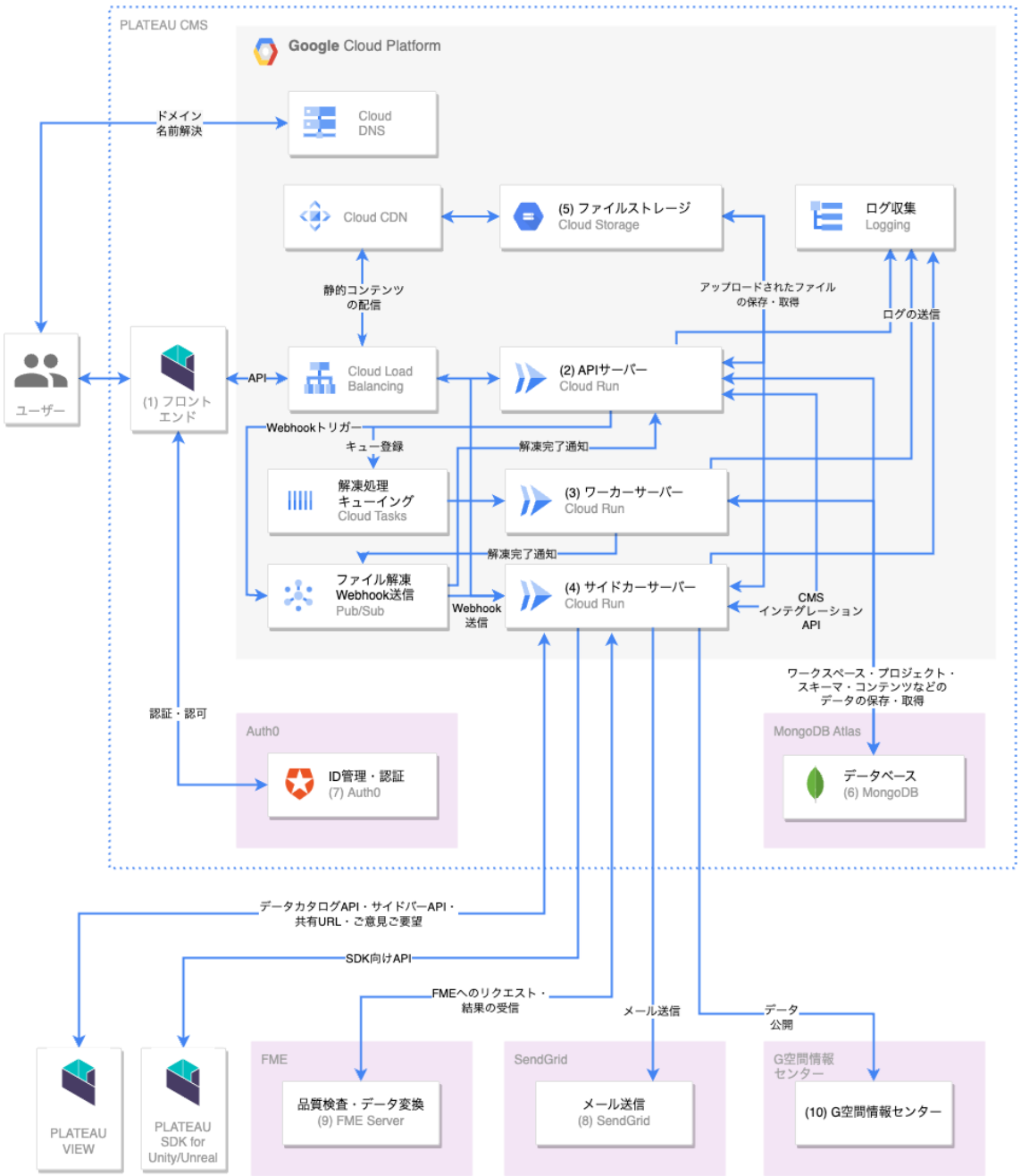
CMSへアップロードするデータの命名規則（2/2）

データ分類	データ	命名規則	記載例	CMSへのアップロード主体
都市モデルデータ	地物型データ（土地利用モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_luse_mvt.Zip	14130_kawasaki-shi_city_2023_1_op_luse_mvt.Zip	FME
	地物型データ（洪水浸水想定区域モデル）	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_fld_natl_[水系]_[河川名]_3dtiles_[l1 or l2].Zip	14130_kawasaki-shi_city_2023_citygml_1_op_fld_natl_tamagawa_tamagawa-asakawa_3dtiles_l1.Zip	FME
	ファイル別最大LOD リスト	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_[地物型名]_maxLod.csv	14130_kawasaki-shi_city_2023_citygml_1_op_bldg_maxLod.csv	FME
	品質検査結果	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション]_[地物型名]_qc_result.Zip	14130_kawasaki-shi_city_2023_citygml_1_op_bldg_qc_result.Zip	FME
ユースケースデータ	ユースケースデータ	uc_[UC番号]_[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_[データ名].[データの拡張子]	uc_11_22130_hamamatsu-shi_acn_2023_shimizunoyaike.json	ユーザー
関連データセット	関連データセット	[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_[landmark,shelterなど].[データの拡張子]	42202_sasebo-shi_city_2023_landmark.czml	ユーザー
G空間情報センター向けデータ	都市モデルデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_citygml_[更新回数]_[オプション].Zip	14130_kawasaki-shi_city_2023_citygml_1_op.Zip	CMS
	3D Tiles, MVT	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_3dtiles-mvt_[更新回数]_[バージョン].Zip	13100_tokyo23-ku_city_2023_3dtiles-mvt_1_2_op.Zip	CMS
	ユースケースデータ	[市区町村コード]_[市区町村名英名]_[提供者区分]_[整備年度]_uc[UC番号]_[バージョン].Zip	13100_tokyo23-ku_acn_2023_UC22-1.Zip	CMS
	オルソ画像	[市区町村コード]_[市区町村名英名]_[提供事業者名]_[整備年度]_ortho_[更新回数]_[オプション]_op.Zip	42202_sasebo-shi_[提供事業者名]_2023_ortho_1_op.Zip	CMS

1.3.2 システム構成の全体像

PLATEAU CMSのシステム構成について説明する。

表　PLATEAU CMSのシステム構成



（1）フロントエンド

Webブラウザ上で動作する、（HTML・CSS・JavaScriptによる）フロントエンドアプリケーション。APIサーバー・Auth0・その他各種サーバーとの通信を行い、UIやプレビュー機能による地図の表示を行う。

（2）APIサーバー（Cloud Run）

PLATEAU CMSのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。MongoDBやGoogle Cloud Storageと連携して、アイテムの保存やプロジェクトの管理・公開などのさまざまなビジネスロジックを実行する。

APIサーバーは、Cloud Run 上で動作する。Cloud Run とは、GCPで利用可能なサーバーレス（CaaS）プラットフォームであり、Dockerコンテナをデプロイすることで、サーバーの保守管理の手間なしに、アプリケーションをクラウド上で動作させることができる。同時接続リクエスト数が規定数以上に達すると自動的にコンテナが増加し、より多くのトラフィックを自動的に分散処理することができる。

Cloud Runは、デフォルト設定では、HTTPリクエストを受信して処理している間のみCPUが動作し、リクエストを処理していない時は動作を停止するため、HTTPリクエストを実際に受信し処理するために動作したCPU時間分のみが課金対象となる。この点が、常時稼働が前提となることが多いAWSのEC2やGCPのGCEとは異なる。（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU CMSのAPIサーバーは、メモリ4GB・CPU2コアの設定で動作している。

（3）ワーカーサーバー（Cloud Run）

PLATEAU CMSにおける非同期バックグラウンド処理を行うサーバーであり、HTTPサーバーとしてAPIサーバーからのリクエストを受信している。Zipファイル等の解凍処理やWebhookの送信などをCloud TasksやCloud PubSubと連携しながら行なっている。ワーカーサーバーは、APIサーバーと同じく、Cloud Run 上で動作する。（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU CMSのWorkerサーバーは、メモリ32GB・CPU8コアの設定で動作している。

（4）サイドカーサーバー（Cloud Run）

PLATEAU CMSと連携しPLATEAU CMSを補助する形で動作するサーバー。外部サービスとの連携や、PLATEAU SDK for Unity/Unreal向けのデータ配信、PLATEAU Editor・PLATEAU VIEWで凡例等の表示を行うサイドバーのデータ等を扱う。（サイドバーについては、3.2.7で詳細を解説）以下の機能を持つ。

- FMEへの品質検査・データ変換処理のリクエスト・結果の受信と保存
- G空間情報センターへのデータ登録処理・カタログ検査
- PLATEAU SDK for Unity/Unreal向けCityGMLデータ配信API
- PLATEAU VIEWの建築検索機能向け検索インデックスの構築
- その他データのランドマーク・鉄道駅・行政界データのCZMLへの変換
- PLATEAU Editor・PLATEAU VIEW向けAPI
    - データカタログAPI
    - サイドバー設定・共有URLデータの保存
    - ご意見ご要望を受け取りSendGridと連携してメール送信

サイドカーサーバーは、APIサーバーと同じく、Cloud Run 上で動作する。

（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU CMSのPLATEAU VIEWサーバーは、メモリ16GB・CPU4コアの設定で動作している。

（５）ファイルストレージ（Google Cloud Storage）

フロントエンドのアプリケーションのソースコードや画像、ユーザーによってアップロードされたアセットファイルを保存する、オブジェクトストレージサーバー。

Google Cloud Storage（GCS）を使用している。容量は無制限、自動的にスケーリングし、バックアップも自動的に行われ、データは複数拠点に分散配置される。巨大なファイルを格納・配信することが可能。

PLATEAU CMSでは、PLATEAU関連データセット等の静的ファイルをGCSに保存している。

（６）MongoDB

MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。
PLATEAU CMSはデータベースとしてMongoDBを使用している。

PLATEAU CMSでは、MongoDB社が提供するクラウドサービス MongoDB Atlas を利用している。保守運用が自動化されたマネージドなMongoDBが利用可能で、自動的に3台以上のサーバーから成るクラスタを構成し、データベース内のデータは自動的に各サーバーに複製され、リクエストは分散処理されるようになっている。M0からM30まで、さまざまなマシンスペックのサーバーによる可用性の高いクラスタを構築することができ、マシンスペックによって料金が変わる。

（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU CMS向けには、M10クラスタを運用しており、PLATEAU Editorと同じクラスタを使用している。
MongoDBには、ユーザーやワークスペース、スキーマ、コンテンツ（データカタログに表示される説明文等）に関する情報が保存される。

（７）Auth0

Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証認可機能を実現している。なお、PLATEAU CMSとPLATEAU Editorでは同じAuth0テナントを利用している。

（2024年3月現在）テナントに対して登録されているユーザー数に応じて課金されるが、7000ユーザーまでは無料となっている。

（８）SendGrid

Twillio社が提供するクラウドサービス。クラウドベースのメール配信プラットフォームで、企業や開発者がアプリケーションやWebサイトから大量のメールを配信するために使用される。PLATEAU CMSではご意見ご要望のメール送信で使用している。

（９）FME Flow

Safe Software社（カナダ） が開発したデータ変換エンジンで、データの品質検査や、CityGMLから3D TilesやMapbox Vector Tilesへのデータ変換などを行う。詳細は「1.3 FME Flow」を参照されたい。

（１０）G空間情報センター

官民問わずさまざまな主体により整備・提供される多様な地理空間情報を集約し、利用者がワンストップで検索・ダウンロードし利用できる、産学官の地理空間情報を扱うプラットフォーム。

PLATEAU CMSでは、3D都市モデルデータ、ユースケースデータの公開時（公開申請承認時）に自動的にG空間情報センターへ登録を行う。

（１１）その他のコンポーネント

表　その他のコンポーネント

コンポーネント名	説明
Cloud CDN	GCP(Google Cloud Platform)のCDN（Content Deliver Network = ウェブコンテンツをインターネット経由で配信するために最適化されたネットワーク）。GCSなどと組み合わせて使用することで、リクエスト元から地理的に近いサーバーにコンテンツのキャッシュを自動的に配置し、コンテンツ配信を高速化・効率化させることができる。
Cloud DNS	GCPのDNS。PLATEAU VIEW 3.0のPLATEAU Editorで使用しているドメインに対応するレコードは全てCloud DNSで管理されている。
Cloud Load Balancing	GCPのマネージドなロードバランサ（負荷分散システム）。PLATEAU VIEW 3.0のPLATEAU Editorで使用されるドメインのIPアドレスは全てCloud Load Balancingに向いており、リクエストのホスト（ドメイン）に応じてAPIサーバーやストレージサーバーに自動的にルーティングされる。
Cloud Logging	GCPのログ収集サービス。Cloud Runなどから出力されるログを閲覧可能。システムのトラブルシューティング時に役立つ。
Cloud PubSub	GCPのメッセージングサービス。アプリケーション間の連携を行うために利用される。PLATEAU CMSでは、APIサーバーとワーカーサーバー間の通信に使用している。
Cloud Tasks	GCPのキューイングサービス。大量の分散タスクの実行を管理できる。 PLATEAU CMSではワーカーサーバーで行われる解凍処理のキューイング及び再試行処理制御のために使用している。
Cloud Build	Cloud Build: GCPのCI/CDパイプライン向けビルドプラットフォーム。CMSでは、大規模圧縮ファイルの解凍処理の基盤として利用している。

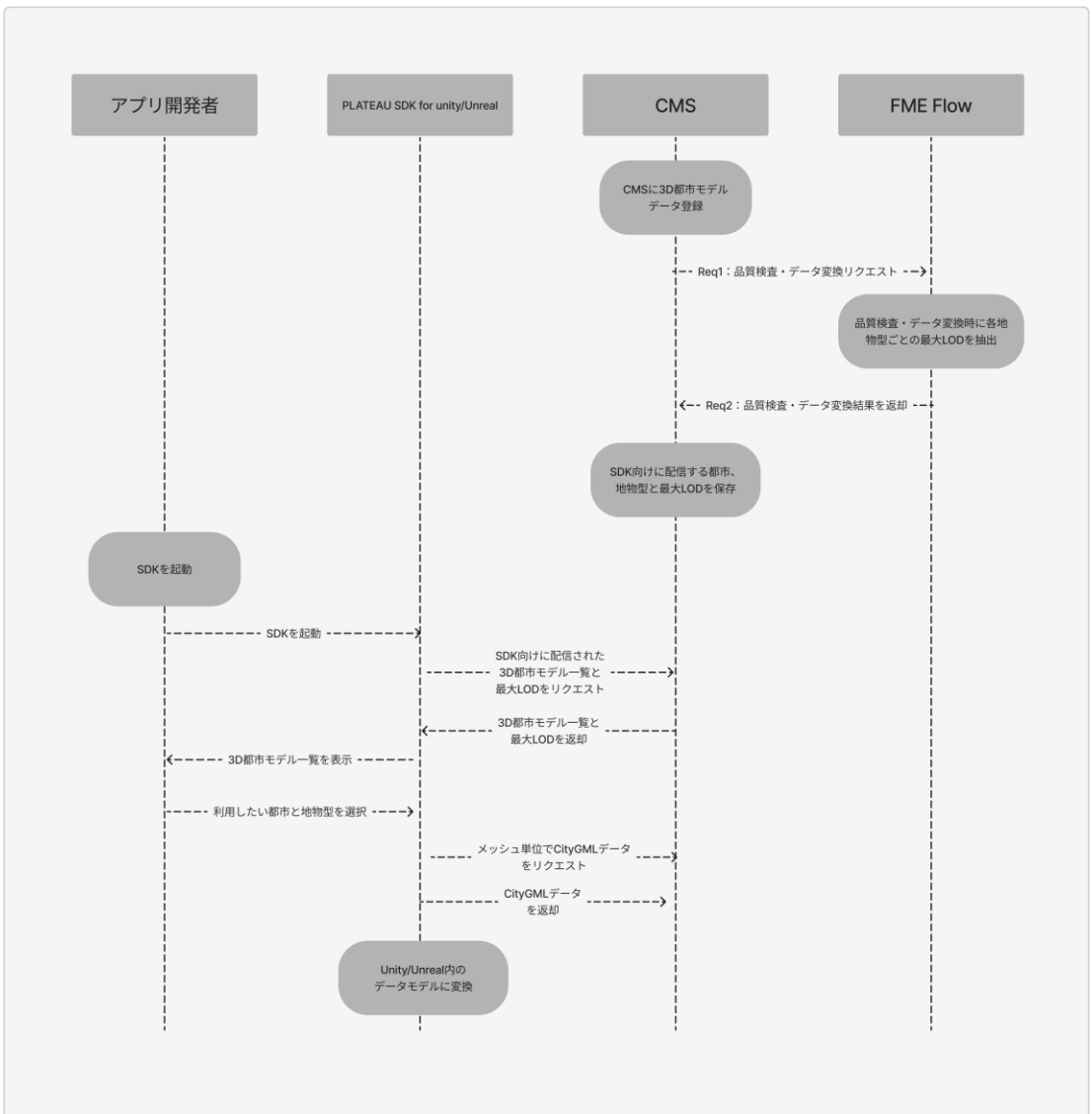
1.3.3 システム間連携におけるAPI仕様

（１）PLATEAU SDK for Unity/Unrealとの連携

1.3.2で述べたように、CMSはいくつかの外部アプリケーションとシステム間連携を行っている。以下に、PLATEAU SDK for Unity/Unrealとの連携仕様を記載する。PLATEAU SDK for Unity/Unrealでは、SDKが3D都市モデルが利用可能な都市、地物型、最大LOD一覧を表示し、アプリ開発者が利用したいデータを選択する方式となっている。これらの連携を実現するために、SDK、CMS、FME Flowでは以下のような連携を行っている。CMSに3D都市モデルデータが登録され、品質検査及びデータ変換を実行する際に、FME FlowがCityGMLファイルから地物型ごとの最大LODを抽出する。抽出された最大LODをCMSに保存し、CMSはSDKから発行される都市、地物型、最大LODの一覧取得をするリクエストに応答する。

図　複数システム連携図

SDKとCMSの連関図



以下にSDKからCMSへ発行されるリクエストと対応するレスポンスを記載する。

（1）共通項目

- Base URL:`https://api.plateau.reearth.io`
- メッシュコードは全て三次メッシュ単位である。
- Access token: （セキュリティの観点から本ドキュメントには記載しない）

全件取得

本エンドポイントでは、SDKから利用可能な都市、地物型が返却される、

```
GET /sdk/datasets
Authorization: Bearer xxx
```

```
{
 "data": [
  {
   "title": "東京都",
   "data": [
    {
     "id": "tokyo-23ku",
     "title": "23区"
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg", "dem"]
    },
    {
     "id": "hachioji-shi",
     "title": "八王子市",
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg", "dem"]
    }
   ]
  },
  {
   "title": "神奈川県",
   "data": [
    {
     "id": "yokohama-shi",
     "title": "横浜市",
     "spec": "2.3", //or "3.0"
     "description": "xxxx", // 建築物モデルの説明文
     "featureTypes": ["bldg"]
    }
   ]
  }
 ]
}
```

ID指定してファイル一覧を取得

本エンドポイントでは、対象都市の各地物型ごとの最大LODと3次メッシュ単位のCityGMLファイルへのURLが返却される。SDKはこのCityGMLをダウンロードしている。

```
GET /sdk/datasets/:id/files
Authorization: Bearer xxx
```

```
{
 "bldg": [
  {
   "code": "12345678",
   "url": "https://xxxx/xxxxx/12345678_hogehoge.gml",
   "maxLod": 2
  },
  {
   "code": "12345679",
   "url": "https://xxxx/xxxxx/12345679_hogehoge.gml",
   "maxLod": 2
  }
 ],
 "veg": [
  {
   "code": "123456",
   "url": "https://xxxx/xxxxx/123456_hogehoge.gml",
   "maxLod": 2
  }
 ],
 "...": []
}
```

（2）その他の公開API

CMSでは、SDK向けのAPIだけでなく試験的にGraphQL形式のAPIを公開している。GraphQLとは、サーバー向けに発行するリクエストをより柔軟に構築するためのOSS及びクエリ言語である。実際のAPIはhttps://api.plateau.reearth.io/datacatalog/graphqlから確認が可能で、開発に伴いこちらのページは適宜アップデートされている。こちらのGraphiQLページから最新のドキュメントも閲覧できる。本ページでは、このGraphQLAPIを利用して、CMSへ保存されたデータがどのように取得可能かを例示する。

データセット検索

「searchTokens」に検索文字列を入力することで、検索文字列を名称に含むデータの一覧を返却する。

```
query dataset {
  datasets(input:{searchTokens:["建築物"]}) {
    id
    name
    groups
    items {
      id
      name
      url
    }
  }
}
```

```
{
  "data": {
    "datasets": [
      {
        "id": "d_01101_bldg",
        "name": "建築物モデル（中央区）",
        "groups": null,
        "items": [
          {
            "id": "di_01101_bldg_lod1",
            "name": "LOD1",
            "url": "https://assets.cms.plateau.reearth.io/assets/b8/314602-4b39-4d5f-be2d-
a0b17a3e3c21/01100_sapporo-shi_city_2020_citygml_6_op_bldg_3dtiles_01101_chuo-
ku_lod1/tileset.json"
          }
        ]
      },
    ]
  }
}
```

都市とそのデーター覧取得

「code」に都市の行政コードを入力することで、対象都市に関連するデータの一覧を返却する。

```
query area {
  area(code:"13229") {
    id
    type
    name
    datasets{
      id
      name
    }
  }
}
```


```
{
  "data": {
    "area": {
      "id": "c_13229",
      "type": "CITY",
      "name": "西東京市",
      "datasets": [
        {
          "id": "d_13229_bldg",
          "name": "建築物モデル（西東京市）"
        },
        {
          "id": "d_13229_tran",
          "name": "道路モデル（西東京市）"
        },
        {
          "id": "d_13229_lsld",
          "name": "土砂災害警戒区域モデル（西東京市）"
        },
        {
          "id": "d_13229_urf_UseDistrict",
          "name": "都市計画決定情報モデル 用途地域モデル（西東京市）"
        }
      ]
    }
  }
}
```

1.4 FME Flow
1.4.1 FME Flow とその稼働環境

稼働環境の制約

CMSと連携して3D都市モデルデータの品質検査、可視化用のデータ変換は Safe Software社 (カナダ) の FME Flow (旧称 FME Server) によって行うため、同ソフトウェアとその稼働環境を用意する。

必要な FME ソフトウェアおよびその稼働環境は次のとおり。

- FME ソフトウェア
  - FME Flow：バージョン 2023.1.2 以降
  - CPU-Time：1式（注1）
- FME Flow の稼働環境
  - OS：次のページに記載されている FME Flow 対応 OS の範囲（注2）
    - FME 2023 の場合：Legacy FME Technical Specifications
    - FME 2024 の場合：FME Platform Technical Specifications
  - 推奨スペック：RAM64GB以上, ディスク容量500GB以上（注: 必要なメモリ、ディスク容量は、品質検査やデータ変換の対象とするデータのサイズや同時処理数に応じて異なり、このスペックであればどんな条件でも対応できるということを保証するものではない）。
  - インターネットにアクセスできること

注1: CMSと連携した処理はいくつかのプロセスを並列で行うことがあるため、FME Flow 1ライセンス（同時実行1プロセス）のみでは対応ができない。そのため、一定のCPU時間数に達するまでの間は任意にエンジン（同時実行プロセス）数を設定できる CPU-Time を導入する必要がある。CPU-Time を利用するため、実行環境からはインターネットにアクセスできる必要がある。

注2: FME 2023.x の推奨 OS (Linux の場合) は Ubuntu 20.04 であったが、本マニュアル作成時点の最新バージョン FME 2024.0 では Ubuntu 20.04 のサポートは終了し、Ubuntu 22.04 が推奨 OS となっている。FME 2023 も Ubuntu 22.04 をサポートしているので、本PLATEAU VIEWのために新たに Linux マシンを用意する場合は、Ubuntu 22.04 とすることを推奨する。

Amazon Web Services S3バケット

- CMSからのリクエストに応じてFME Flow が行った品質検査や可視化用データ変換処理の結果やログファイル等は Amazon Web Services（以下「AWS」と言う）S3 経由で CMS に渡す仕組みであるため、それに必要な AWS S3 バケットをひとつ用意し、データ書き込み権限のある「アクセスキーID」及び「シークレットアクセスキー」を取得する。
- バケット名やリージョンについての制約はないので、データ書き込み権限を取得できるものであれば、既存のバケットを利用しても良い。

1.4.2 FME Flow プロジェクトファイル

FME Flow によって実行する品質検査、可視化用データ変換のフローを定義したワークスペース、CMSと連携してそれらを自動実行するための構成内容、その他の関連ファイルは、すべて次の FME Flow プロジェクトファイルに含まれている。

plateau-2023-fme-flow-project1.0.0_2024-3-20-T055500_b23636.fsproject

“-project”と拡張子 .fsproject の間の部分（バージョン、日付時刻、作成時にFMEバージョン）は、プロジェクトファイルの更新に伴って変わることがある。

後述するように、FME Flow のウェブインターフェースの操作によってこのプロジェクトファイルを FME Flow にインポートし、いくつかのの設定を行うことによってCMS 連携処理に必要なすべてのファイル、構成が再現される。

参考: 表　プロジェクトファイルに含まれるリポジトリとCMS連携処理で使用するワークスペース

リポジトリ名	CMS連携処理で使用するワークスペース
PLATEAU 2023 品質検査	• PLATEAU3 品質検査01 共通.fmw • PLATEAU3 品質検査02 建築物.fmw • PLATEAU3 品質検査03 道路等の交通モデル.fmw • PLATEAU3 品質検査04 都市設備・植生.fmw • PLATEAU3 品質検査05 土地利用・都市計画決定情報・区域.fmw • PLATEAU3 品質検査06 浸水想定区域.fmw • PLATEAU3 品質検査07 土砂災害警戒区域.fmw • PLATEAU3 品質検査08 地形.fmw • PLATEAU3 品質検査09 橋梁・トンネル・地下街.fmw • PLATEAU3 品質検査10 その他の構造物 • PLATEAU3 品質検査11 水部.fmw • PLATEAU3 品質検査12 地下埋設物.fmw • PLATEAU3 品質検査13 汎用都市オブジェクトfmw
PLATEAU 2023 可視化用データ変換	• PLATEAU3 可視化用データ変化01 建築物.fmw • PLATEAU3 可視化用データ変換02 道路・鉄道・徒歩道・広場・航路.fmw • PLATEAU3 可視化用データ変換03 都市設備・植生.fmw • PLATAEU3 可視化用データ変換04 土地利用・土砂災害警戒区域.fmw • PLATEAU3 可視化用データ変換05 浸水想定区域.fmw • PLATEAU3 可視化用データ変換06 都市計画決定情報・区域.fmw • PLATEAU3 可視化用データ変換07 橋梁・トンネル・その他の構造物.fmw • PLATEAU3 可視化用データ変換08 地下街.fmw • PLATEAU3 可視化用データ変換09 地下埋設物.fmw • PLATEAU3 可視化用データ変換10 水部.fmw • PLATEAU3 可視化用データ変換11 汎用都市オブジェクト.fmw
plateau-utilities	• processing-status-notifier.fmw
plateau2023-cms	• cms-error-logger.fmw • cms-job-submitter_conv.fmw • cms-job-submitter_qc.fmw • cms-request-receiver.fmw

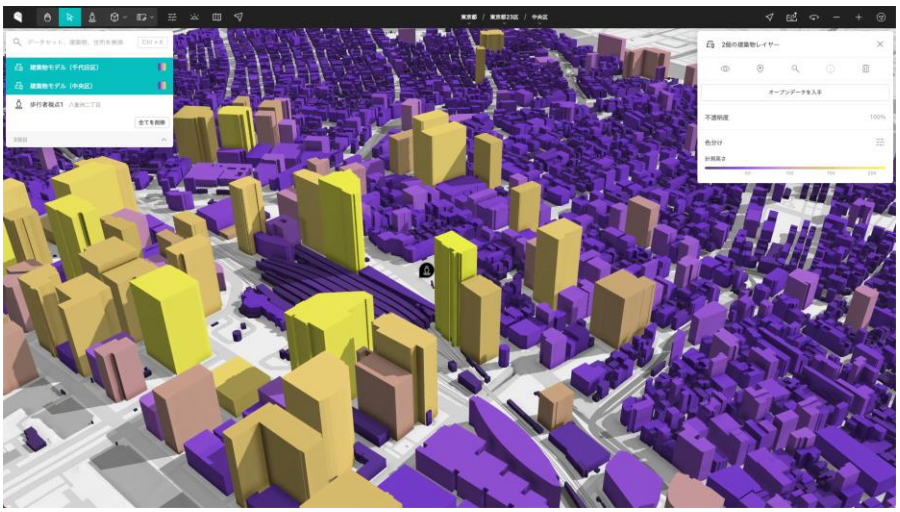
1.5 PLATEAU Editor・PLATEAU VIEW
1.5.1 PLATEAU Editor・PLATEAU VIEWとは

1.1のシステム全体構成で述べたように、PLATEAU VIEW 3.0のシステムの機能は、大きく「データの管理」と「データの可視化」に分かれる。

PLATEAU EditorとPLATEAU VIEWは、このうち「データの可視化」を担うシステムのことを指す。具体的にはこの可視化システムは、更に以下のように分かれる。

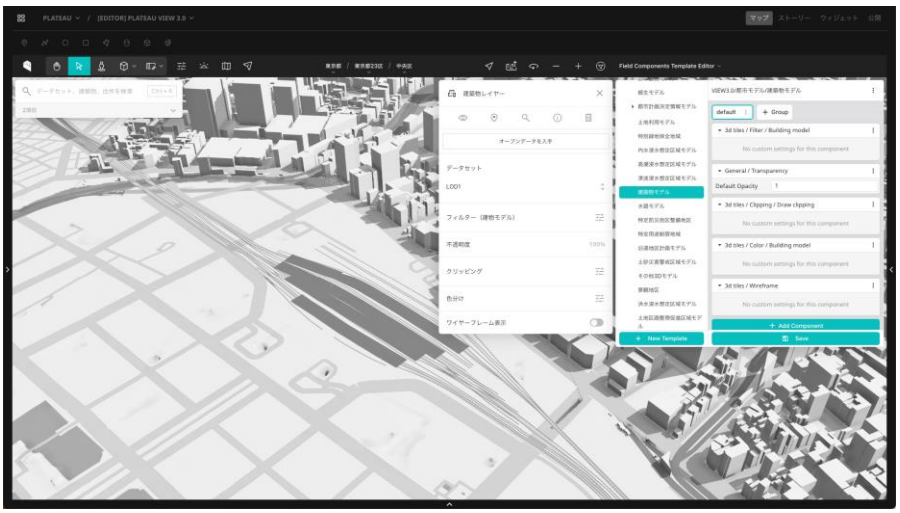
PLATEAU VIEW

- ユーザーが、3D都市モデルデータを初めとするさまざまなGISデータの可視化をWeb上のデジタル3D地球儀上で行うことができる、Webアプリケーション。下記に述べるPLATEAU Editorによって自動的に管理・運用されている。



PLATEAU Editor

- 上記のPLATEAU VIEWそのものを作成及び公開するための、Webアプリケーション。目的に応じてさまざまなWebアプリケーションを作成・公開することが可能であるが、PLATEAU VIEW 3.0では、1つのWebアプリケーションをPLATEAU Editor経由で一般向けに公開し、それをPLATEAU VIEWと呼称している。



（１）使用ソフトウェア・サービス

上記システムを構築するために、オープンソースソフトウェア（OSS）と、有償のクラウドサービスを組み合わせて利用している。クラウドサービスの詳細については次項の各コンポーネントの説明で述べる。

表　使用ソフトウェア・サービス一覧

項目	項目	説明
Re:Earth	OSS	Eukarya社が開発・公開しているOSS。CesiumJSを内包しており、ノーコードで地図やデジタル地球儀を使用したWebアプリケーションを作成・公開することができるWebアプリケーション。プラグインによる機能拡張にも対応。
Cesium Ion	有償クラウドサービス	Cesium社が提供するクラウドサービス。地球儀上の日本における地表面を表現する3Dのデータ（テラインデータ）を配信するために使用している。
Auth0	有償クラウドサービス	Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証・認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証・認可機能を実現している。
Google Cloud Platform（GCP）	有償クラウドサービス	Google社が提供するクラウドコンピューティングプラットフォーム。PLATEAU Editorを動作させるためのサーバーや、ファイルを保存するためのサーバーをGCP上に構築している。
MongoDB Atlas	有償クラウドサービス	MongoDB社が提供するクラウドサービス。保守運用が自動化されたマネージドなMongoDBを提供している。MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。PLATEAU CMSはデータベースとしてMongoDBを使用している。
Google Street View	有償API	Google社が提供するGoogle Maps及びGoogle Earthで提供される技術で、ある地点における、パノラマ画像を閲覧することができるサービスである。PLATEAU VIEWでは、この機能を別アプリケーションから利用するための、Street View Static APIを利用している。

（2）使用技術・ライブラリ

PLATEAU Editorはその他さまざまな技術を組み合わせて構築されている。

PLATEAU EditorとPLATEAU VIEWでは共通のソースコードが多く存在するが、地図のレンダリングに関連するものは、PLATEAU VIEWの技術として紹介する。

PLATEAU Editorで内部的に利用されている技術・ライブラリ等

項目	説明
Go	プログラミング言語の1つで、Googleによって開発された。構文がシンプルであり、かつ処理が高速な言語で、主にバックエンド開発に用いられる。PLATEAU Editorのサーバー実装に利用されている。
TypeScript	プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU Editorではフロントエンド向け開発に利用されている。
React	Meta（旧Facebook）社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU Editorではフロントエンド向け開発に利用されている。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU Editorではバックエンドとフロントエンド間の通信に利用されている。
Ant Design	Alibaba社が開発したUIフレームワーク及びUIライブラリ。これによりPLATEAU Editorでは統一的なデザインを提供している。
Terraform	HashiCorp社によって開発された、infrastructure-as-codeを実現するためのソフトウェアであり、サーバー構成をコードとして宣言的に管理をできるようにする。
WebAssembly	モダンなブラウザで動作するプログラミング言語の一種。Rustなどその他のプログラミング言語からコンパイルされ、ブラウザ上の隔離環境でプログラムを実行することで、安全で高速に実行される。PLATEAU Editorでは、プラグインシステムなどの動作に利用されている。

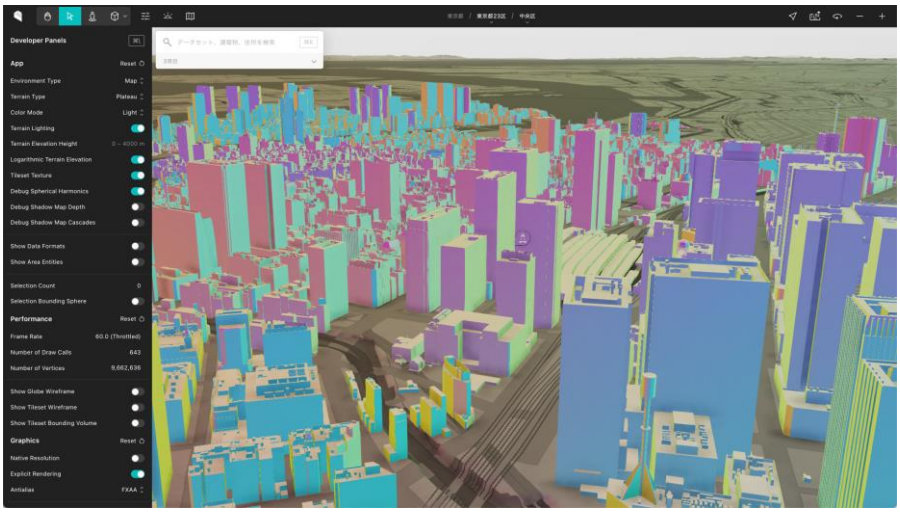
PLATEAU VIEWでは、以下のライブラリ等と、3DCG技術を利用している。

PLATEAU VIEWで内部的に利用されている技術・ライブラリ等

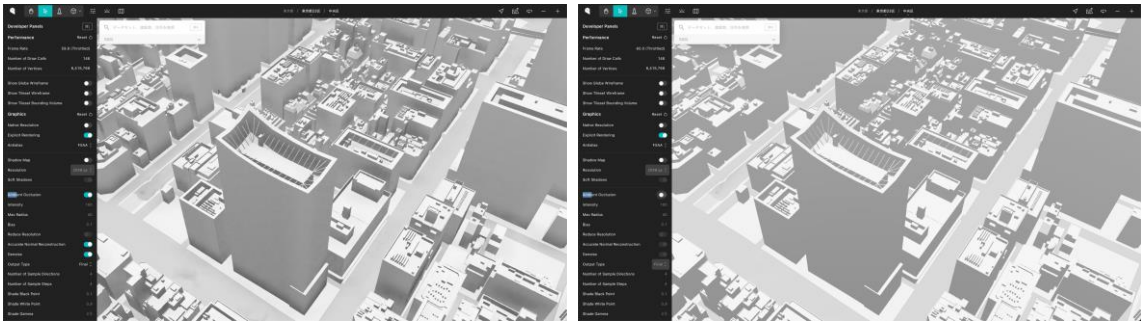
項目	説明
TypeScript	プログラミング言語の1つで、Microsoft社によって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。PLATEAU Editorではフロントエンド向け開発に利用されている。
React	Meta（旧Facebook）社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU Editorではフロントエンド向け開発に利用されている。
CesiumJS	デジタル3D地球儀上にさまざまな情報を描画することができる地図エンジン。Webブラウザ上で動作し、WebGLを用いて描画を行うため、PCやスマートフォンで閲覧することができる。PLATEAU Editor上でデータを表示するために使用している。
Resium	React上でCesiumJSを手軽に利用可能にするコンポーネントを提供するライブラリ。Eukarya開発。PLATEAU Editor上でCesiumJSと共に使用している。
GraphQL	API向けに作られたクエリ言語及びランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU Editorではバックエンドとフロントエンド間の通信に利用されている。
MUI	Google社のMaterial Designを実装したUIライブラリ。PLATEAU VIEWのUI部分はこのライブラリによって統一的なデザインを実現している。
Mapbox GL JS	Mapbox社が開発したライブラリで、ベクトルタイルを描画するための機能を提供する。PLATEAU VIEWでは、ベクトルタイル形式の「地理院タイル」をCesiumJSで利用するために、ベクトルタイル形式からラスター形式へ変換するために利用している。

（３）使用3DCG技術

球面調和関数
球面上の関数を表すのに用いられる数学的関数。3Dグラフィックスでは、光の反射や放射など、環境内の光の複雑な挙動を効率的に近似し、シミュレートするために使用される。この技術により、リアルタイムレンダリングにおける計算負荷を減少させつつ、高品質なビジュアルエフェクトを実現できる。
以下の画像は、球面調和関数の適用がわかりやすいよう、上下前後左右から別の色を当てている。建築物の各面にグラデーションのような色が適用されているように、環境における光の反射を再現しつつ、計算負荷の最適化を行っている。

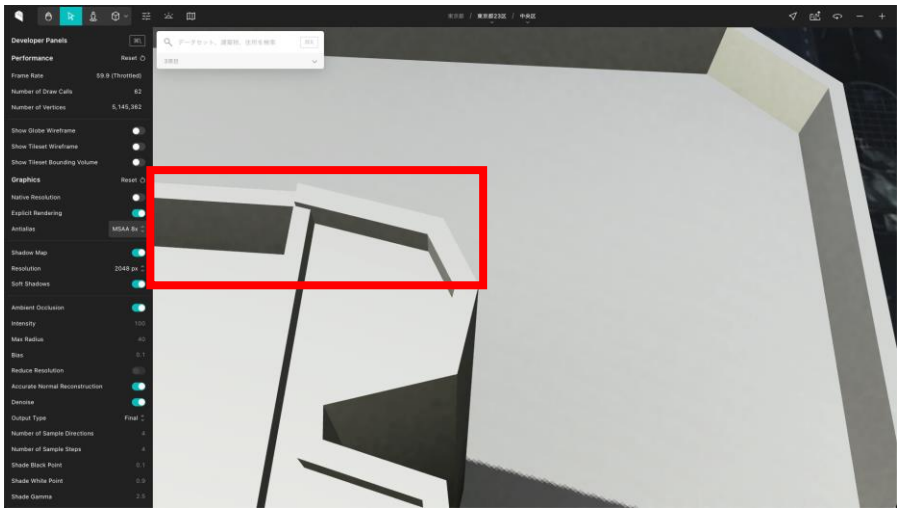


アンビエントオクルージョン
シーン内の各点での局所的な環境光の遮蔽を計算するレンダリング技術。物体の接近する部分や隅など、光が届きにくい場所に陰影を加えることで、よりリアルな3Dシーンを生成する。アンビエントオクルージョンの代表的なものには、SSAO（Screen Space Ambient Occlusion）やHBAO（Horizon-Based Ambient Occlusion）などがある。
以下の画像（左：アンビエントオクルージョンあり、右：アンビエントオクルージョンなし）を比較すると、アンビエントオクルージョンを適用することで、より立体的な視覚表現ができることが分かる。



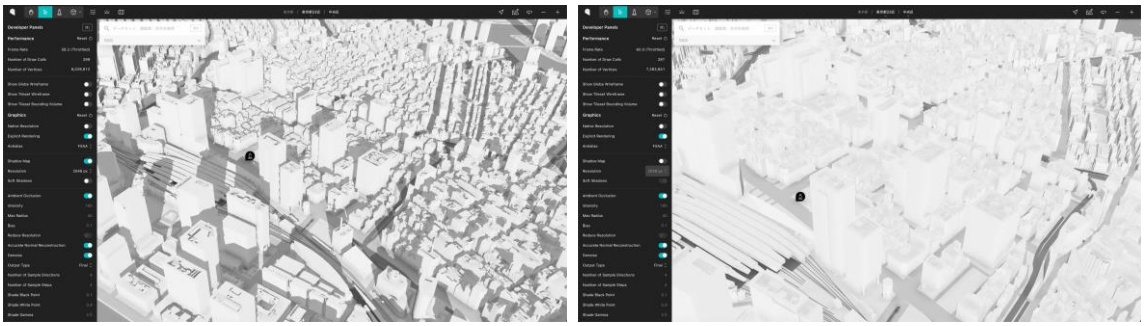
アンチエイリアス
3DCGにおいて、特にエッジ部分に現れるジャギー（エイリアス）をなめらかにする手法。ピクセルの色や輝度を隣接ピクセルとの間で調整し、エッジの滑らかさを向上させることでジャギーを改善する。アンチエイリアスにはいくつか種類がある。
- MSAA（Multi-sample Anti-aliasing）：複数のサンプルポイントを持つピクセルにおいて、エッジ周辺の色を平均化する。
- FXAA（Fast Approximate Anti-aliasing）：ピクセルレベルでエッジを検出し、テクスチャをぼかす。

スクリーンショットではわかりにくいが、アンチエイリアスによって建築物などのエッジのジャギーが緩和されている。



シェーダー
3Dグラフィックスにおいて、光の影響を受けた物体の表面の見え方を決定するためのプログラムまたはコードのセットを指す。シェーダーは、GPU（グラフィックス処理ユニット）上で実行され、リアルタイムで高速な計算を可能にする。PLATEAU VIEW 3.0では、CesiumJSに加え、独自のシェーダーを実装することで、リッチな立体表現を実現している。
特にメインのシェーダーは以下の2つである。
- 頂点シェーダー：3Dモデルの拡張点に対する操作を行う。頂点の位置、光の影響などの計算に利用する。
- フラグメントシェーダー：画面上の各ピクセルに対する色やテクスチャの計算に利用する。

Shadow mapping
3DCGにおいて影を生成するための手法。光源から見たシーンをレンダリングして、シャドウマップ（深度マップ）を生成し、実際のカメラ位置からシーンをレンダリングする際に、深度マップを参照して、各ピクセルに光が直接照らされているか、影になっているかを判定する。ソフトシャドウを利用して、より自然な影表現をすることも可能。
以下の画像（左：Shadow mappingあり、右： Shadow mappingなし）を比較すると、影表現が適切にされていることが分かる。



（４）PLATEAU Editorの主な機能

PLATEAU Editorでは主に以下の機能が利用可能である。詳しい使い方は、3.2を参照されたい。

- ワークスペースの作成・ユーザーの招待
- プロジェクトの作成
- レイヤーの配置・スタイルの設定
- GISデータの読み込み・表示
- シーンの設定変更（ベースマップ・テライン・カメラなどの各種設定）
- インフォボックス（レイヤーの詳細を表示する画面領域）の作成・編集
- ウィジェット（地球儀の上に表示されるさまざまなUI）の配置・編集
- プラグインのインストール（ウィジェットなどを機能拡張可能）
- プロジェクトの公開

（５）PLATEAU VIEWの主な機能

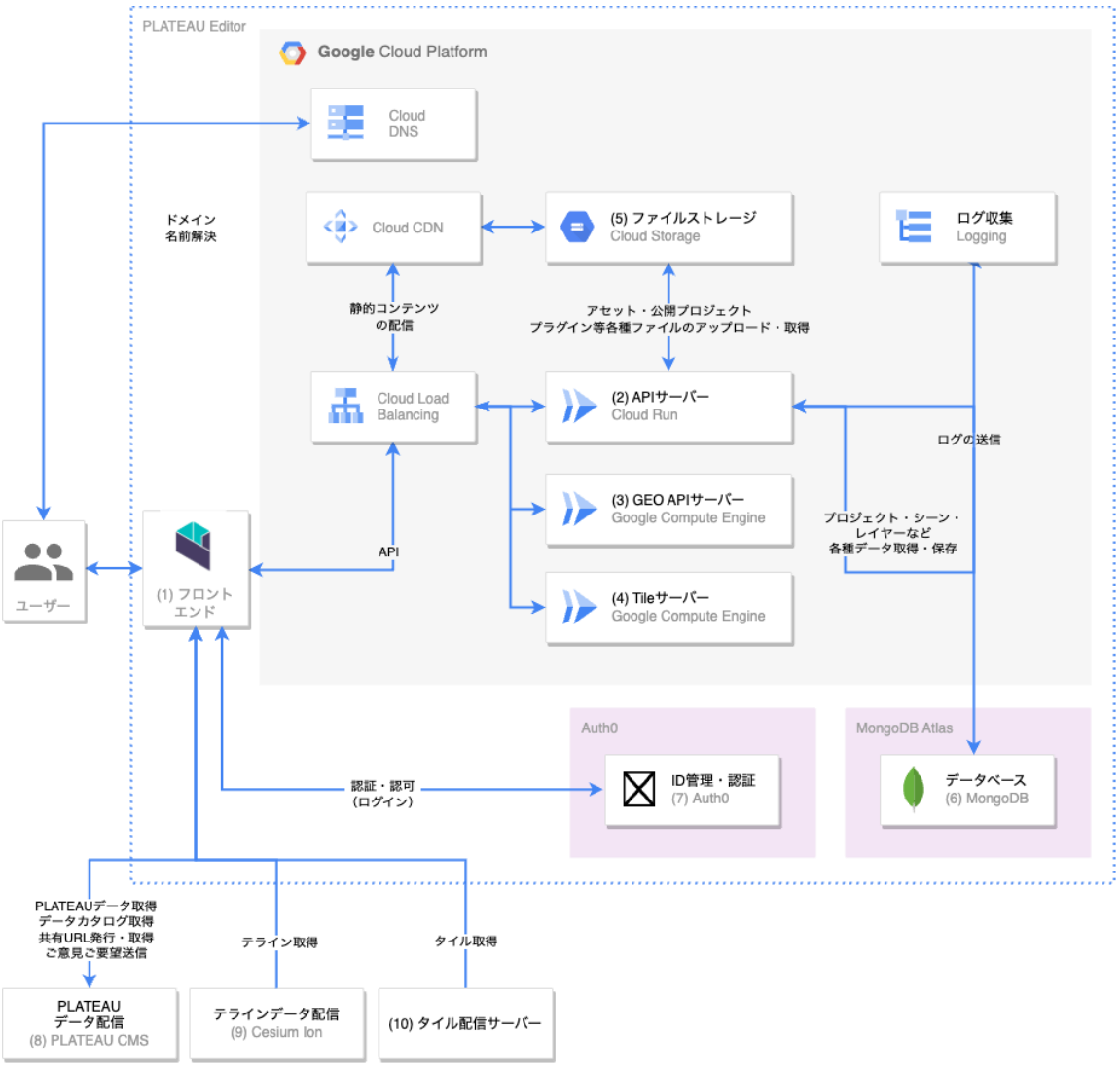
PLATEAU VIEWでは主に以下の機能が利用可能である。詳しい使い方は、3.3を参照されたい。

- ヒエラルキーウィンドウ
    - データセットの一覧表示
    - データセット及び住所の検索
    - 地図上へ追加済みレイヤーの表示及び選択
- 凡例の表示・絞り込み表示などの操作・地物の属性表示
- 3D図形の作図
- Google Street Viewとの連携
- 地図の設定の変更（ベースマップの切り替え、地図ラベルの表示）
- タイムラインによる時系列データの表示
- ストーリーテリングの表示と編集
- カメラ操作・現在地の表示
- 統計データ及び標高値によるヒートマップの表示
- 建築物検索機能・建築物クリップ機能
- 共有URLの発行
- ご意見ご要望

1.5.2 システム構成の全体像

PLATEAU VIEW 3.0におけるPLATEAU Editorのシステム構成について説明する。

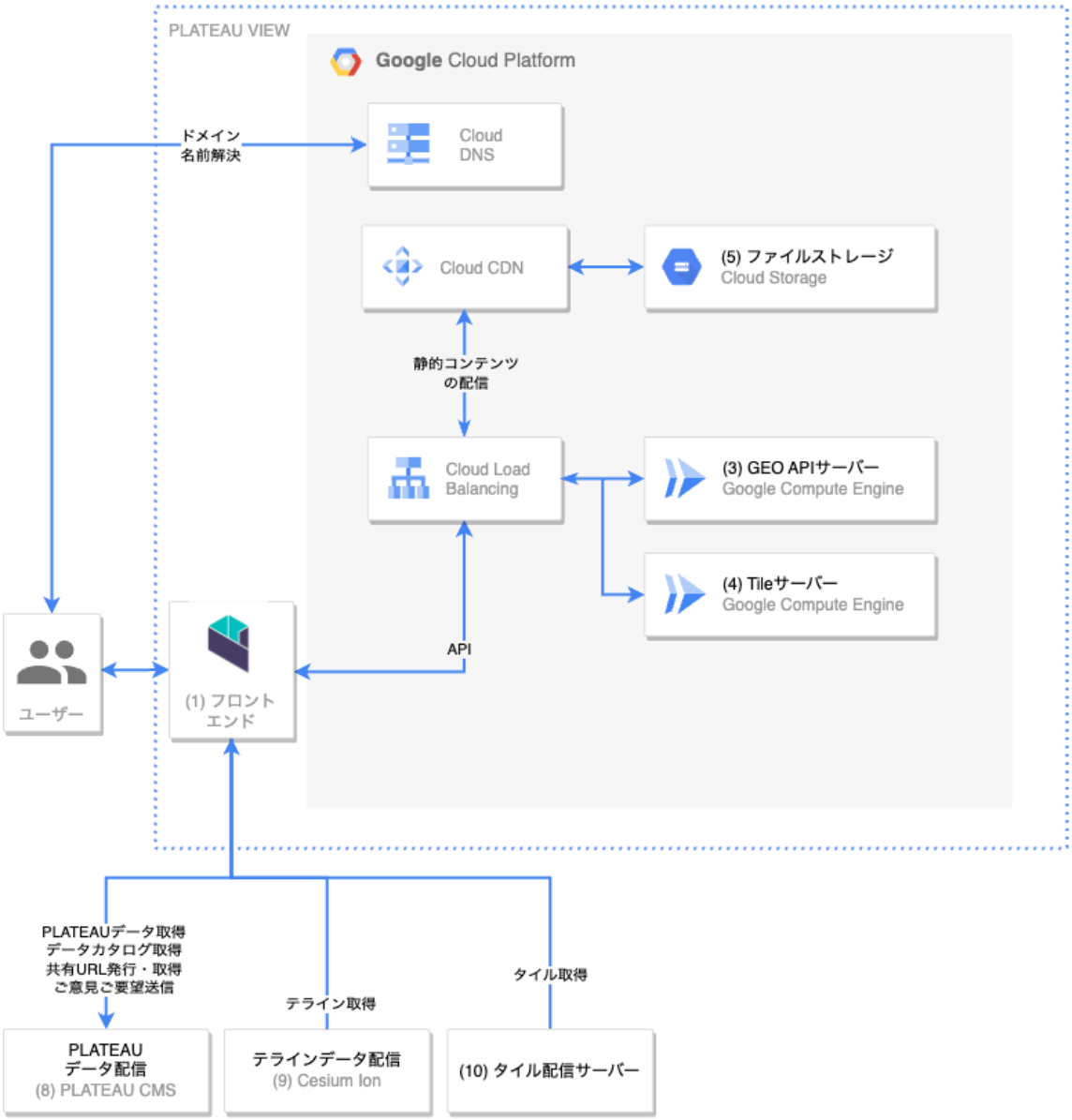
表　PLATEAU Editorのシステム構成



PLATEAU VIEWのシステム構成

PLATEAU VIEWのシステム構成について説明する。

表　PLATEAU VIEWのシステム構成



以下、上記構成図の各要素についてそれぞれ説明する。

なお、各コンポーネントは、保存データ量、ネットワーク転送量、CPU時間、マシンスペックなどに基づいて料金が発生する。

（１）フロントエンド

Webブラウザ上で動作する、（HTML・CSS・JavaScriptによる）フロントエンドアプリケーション。APIサーバー・Auth0・その他各種サーバーとの通信を行い、UIや地図を表示する。

（２）APIサーバー（Cloud Run）

PLATEAU EditorのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。MongoDBやファイルストレージと連携して、レイヤーの保存やプロジェクトの管理・公開などの、さまざまなビジネスロジックを実行する。

APIサーバーは、Cloud Run 上で動作する。Cloud Run とは、GCPで利用可能なサーバーレス（CaaS）プラットフォームであり、Dockerコンテナをデプロイすることで、サーバーの保守管理の手間なしに、アプリケーションをクラウド上で動作させることができる。同時接続リクエスト数が規定数以上に達すると自動的にコンテナが増加し、より多くのトラフィックを自動的に分散処理することができる。

Cloud Runは、デフォルト設定では、HTTPリクエストを受信して処理している間のみCPUが動作し、リクエストを処理していない時は動作を停止するため、HTTPリクエストを実際に受信し処理するために動作したCPU時間分のみが課金対象となる。この点が、常時稼働が前提となることが多いAWSのEC2やGCPのGCEとは異なる。

（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU EditorのAPIサーバーは、メモリ1GB・CPU2コアの設定で動作している。

（３）GEOサーバー（Cloud Run）

Editor及びViewerに必要な地理情報を処理するためのAPIサーバーであり、HTTPサーバーとして外部からのリクエストを受信している。住所検索や逆ジオコーディングなどEditor・Viewer上必要な機能を提供する。

（４）Tileサーバー（Google Compute Engine）

PLATEAU VIEW 3.0では、国土地理院が実験的に提供しているベクトルタイル形式の「地理院タイル」を利用している。フロントエンドで利用している、CesiumJSで利用できるよう、Tileサーバーにおいてベクトルタイルをラスター化し、Slippy Map Tilenames形式で配信している。

（５）ファイルストレージ（Goole Cloud Storage）

フロントエンドのアプリケーションのソースコードや画像、ユーザーによってアップロードされたアセットファイルや、プロジェクト公開時にビルドされる情報、インストールされたプラグインのファイルを保存する、オブジェクトストレージサーバー。

Google Cloud Storage（GCS）というサービスを使用しており、容量は無制限であり、自動的にスケーリングし、バックアップも自動的に行われ、データは複数拠点に分散配置される。巨大なファイルを格納・配信することが可能。

なお、PLATEAU VIEW 3.0のPLATEAU関連データセットなどのGISデータは、主にPLATEAU CMSのストレージサーバーで保存・配信されており、PLATEAU Editorで保存されているファイルの量はそれに比較してさほど大きくない。PLATEAU VIEW上で可視化されるGISデータは主にPLATEAU CMSから配信されるデータを使用している。

（６）MongoDB

MongoDBとは、ドキュメント指向のNoSQLデータベースで、データの柔軟性と拡張性が特徴。
PLATEAU EditorはデータベースとしてMongoDBを使用している。

PLATEAU Editorでは、MongoDB社が提供するクラウドサービス MongoDB Atlas を利用している。保守運用が自動化されたマネージドなMongoDBが利用可能で、自動的に3台以上のサーバーから成るクラスタを構成し、データベース内のデータは自動的に各サーバーに複製され、リクエストは分散処理されるようになっている。M0からM30まで、さまざまなマシンスペックのサーバーによる可用性の高いクラスタを構築することができ、マシンスペックによって料金が変わる。

（2024年3月現在）PLATEAU VIEW 3.0のPLATEAU Editor向けには、M10クラスタを運用している。

（７）Auth0

Auth0社が提供するクラウドサービス。アカウントの管理・認証・認可を行うIDプロバイダを提供する。Auth0を使用することで、開発者はアプリケーションに安全で使いやすく信頼性の高い認証認可機能を組み込むことができる。PLATEAU EditorもAuth0を使用して認証認可機能を実現している。

（2024年3月現在）テナントに対して登録されているユーザー数に応じて課金されるが、7000ユーザーまでは無料となっている。

（８）PLATEAU CMS

PLATEAU VIEWで利用可能なデータカタログや、3D都市モデルデータをはじめとする各種GISデータ等を配信しているシステム。詳しくは1.3を参照されたい。

（９）Cesium Ion

Cesium社が提供するクラウドサービス。PLATEAU VIEW 向けに独自に作成された、地球儀上の日本における地表面を表現する3Dのデータ（テラインデータ）を配信するために使用している。

（１０）タイル配信サーバー

PLATEAU VIEWの地図の設定で選択可能なベースマップについて、それぞれ以下のサーバーから配信されるタイルデータを使用している。

タイルデータとは、ユーザーからリクエストされた地図表示範囲に対して、あらかじめタイル状に分割された画像データのことであり、それらをサーバーから配信するサービスを使用することでタイルデータを取得してCesiumJS上で描画している。

- 全国最新写真 （シームレス）：PLATEAU VIEW 向けに独自に作成されたタイルデータ。
  PLATEAU VEIW 3.0向けに構築されたGCPのGCSから配信されている。

（１１）その他のコンポーネント

表　その他のコンポーネント

コンポーネント名	説明
Cloud CDN	GCPのCDN（Content Deliver Network = ウェブコンテンツをインターネット経由で配信するために最適化されたネットワーク）。GCSなどと組み合わせて使用することで、リクエスト元から地理的に近いサーバーにコンテンツのキャッシュを自動的に配置し、コンテンツ配信を高速化・効率化させることができる。
Cloud DNS	GCPのDNS。PLATEAU Editorで使用しているドメインに対応するレコードは全てCloud DNSで管理されている。
Cloud Load Balancing	GCPのマネージドなロードバランサ（負荷分散システム）。PLATEAU VIEW 3.0のPLATEAU Editorで使用されるドメインのIPアドレスは全てCloud Load Balancingに向いており、リクエストのホスト（ドメイン）に応じてAPIサーバーやストレージサーバーに自動的にルーティングされる。
Cloud Logging	GCPのログ収集サービス。Cloud Runなどから出力されるログを閲覧可能。システムのトラブルシューティング時に役立つ。

1.6 PLATEAU VIEW におけるWebAR検証

1.6.1 現状課題と課題解決のアプローチ

スマートフォン等から取得されるGPS情報等を用いたAR体験を提供するアプリケーションは、一般的にはクライアントアプリとしてシステムとデータをスマホ本体に保持する形で提供されている。これは、高精度の位置測位のためにIMUやVPS（Visual Positioning System）などクライアント側の処理を必要とする位置測位技術を用いるため、あるいはリッチなユーザー体験を提供するために大容量のコンテンツを保持するためといった要因による。

一方で、PLATEAUが提供する3D都市モデルは全国スケールで提供されるビッグデータであるため、3D都市モデルを利用したAR体験をグローバルに提供するためには、ウェブアプリとの組合せが最適となる。インストール不要でブラウザから利用できるWebARアプリケーションはこれまでもいくつか存在しているが、位置精度が低いことや、リッチなユーザー体験の提供が難しいことなど、未だ多くの課題を抱えている。

これら既存の課題を解決しつつ、3D都市モデルを用いたWebARアプリの構築を行うため、2023年度のProject PLATEAUでは、ストリーミングによって取得される3D都市モデルのデータをブラウザベースのウェブアプリによってAR表示する「PLATEAU WebAR（仮称）」のプロトタイプ版を開発した。このプロジェクトでは、ストリーミングに最適化された3D都市モデルのレンダリングフォーマットである3DTilesを用いた（CesiumJSエンジンを用いた）WebARアプリを開発することで、ウェブベースでのAR体験の価値検証や、位置精度向上の方策の検討、PLATEAU VIEWと連携したWebGISとWebARの統合を実現するための技術的ナレッジの獲得等を目的としている。

1.6.2 創出価値

PLATEAUが提供する3D都市モデルの表示やこれをベースにしたコンテンツの重畳等が可能なWebARアプリケーションの開発ナレッジを蓄積することで、 WebGISとWebARを組み合わせた新たなユーザー体験の提供を目指す。

1.6.3 実証システム

実証に用いたシステムアーキテクチャ、ハードウェアにおいては以下のとおりである。ハードウェアは、①国内における高いシェアを保持していること、②各メーカー製品の代表的な端末であること、③WebARを実現する十分なスペックであることから選定を行った。

1.6.4 システム機能一覧とソフトウェア・ライブラリ

実証で用いた機能およびソフトウェア・ライブラリは以下のとおり。

表　機能一覧

分類	ID	機能名	機能説明
基本機能	FN101	3D表示	• 選択されたデータを3Dでカメラビューとともに表示。
	FN102	レイヤ切替	• 表示対象のデータを一覧で表示し、選択することにより3D表示で表示。
	FN103	属性表示	• 3D表示上で選択したデータの属性を表示。
検索機能	FN105	データ検索	• ユースケースや地点からデータカタログを検索可能。
ビュー調整機能	FN106	FOV調整	• 端末により異なるFOVを調整する機能。
	FN107	コンパス補正機能	• コンパスがずれている際に補正する機能。

表　利用したソフトウェア・ライブラリ

ID	項目	説明
SL001	TypeScript	• プログラミング言語の1つで、Microsoftによって開発された。JavaScriptに静的型付けを加えたスーパーセットであり、大規模システムの開発に用いられる。フロントエンド向け開発に利用。
SL002	React	• Meta社によって開発されたUI構築のためのJavaScriptライブラリ。特に大規模かつ複雑なUI実装において利用される。PLATEAU CMSではフロントエンド向け開発に利用されている。
SL003	CesiumJS	• デジタル3D地球儀上にさまざまな情報を描画することができる地図エンジン。Webブラウザ上で動作し、WebGLを用いて描画を行うため、PCやスマートフォンで閲覧することができる。
SL004	Vite	• Typescript/Reactベースの本アプリケーションをブラウザで動作するようにするためのビルドツール。Evan Youによって開発されたOSSツール。
SL005	GraphQL	• API向けに作られたクエリ言語およびランタイムを指す。WebAPIの開発において、RESTなどの方式と比較して、より柔軟かつ効率的なAPIの提供を可能にする。PLATEAU CMSではバックエンドとフロントエンド間の通信に利用されている。
SL006	Apollo	• Meteor Development Groupによって開発された。GraphQL APIを簡単かつ効率的に行うことができるツール。

図　システムアーキテクチャ

WebARシステムはフロントエンドのみで構成されたアプリケーションであり、Google Cloud Storage上にホスティングされている。

データの取得には、アクセスされたURLのパラメータもしくはユーザーがUIを操作することでPLATEAU CMSからGraphQL（Apollo）を利用して3D Tilesデータを取得している。

データの表示にはフロントエンドのUIライブラリとしてReactを用い、3D都市モデルのデータ形式である3D TilesのレンダリングエンジンとしてCesiumJSを用いた。また、これらの開発環境として開発言語にTypescript、ビルド環境にViteを用いている。

3D都市モデルのWebAR上でのレンダリングエンジンとしてCesiumJSを用いた理由として、PLATEAUエコシステムにおいて、すでに3D TilesのデータがPLATEAU CMSを介して整備されており、CesiumJSを使うことでスマートフォンのブラウザ上でもデータが描画できることを確認済みであったことに加え、スマートフォンの姿勢情報をCesiumJSのカメラと連動させることで擬似的にARとして動作する仕組みを確立できたためである。

以下、ライブラリ選定の際に検討したものの最終的な利用には至らなかったライブラリと非選定の理由は以下のとおり。

表　利用には至らなかったライブラリ

名前	説明	非選定理由
A-Frame	Mozzila社製のHTMLで簡単にモバイルやヘッドマウントディスプレイ用WebVRを作成するために開発されたフレームワーク。 https://aframe.io/	AR開発機能は同梱していないので別途AR.js等を組み合わせる必要があり、これだけでは完結しないため。
Babylon.js	Microsoft製オープンソース3DCGエンジン。もとの設計はSliverlight/WPFベースのゲームエンジンに基づいており、全てのモデリング、プログラミングはjs上で行える。 https://www.babylonjs.com/	3D Tilesのデータを空間座標系に変換するための実装が別途必要になるため。
Three.js	オープンソースのWebGLラッパーライブラリ。レンダラーはWebGLが基本だが、他にも Canvas、SVG、CSS、DOM が選択できそれらの実装を吸収してくれる。 https://threejs.org/	3D Tilesのデータを空間座標系に変換するための実装が別途必要になるため。
Immersal	APIを通してVPS機能にアクセス可能なWeb ARフレームワーク。VPSを使うことでより正確な位置測位が可能。しかしVPSを利用するにはマップを自作する必要あり。 https://immersal.com/	VPS機能を使うために全国的にImmersalのマップを作ることが現実的ではない。また座標系を緯度経度等のグローバルな座標系に直す必要があるため。

WebARとして3D都市モデルのデータを端末の動きに合わせて現実世界に重畳させるためには以下の要件が必要となる。以下では、各要件の実現方法について詳細を記載する。

- カメラビューの利用
- GPSによる現在地（緯度・経度・高さ）の取得
- 端末のIMU（Inertial Measurement Unit=慢性計測装置）の動きに合わせて現在地から見える3Dモデルを移動させる
- 端末ごとによる視野角（FOV）とコンパスのズレを補正し、ビューに3D都市モデルをあわせる

カメラビューの利用

カメラビューの利用にはCesiumJSのskyboxをオフにし、透明化された空間とブラウザのcanvas要素を重ね合わせ、ブラウザのカメラAPIを用いてcanvas上にカメラの映像を出すことで実現した。
描画の負荷としては、LOD2のテクスチャつき3D都市モデルであれば快適に描画される。これはCesiumJSが表示範囲をカメラの位置により最適化していることの恩恵も受けている。

一方で、LOD4等データの転送量が桁違いのものに関してはフレームレート、描画速度ともに実用に耐えるレベルではなかった。

図　Cesiumを透明化する該当コード

```
// Initialize the Cesium Viewer in the HTML element with the `cesiumContainer` ID.
cesiumViewer = new Cesium.Viewer("cesium_container", {
 animation: false,
 baseLayerPicker: false,
 fullscreenButton: false,
 geocoder: false,
 homeButton: false,
 infoBox: false,
 sceneModePicker: false,
 selectionIndicator: false,
 timeline: false,
 navigationHelpButton: false,
 navigationInstructionsInitiallyVisible: false,
 skyBox: false, // スカイボックス無効化
 skyAtmosphere: false, // 空を非表示
 contextOptions: {
 webgl: {
  preserveDrawingBuffer: true,
  alpha: true, // 透過
 },
 },
 globe: false, // 地球を非表示
});
// 背景透過
cesiumViewer.scene.backgroundColor = Cesium.Color.TRANSPARENT;

// Cesiumのカメラを取得
cesiumCamera = cesiumViewer.camera;
```

図　透明化したCesiumとカメラを重ね合わせる該当コード

```
<div>
 <video
 id="device_camera_preview"
 autoPlay muted playsInline
 className="absolute top-0 left-0 w-full h-full object-cover">
 </video>
 <div
 id="cesium_container"
 className="absolute top-0 left-0 w-full h-full">
 </div>
 ...
</div>
```

図　カメラビューの利用

GPSによる現在地（緯度・経度・高さ）の取得

GPSの取得にはWebブラウザのAPIであるGeolocation APIを用いて取得した。

GPSでは屋内や高層ビルに囲まれた街などでは精度が出ないことが予測されるため、Immersalなどの VPSを用いた位置測位も検討した。

しかし、VPSシステムでは対象地域のマップを手動で作成する膨大な作業が必要であったり、3D都市モデルデータから作成する手法も本マニュアル作成時点では精度が高く出ないことがわかっている。

そのため本検証ではGPSを利用した開発にとどめている。

端末のIMUの動きに合わせて現在地から見える3Dモデルを動かす

端末から加速度・ジャイロ・コンパスを取得することで、端末の姿勢計測が可能になる。この値とGPSを合わせることで、地球上のどこにいて、端末がどこを向いているのかが測位できる。

加速度・ジャイロ・コンパスを取得するためにWebブラウザから提供されているDevice Orientation APIを利用した。（iOSではdeviceOrientationAbsoluteが存在しないため、代わりに deviceorientation/webkitCompassHeadingを利用している）

このAPIから取得できるalpha/beta/gammaの値は以下のとおり。

- alpha：画面平面中心と直交する軸z（画面より手前が正）を中心として、デバイスを地面に対して水平にしたとき北向きを0とし、軸を正の方向に見て時計回り（画面を見る者からは反時計回り）に全周で360までの値を返す。
- beta：画面平面中心を原点とする妻手方向の軸x（画面右方向が正）を中心として、デバイスを地面に対して水平にしたときを0とし、機首上げ方向（軸を正の方向に見て時計回り）にピッチをとると+180、機首下げ方向（軸を正の方向に見て反時計回り）にピッチをとると-180までの値を返す
- gamma：画面平面中心を原点とする長手方向の軸y（画面上方向が正）を中心として、デバイスを地面に対して水平にしたときを0とし、軸を正の方向に見て時計回りにロールさせると+90、軸を正の方向に見て反時計回りにロールさせると-90までの値を返す （裏返しの場合も同様）

alpha/beta/gammaの値を素直に変換できるのは、端末を地面に対して水平にしているときのみであり、Heading/Pitch/Rollの値に変換することはできない。それ以外の場合は、alpha, betaの性質（それぞれの辺の端点の地面からの距離の差が値になること）が、Heading/Pitch/Rollへの変換には不適切になる。また、alpha/beta/gammaのオイラー角のままでは、betaの値が90度になるとalpha/gammaが180度飛んでしまうジンバルロックが発生する。
そのため、Heading/Pitch/Rollのorientationではなく、alpha/beta/gammaの値を回転行列または四元数に変換してから別の方法でカメラに適用する方が適切となる。
このとき、endTransformを用いて回転行列でカメラの向きを制御する方法も検討しうるが、動作が不安定であったことから、今回は使用せず、direction/upベクトルを指定する方法をとった。
[参考]https://community.cesium.com/t/control-cesium-camera-with-device-orientation/6844
https://groups.google.com/g/cesium-dev/c/cr2P2wfOwl4

上記より、 alpha/beta/gammaのオイラー角を3次元回転行列に変換していく。
W3Cの定義によると、デバイスオリエンテーションの一連のローテーションは、intrinsic Tait-Bryan angles （オイラー角） of type Z-X'-Y" と定められている。これは、intrinsicなのでデバイスに追従するデバイス座標系であり、X'は最初にZ軸周りの回転を適用した後のX軸、Y"はさらにX'軸周りの回転を適用した後のY軸を表す。

なお右手左手系については、軸の正な方向に向かって眺めたときに、時計回りが軸周りの正な回転であるとするため、右手系とする。
[参考] https://www.w3.org/TR/orientation-event/#device-orientation-model
Tait-Bryan angles （オイラー角） から回転行列への変換は、下記のW3Cのドキュメント内でも取り扱われているので、それに従いつつCesiumJSを使用したバージョンとして実装した。
[参考] https://www.w3.org/TR/orientation-event/#worked-example-2

この実装を通してCesiumJSのカメラの動きをデバイスの姿勢情報と連携し、モデルがカメラの動きによって動くようになる。

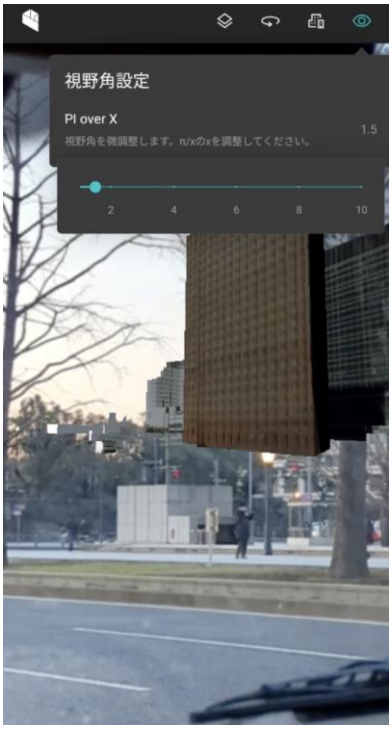
端末ごとによる視野角（FOV）とコンパスのズレを補正し、ビューに3D都市モデルをあわせる

スマートフォンの端末によってカメラのビューの視野角（FOV： Field of View）が変わる。視野角はAPIを通しても取得することはできず、昨今の複数眼カメラを持つようなデバイスもあるが、WebのカメラAPIにおいてはデフォルトの１つしか取得することができない。

そのため、ユーザー側で使用している端末によって変わる視野角を補正してもらう必要がある。カメラのビュー自体は制御できないため、3D都市モデルのパースをCesiumJS側のカメラを修正する形で補正している。

ユーザーはUI上のスライダーを操作することでFOVを連続的に変更可能となっている。

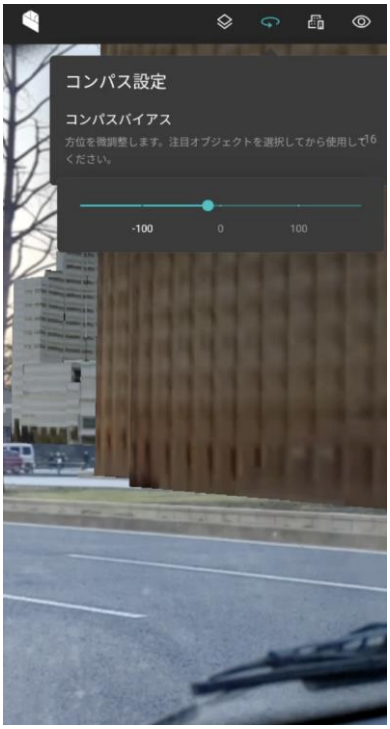
図　スライダーによる裾野角の変更



また、描画のズレにはFOVだけでなく、コンパスのズレによっても水平方向に起きてしまう。

コンパスのズレは周りの地磁気などの影響を受けて起きてしまうため、こちらもソフトウェア側で自動で補正することが難しい。そのため、こちらもユーザーの操作によりコンパスのズレによるモデル表示のズレを補正するUIを開発した。ズレは水平方向のみのため、alphaにUIからの補正値を加えることでズレを補正する機能を実現している。

図　スライダーによるコンパス設定



1.6.5　実証に用いたデータ

フォーマット	説明	対応する地物
3D Tiles	CesiumJSによって開発された、Web上で3D地理空間コンテンツを効率的にストリーミングするためのデータ形式。 ドキュメント https://github.com/CesiumGS/3d-tiles	建築物モデル

1.6.6 ユーザーインターフェース

本検証では、メイン画面よりFN101~107の各機能を呼び出すよう実装を行った。画面仕様の詳細、及び画面遷移は以下のとおりである。

【SC001】メイン画面

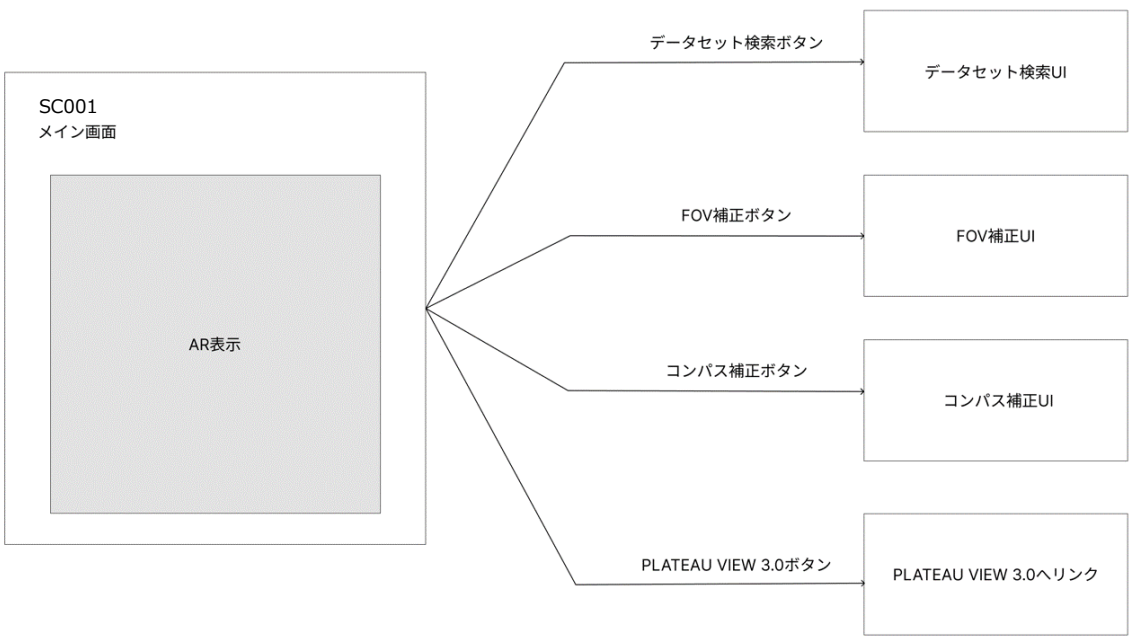
画面の目的・概要

- 3D都市モデルのARビューワと、検索やカメラ・コンパス補正が可能な画面。
- 3D都市モデルARビューワでは現実世界をカメラを通して表示し、そこに位置情報に基づいて3D都市モデルを表示することが可能。
- 画面上部の検索プルダウンから、表示するデータを選択可能。
- 画面上部のFOV設定からデバイスごとのFOVの違いによる3Dモデルのズレを補正可能。
- 画面上部のコンパス設定からコンパスのブレによる3Dモデルのズレを補正可能。

表　画面一覧

ID	画面名	画面説明	画面を表示した機能（ID）
SC001	メイン画面	• 都市モデルをARビューに表示し、現実世界に重畳する画面。また、データ検索やFOV、コンパスの設定が可能。	FN101-107

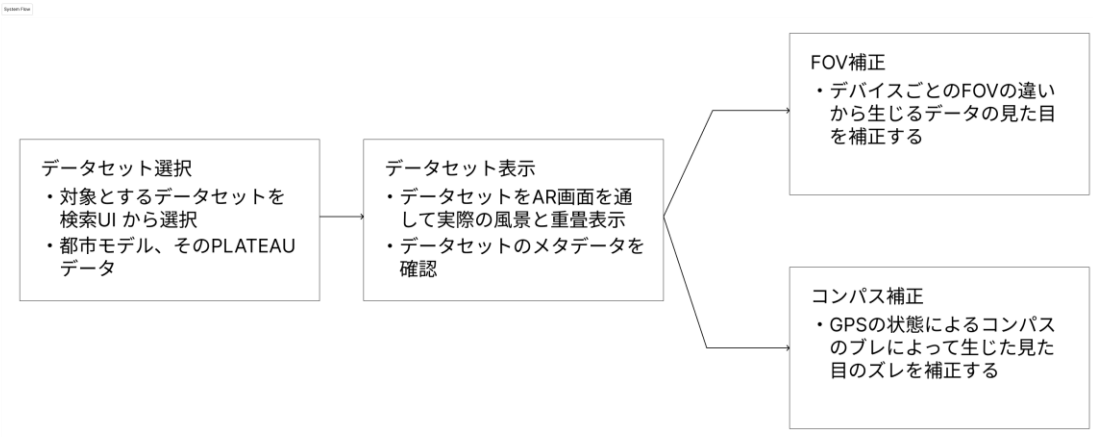
図　画面遷移図



1.6.7 実証システムの利用手順

対象とする建物を検索し、AR画面に表示・重畳し、データを現実世界とともに閲覧しながら施策の評価や検証を行う。

表　実証システムの利用フロー



利用フロー詳細

1. PLATEAU VIEWからARビューへ遷移する。
    - PLATEAU VIEW 3.0をスマートフォンで開くとARボタンが表示される。
    - データを選択した状態でこちらのボタンを押下することで、選択されているデータをARビューで確認が可能となる。

図　PLATEAU VIEWからARビューへの遷移

2. ARビュー中でのデータを選択する。
    - ARビューに遷移後は、実際に選択したデータがある場所まで行き、データをカメラビューに重畳する形で確認することができるようになる。
    - ARビューはURL（http://ar.plateauview.mlit.go.jp）から直接アクセスすることも可能である。
      注：URLからアクセスする際にはデータは何も選択されていない状態でのアクセスとなる。
    - ARビューに遷移後もPLATEAU VIEW 3.0と同様にデータを選択、表示でき、データ選択のアイコンをタップし、データを選択、表示することで現地でカメラビューに重畳することができる。
    - 表示したデータをタップすることで、そのデータのメタデータなども閲覧可能である。

図　ARビュー中のデータ選択

3. 見た目を補正する。
    - コンパスの補正
        - ARビューではコンパスのブレによる見た目の補正を行うことができる。画面右上のコンパスのアイコンをタップすることで、コンパスのブレによる見た目を補正するスライダーが表示される。こちらを動かすことで表示中の3D都市モデルが移動し、現実世界に合うように補正することが可能となる。
    - FOVの補正
        - スマートフォンは端末によりFOV（Field Of View：視野のこと）が異なるため、FOVの違いによる補正をしないと表示している3D都市モデルがカメラの映像に合わなくなってしまう。
        - 画面右上のFOVのアイコンをタップすることで、FOVの違いによる見た目を補正するスライダーが表示される。こちらを動かすことで表示中の3D都市モデルが動き、現実世界に合うように補正することが可能となる。

図　コンパスの補正

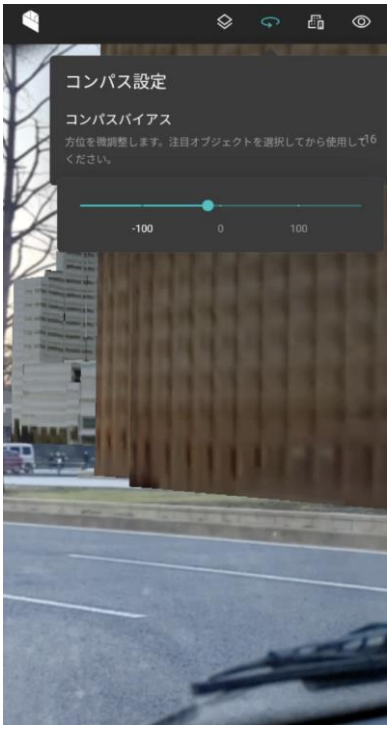
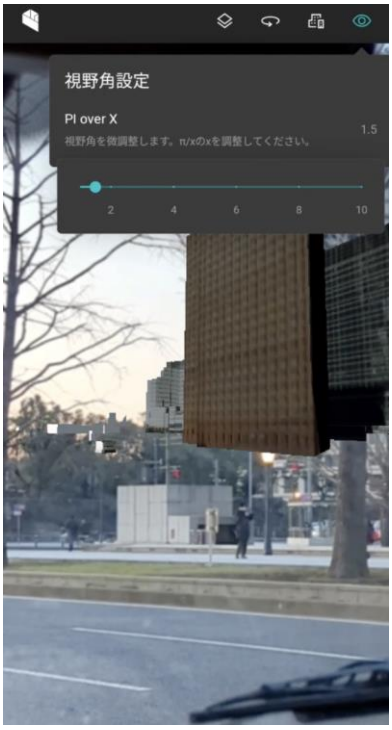


図　FOVの補正



1.6.8 実証の成果

実証実験を通じた成果として、3D都市モデルを活用することによる以下のような技術面での優位性が示された。

表　3D都市モデルの技術面での優位性

大項目	小項目	詳細
システム・機能	WebARでの現在地に基づいたモデル表示	3D都市モデルにより現在地におけるリアルな3D都市モデルが現実の映像とマッチし、動きにスムーズに追従することが可能。 Webブラウザ上でデバイスへのインストールなしで3D都市モデルを扱う基盤を作ることが可能。
アルゴリズム	デバイスの姿勢に応じたCesiumJS内カメラの追従	CesiumJS上のカメラをデバイスの姿勢情報と方角をもとに同期させ、現実のカメラに合わせて3D都市モデルが滑らかに動くようになった。
	FOV補正	デバイスごとに違うFOVを補正できるスライダーを実装し、見た目の補正を行うことが可能。
	コンパス補正	場所によって変化してしまうコンパスの精度のズレが原因で起きてしまう3D都市モデルの表示のズレを補正することが可能。

1.6.9 今後の展望

本プロジェクトにて開発したPoCではこれまで難しいとされていたロケーションベースのWebARにおける3D都市モデルの利活用の可能性を検証することができた。他方、データを検索するUI/UXやモデル表示の精度についてさらなる改善の余地も認められた。

例えば、今回のシステムでは自己位置測位をスマートフォンのGPSのみによって行ったが、位置測位の不安定性が明らかとなった。この点については、VPS(Visual Positioning System) など他の位置測位技術との組合せによって解消可能であり、ウェブで利用可能なスケーラブルなVPSの研究を進める必要がある。

また、表示モデルのクオリティについてはも改善の余地がある。この点については、PLATEAU VIEW 3.0同様、モデルへのシェーダー処理を通して見た目を補正することがあり得る。他方、スマートフォンでの動作を前提とすると、パフォーマンスについても配慮する必要があり、スマートフォン描画のための最適化（現在地周辺のみの描画にとどめたフィルタリング、スマートフォン用データの最適化等）を検討する必要がある。

対応データについても、今後はユーザーによるデータのアップロード機能を拡充することで、よりユーザーフレンドリーなアプリケーションとすることができる。

これらの課題解決を図ることで、都市計画や防災などの分野でWebARの活用を拡大し、今までにない体験価値を提供できると期待される。

第２章　PLATEAU VIEWの提供機能

2. PLATEAU VIEWの提供機能

本章では、PLATEAU CMS、Editor、VIEWそれぞれの主要機能の解説と、それらによって実現できることを解説する。

各機能の利用手順は、「PLATEAU VIEW 3.0 データ登録マニュアル」にそれぞれのユーザー種別向けに記載があるため、参照されたい。

また、PLATEAU VIEWに関しては、

- 「TOPIC 2｜PLATEAU VIEWで体験する[1/2]｜3D都市モデルをブラウザで利用」
- 「TOPIC 2｜PLATEAU VIEWで体験する[2/2]｜他の地理空間情報を重ねて確認」

にチュートリアル形式で操作方法が紹介されている。そのため、本章では機能の利用方法ではなく、機能の実現にあたって検討された背景課題や検証結果を記載することとする。

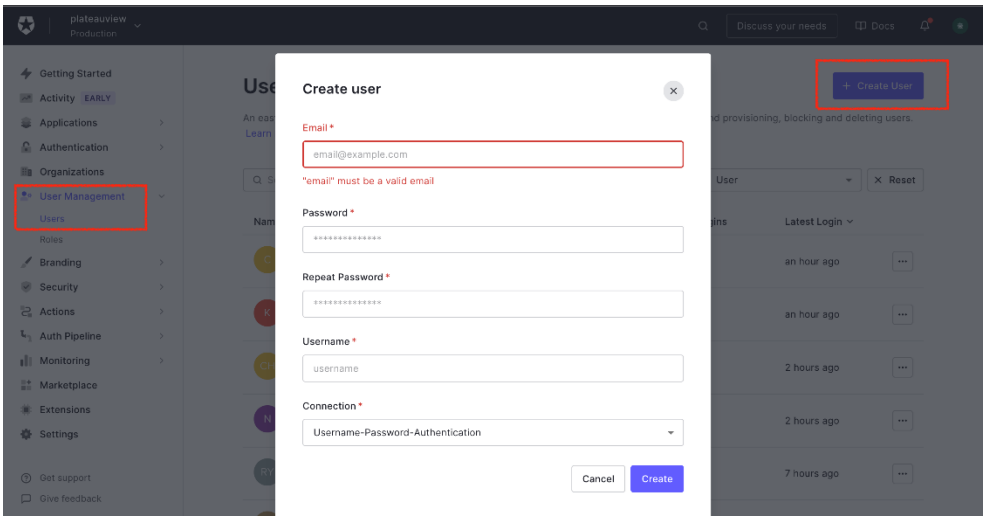
2.1 PLATEAU CMSの機能
2.1.1 管理者向け機能

（1）アカウント管理機能
アカウント作成機能
CMSでは、管理者としてCMSを利用するユーザーの管理を行うことができる。
ワークスペースへの追加対象者の情報を準備する。必要となる情報は次のとおり。

- 所属会社
- メールアドレス
- 対象者氏名

CMSの管理者は、Auth0のテナントで、招待するメンバー作成することができる。その後、以下のURLからCMSの利用が可能になる。

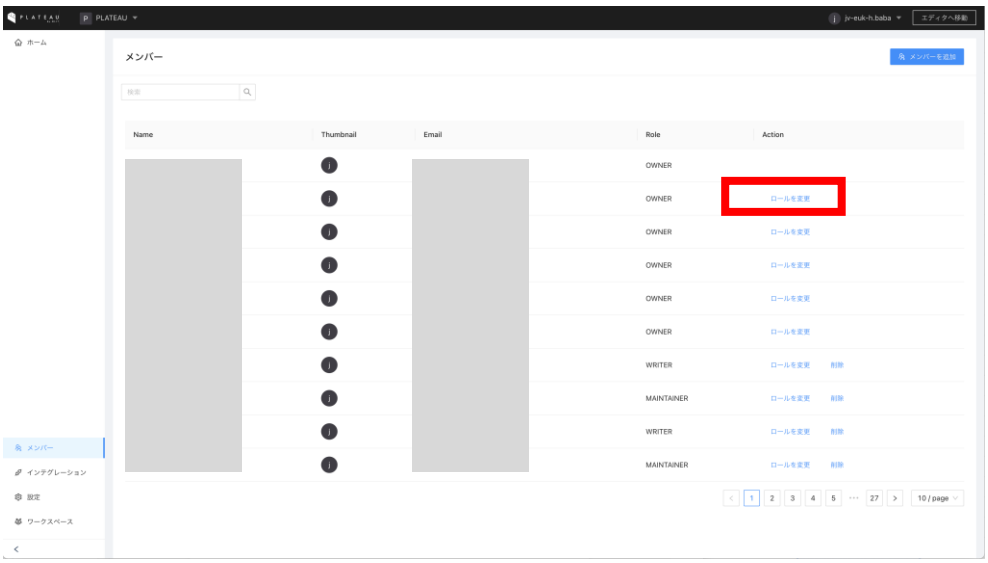
- URL: https://cms.plateauview.mlit.go.jp/
- ID: 個人のメールアドレス
- パスワード: 入力したパスワード



アカウント権限変更機能

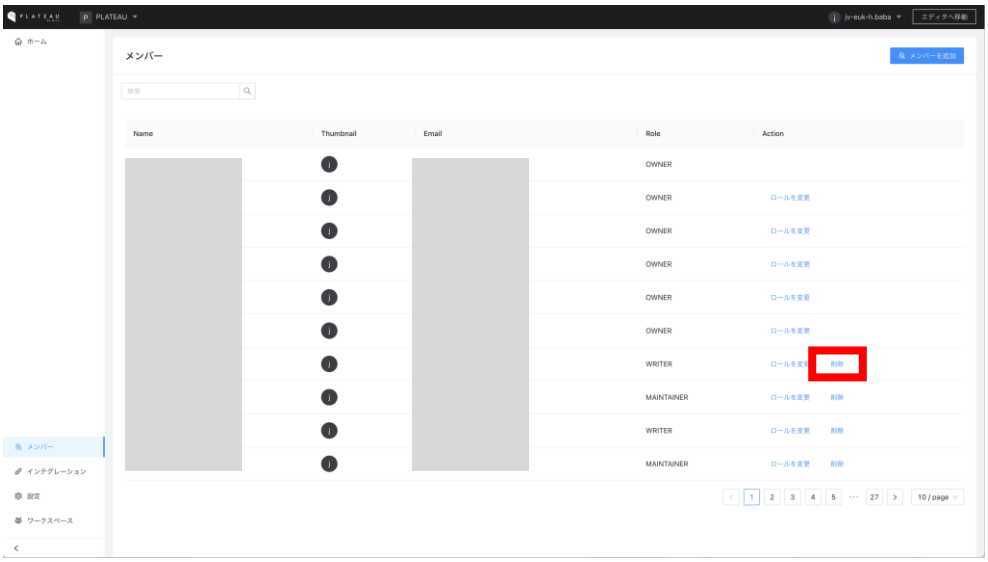
CMSでは、管理者はワークスペース内の権限管理を行うことができる。これによって、多数のユーザーが共同で取り組むプロジェクトにおいても、適切な操作のみをユーザーに許容することができる。権限の種類は以下のとおり。

- オーナー: ワークスペースへのメンバーの招待や削除含めて全ての操作が可能なユーザー
- メインテイナー: ワークスペースへのメンバーの招待や削除以外の操作が可能なユーザー
- 編集者: コンテンツ・アセット・リクエスト・コメントの作成・編集が可能なユーザー
- 閲覧者: 閲覧権限のみ付与されたユーザー



ワークスペースからアカウントの削除

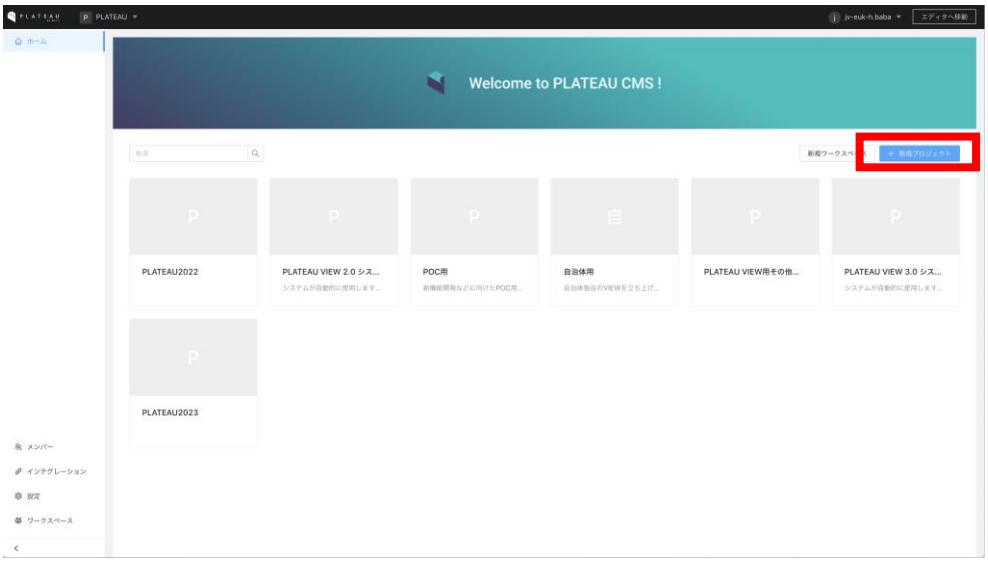
PLATEAU CMSでは、管理者はワークスペース内のユーザーを削除することができる。



（2）プロジェクトの編集機能

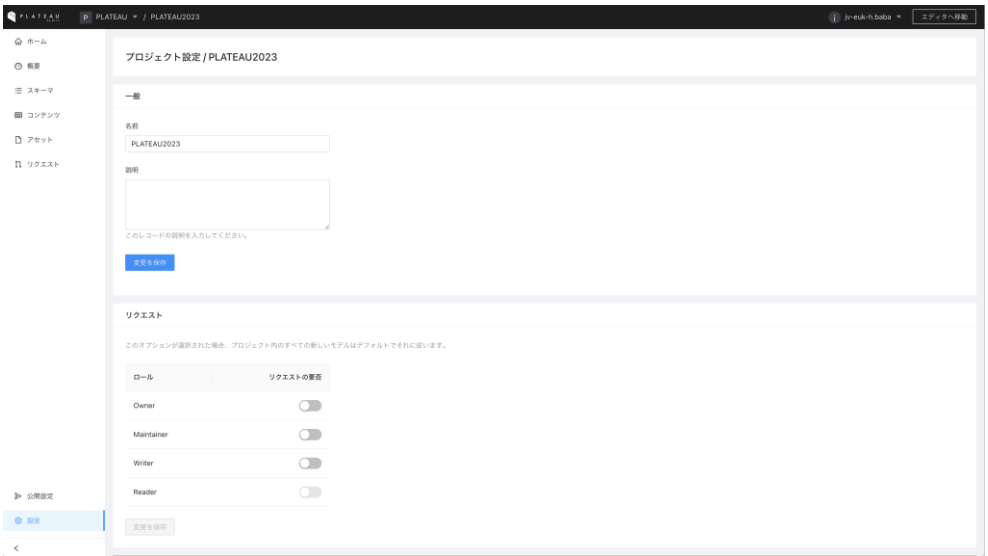
プロジェクトの作成

CMSでは、プロジェクトという単位でデータの管理を行っている。プロジェクトは、任意の目的に応じて管理者が作成・管理することができる。以下の画面では、プロジェクトの一覧表示や新規作成を行うことができる。



プロジェクト設定の変更

プロジェクトの設定ページでは、プロジェクト名や説明の変更ができる。また、リクエストの設定を変更することで、各権限を持ったユーザーがアイテムを直接公開するか、リクエスト機能を経て公開するかを設定できる。リクエスト機能を利用する場合には、レビュワーが承認をすることで、データをCMSから一般公開することができる。

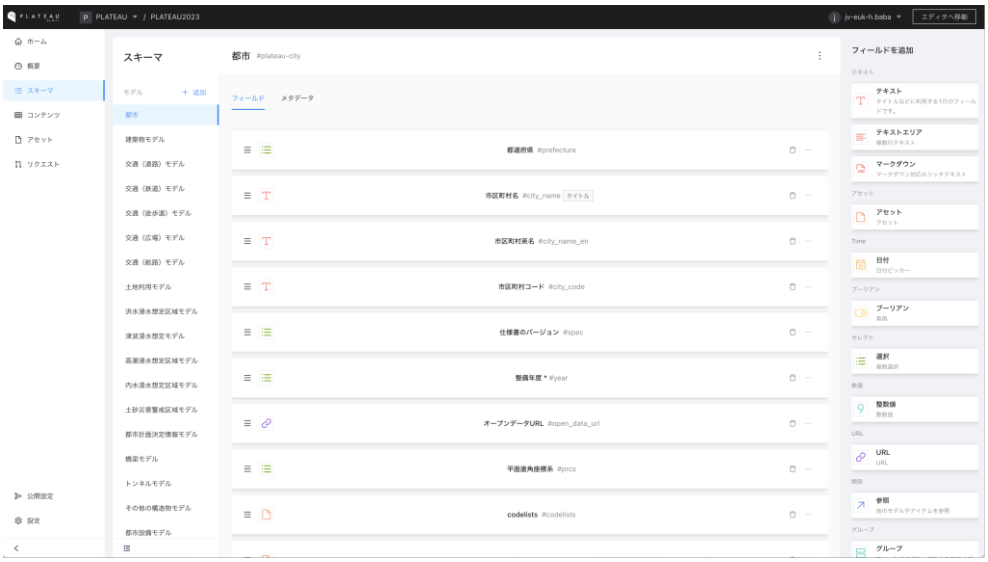


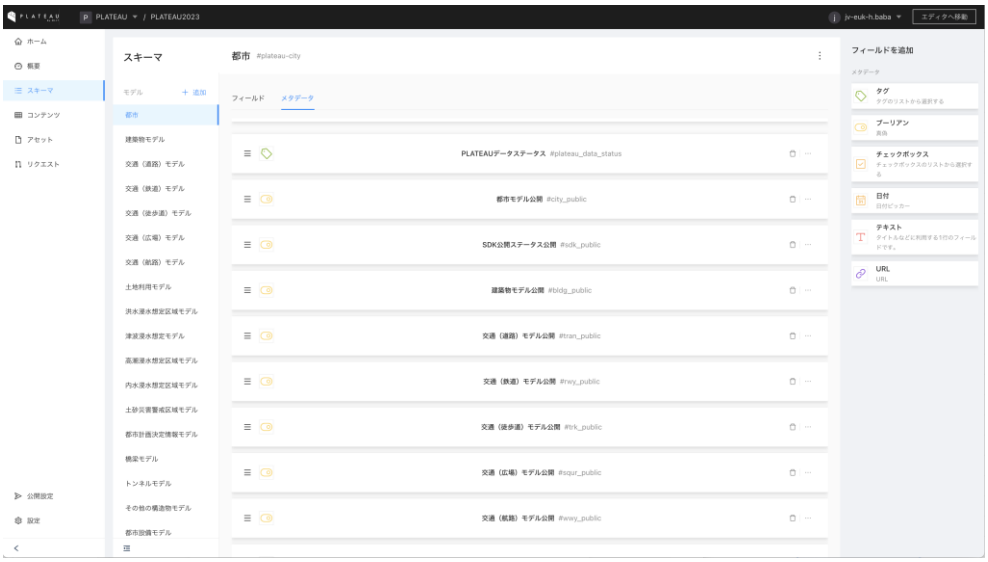
（3）スキーマ編集機能

スキーマの作成

プロジェクト内には、スキーマが存在し、管理者は自由にスキーマを変更することができる。スキーマとは、CMSへ登録するデータ自体のデータ構造を定義するもので、これにより多数のユーザーが共同でデータ登録作業を行なっても、決まった形式でデータを登録することができる。2023年度プロジェクトにおけるスキーマの詳細は、3.1.6で解説するため、ここではスキーマ機能の概要を述べる。スキーマには大きく2種類あり、それぞれ目的に応じて設定をする。2023年度版CMSでは、メタデータにはデータ登録のステータス管理するフィールドなどを設定している。

- フィールド: 登録するデータ自体のデータ構造を定義する。（画像1枚目）
- メタデータ: 登録するデータに対するメタデータ構造を定義する。（画像2枚目）

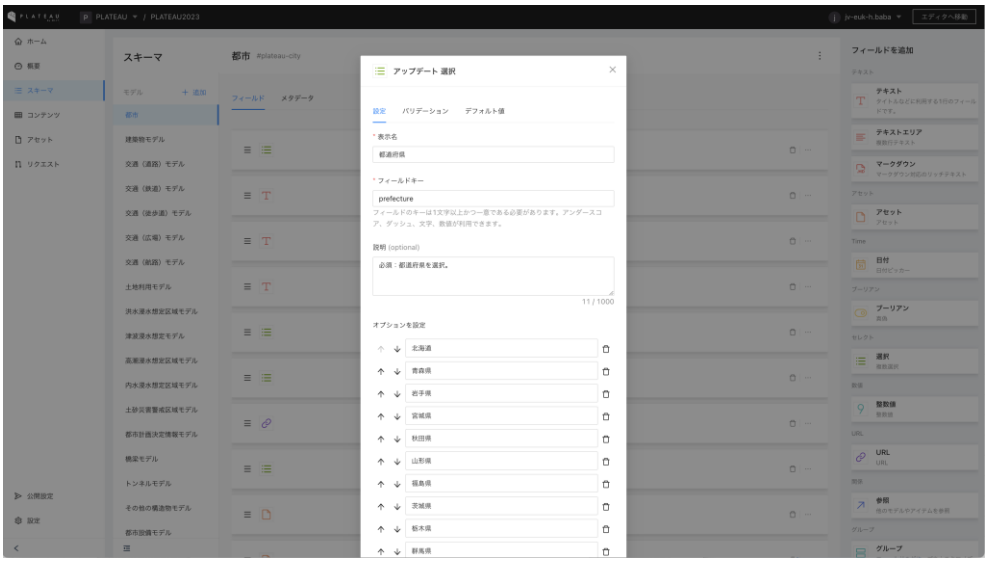




スキーマの変更

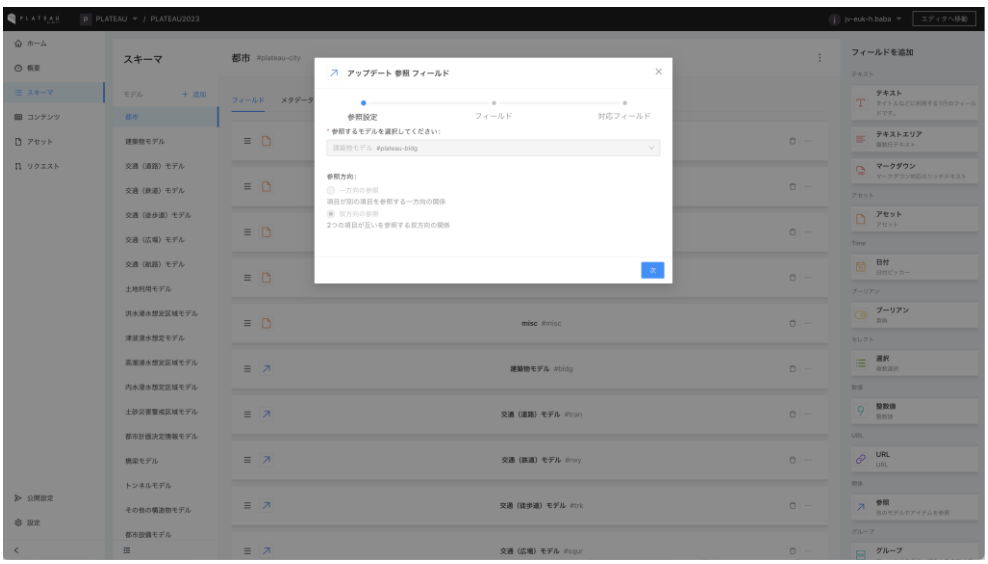
スキーマは後から変更することもできる。スキーマのフィールドの設定変更、フィールドの並び順変更、フィールド自体の削除などが可能である。

また、特殊なフィールドとして、「参照フィールド」と「グループフィールド」が存在する。これらについて、以下で解説する。



参照フィールド

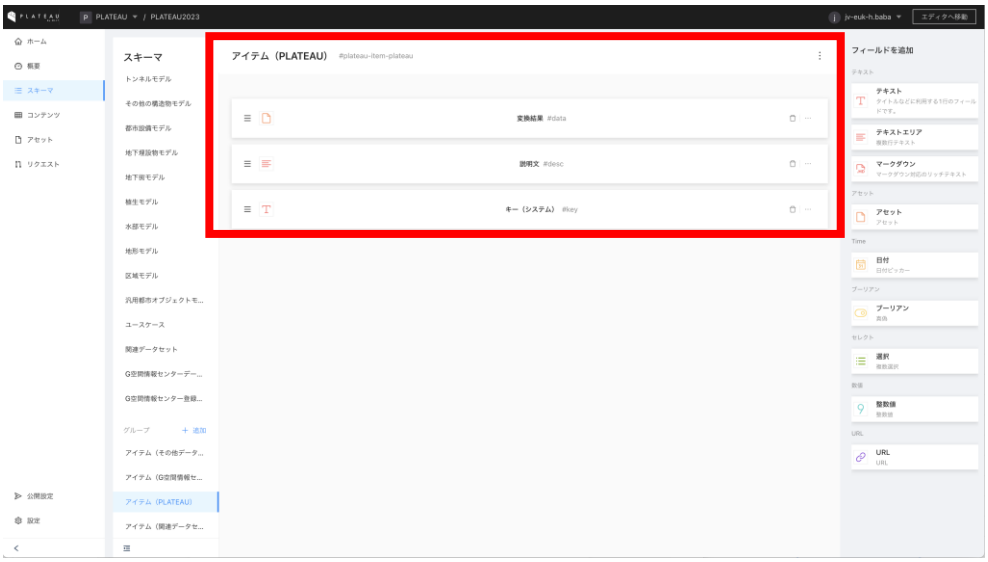
参照フィールドとは、Excelの参照機能のようなもので、モデルを跨いでひも付けをすることができる機能である。例えば、「都市」モデルと「建築物モデル」モデルをひも付けることで、各地物と都市をまとめている。参照フィールドは、以下のように参照先のモデルをスキーマ設定で選択することで設定可能。



グループフィールド

グループフィールドとは、複数のフィールドを任意の組み合わせでまとめて1つのフィールドとして設定する機能である。データをCMSへ登録する際に、データ本体（アセットフィールド）とそれに付随する説明文（テキストエリアフィールド）を1つの組み合わせとして登録したいケースなどで利用される。CMSでは、洪水浸水想定区域モデルなどにおいて、河川のデータ本体と、各河川の説明文を登録するために利用している。

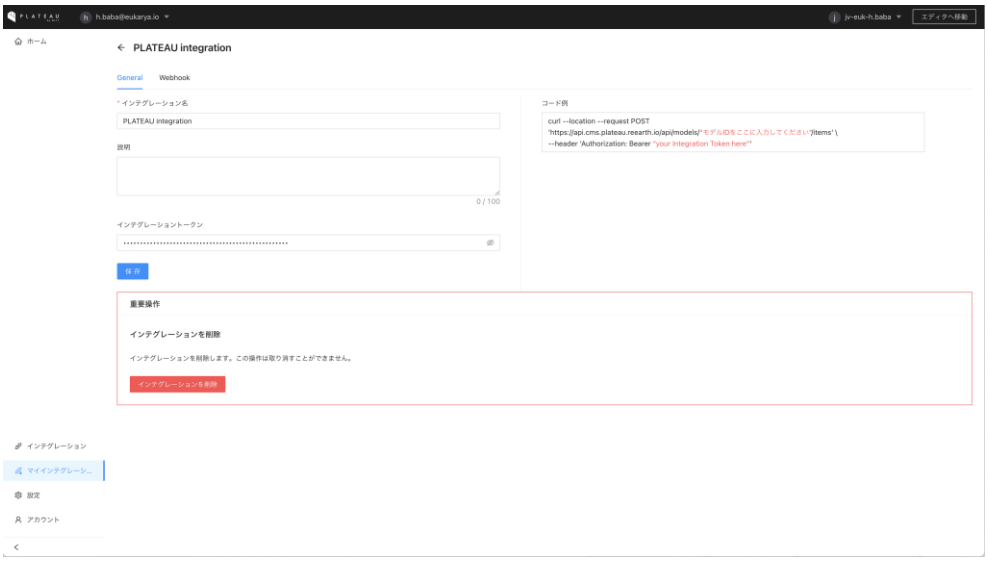
以下の例では、グループフィールドとして「変換結果（アセットフィールド）」、「説明文（テキストエリアフィールド）」、「キー（システム）（テキストフィールド）」の3つのフィールドを1つのグループとして設定している。下の画像は、これらを合わせたグループフィールドにおけるデータ入力画面の例である。



（4）インテグレーション機能

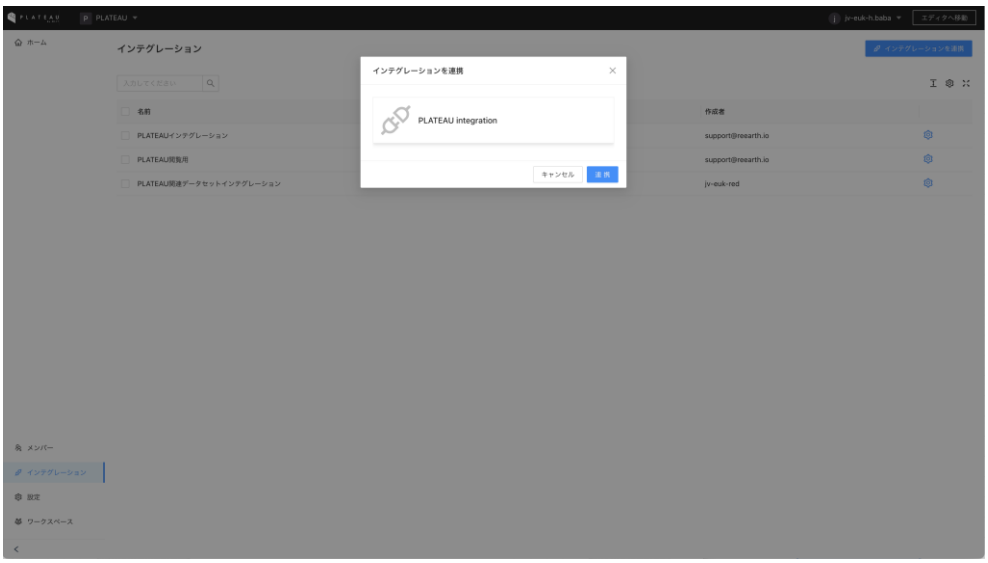
インテグレーションの作成

インテグレーション機能は、CMSが外部サーバーと連携するための仕組みである。同様の事例としては、SlackのApp機能がある。インテグレーションは、管理者が作成と連携をすることができ、APIキーを利用して、通常ユーザーと同様にアイテムの追加、編集、削除等を行うことができる。
PLATEAU CMSでは、このインテグレーション機能を利用して、CMSとFMEの連携を行っている。インテグレーションは個人アカウントにひも付き、そのインテグレーションを利用したいワークスペースへ招待することで利用が可能になる。



インテグレーションの連携

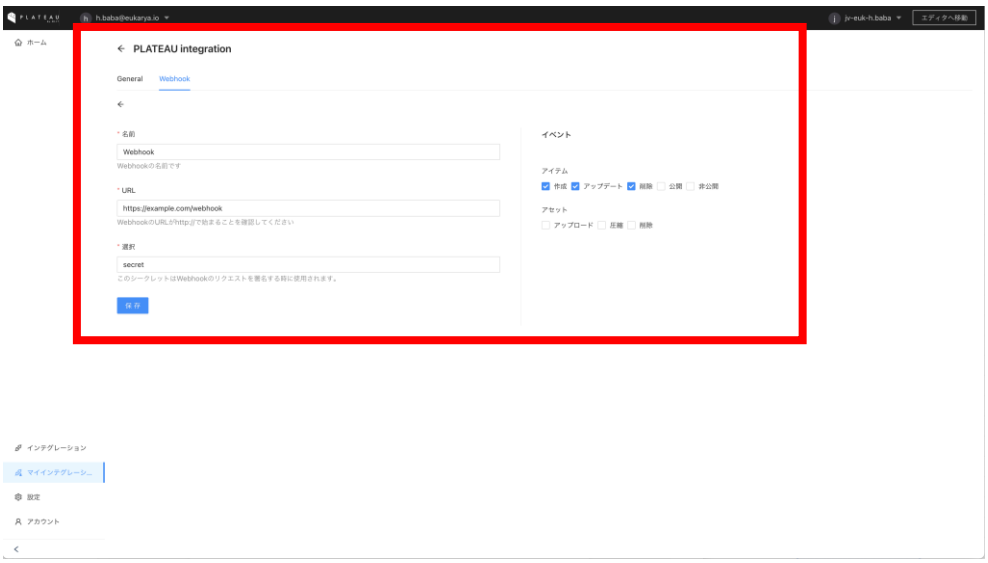
インテグレーションを作成すると、管理者は自身が所属しているワークスペースにインテグレーションを連携することができる。インテグレーションは、通常ユーザーと同様に見なすことができ、通常ユーザーと同様にロールが存在する。適切なロールを割り当てることで、CMSで管理されているデータへの操作をインテグレーションが行うことができる。



（５）Webhook機能

Webhookの設定

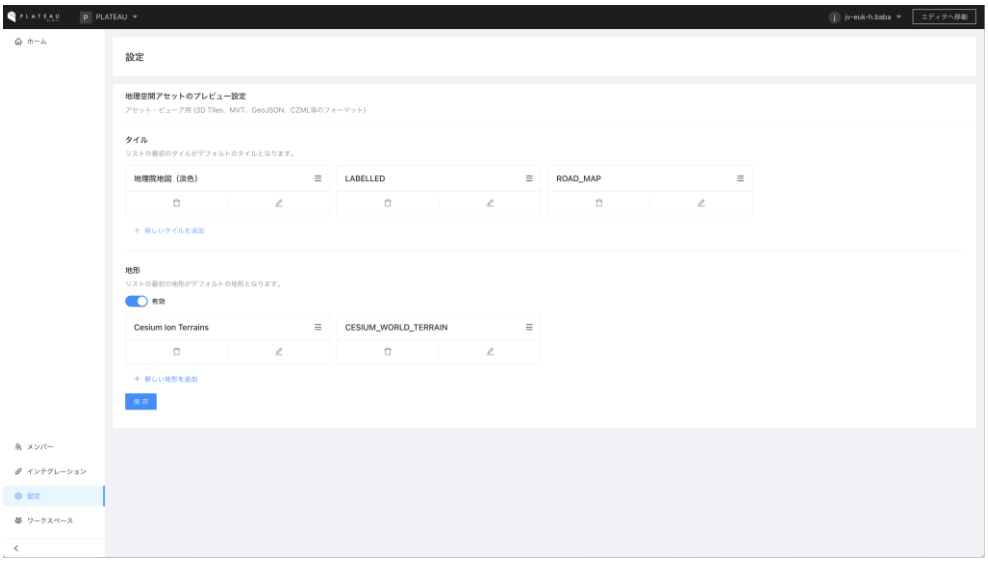
Webhookとは、Webアプリケーションにおいてユーザー定義のHTTPコールバックを設定できる機能である。例えば、「アイテムが新規に登録された時」、「アセットがアップロードされた時」など任意のイベント発生時に、そのイベントを外部アプリケーションへ通知することができる仕組みである。CMSでは、アイテムが更新された際にWebhookをサイドカーサーバーへ通知し、その内容をFMEへ連携することで、3D都市モデルの品質検査やデータ変換を行っている。
以下のように、通知したいイベントと通知先サーバーのエンドポイントを設定する。



（6）アセットプレビュー設定機能

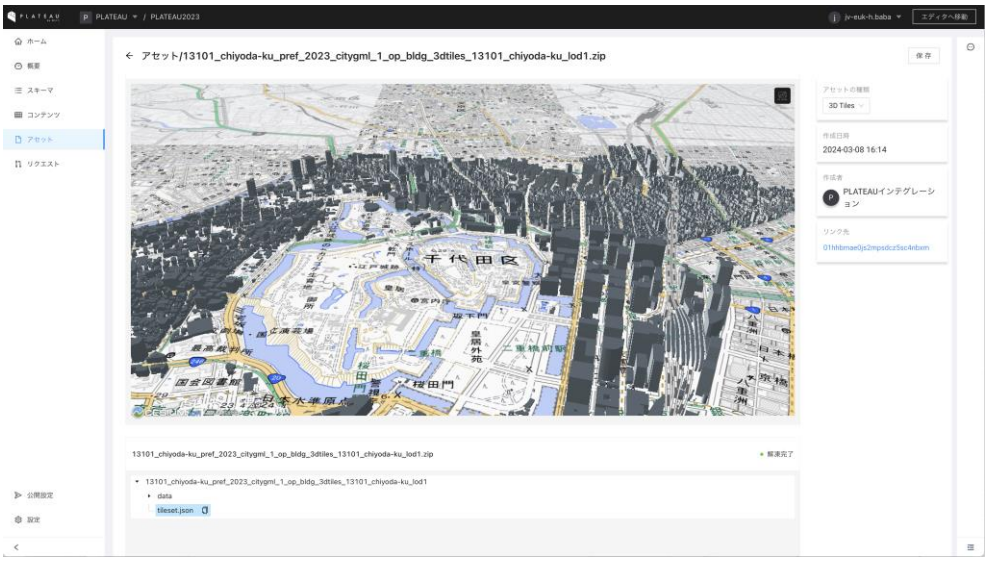
アセットプレビュー設定例

CMSではアセットとしてアップロードされたGISデータの簡易的なプレビュー機能を提供している。管理者はアセットプレビュー設定機能を通して、プレビューで利用する背景地図や地形データの設定を変更することができる。以下の例では、地理院地図（単色）やPLATEAU Terrainを利用するための設定をしている。



アセットプレビュー設定結果

上記のように設定をすると、以下のようにアセットのプレビューをすることができる。地理院地図（単色）を利用することで、閲覧中のエリアなどがわかりやすくなっている。



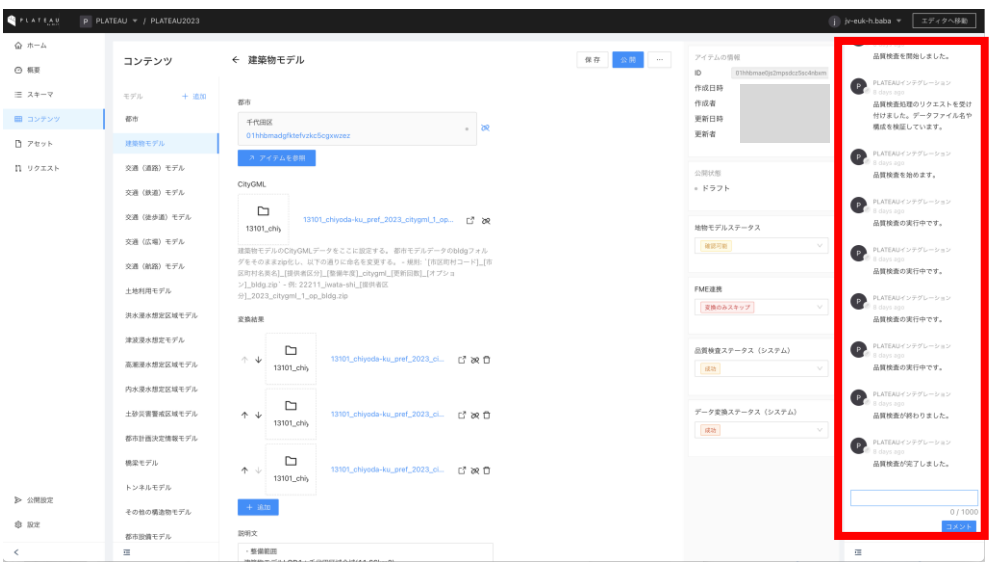
2.1.2 データ登録者向け機能

（1）データ登録機能

3.1.1で解説したように、データ登録者はスキーマに沿ってデータの登録を行うことができる。定義されたスキーマに沿ってデータを登録後、保存をする。

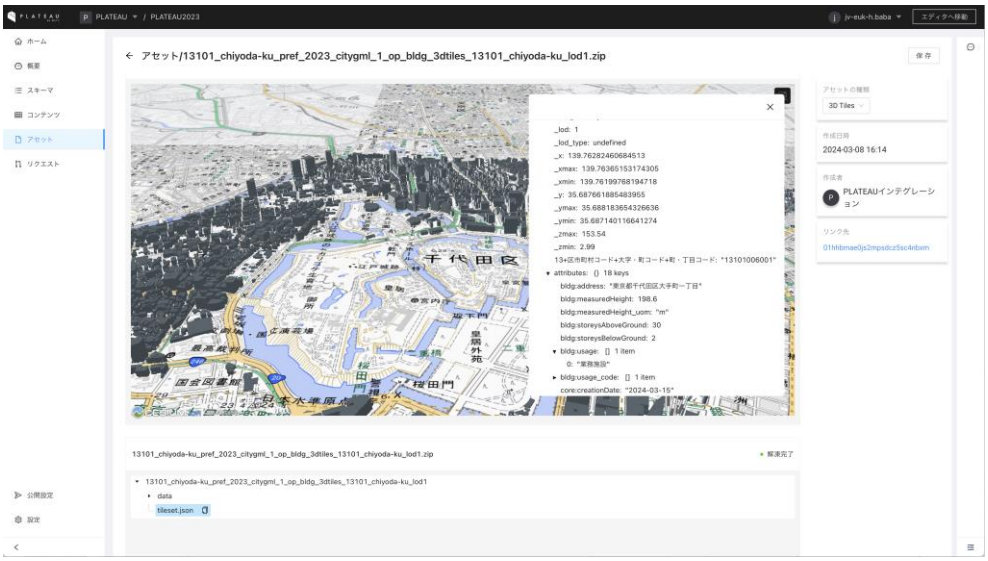
（2）コメント機能

作成したアイテムにはコメントを残すことができる。また、FMEとの連携におけるシステム側からのログもこのコメントに投稿がされる。



（3）アセットプレビュー機能

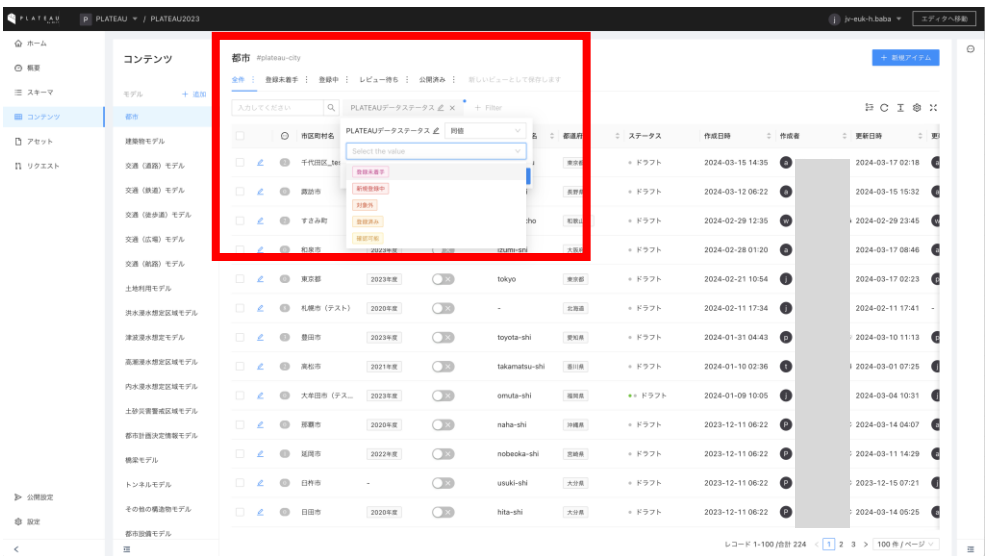
アセットプレビュー機能を利用することで、CMS上でGISデータの簡単な確認をすることができる。地物を押下すると、その地物の属性が表示される。ここに表示される属性は、日本語訳などされていない変換後データが持っている属性である。FMEによるデータ変換が適切に行われているかなどの確認に利用することができる。



（4）ビュー設定機能

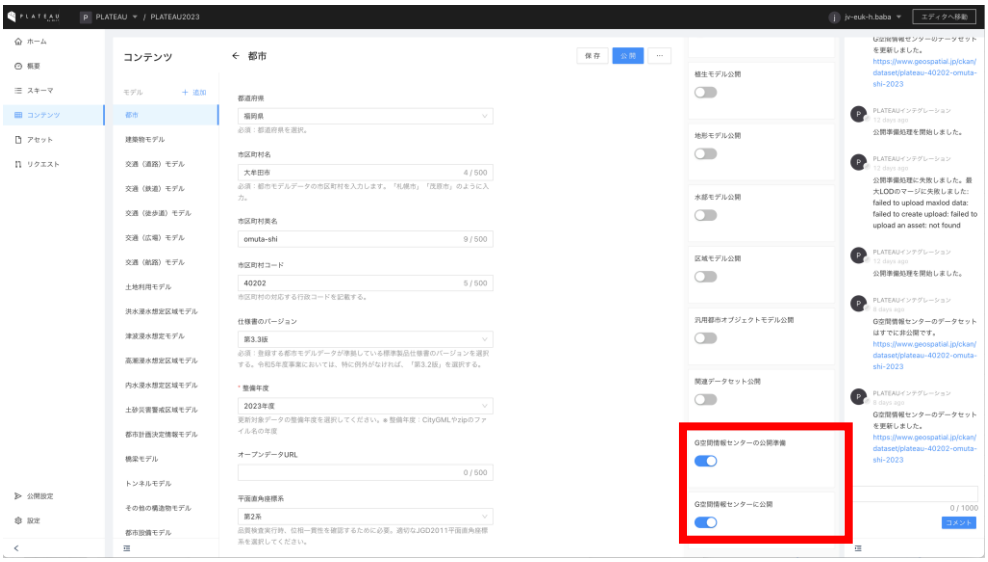
ビュー設定機能とは、アイテムの一覧表示方法をカスタマイズする機能である。類似例はExcelのカスタムビュー機能である。CMSでは、デフォルトでスキーマに定義された順でフィールドを表示する。ビュー機能を利用し、アイテムのフィルター条件や並べ替え条件を設定することで、表示条件を保存し、次回以降の利用時にも同様の表示をすることができる。カスタマイズできる内容は以下のとおり。

- フィルター：アイテムのフィルター条件を設定
- ソート：アイテムの並び変え条件を設定
- カラム：表示するカラムや順序を設定

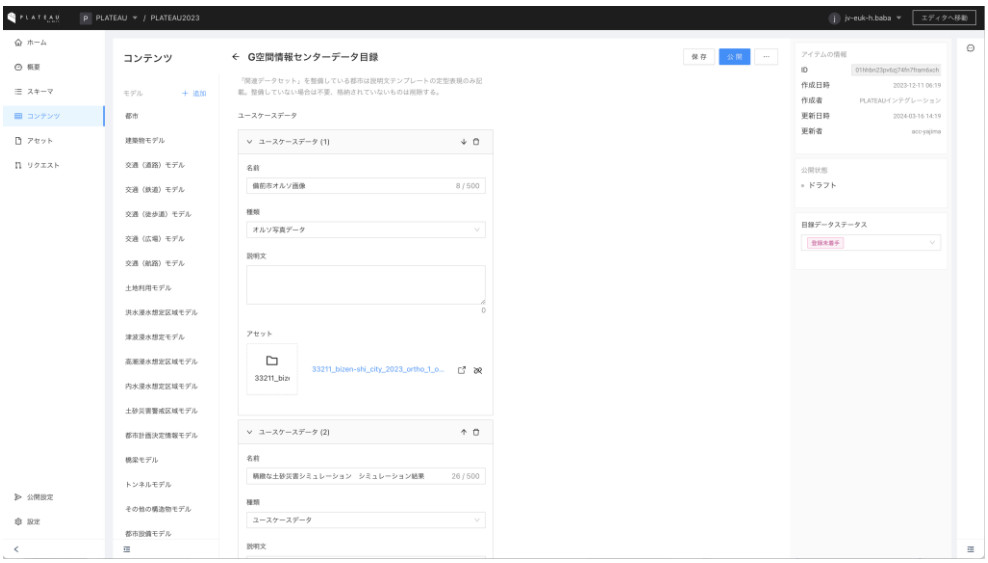


（５）G空間情報センター連携機能

CMSはG空間情報センターとの連携機能を有する。G空間情報センターへの公開設定は「都市」モデルから行う。「G空間情報センターの公開準備」、「G空間情報センターへ公開」をONにすることで、公開処理が実行される。 「G空間情報センターの公開準備」をONにすると、対象都市にひも付けされた地物型ごとのデータをCMSがまとめ、1つのZipファイルを生成する。合計ファイルサイズが大きい場合には、時間がかかることもある。（60GB程度のZipファイルの結合には約4時間程度必要）公開準備が完了し、 「G空間情報センターへ公開」をONにすることで、「G空間情報センターデータ目録」モデルに登録されたテキスト情報と合わせてG空間情報センターへの公開が行われる。なお、デフォルトではG空間情報センターへはプライベートページとして公開されるため、一般公開するにはG空間情報センターで公開設定を変更する必要がある。



また、ユースケースデータやオルソ画像など追加のデータをG空間情報センターへ登録したい場合には、「G空間情報センターデータ目録」モデルにアセットとして登録する。



2.2 PLATEAU Editorの機能

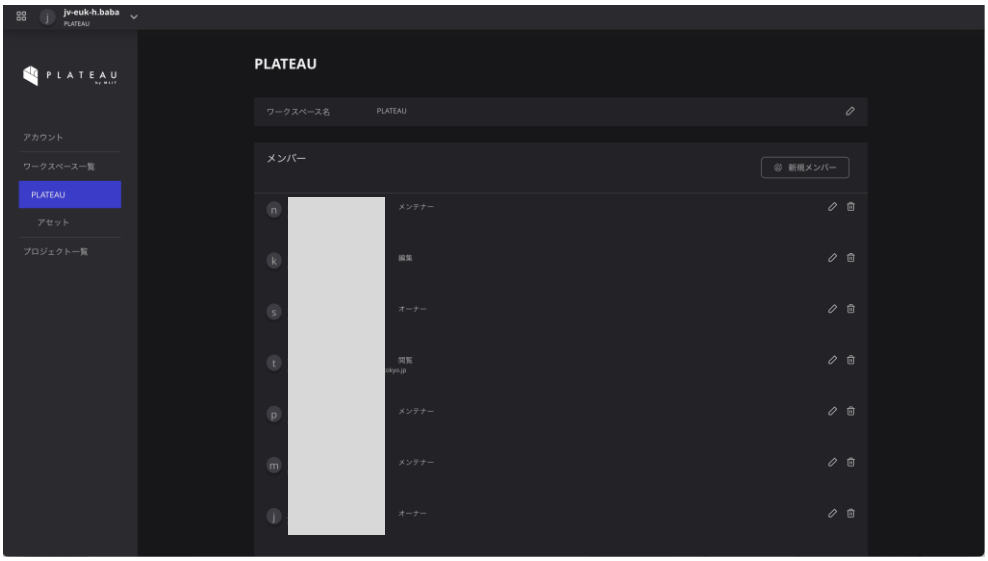
ここではPLATEAU Editorの使用方法について述べながら、PLATEAU VIEW 3.0におけるPLATEAU VIEWをPLATEAU Editor上で作成・公開する手順を解説する。なお、PLATEAU Editorで採用されているOSSであるRe:Earthは非常に多くの機能を持っているが、ここではPLATEAU VIEWを構築するために関係する項目のみを説明する。それ以外のRe:Earthの全ての機能をここで紹介することはできないため、必要に応じて以下のURLも併せて参照されたい。

URL:https://docs.reearth.io/ja/

2.2.1 管理者向け機能

（1）アカウント管理機能

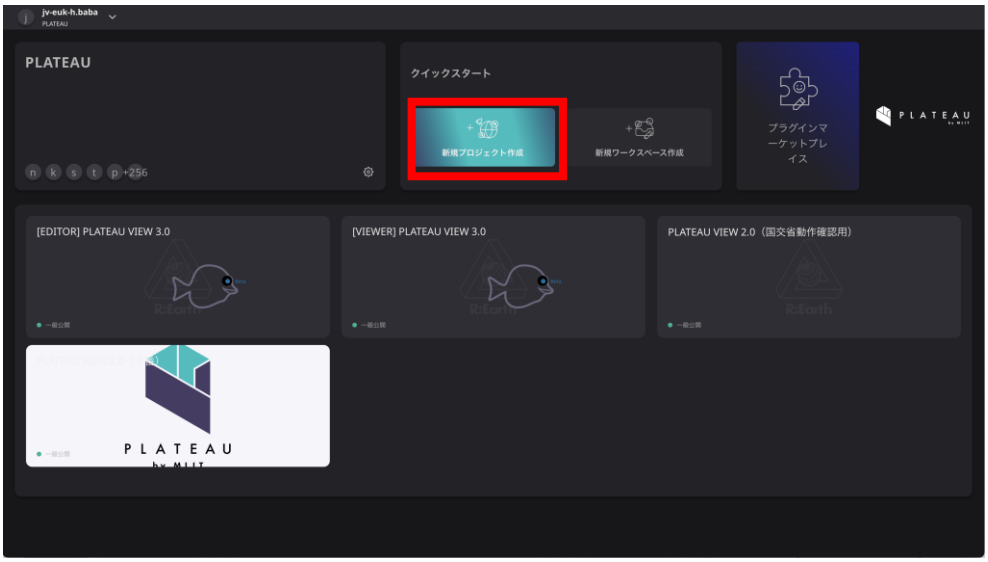
Editorのワークスペースは複数のユーザーが共同でプロジェクトを管理することができる。なお、EditorとCMSではユーザーアカウント及びワークスペースは同じデータベースを参照しており、両者のデータは同期されている。アカウント管理機能の全体像はCMSと同じである。以下のページからワークスペース内のメンバーの権限変更、削除等が可能である。



（2）プロジェクトの編集機能

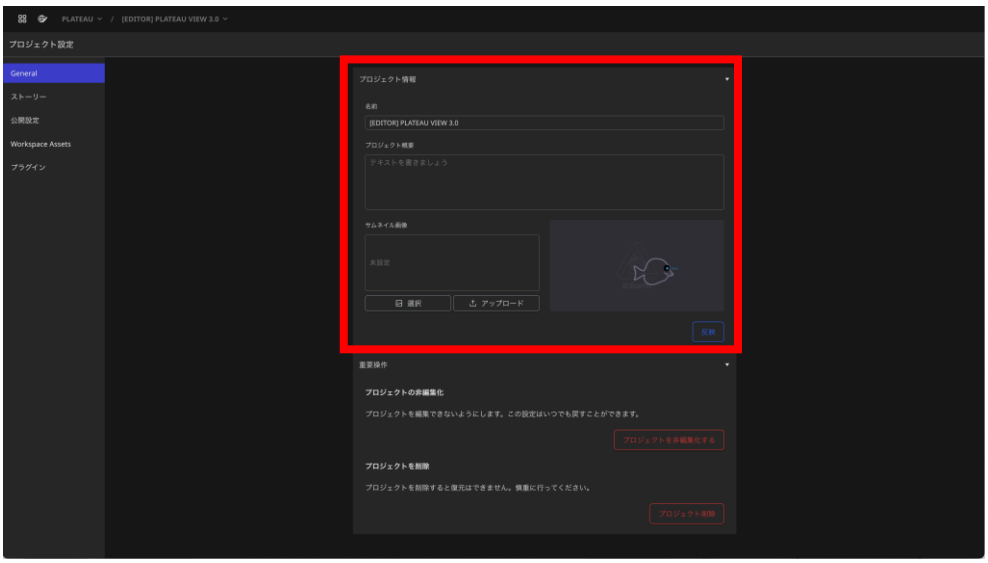
プロジェクト作成機能

Editorでは、CMSと同様にプロジェクトという単位で可視化設定を行っている。プロジェクトは、任意の目的に応じて管理者が作成・管理することができるものである。以下の画面では、プロジェクトの一覧表示や新規作成を行うことができる。

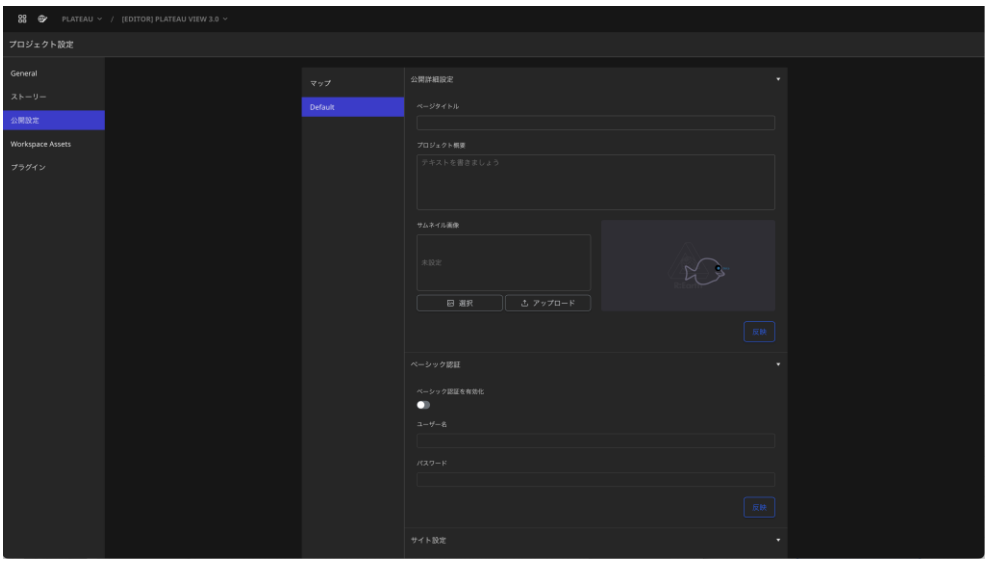


プロジェクト設定変更機能

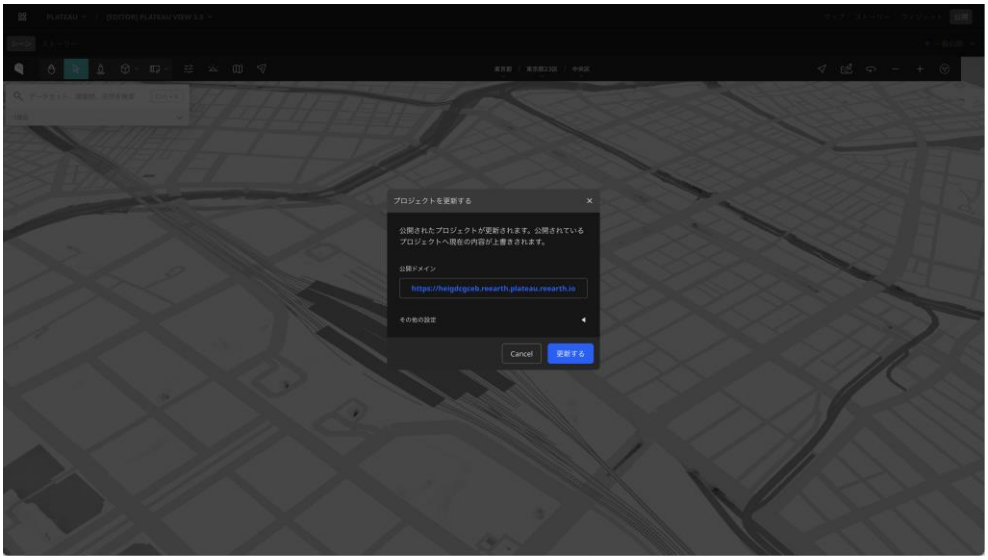
プロジェクトの設定ページでは、プロジェクト名や説明、サムネイルの変更ができる。



プロジェクトの公開設定機能1
プロジェクトの公開設定では、公開後のプロジェクトのページタイトルやベーシック認証の設定等ができる。

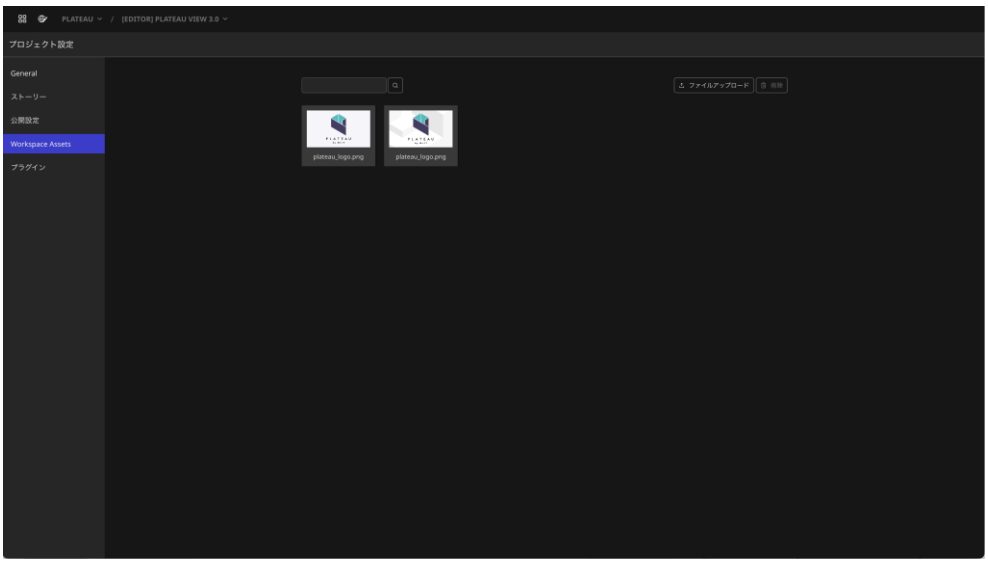


プロジェクトの公開設定機能2
プロジェクトにおいて可視化設定を完了し、公開準備が整ったら、プロジェクト編集ページにから一般公開することができる。一般公開をすると、専用のURLが発行され、URLを通して誰もがプロジェクトの閲覧できるようになる。



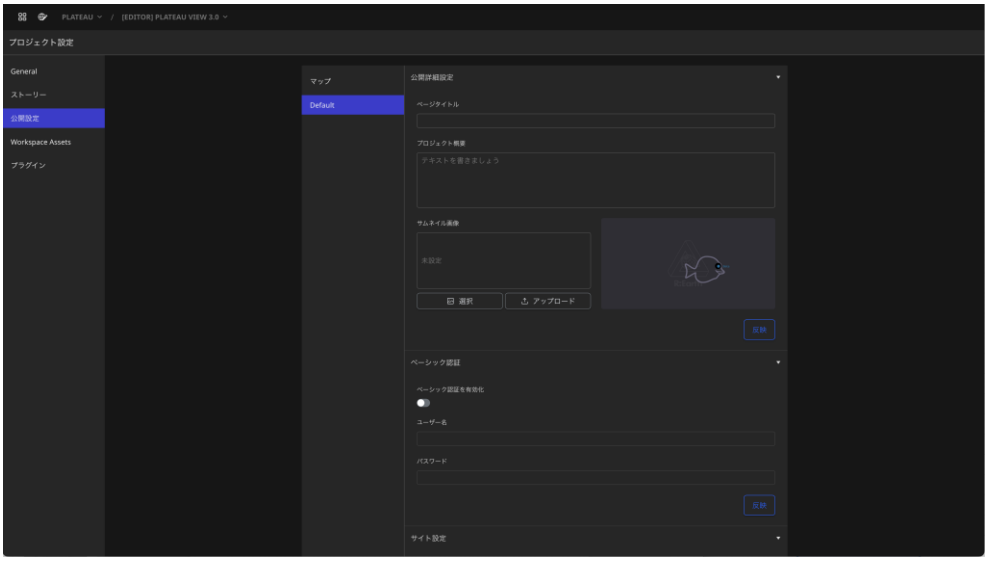
アセット管理機能

アセット管理機能では、Editorにアップロードされたアセットの管理ができる。PLATEAU VIEW 3.0では、3D都市モデルデータをCMSから直接配信しているため、Editorでアセットをアップロードすることは少ないが、以下のようにサムネイル画像などの管理をすることができる。



プラグイン管理機能

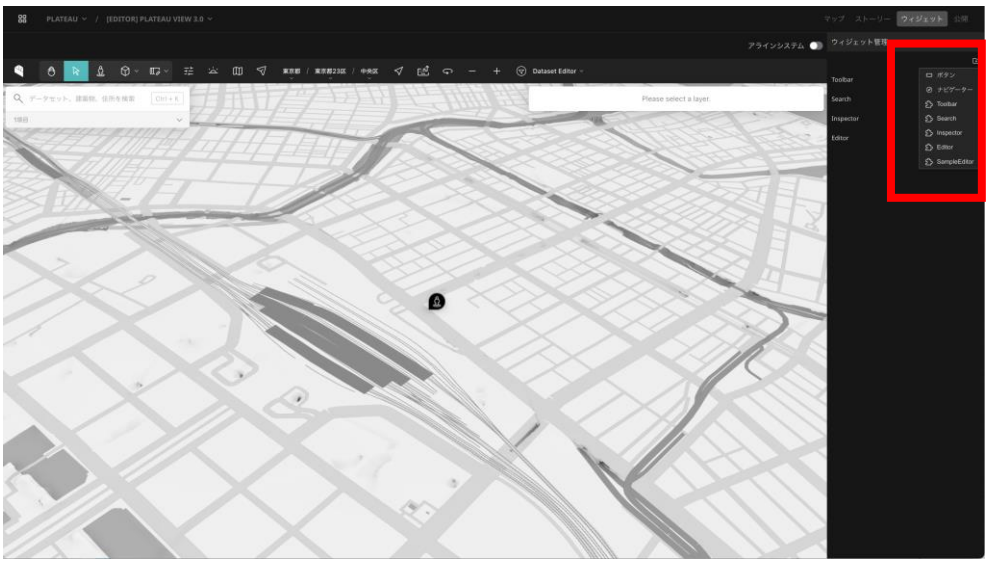
プロジェクトのプラグイン設定では、プロジェクトにインストール済みのプラグインの管理や新規インストールが可能である。プラグインを利用することで、Editor上でデフォルトには存在しない拡張的な機能を利用することができる。利用可能なプラグインの一覧は、Re:Earth Marketplaceで公開されている。



（3）ウィジェット機能

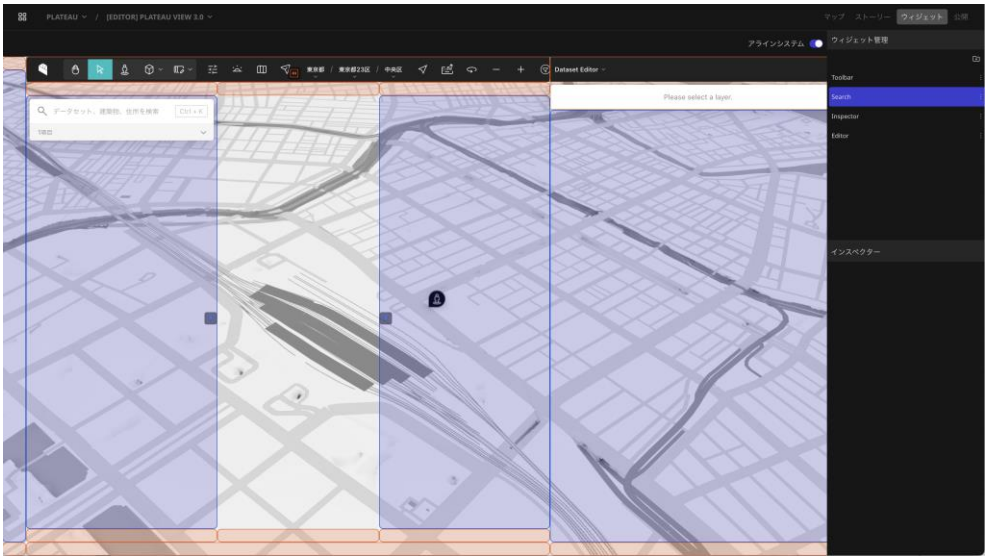
ウィジェット追加機能

Editorではウィジェット（デジタル地球儀とは画面に表示されるUI）の追加や変更をすることができる。



ウィジェット配置変更機能

また、追加したウィジェットの配置を変更することもできる。これにより、UIの自由なカスタマイズが可能である。

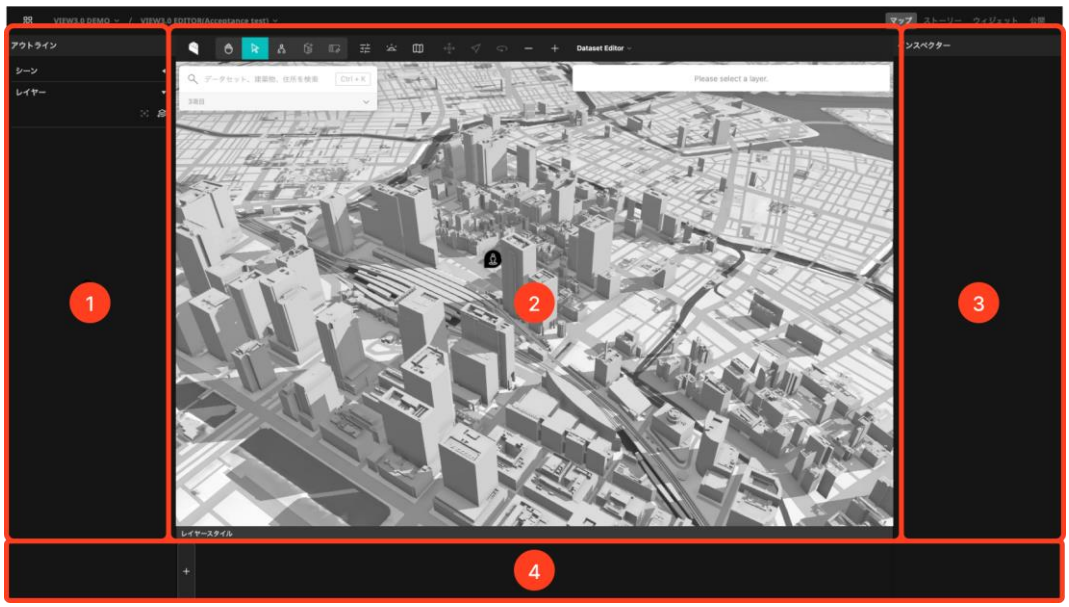


2.2.2 データ登録者向け機能

まずEditorーのUIの全体感について説明をする。

1. 左サイドバー
2. 地図Viewer
3. 右サイドバー
4. 下部バー

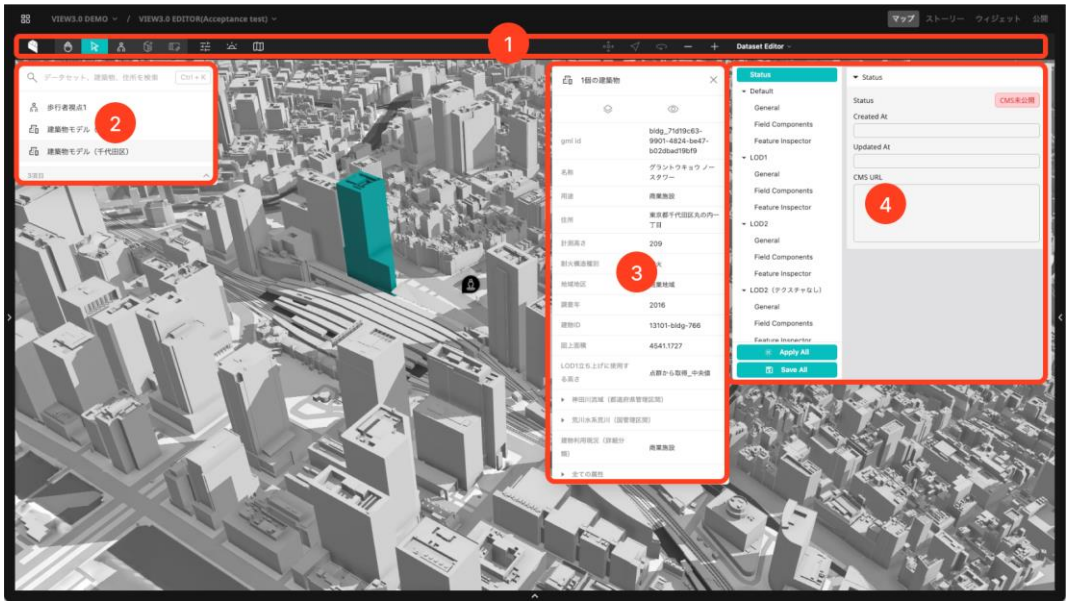
で構成されている。主に2の地図Viewerーを頻繁に利用するため、必要に応じて1、3、4については縮小表示することを推奨する。



各バーの境界をドラッグアンドドロップすることで縮小表示が可能。

地図ViewerにおけるUI

地図Viewerでは、地図上へのデータの追加と、対象データへの色分けや凡例設定等を行うことができる。まずは、地図ViewerにおけるUIのレイアウトを解説する。



（1）ツールバー

データを閲覧する際のモードの切り替えや、 歩行者モードの利用、作図機能の利用、ベースマップの切り替え等を行うことができる。Editorの設定では基本的には、

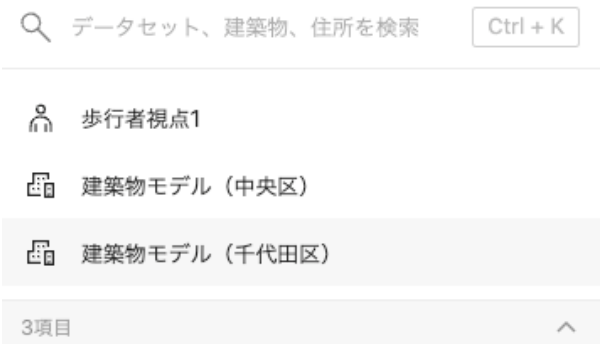
1. 移動モード
2. 地物選択モード

の2つのみを利用する。移動モードでは、地図をマウス操作しながら移動することができる。一方で、地物選択モードでは、押下して地物を選択し、地物の属性情報を閲覧することなどが可能。（うまく地物が選択できない場合や、地図を移動できない場合は現在のモードを確認すること）

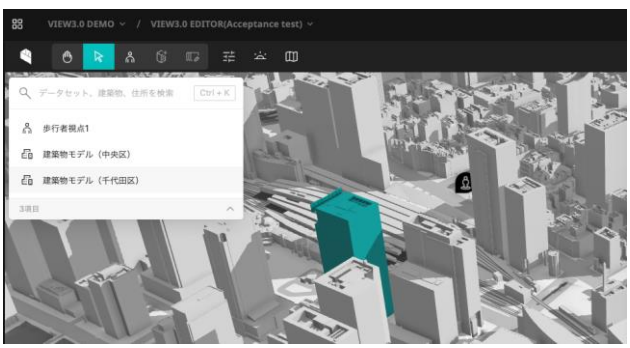
（2）ヒエラルキーウィンドウ

ヒエラルキーウィンドウでは、地図上に追加するデータを選択することができる。 3つのタブから構成され、検索、都道府県一覧から選択、カテゴリーから選択することができる。 Editorでのデータ確認時には、都道府県一覧から選択することをお勧めする.

マウスのフォーカスを外すと 項目を選択できるタブが存在し、そこを押下すると、現在追加されているデーター一覧が表示される。

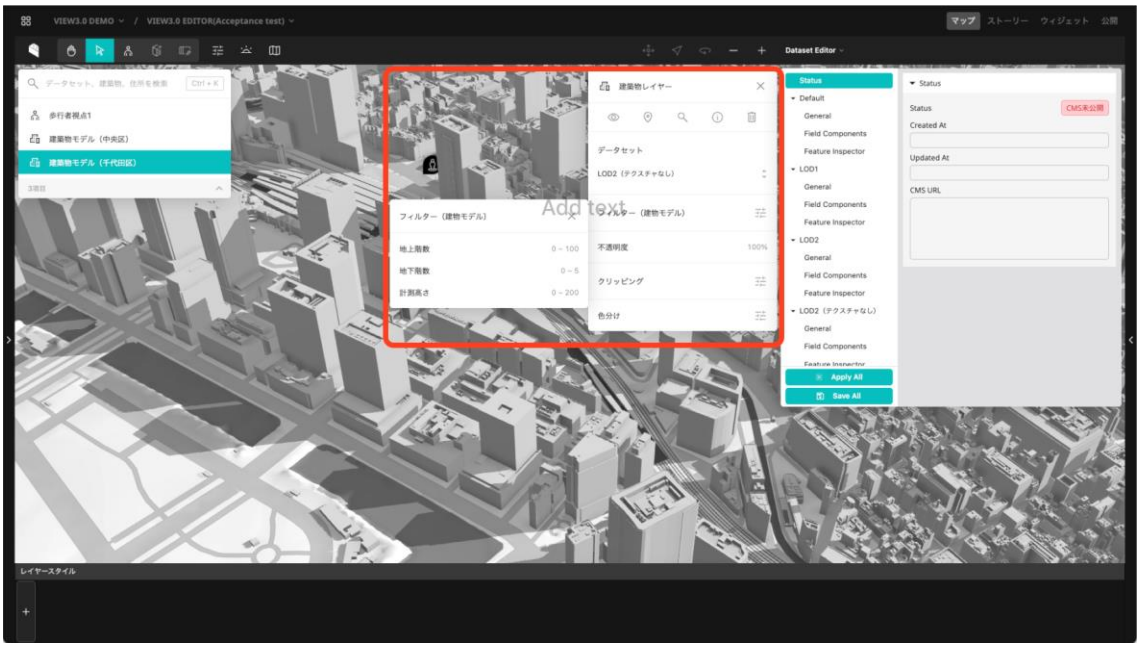


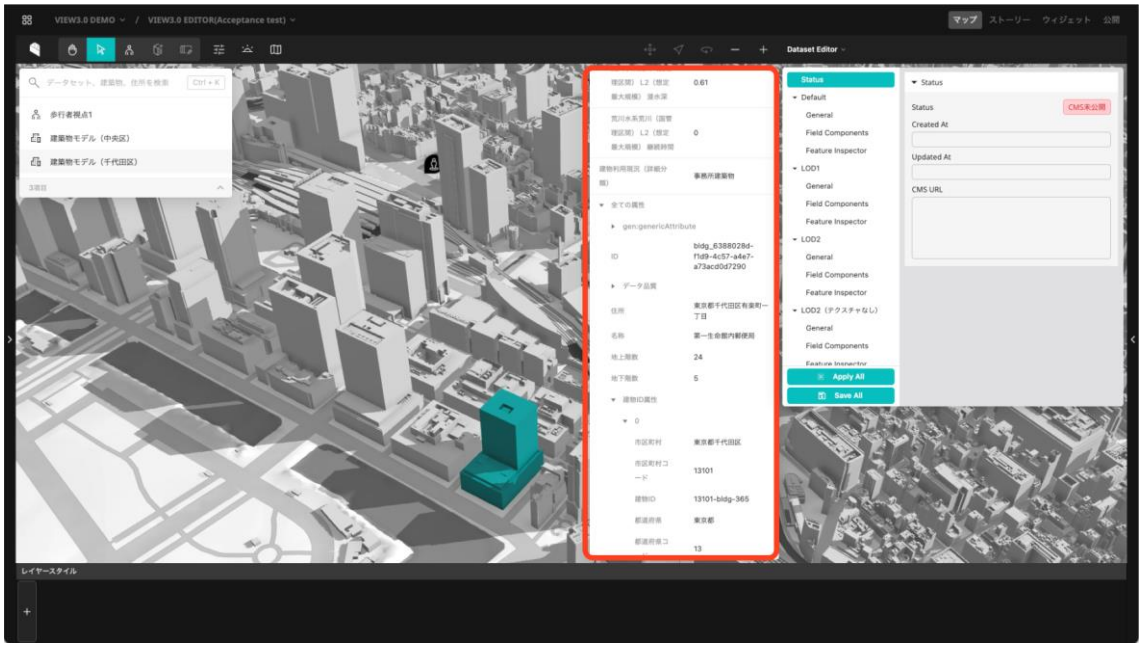
地図上における選択状態には2種類ある。レイヤーの選択と地物の選択である。 レイヤーを選択するには、データー一覧からレイヤーを押下する。 レイヤーの選択状態と地物の選択状態は、後述する。インスペクターの使い方に関わるので注意すること。



（3）インスペクター

上述したように、地図上における選択状態には、レイヤーと地物の選択状態が存在する。インスペクターは、レイヤー選択時には、レイヤーに関する情報を、 地物の選択時には、地物の属性情報を表示する。レイヤーの選択状態のインスペクターからは、LODの 色分け、クリッピング等の機能をすることができる。 一方で、地物の選択状態では 属性情報 閲覧することができる。 データが正しいか確認するときには、この地物の情報をよく確認すること。



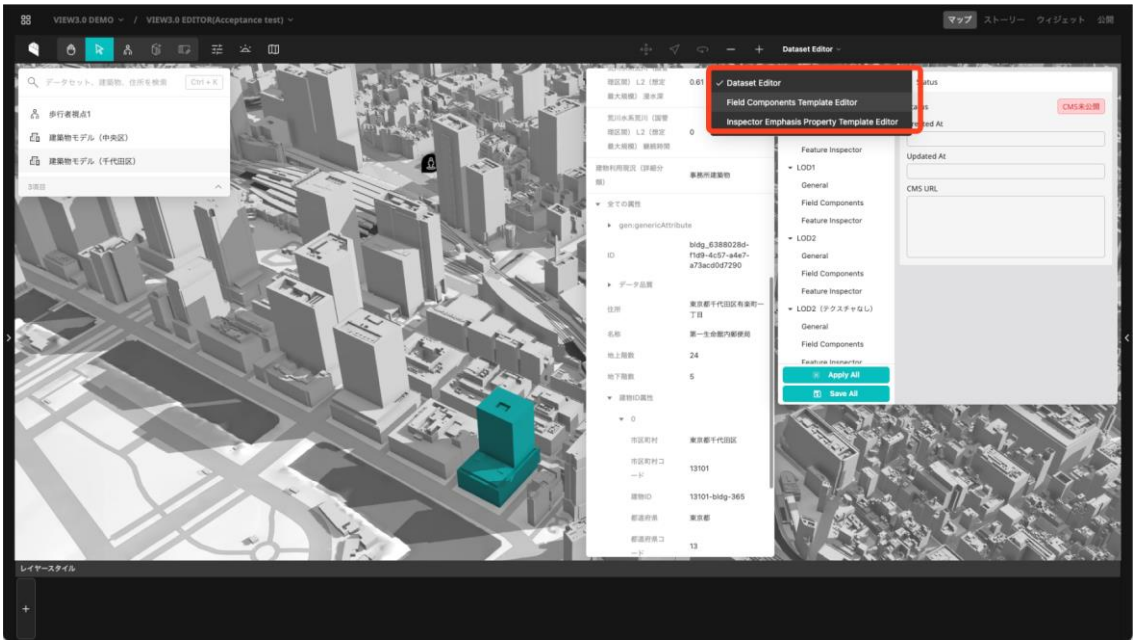


（4）データ編集パネル

コンポーネント設定パネルは、主に3つの設定ものが存在し、それぞれの設定の切り替えは画面スクショのタブから変更可能である。

1. コンポーネントEditor
2. テンプレートEditor
3. インスペクターEditor

以下にそれぞれの詳細を説明する。



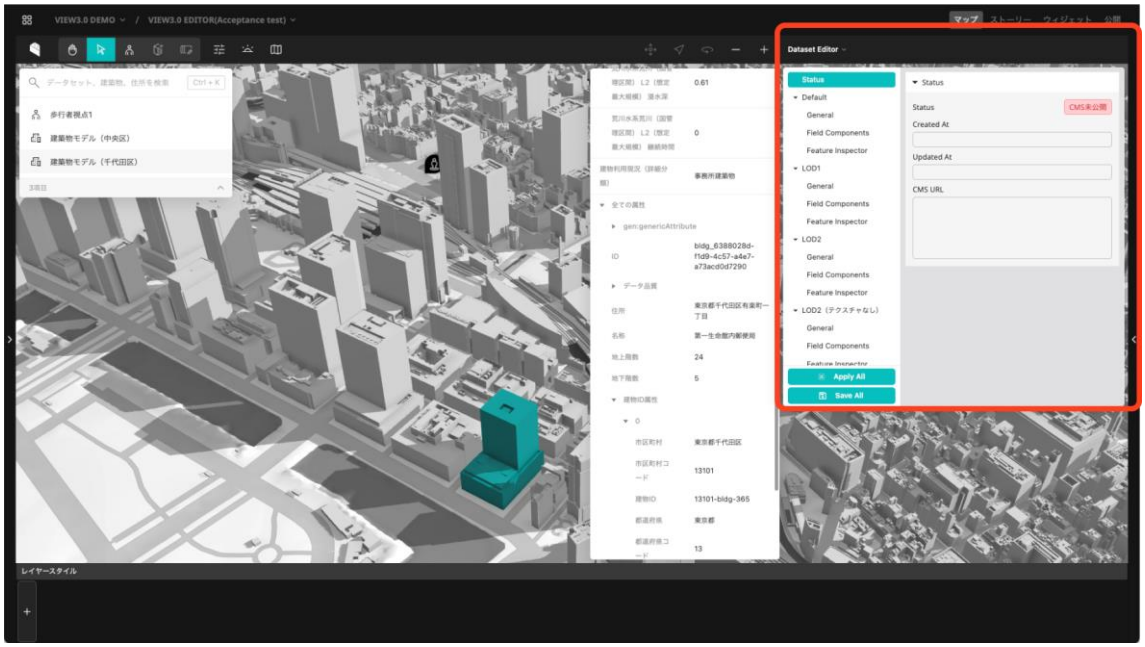
1.　コンポーネントEditor

Editorでは、データカタログから追加可能なそれぞれのデータセットに対し、UI操作によって凡例の表示や地図上のデータのスタイルを細かく設定することができる。それを可能にしているのが「コンポーネント機能」と「テンプレート機能」である。

コンポーネントとは？

コンポーネントとは、データセットに対し「凡例の表示」「スタイルの変更」など、Viewerにおける追加の機能を付加する仕組みである。データカタログで追加可能なそれぞれのデータセットに対し、複数のコンポーネントをひも付けることができる。
それぞれのコンポーネントは設定項目を持ち、VIEWではその設定項目を編集することはできないが、EditorではコンポーネントEditorで編集することができる。

　コンポーネントEditorは 最も多く使われる機能の1つであり、 凡例の表示は、地物の色分けの設定を行う。 設定方法は多岐にわたるため、本節では概要にとどめ、詳細な設定方法に関しては、 3.2.4 コンポーネントの種類と設定方法で解説をする。

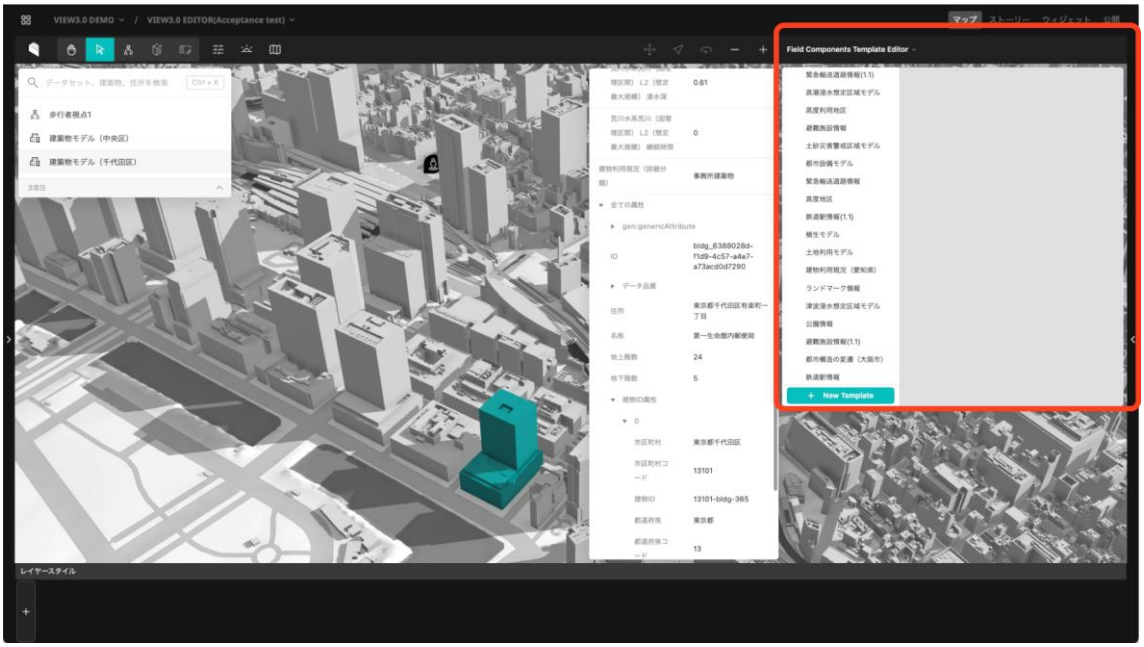


2.　テンプレートEditor

テンプレートEditorは、 利用可能なテンプレートの管理を行う場所である。

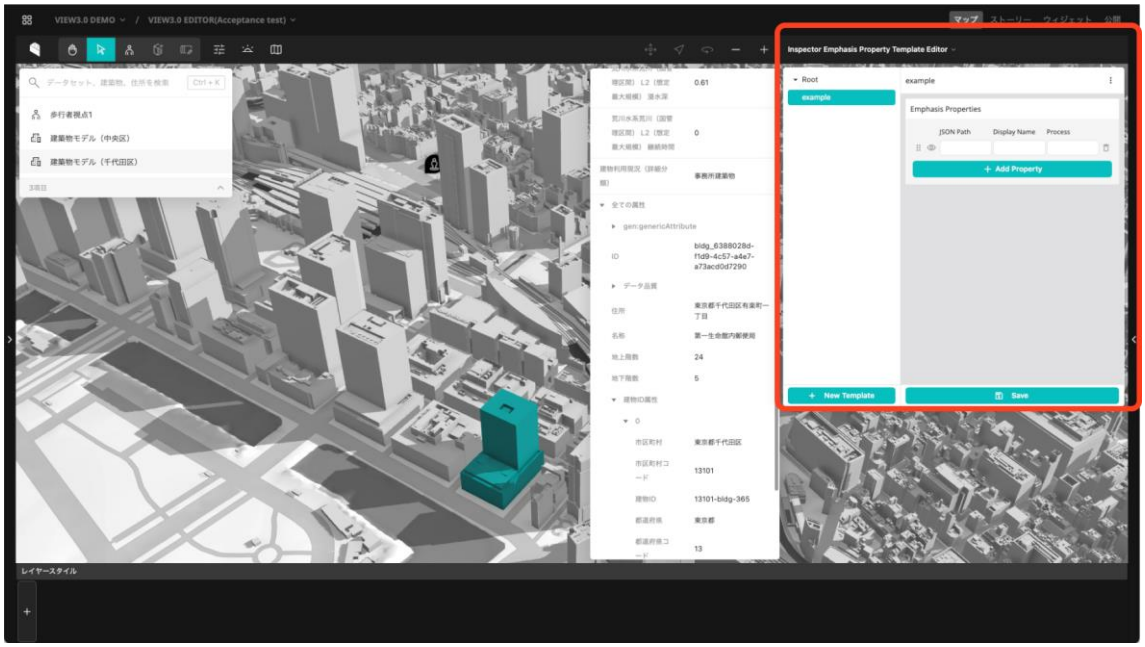
テンプレートとは？

　テンプレートとは、複数のデータセットに対し同じ設定のコンポーネントを一括で登録するための機能。データの種類に応じたテンプレートを作成することができ、データカタログからデータセットを追加すると、自動的にテンプレートで設定されたコンポーネントが反映される。
コンポーネントは複数の種類を同時に組み合わせることができ、どのような効果が発生するかはコンポーネントごとに異なる。例えばインスペクター上に凡例表示するための「凡例コンポーネント」は、地図上のデータのスタイルには何も影響を与えない。それに対し「ポイント>色コンポーネント」を使うと、サイドバーには特にUIは表示されないが、データの地図上での見た目を変更することができる。こうした異なる機能を持つコンポーネントを組み合わせることで、データセットに対しさまざまな表示のカスタマイズを行える仕組みとなっている



3. インスペクターEditor

インスペクターEditorとは、地物の属性表示方法をカスタマイズするための機能である。 データ整備事業者にとっては、利用機会が少ないが、 特定の属性を英名ではなく、日本語名で表示したい。また、特定の属性を表示にしたい等のケースにおいて利用できる主にユースケースデータでの利用を想定している。



2.2.3 コンポーネント設定方法

（1）コンポーネント設定の全体像

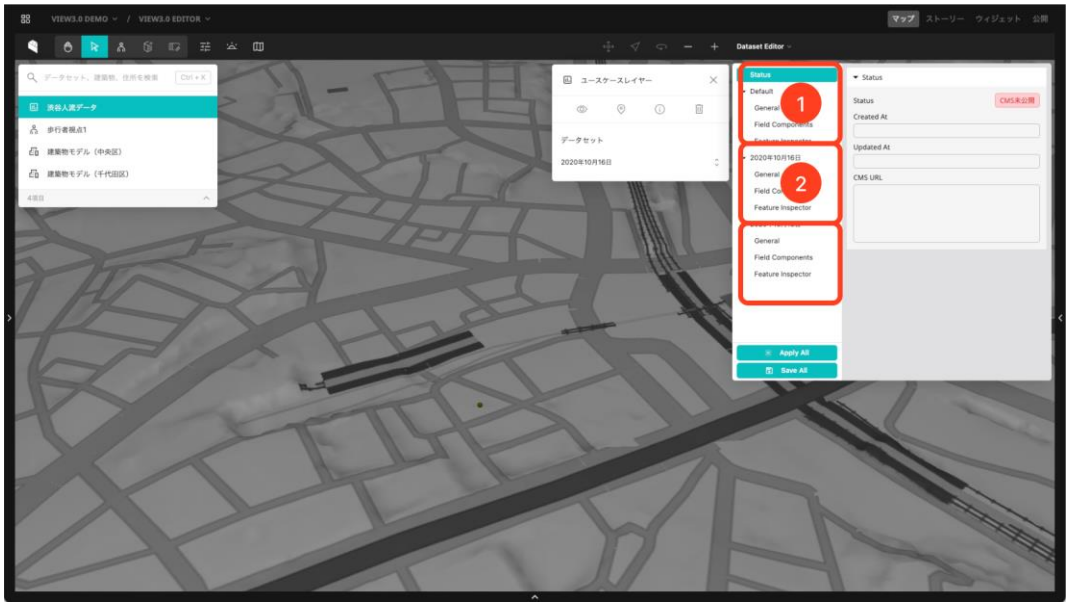
コンポーネントとは、データセットに対し「凡例の表示」「スタイルの変更」など、Viewerにおける追加の機能を付加する仕組みである。データカタログで追加可能なそれぞれのデータセットに対し、複数のコンポーネントをひも付けることができる。
それぞれのコンポーネントは設定項目を持ち、Viewerではその設定項目を編集することはできないが、EditorではコンポーネントEditorで編集することができる。

（2）コンポーネント設定をするデータ

コンポーネントは、各データ（データカタログに表示される単位）ごとに、それぞれ異なるコンポーネントを設定する。コンポーネントを設定する箇所は、大きく2つ存在する。

1. デフォルト
2. 個別データ

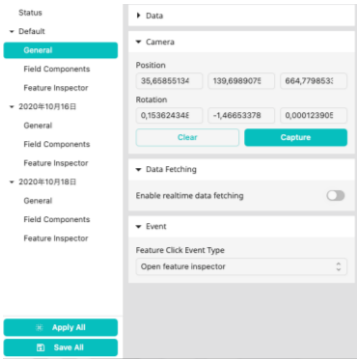
データカタログに表示されるデータは単一のファイルから構成されるデータもあれば、CMS上で複数のファイルを登録することで、複数ファイルから構成されるデータも存在する。（例：LODの切り替えなど）
原則としては、**「1.デフォルト」**でコンポーネント設定を行う。（多くのケースで「2.個別データ」への設定は不要）「1.デフォルト」で設定したコンポーネントの設定は、そのデータを構成する複数ファイル全てに適用される。一部、データを切り替えた際に適用するコンポーネントを変えたい場合のみ、対象のデータに対してコンポーネントを設定する。

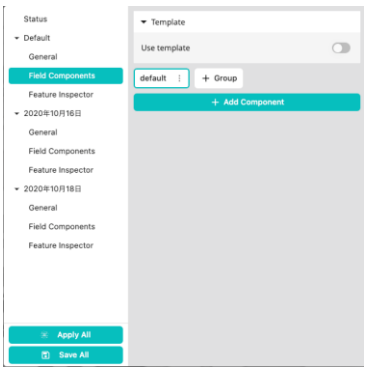


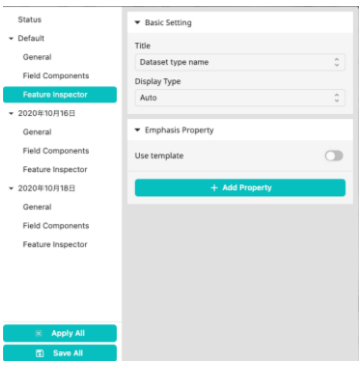
（3）コンポーネント設定の種類

さらにコンポーネントの設定には、３種類存在する。

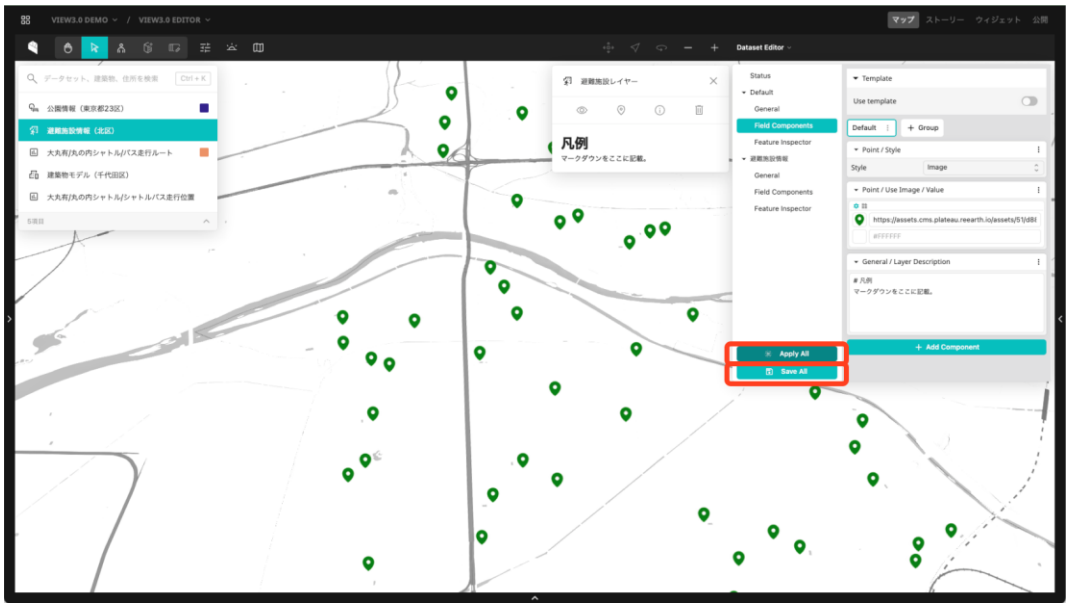
1. 一般設定: カメラの設定や、CMSから配信されるデータのJSON表記、地物押下時のイベントの設定などを行う。
2. フィールドコンポーネント: 地物の色分けや、凡例等の設定を行う。
3. 地物インスペクター: 地物の属性表示の設定を行う。







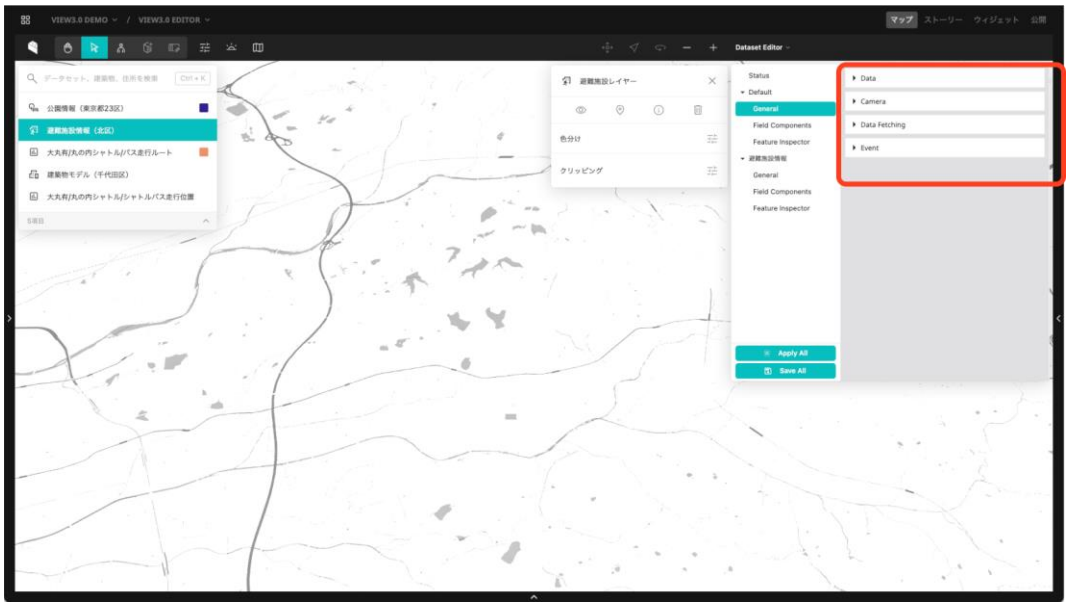
それぞれ設定が完了したら、「Apply All」を押下して設定を地図上の地物へ反映、「Save All」を押下して設定を保存する。（自動保存ではないため必ず保存を押すこと）



1.　一般設定

一般設定は、以下の4設定から構成される。

- データ
- カメラ設定
- データ取得設定
- イベント設定

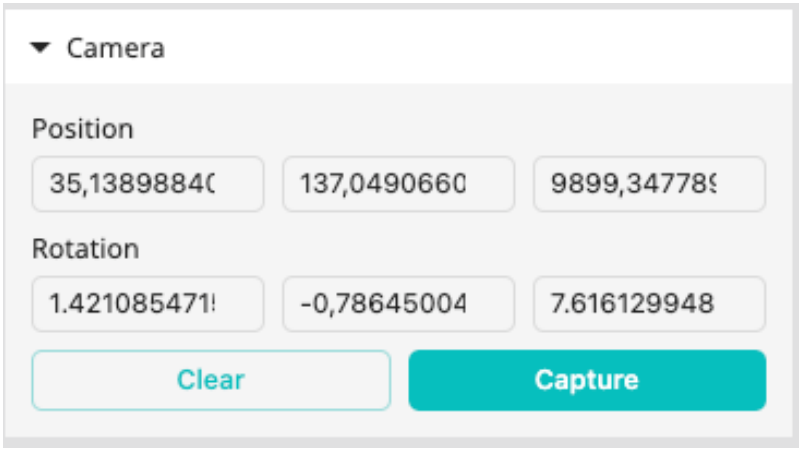


データ

CMSから配信されているデータの生データを確認することができる。主にデバッグ向け。

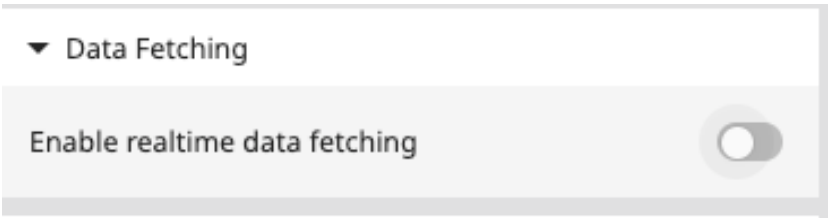
カメラ設定

「カメラ」ボタンで移動するカメラの位置を設定することができる。このコンポーネントがない場合は自動的にカメラ位置が計算される（MVTやWMSなどカメラ位置の自動計算が行われないデータフォーマットもある）。



データ取得設定

リアルタイムデータ取得を有効にする。GTFS Realtimeなどリアルタイムなデータ取得が必要な場合のみ有効にする。



イベント設定

地物押下時のイベントを設定する。

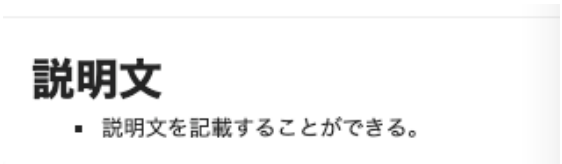
- Open feature inspector: 通常とおり、地物選択時に地物の属性情報表示をインスペクターから行う。
- Open new tab: 地物選択時、外部サイトへリンクしたい際に利用する。（利用頻度はあまり高くない）

2. フィールドコンポーネントの種類

イベント設定

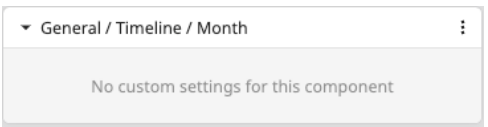
地物押下時のイベントを設定する。

- Open feature inspector: 通常とおり、地物選択時に地物の属性情報表示をインスペクターから行う。
- Open new tab: 地物選択時、外部サイトへリンクしたい際に利用する。（利用頻度はあまり高くない）



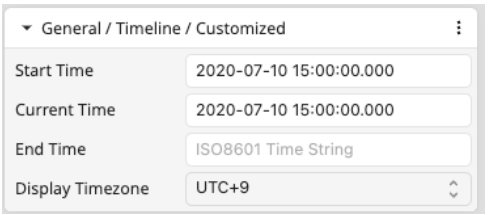
General > Timeline > Month

時系列データ表示用のタイムラインを表示する。1ヶ月前を開始日時、現在を終了日時として自動的に設定する。



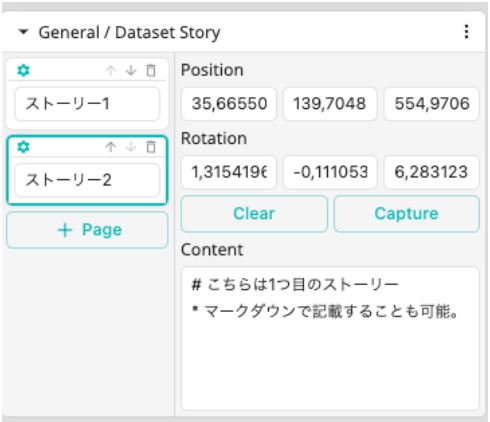
General > Timeline > Customised

時系列データ表示用のタイムラインを表示する。開始日時、現在日時、終了日時をISO8601形式で入力することで、再生バーの表示を行うことができる。



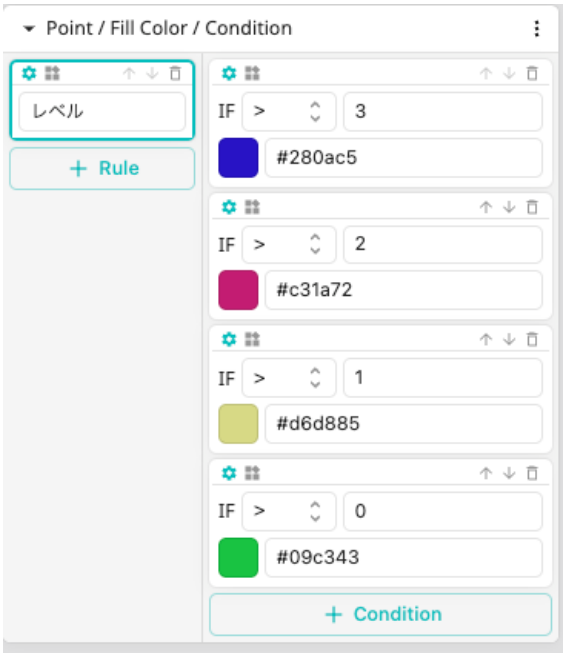
General > LinkButton
リンクボタンを表示する。タイトルとURLを設定することで、指定したリンクへ飛ぶリンクボタンを表示することができる。

General > Dataset Story
データセットにひも付くストーリーを作成可能。ストーリーは、複数のストーリーから構成され、それぞれのストーリーはカメラキャプチャとテキストを保持する。

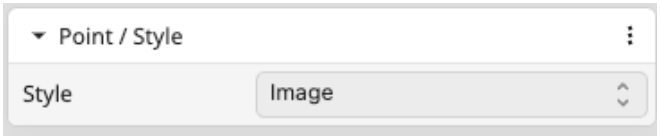


Point > Fill Color > Value
ポイント形式のデータにおいて、地物の色をカラーコードで設定する。（Line、Polygon形式のデータも同様のため省略）

Point > Fill Color > Condition
ポイント形式のデータにおいて、地物の特定の属性の値によって色を設定する。（Line、Polygon形式のデータも同様のため省略）

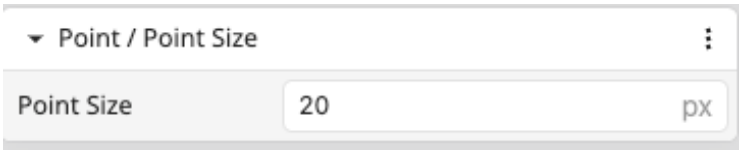


Point > StyleValue
ポイント形式のデータにおいて、地物の表示をポイントにするか画像にするかを選択する。データの表示においては必ず設定が必要。

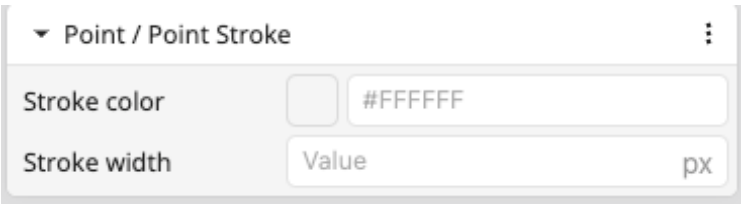


Point > Use Image > Value
ポイント形式のデータにおいて、地物表示に利用する画像のURLを指定する。

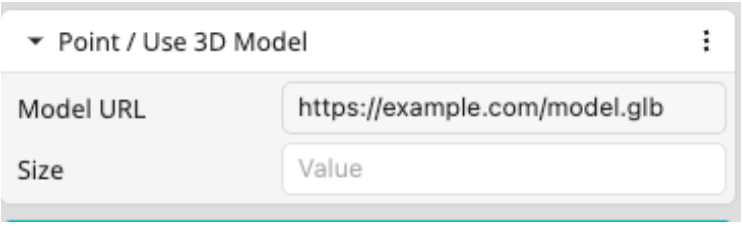
Point > Point Size
ポイント形式のデータにおいて、地物のサイズを指定する。



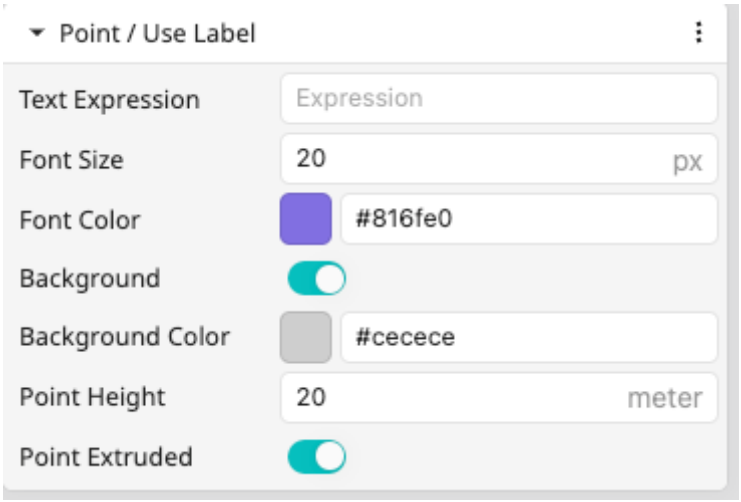
Point > Point Stroke
ポイント形式のデータにおいて、線の太さと色を設定する。



Point > Use Image > Value
ポイント形式のデータにおいて、3Dモデルを地物表示に利用する際に設定する。3Dモデル（glb形式）へのURLとサイズを設定する。

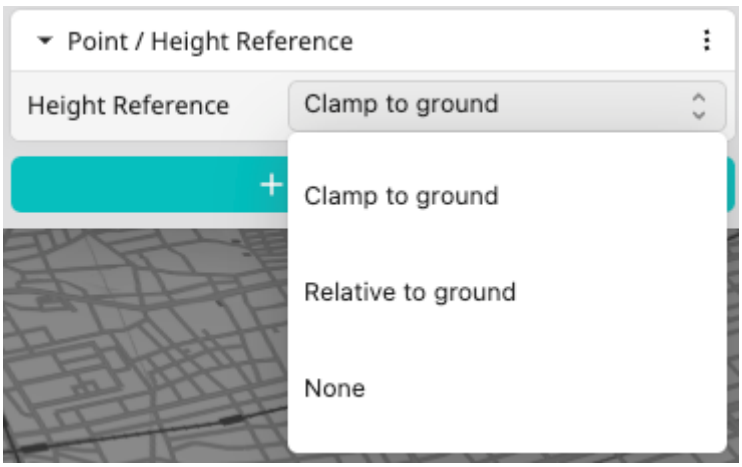


Point > Use Label
ポイント形式のデータにおいて、ラベル表示を行う。ラベル表示に利用する属性のキーと、その他フォント設定を行う。



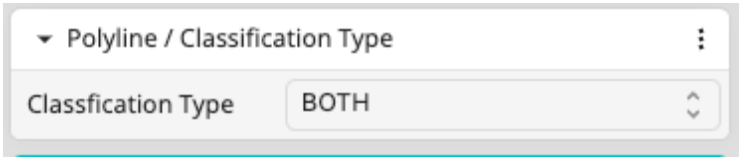
Point > Convert from CSV
CSV形式のデータをポイント形式で表示する際に利用する。緯度、経度、高さに利用する属性のキーを設定する。

Point > Height Reference
ポイント形式のデータにおいて、高さ位置設定を行う。
・Clamp to ground: 地表面にくっつくように高さが設定される。
・Relative to ground: 地表面から相対的な高さに設定される。
・None: 楕円体からの相対的な高さに設定される。

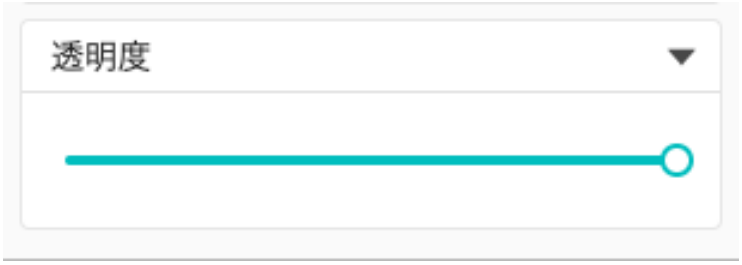


Line > Classification Type
ラインデータにおいて、地形や3Dモデルと位置的に重なるラインをどのようにドレープ表現するかを設定する。

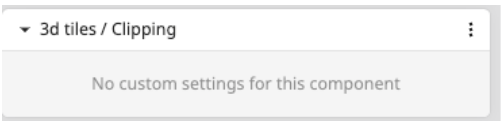
- Both: 3Dモデルと地形の両方に被るように表現する。
- Cesium 3D Tiles: 3D Tilesデータに被さるように表現する
- Terrain: 地形データに被さるように表現する。



3D Tiles > 透明度
3D Tilesの透明度を変更可能にする。

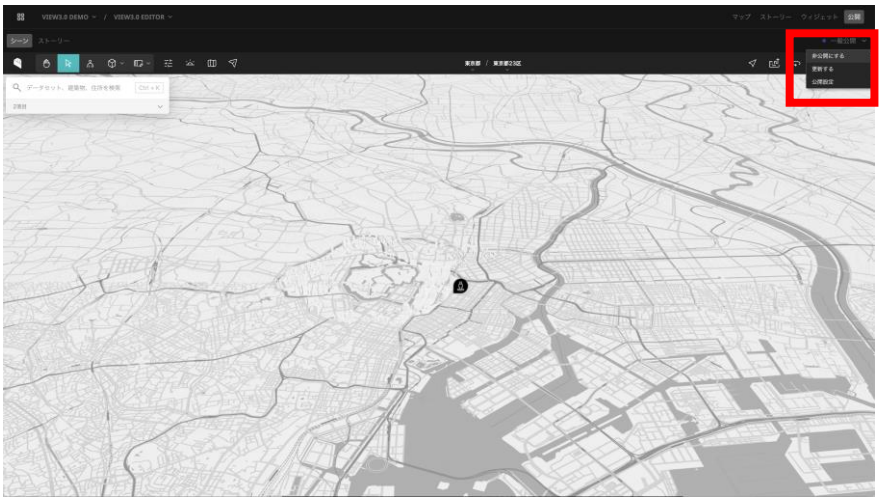


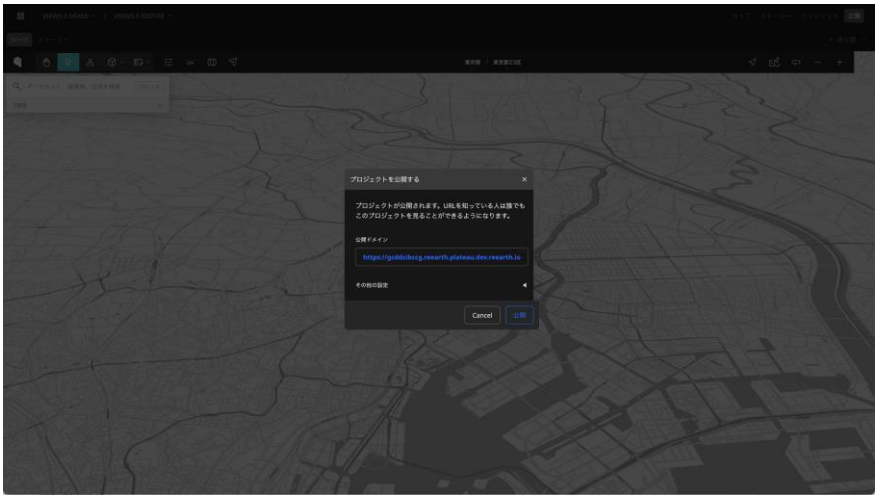
3D Tiles > Clipping
3D Tilesのクリッピング機能（マウスで操作可能な箱で建物などをくり抜いて表示する）を有効化する。

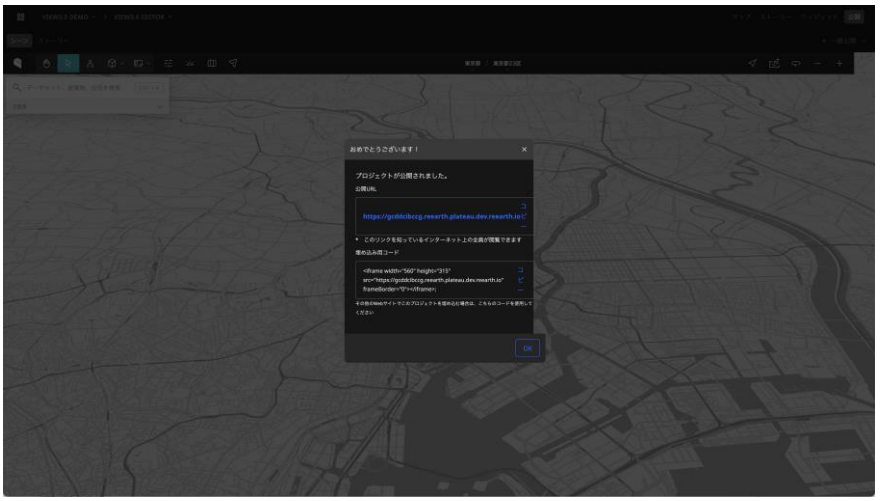


2.2.5 プロジェクトの公開

プロジェクト編集画面の右上の「公開」からプロジェクトを公開することができる。







2.2.6 データの可視化

PLATEAU VIEWで表示可能なデータフォーマットとそれらの表現方法のパターン、及びコンポーネントを概説する。

（1）主な対応データフォーマット

パフォーマンスの観点からWebで表示するのに適したフォーマットの使用を推奨する。Shapefileは Web用に最適化されたフォーマットではないため、表示が遅いなどの理由で非推奨。

表　主な対応データフォーマット

フォーマット	説明	MyDataでローカルファイルから追加	MyDataでURLから追加	表示レイヤーの指定	対応する地物
GeoJSON	JSONで記述される、2Dの空間データを扱うためのフォーマット。 ドキュメント：https://geojson.org/	✅	✅	不要	ポイント・ポリライン・ポリゴン
KML	空間データをXMLで記述するフォーマット。 ドキュメント：https://developers.google.com/kml/documentation/?hl=ja	✅	✅	不要	ポイント・ポリライン・ポリゴン
CZML	JSONで記述される、CesiumJSでの空間データを表現するのに適したフォーマット。時系列データも表現可能。ドキュメント：https://github.com/AnalyticalGraphicsInc/czmlwriter/wiki/CZML-Structure）	✅	✅	不要	ポイント・ポリライン・ポリゴン
CSV	テキストファイルのフォーマットの1つ。PLATEAU VIEWでは経度・緯度・高さの情報を持ったポイントデータとして取り込むことができる。読み込むためには少なくとも経度・緯度を表す列が必要。	✅	✅	不要	ポイント・ポリライン・ポリゴン
3D Tiles	Cesium社によって開発された、Web上で3D地理空間コンテンツを効率的にストリーミングするためのデータ形式。 ドキュメント：https://github.com/CesiumGS/3d-tiles	❌	✅	不要	3D Tiles
MVT	Mapbox Vector Tileの略称。Mapbox社が開発。Web上で高速かつ効率的なベクタータイルの表示を実現 ドキュメント：https://docs.mapbox.com/data/tilesets/guides/vector-tiles-standards/	❌	✅	必須	ポリゴン
glTF	3Dモデルのデータフォーマットの1つ 3DモデルをWeb上で簡単に共有・表示するための標準形式として、Khronos Groupによって策定。 ドキュメント：https://github.com/KhronosGroup/glTF	✅	✅	不要	ポイント※
WMS	Webマップサービスの1つであり、OGCによって定義された標準規格。地理情報をリクエストし、画像として返す。 ドキュメント：https://www.ogc.org/standard/wms/	❌	✅	必須	ラスター

※表示には別途ポイントデータが必要なためCSVなど他のデータと組み合わせることを推奨

（2）補遺

•3D データの高さ方向の座標値について
•PLATEAU VIEWで表示する3Dデータを作成する場合、z座標は楕円体高（ジオイド高 + 標高）とする必要がある。
•PLATEAU VIEW 1.1からの変更点
•1.1では内部システムにTerriaJSが採用されており、データカタログを記述したJSONを作成することで、凡例などのカスタマイズを可能にしていた。
•2.0以降では独自のEditor及びVIEWを開発し、UI操作のみで凡例やスタイルのカスタマイズが可能になった。
•これに伴い、1.1で使用されていたツール「plateau-catalog-generator」は使用する必要がない。また、それを使用して生成したJSONはPLATEAU VIEW 2.0以降では使用できないことに注意。

2.3 PLATEAU VIEWの機能及びUIの解説

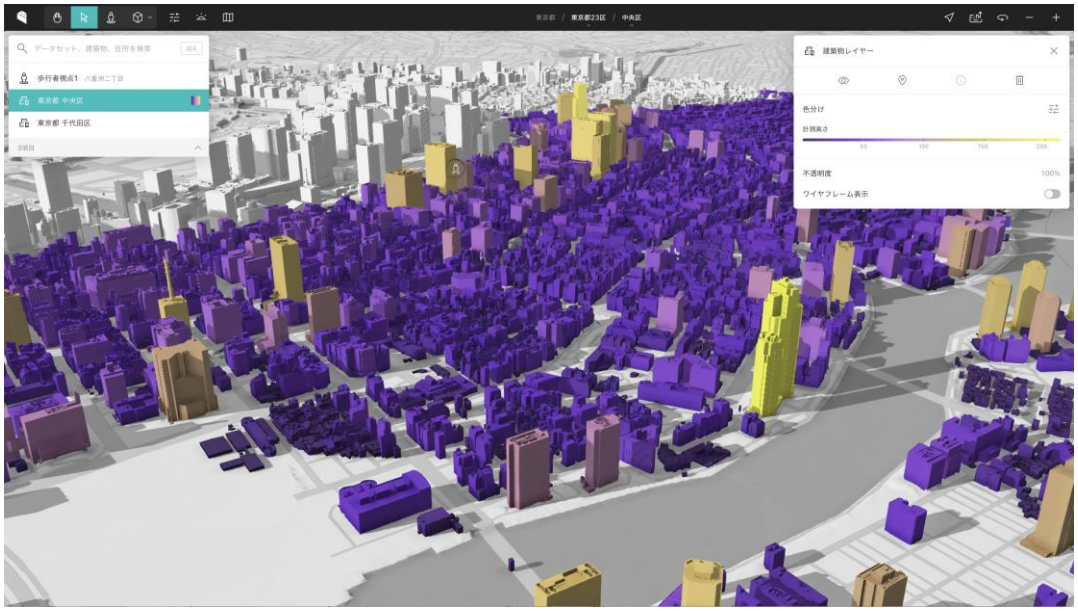
ここではでは、PLATEAU VIEW 3.0の機能及びUIについて、PLATEAU VIEW 2.0と比較してどのような改善を行なったかを解説する。なお、本章冒頭でも紹介したが、チュートリアル形式での使い方の解説は、

- 「TOPIC 2｜PLATEAU VIEWで体験する[1/2]｜3D都市モデルをブラウザで利用」
- 「TOPIC 2｜PLATEAU VIEWで体験する[2/2]｜他の地理空間情報を重ねて確認」

を参照されたい。

本節では、PLATEAU VIEW 3.0の改修において検討した6点を重点的に解説する。

- UIUXの再設計
- レンダリング品質の改善
- 新しい地図表現手法の導入
- 歩行者モードの改善
- 時系列表現の改善
- クリッピング機能
- ストーリー機能



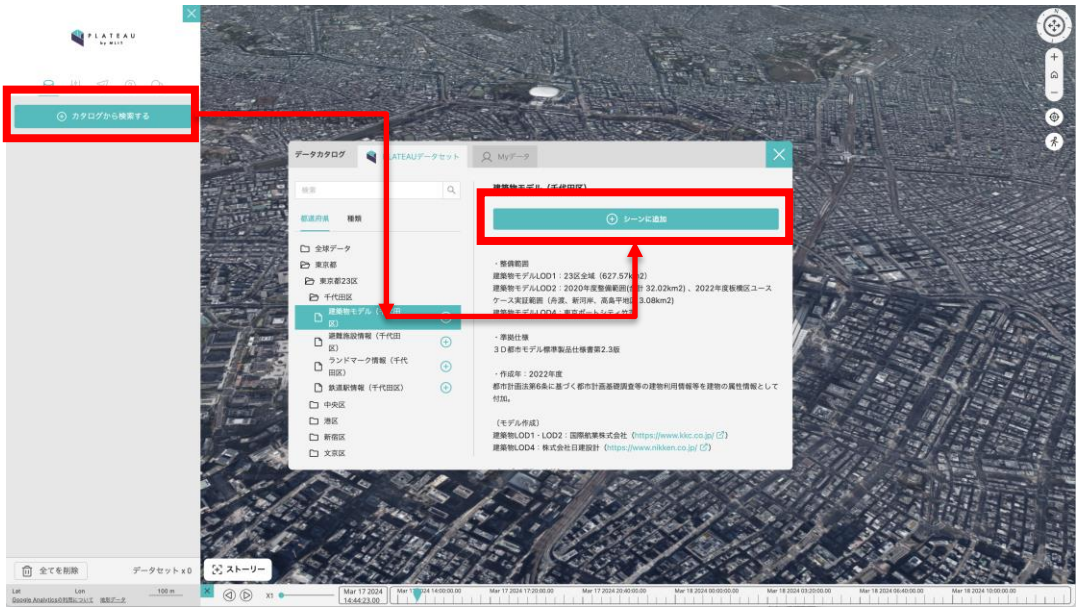
2.3.1 UIUXの再設計

（1）PLATEAU VIEW 2.0でのUIUX設計の課題

PLATEAU VIEW 2.0は、初見のユーザーが離脱しうる多くの課題があった。

データ追加の難しさ

VIEW 2.0では、閲覧可能なPLATEAU関連データセットが大量に存在するにも関わらず、1つ目のデータ追加までの押下数が多く、1つのデータを閲覧するまでのハードルが高かった。



レイヤーに関する操作と表示の混在

VIEW 2.0では、レイヤーを選択すると、レイヤーに関する凡例、絞り込みなどが全てサイドバーに表示されていた。これにより、複数レイヤーを追加した際にサイドバーの役割が複雑化し、非常に操作しにくいUIになっていた。

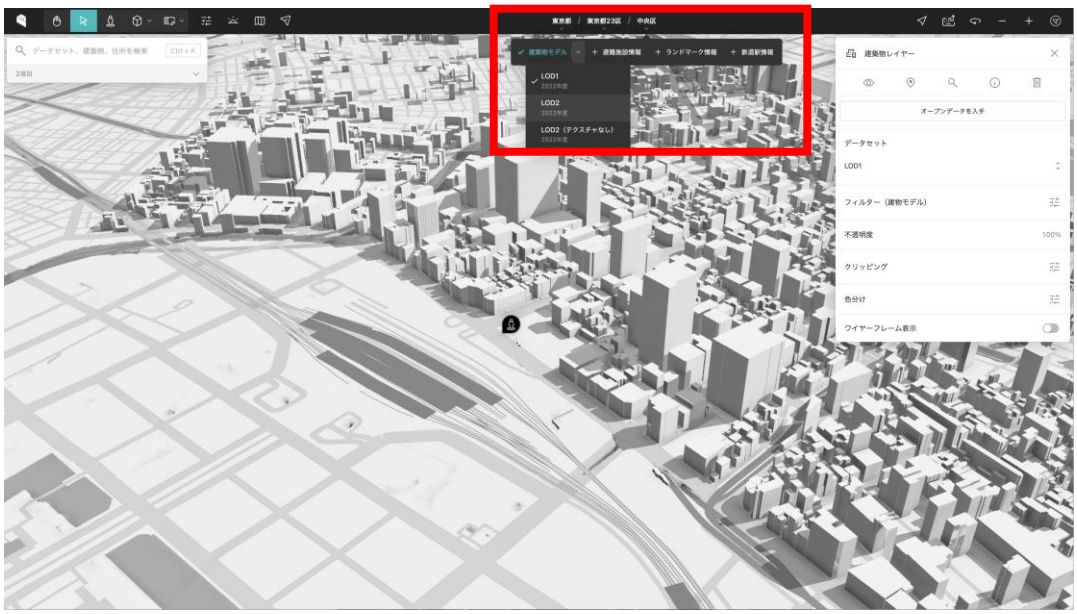
（2）PLATEAU VIEW 3.0でのUIUX再設計

上記の課題をふまえ、PLATEAU VIEW 3.0では以下のUIUX改善策を講じた。

カメラ位置に応じたデータの追加（レイヤーショートカット機能）

上述したデータの追加における課題を解決するため、一覧からデータを選ぶのではなく、今ユーザーが見ている地図領域に対応したデータ追加をできるようにした。

この機能は、カメラ位置が変更された際に、カメラの中心点の緯度経度から住所を検索し、その住所に対応した都市のデータを追加可能にする機能である。この機能によって、ユーザーは最短1押下でデータを追加することができるようになった。



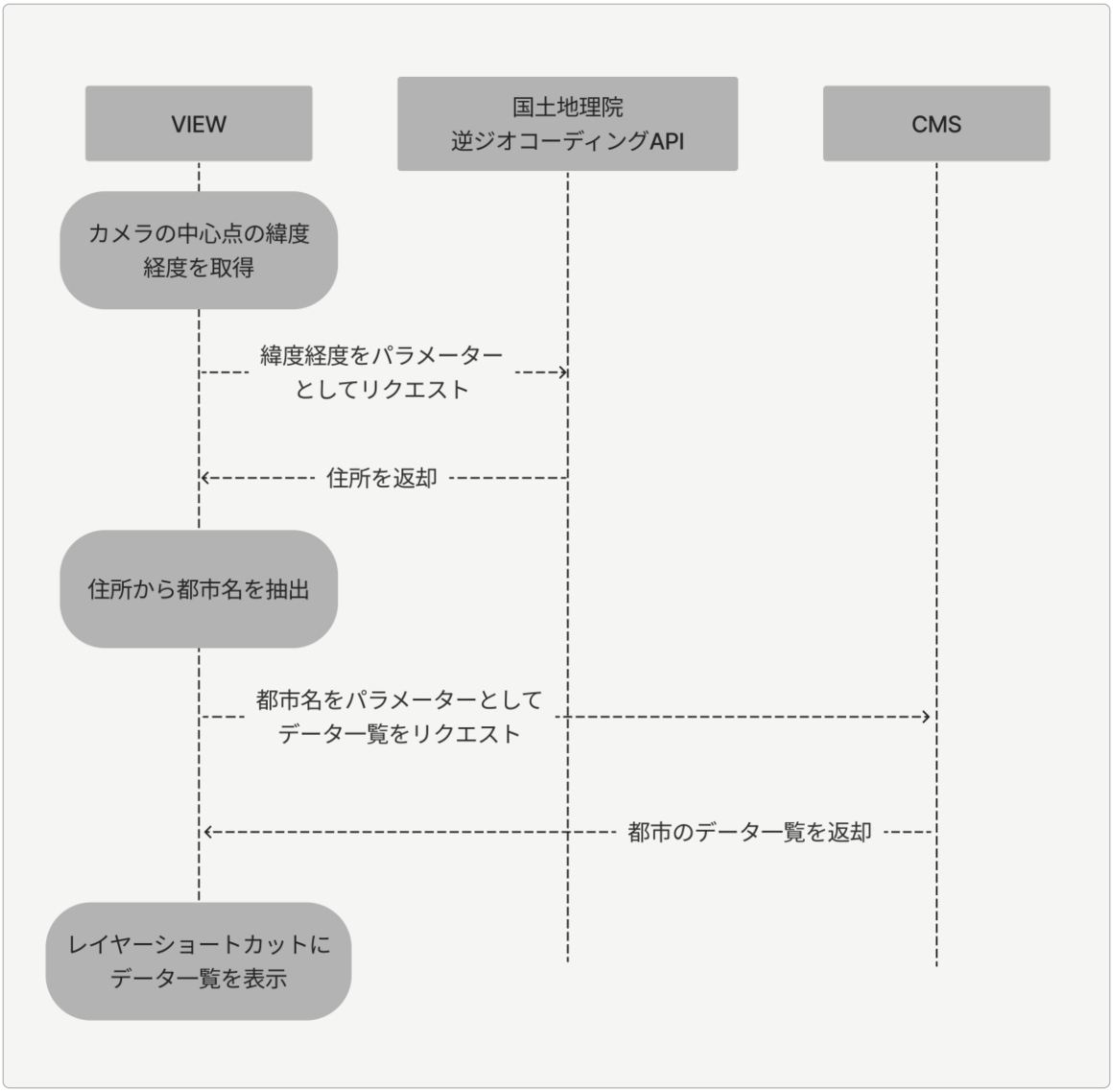
レイヤーショートカット機能の実装

上述したレイヤーショートカット機能を実現するため、以下の技術を利用している。

- CesiumJSのカメラ中心点の緯度経度取得
- 国土地理院逆ジオコーディングAPI

レイヤーショートカットでは、CesiumJS上でのカメラ移動イベントをトリガーとして、カメラの中心点の緯度経度を取得する。その後、国土地理院逆ジオコーディングAPIを利用して緯度経度から住所への変換を行っている。住所から都市名を抽出し、1.3.3で紹介したCMSのAPIを利用して都市のデーター一覧を取得して表示している。

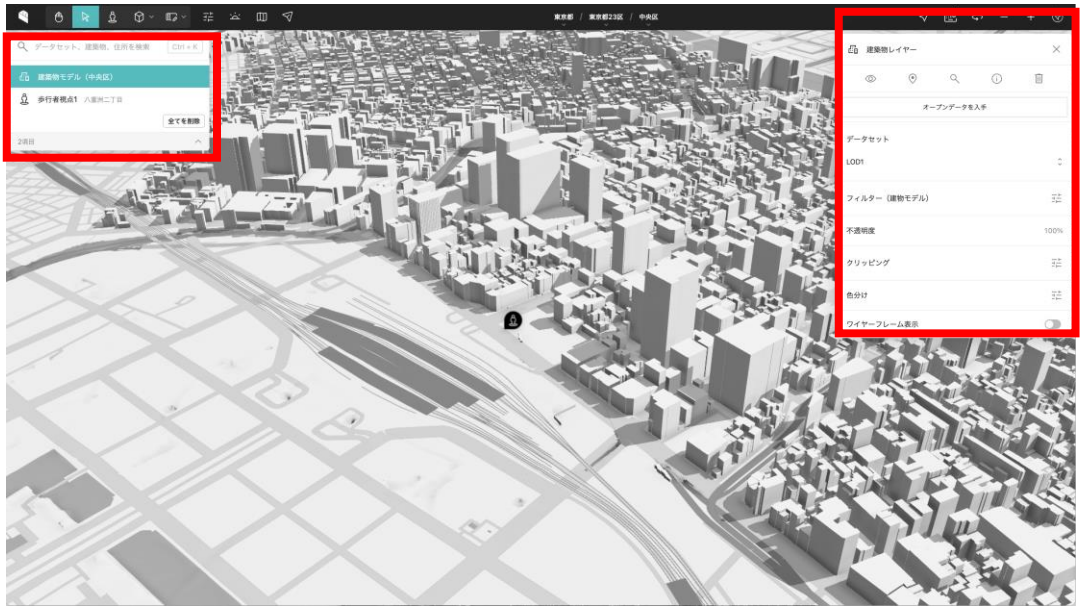
図　レイヤーショートカットの仕組み



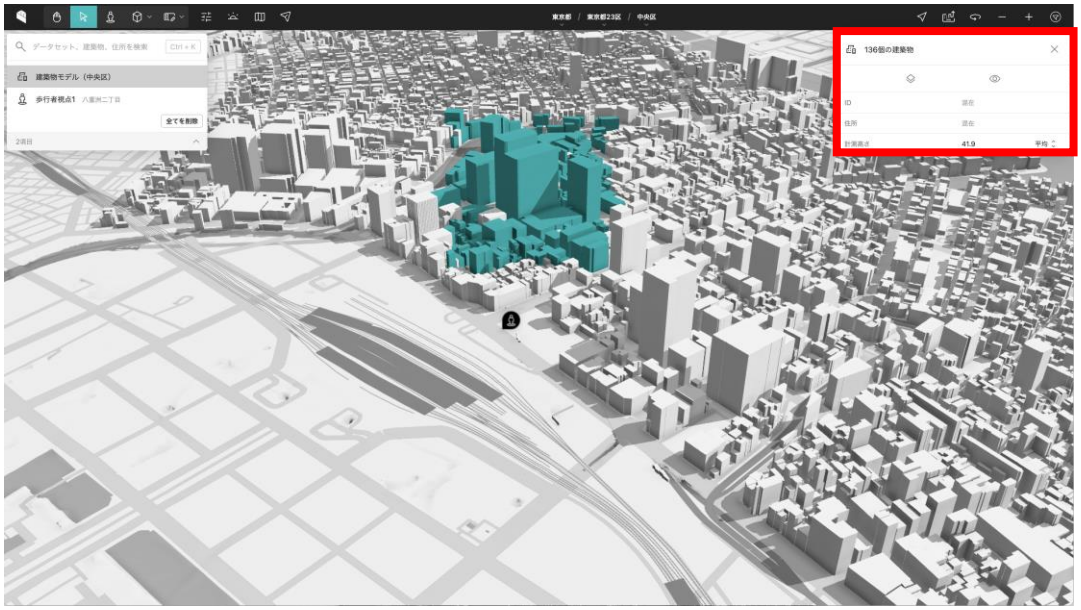
選択モデルの改善

また、VIEW 2.0においてサイドバーが複雑化する問題に対応するために、そもそもの選択モデルを見直した。選択モデルとは、ユーザーにとって「何を今選択しているのか」という概念の見直しである。VIEW 3.0では、レイヤー、地物（複数）、凡例それぞれが全て選択可能な対象であり、選択した対象に対するアクションをインスペクターから行うことができる。

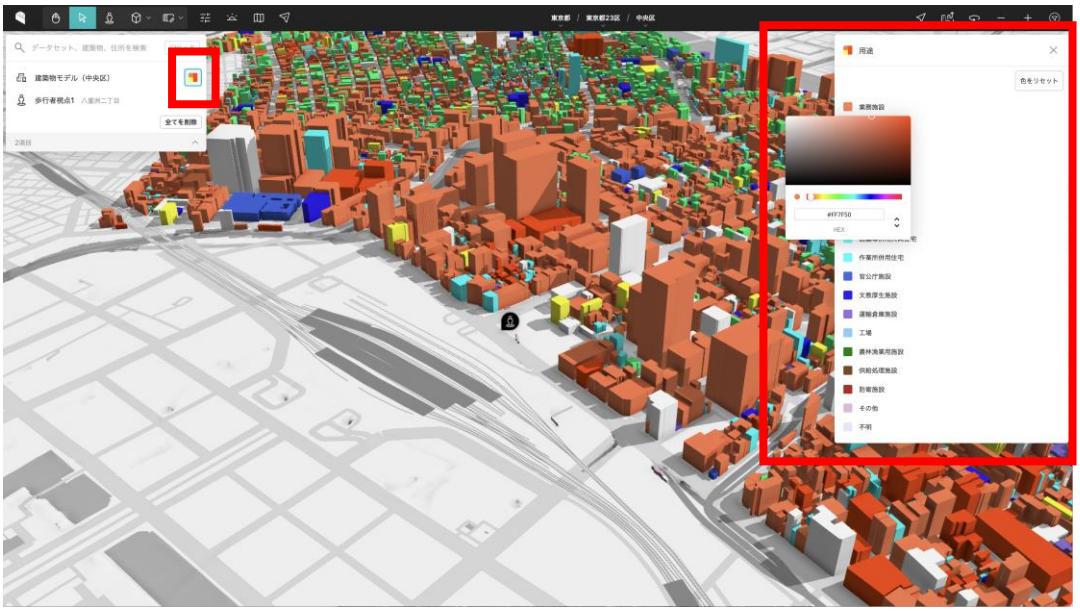
以下の画像のように、画面全体において、左側ヒエラルキーウィンドウで対象を選択し、右側インスペクターでアクションを行うという設計になっている。レイヤー選択時には、インスペクターでそのレイヤーに対するフィルターや色分けなどができる。



地物を選択時には、選択中の地物に関する属性情報の表示ができる。



凡例の選択時には、凡例の設定を変更することができる。



このように、ユーザーが選択できる対象の整理と、選択対象が違ってもそれらに対するアクションの仕方を統一することで、使いやすいデザインを実現した。

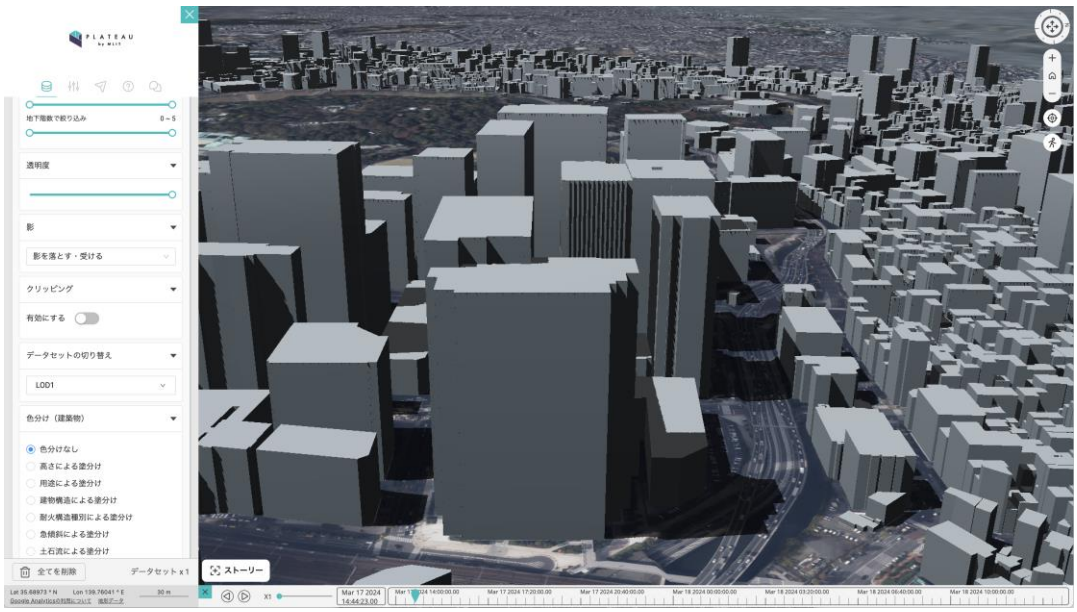
2.3.2 レンダリング品質の改善

（1）PLATEAU VIEW 2.0でのレンダリング品質の課題

VIEW 2.0は、3D都市モデルデータのレンダリングという観点において、以下のような課題があった。

陰影表現の不自然さ

VIEW 2.0では3D都市モデルデータや地形データにおける陰影表現に特別な処理をしておらず、単に影がそのまま地面に落ちるだけであった。しかし、現実世界では環境光は周囲に反射し、本来影になる箇所も照らすため、ぼやけた影になるはずである。



3D Tiles、MVT形式データのレンダリングパフォーマンス

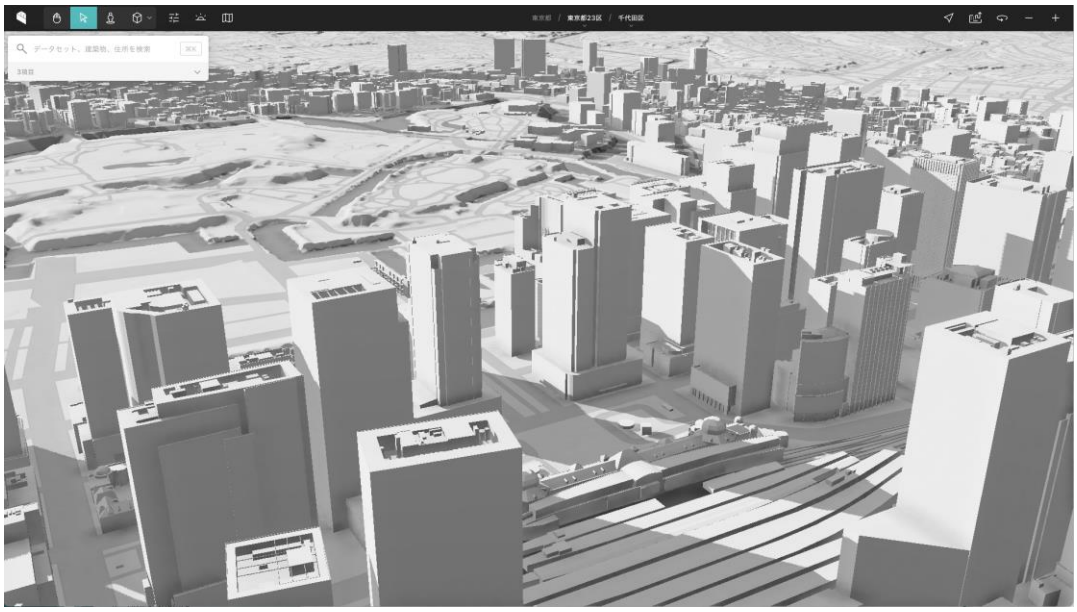
VIEW 2.0では頂点数が多いデータの描画に時間がかかっていた。特に、洪水浸水想定区域モデルのデータには10秒近く必要なケースも見られた。

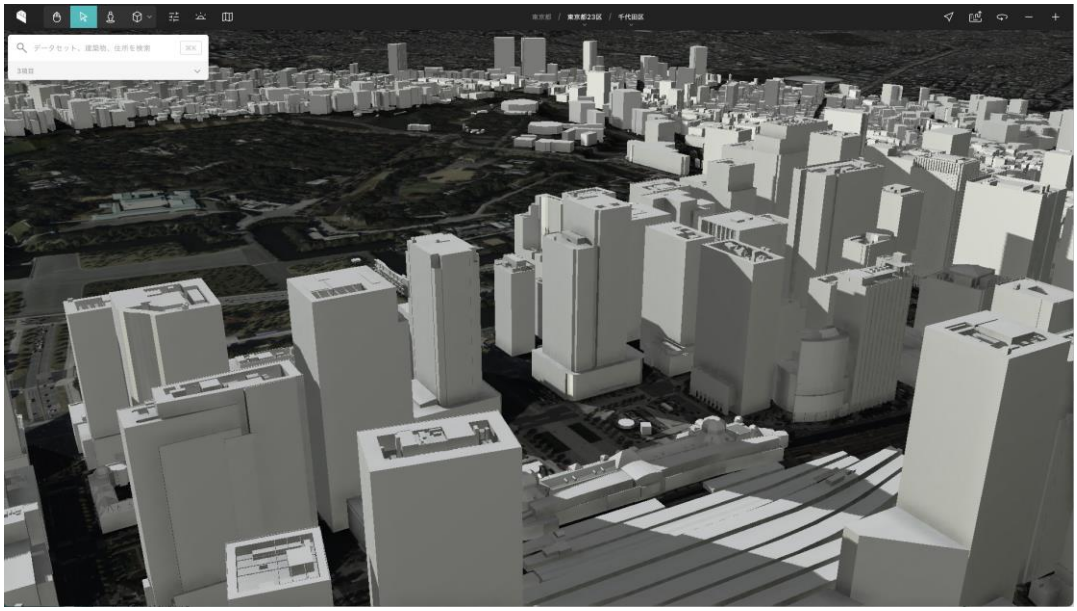
（2）PLATEAU VIEW 3.0でのレンダリング品質の改善

上記の課題をふまえ、VIEW 3.0では以下の観点からレンダリング品質の向上を行なった。

3DCG技術を用いた自然なライティングの実現

球面調和関数を用いた自然光の再現を行なった。球面調和関数についての説明は、1.5.1で解説しているためそちらを参照する。以下の画像は球面調和関数を適用して、建築物モデルなどに自然なライティングが適用されていることを表している。球面調和関数には設定すべき係数があり、それらによって色味が変わる。VIEW 3.0ではそれぞれのベースマップごとに球面調和関数の係数を調整している。以下の画像では、白地図の時には白に近い光が建築物当たり、衛生写真の時には地面の色に調和した色が当たっていることを表している。

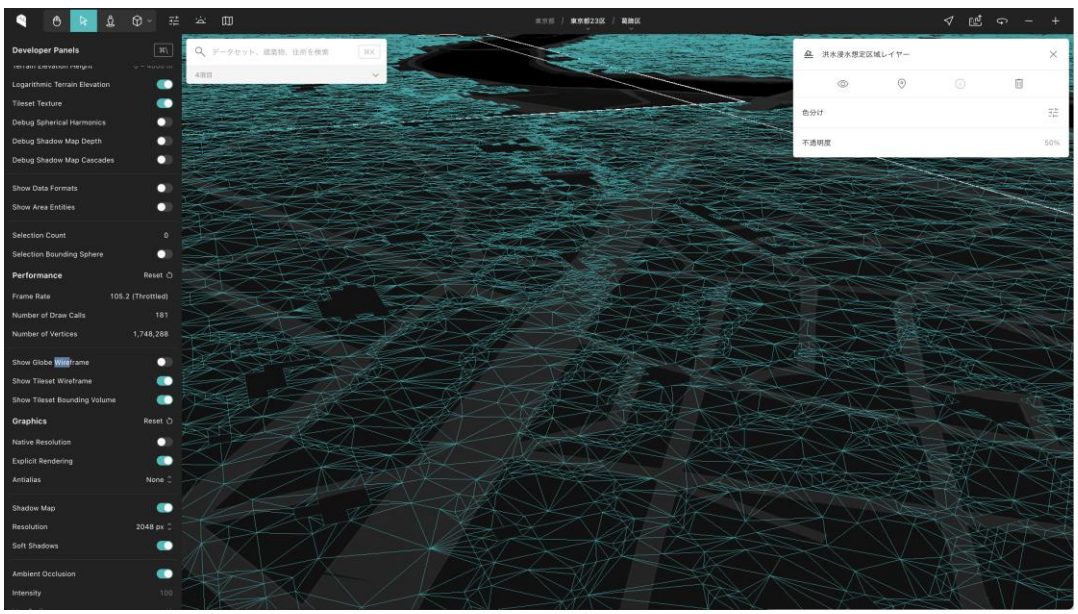




さらに、Terrainの法線を利用した影表現を行なった。これによりTerrainにも3D都市モデル同様の影表現を行うことができ、地形の起伏などをリアルに再現することができた。以下は富士山の陰影表現の例である。

3D Tilesの頂点数削減

頂点数とは、3D空間上に存在するオブジェクトの頂点の総数を指す。PLATEAU VIEW上で閲覧可能な3D Tiles形式の3D都市モデルデータは、1.4記載のFME FlowによってCityGMLから3D Tilesへデータ変換が行われている。PLATEAU VIEWでの表示においては、この変換後3D Tilesデータの頂点数が多いほど詳細なデータ表現ができるが、その分パフォーマンスも劣化する。洪水浸水想定区域モデルのようなデータは水面の高さを表現するために大量の頂点を有している。しかし、ユーザーに認知できる範囲を超えて詳細度を保持しても、パフォーマンスが劣化するため、頂点数の計測とパフォーマンス改善を行なった。結果として、ユーザーに提供すべきデータの詳細度を損なわない程度に頂点の間引き処理を行い、データのサイズを半減させることができた。以下は、利根川水系利根川の水面の頂点を表している。

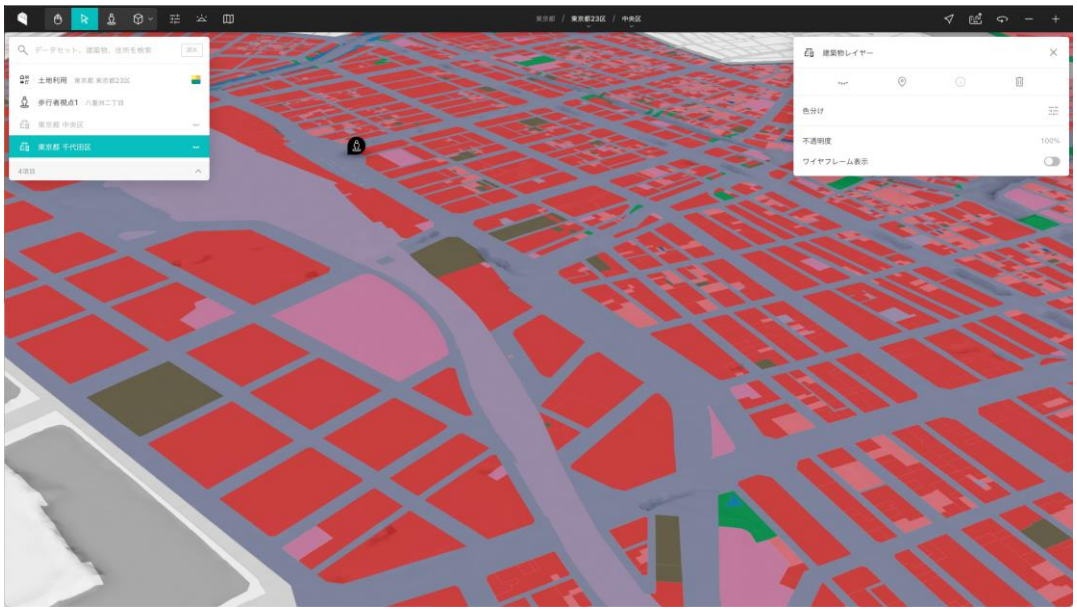


最適な3D Tilesのタイル化

3D Tilesは文字通りタイル化されたデータであり、カメラの位置によって読み込むべきデータを動的に判別している。この際に、タイル化する基準によって本来不要であるタイルが読み込まれたり、逆にタイルを細かく分割しすぎることで、ドローコール（GPUに対しての描画命令）が多くなりパフォーマンスが劣化する。PLATEAU VIEW 3.0では、以下のように3D Tilesのタイル化基準を変えてドローコールの変化を計測し、FME Flowで3D Tilesへの変換時の最適なパラメーターを導出した。（以下は中央区の建築物モデルがどのようにタイルとして分割されているかを表している。）

MVTデータのレンダリング改善

PLATEAU VIEWが内部的に利用しているCesiumJSは、標準ではベクトルタイル形式のデータの描画をサポートしていない。そこで、MVTに変換された3D都市モデルデータを高品質でレンダリングするため以下の実装を行なった。



ベクトルタイルの動的ラスター化

CesiumJSはラスター形式のタイルデータの描画のみを標準でサポートしている。そこで、PLATEAU VIEWでは、CesiumJSでMVTデータを表示するため、cesium-mvt-imagery-providerというJavaScriptライブラリを開発した。このライブラリは、内部的にMVTデータをラスター化する処理を行っており、カメラ位置が変更された際に発火するCesiumJSのイベントごとに処理を行っている。

ウェブワーカーを利用したレンダリング

また、ベクトルデータのラスター化は計算負荷が大きいため、通常のスレッドで描画処理を実行をするとUIのレスポンスが悪化する。この問題を解消するため、PLATEAU VIEWでは、MVTデータの描画をウェブワーカーを利用して行っている。ウェブワーカーとは、処理をバックグラウンドのスレッド実行するための手法である。このように、計算負荷が大きい処理をウェブワーカーで実行することにより、ユーザーにとってのUIレスポンスを維持したまま多くの地物を含んだMVTデータのレンダリングを実現している。

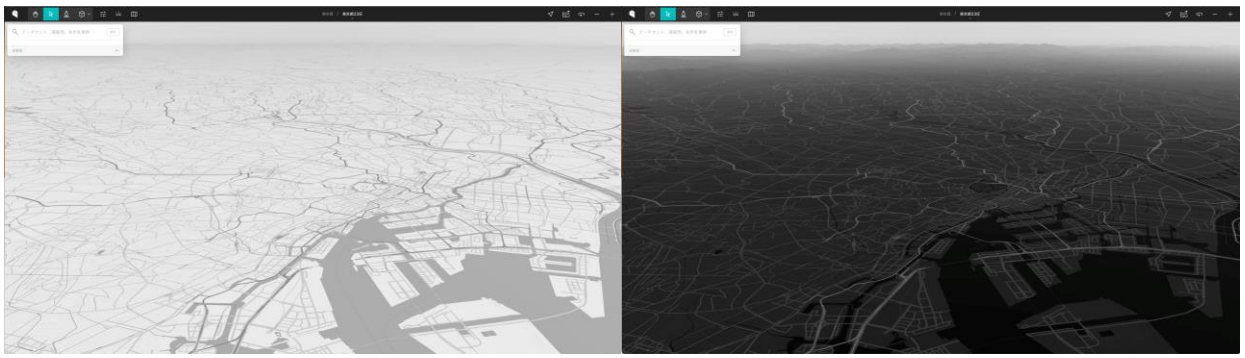
MVTデータの高解像度描画

MVTデータはタイル化されだデータであるため、データ本体に最小最大のズームレベルが存在する。PLATEAU VIEWでは地表面に近づいた時（高ズームレベル）、データ本来が持つズームレベルを超えてもラスターデータの解像度を高くしている。例えば、現在のカメラ位置のズームレベルが19、データ本体はズームレベル16までの場合でも、ズームレベル19相当の解像度でベクトルデータをラスター化することで高解像度で表示を行っている。

地理院地図Vectorを利用した高解像度ベースマップの生成

PLATEAU VIEWの背景地図には、地理院地図Vectorを利用して独自のスタイルを適用した白地図・黒地図が利用可能である。これらは、試験公開されているベクトルタイル形式の地理院地図をサーバーサイドでラスター化して配信している。サーバーサイドでは、事前に定義された白地図・黒地図それぞれのスタイル設定に基づき、mapbox-gl-jsというJavaScriptライブラリを利用してラスター化を行っている。また、サーバーはGPU搭載のGoogle Compute Engineでレンダリングを行っており、一度レンダリングされたラスタータイルはストレージへキャッシュされる。ベクトルタイル形式の地図データを独自にラスター化することで、PLATEAU VIEW独自のスタイル適用と、高ズームレベルでも高解像度を保つことを可能にしている。以下のコード例は白地図において、それぞれどのような色を設定しているかを表している。

```
export const lightStyle = createStyle(
  createLayerStyles({
    landColor: gray(1),
    waterColor: gray(0.75),
    seaRouteColor: gray(0.675),
    coastlineColor: gray(0.7),
    boundaryColor: gray(0.33),
    roadColor: gray(0.9),
    majorRoadColor: gray(0.9),
    highwayColor: gray(0.66),
    roadOutlineColor: gray(0.85),
    majorRoadOutlineColor: gray(0.85),
    highwayOutlineColor: gray(0.6),
    railwayColor: gray(0.6),
    railwayBackgroundColor: gray(0.92),
    railwayJrDashColor: gray(1),
    stationColor: gray(0.6),
  }),
);
```



2.3.3 新しい地図表現手法の導入

（1）PLATEAU VIEW 2.0での地図表現手法の課題

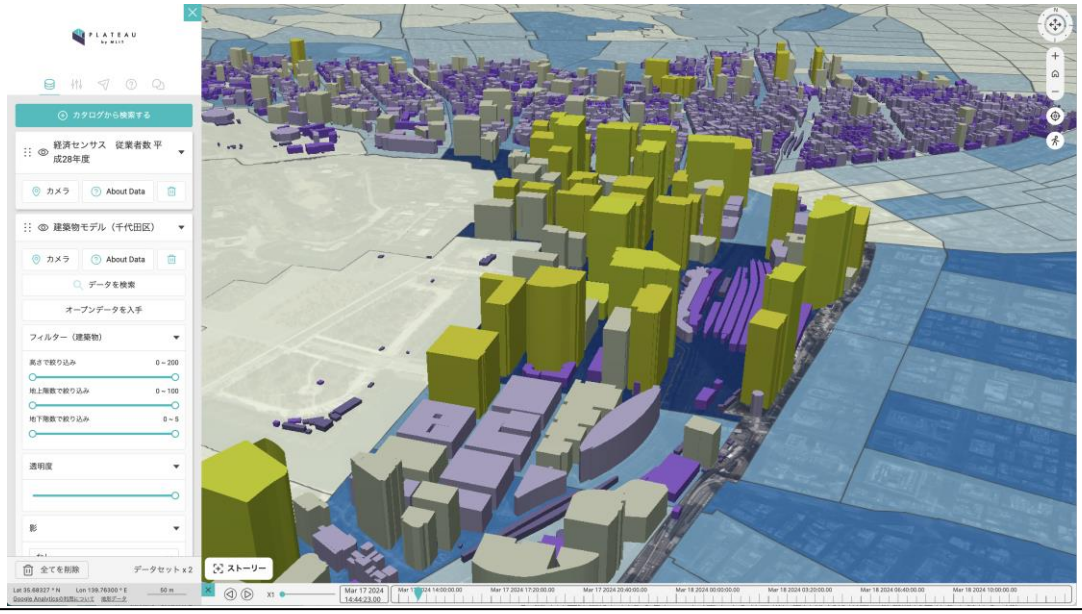
VIEW 2.0は、地図表現手法に限りがあり、ユーザーの目を引くような表現ができていなかった。

連続値の色分けにおける課題

例えば、建築物モデルの高さによる色分け機能では、ある閾値によって不連続に色分けが行われていた。以下はVIEW 2.0における高さによる色分けを表している。このように、連続値に対して非連続的な色分けがされることで、本来伝えるべき情報を伝えきれていなかった。

その他のデータとの重畳のしにくさ

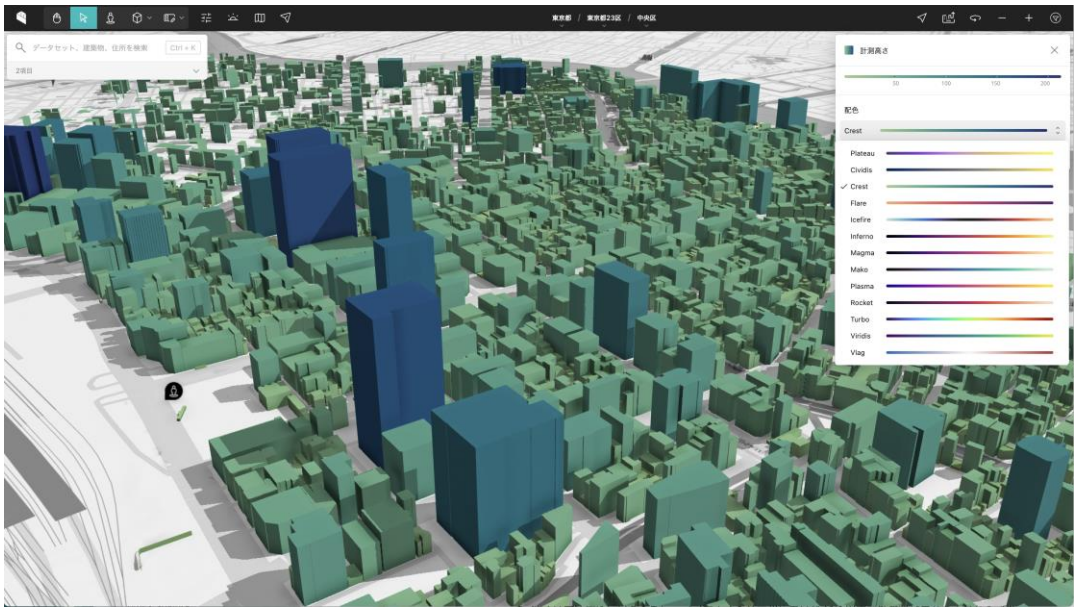
GISデータは、その他のデータと重畳して閲覧することで新たな知見を得ることができる。以下は、千代田区の建築物モデルと経済センサス従業員数のデータをVIEW 2.0で重畳した様子である。VIEW 2.0では全国をカバーする形でこうした統計データを閲覧することができなかったことと、ポリゴン等の単純な重ね合わせしか表現できず、データの重畳による知見が得られにくかった。



（2）PLATEAU VIEW 3.0での新しい地図表現手法の導入

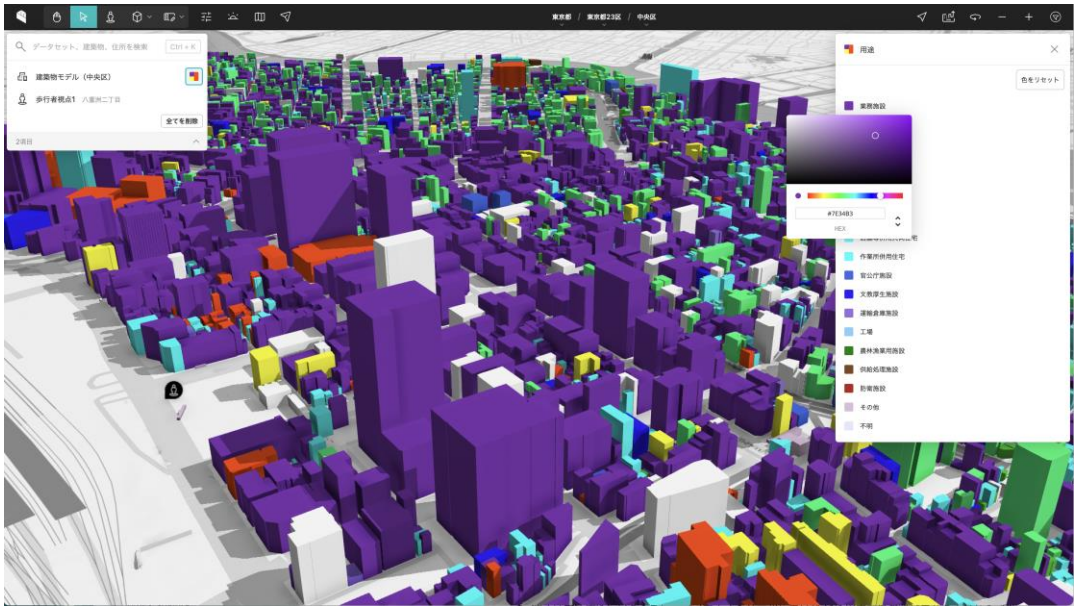
グラデーションを用いた連続値の色分け

3D都市モデルデータの属性には、高さのような連続値の属性と、用途のような離散値の属性が存在する。これらをそれぞれの特性に分けて、色分けができるようにすることで、より詳細な色分けを実現した。以下は高さによる色分けの例である。連続値においては、グラデーションで色分けするようにし、カラーマップをユーザーが選択できるようにすることで、重畳するデータやベースマップの色味に合わせて閲覧しやすくした。



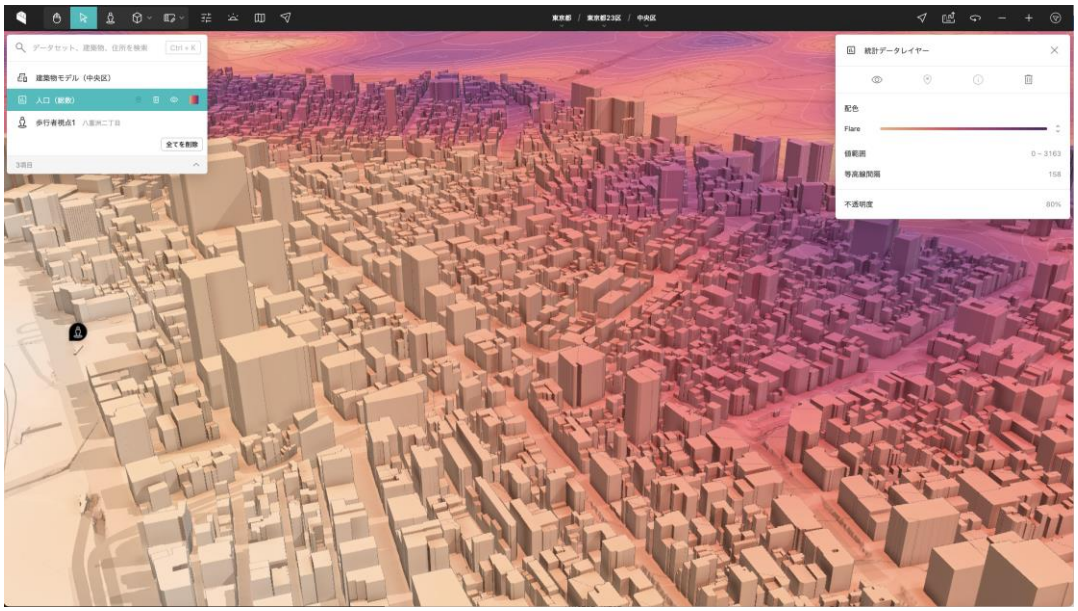
カラーパレットを用いた離散値の色分け変更

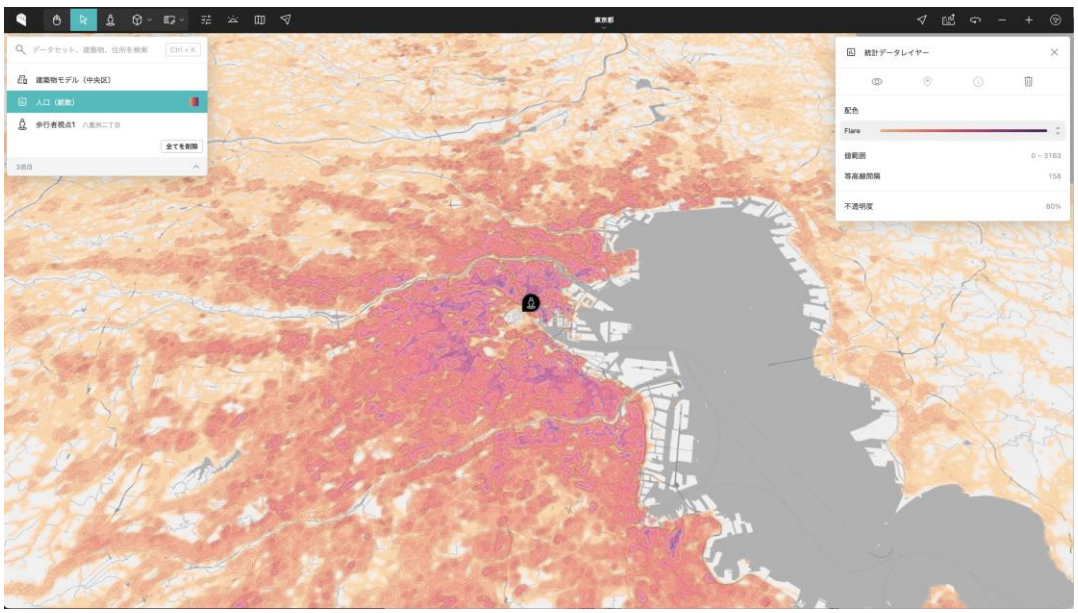
一方で、用途などの離散的な属性値に決まった色を適用し、ユーザーがそれぞれの色をカラーパレットから変更できるようにした。これにより、ユーザーは関心のある属性値をより目立たせて表示することが可能になった。



ヒートマップ表現の導入：e-Stat国勢調査地域メッシュ統計データ利用

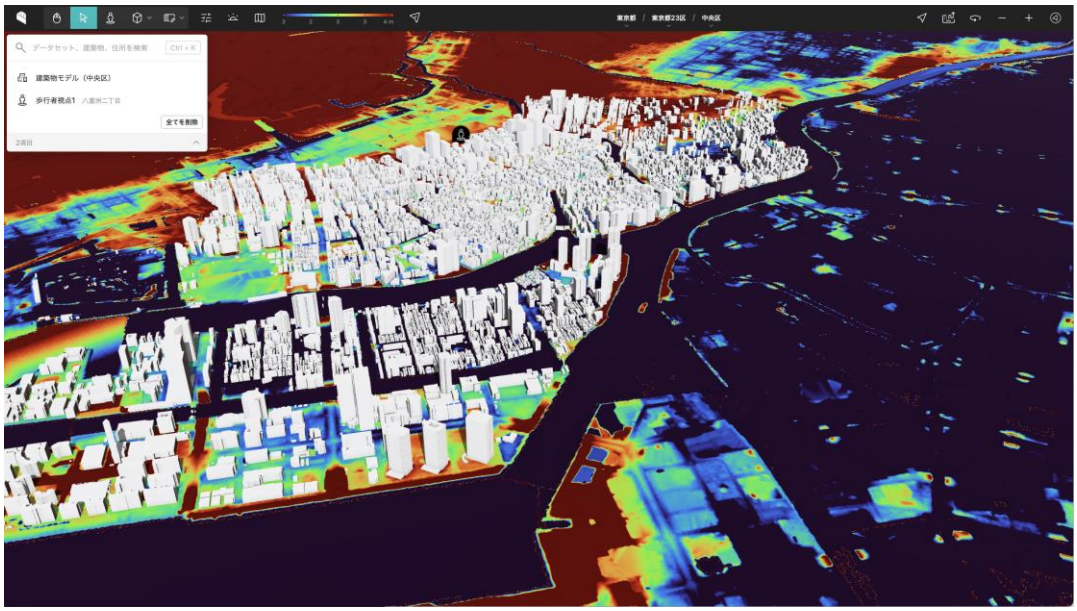
また、統計データをe-Stat国勢調査地域メッシュ統計データを用いてヒートマップ表現を導入した。これにより、全国をカバーする形で統計データと3D都市モデルデータの重畳が可能になった。さらに、ヒートマップとして表現することで、広域地図として見た場合にもわかりやすい表現になっている。

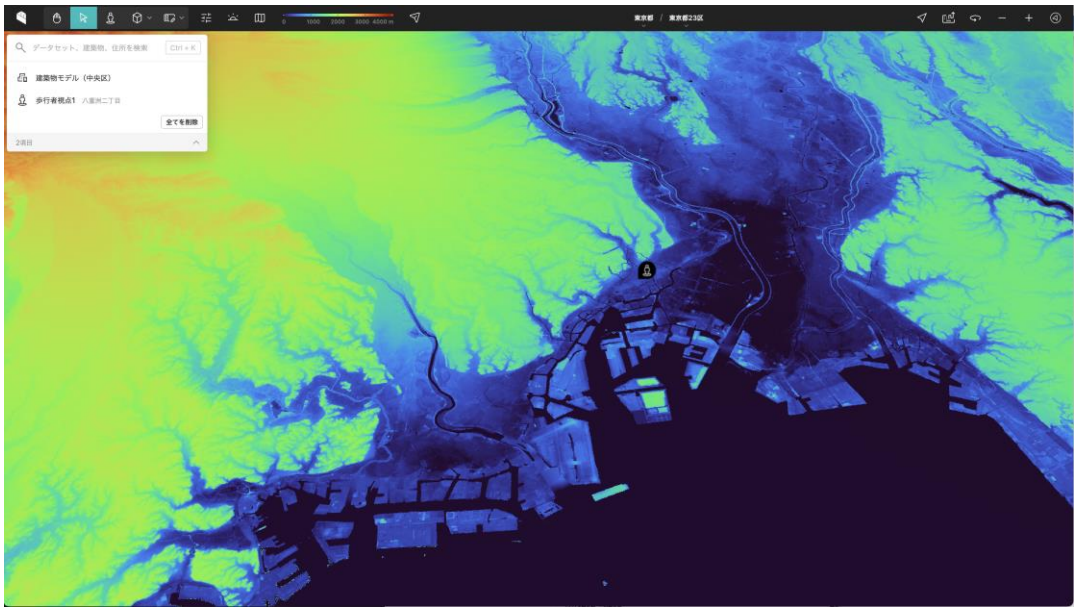




ヒートマップ表現の導入：標高値利用

統計データを用いたヒートマップに加え、標高値を用いたヒートマップ表現も追加した。標高値によるヒートマップ表現では、国土地理院の標高タイルを利用した。国土地理院の標高データは、符号付き24bit整数で保存されており、この標高値をピクセル単位の色を決定する際に利用してヒートマップ表現を行なっている。





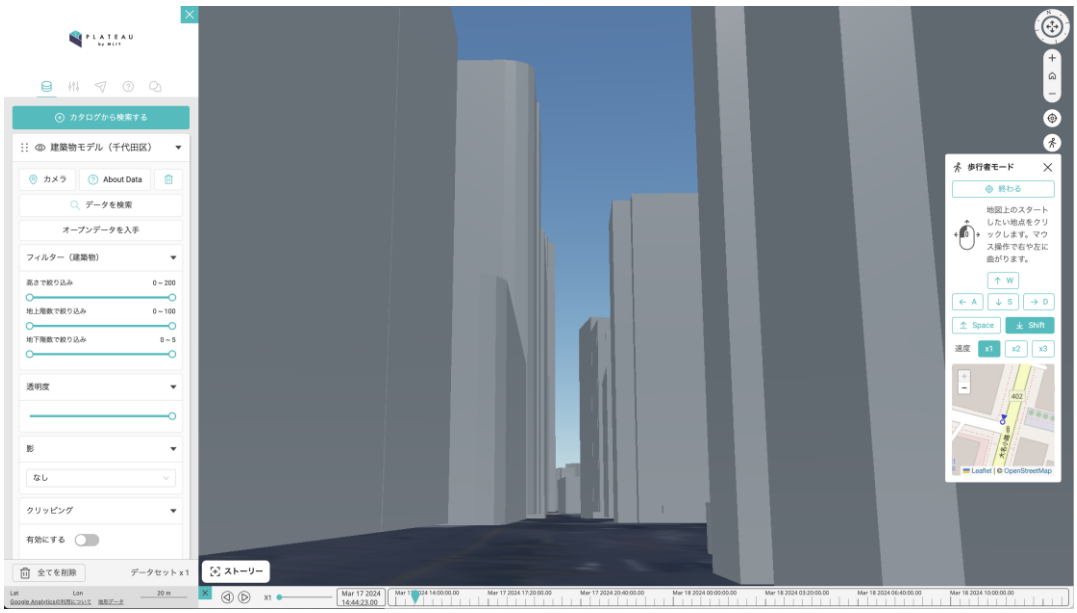
2.3.4 歩行者モードの改善

（1）PLATEAU VIEW 2.0での歩行者モードの課題

VIEW 2.0は、歩行者モードには以下のような課題があった。

現実世界との乖離

VIEW 2.0では歩行者モードを利用して、一人称視点での地図の閲覧が可能であった。しかし、3D都市モデルデータと重畳して閲覧すると、現実世界における場所が特定しにくく、一人称目線での回遊によって得られる情報が限定的であった。また、進行はキーボード操作のみで行うため、適切な距離移動することが難しく、操作性に問題があった。

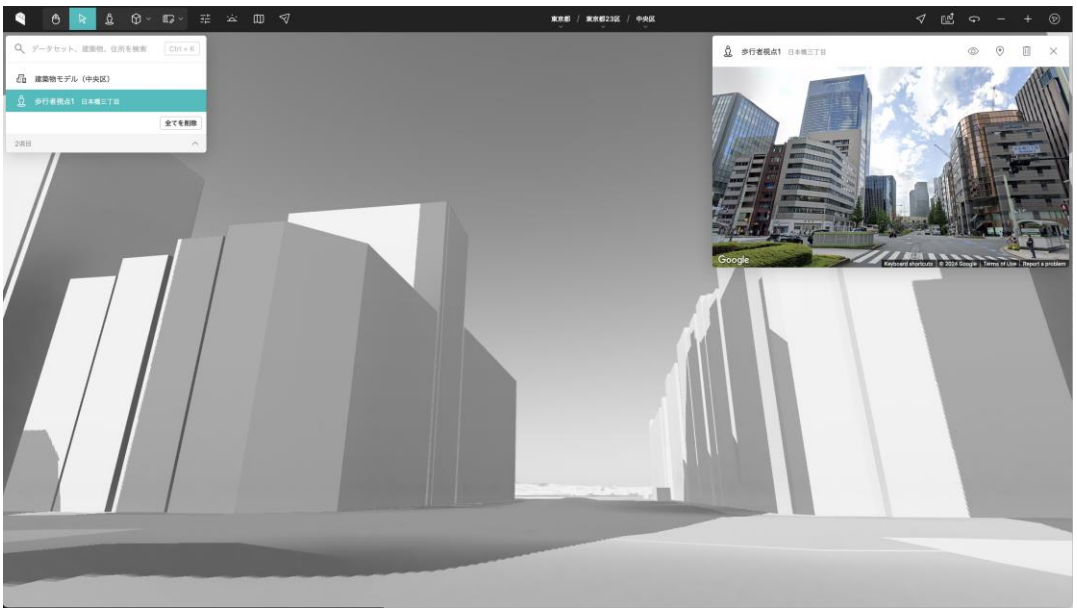


（2）PLATEAU VIEW 3.0での歩行者モードの改善

上記の課題をふまえ、VIEW 3.0では以下ように歩行者モードを改善した。

Google Street Viewとの連携

Google Street Viewと連携して現実世界の様子を表示することで、3D都市モデルデータと実際の様子を照らし合わせながら見ることが可能になった。また、Google Street Viewの視点と地図内のカメラの視点は同期している、という設計をすることで、2つの視点を1度に操作できるようにした。この設計に基づき、カメラの画角も同期しているため、ユーザーにとっても直感的に操作ができる。



地図上に表示されたフラスタム（視野を表現した四角推）はその地点の歩行者におけるカメラの画角を表しており、Google Street Viewのカメラ画角とも連動している。

2.3.5 時系列表現の改善

（1）PLATEAU VIEW 2.0での時系列表現の課題

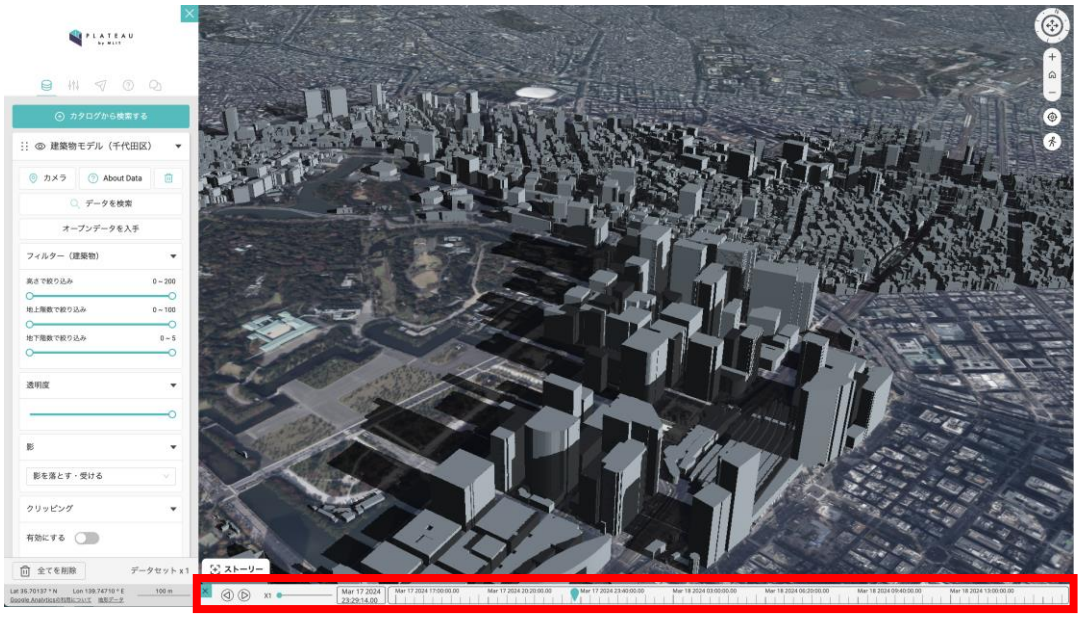
VIEW 2.0は、時系列データに以下のような課題があった。

時系列表現における異なる目的の混在

VIEW 2.0では地図における時間情報を変更するために、画面下部のタイムラインバーを操作することで変更をしていた。しかし、

- 太陽光のシミュレーションをするために時間を変更したい。
- 時系列データを閲覧するために時間を変更したい。

という異なる目的を同一のUIによって実装することで操作上の競合が起きていた。また、複数の時系列データが地図に追加された際には、最後に追加された時系列データしか再生できないという課題も変えていた。

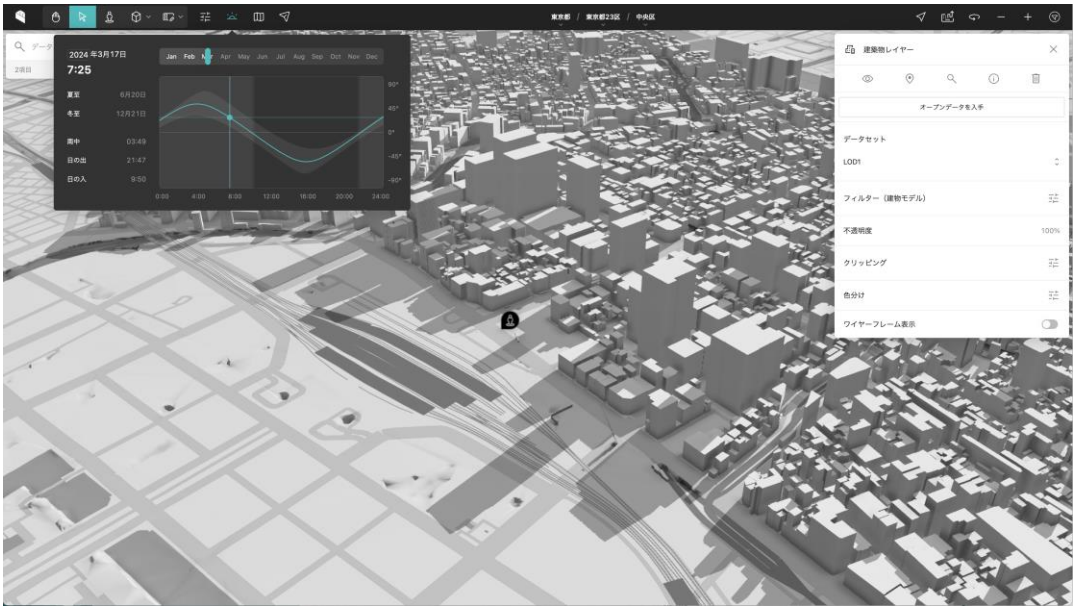


（2）PLATEAU VIEW 3.0での時系列表現の改善

上記の課題をふまえ、VIEW 3.0では以下ように時系列表現を改善した。

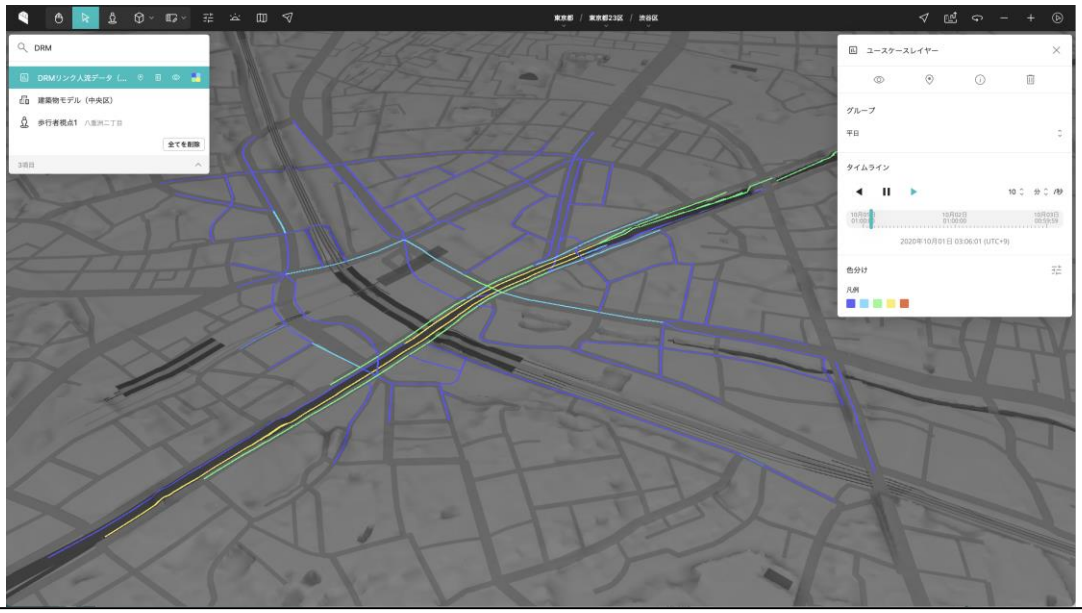
太陽光シミュレーション用UIの追加

上述したように、太陽光のシミュレーションと時系列の再生という2つの異なる操作が同一のUIによって行われる課題を解決するため、太陽光シミュレーションをするための専用UIを開発した。これによりユーザーは任意の日時における太陽光の位置を簡単にシミュレーションできるようになった。



時系列データ再生用UIの追加

さらに、時系列データを再生するためのタイムラインバーを新たに開発し、対象のレイヤーにひも付く形で再設計をした。これにより、複数の時系列データが追加されても、再生したいデータを選んで閲覧できるようになった。



2.3.6 クリッピング機能

（1）PLATEAU VIEW 2.0でのクリッピング機能の課題

VIEW 2.0は、クリッピング機能に以下のような課題があった。

クリッピングボックス自体の編集・移動が競合する

VIEW 2.0のクリッピング機能では、クリッピングボックスの

- 位置を変更する
- サイズを変更する
- 回転する
- 中の地物を選択する

というそれぞれの操作が競合し、非常に操作しにくいものであった。このため、クリッピングボックスを移動しようとした際に誤ってサイズを変更してしまったり、想定していない位置に移動してしまうケースが多々あった。

（2）PLATEAU VIEW 3.0でのクリッピング機能の改善

上記の課題をふまえ、VIEW 3.0では以下ように時系列表現を改善した。

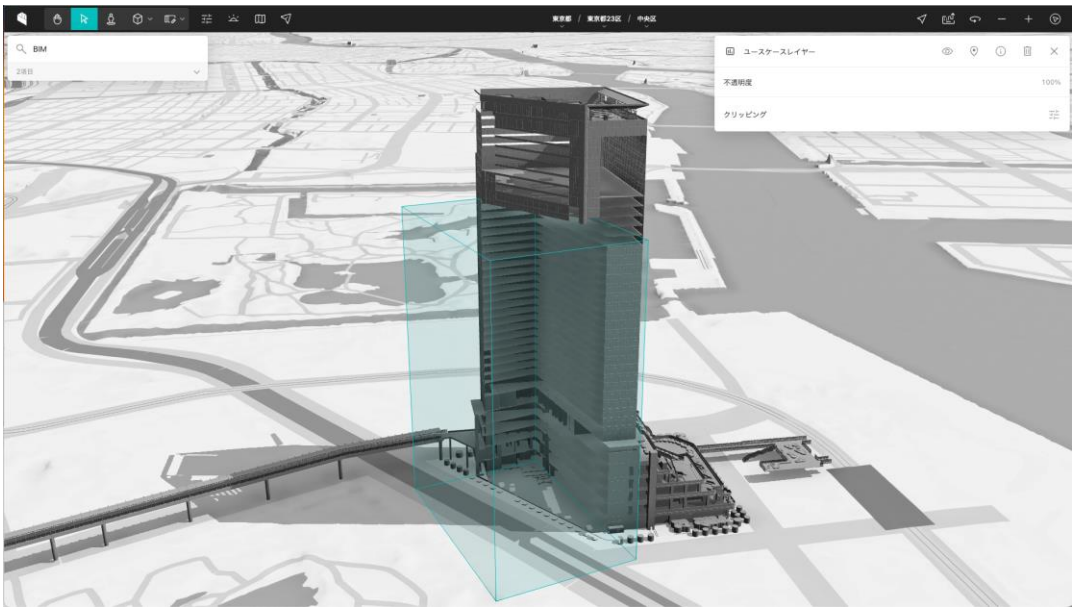
作図モードを踏襲したクリッピング機能

VIEW 3.0では作図モードを用いて3D図形の作図を行うことができる。クリッピング機能にも同様の案を採作図後の移動はできない

用し、クリッピングボックスを以下の通りに改善した。

- 作図時は自由に形状を作る
- クリッピングボックスを非表示にするオプションを追加することで内部の地物を選択できる

これらによって、ユーザーは自由な形状でクリッピングボックスを作図することができるようになり、例えばLOD4建築物等の内部構造の閲覧などがよりしやすくなった。



クリッピング機能の実装

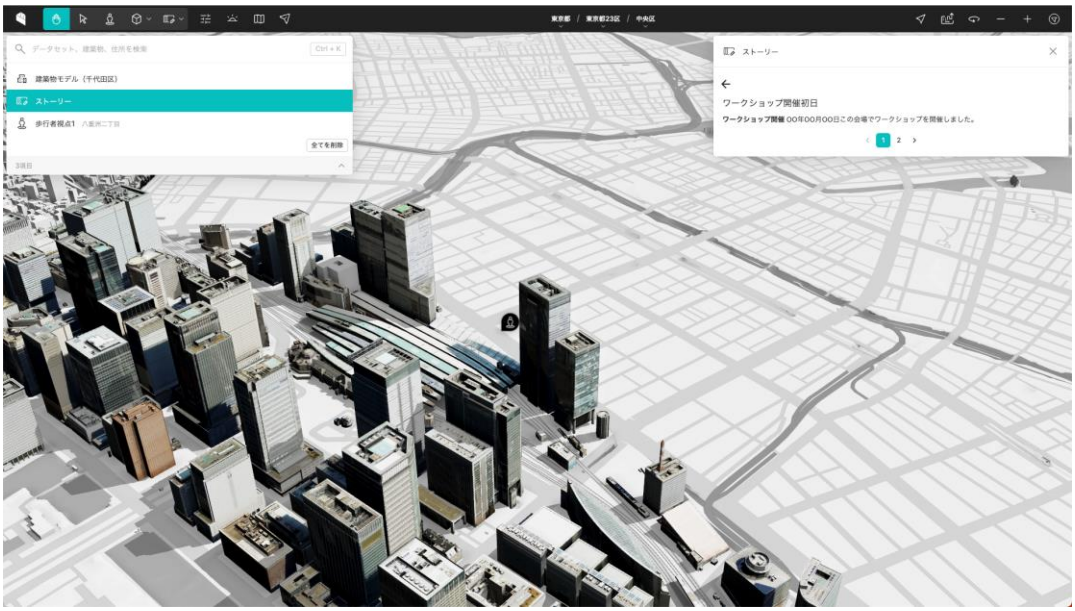
上述したクリッピング機能を実現するため、PLATEAU VIEWではCesiumJSのクリッピングAPIを内部的に利用している。CesiumJSでは、ClippingPlaneCollectionというクラスを設定し、3D Tilesのプロパティとして設定することでクリッピングを行うことができる。以下はCesiumJSのドキュメントに記載のクリッピング例である。以下の例では、Cartesian3というクラスにxyz座標を設定してインスタンス化を行っている。同様に本クリッピング機能では、作図時に取得された座標値からCartesian3クラスをインスタンス化し、3D Tilesを表すクラスのプロパティとして設定することでクリッピングを可能にしている。

```
const clippingPlanes = new Cesium.ClippingPlaneCollection({
   planes : [
      new Cesium.ClippingPlane(new Cesium.Cartesian3(0.0, 1.0, 0.0), 5.0)
   ],
});
```

2.3.7 ストーリー機能

ストーリー機能の改善

ストーリー機能はVIEW 1.1から存在した機能だが、VIEW 3.0ではストーリー機能のデザインもリニューアルした。歩行者モードと合わせた体験とするため、各ストーリーにおいてキャプチャされた位置をカメラアイコンで表現している。また、テキストエディタを利用することで、ストーリー内のコンテンツにリッチなスタイルで内容を組み込むことができる。



ストーリー機能の実装

ストーリー機能は、内部的にはCesiumJSのカメラ座標とそれにひも付くテキストコンテンツの組み合わせを配列の形式で格納することで順序立てた再生を可能にしている。マークダウン形式のテキストエディタにはOSSのeasy-markdown-editorを利用している。以下はCesiumJS内の現在のカメラ座標を取得するコードサンプルである。

```
const cameraPosition = viewer.scene.camera //シーン内のカメラ座標を取得
console.log(cameraPosition.position)
```

第３章　PLATEAU VIEWの構築手順

3.1 PLATEAU CMS・PLATEAU Editorの環境構築
3.1.1 システム構成の全体像

本章では、システム構築を担当するエンジニアを想定読者として、CMS及びEditorのシステムの構築を主にGCP及びAWS上で行う。

本章で構築するシステム構成の全体像は、1.2~1.4を参照すること。ただし、MongoDB Atlas、Auth0、Cloud DNSは両者で同じものを共用する。

システム構築においては、GCPを利用する場合と、AWSを利用する場合で手順が異なる。3.1.2を参照し、共通する各種サービスのセットアップを行なった後、GCPの場合には3.1.3を、AWSの場合は3.1.4を参照すること。

3.1.2 各種サービスのセットアップ（共通）

前項で示したシステムを構築するためのセットアップ手順を示す。事前に以下の準備が必要である。

- クレジットカード（GCPなどで使用）
- 本システムで使用するドメイン（管理画面にアクセスしてネームサーバーの変更ができること）

本項で示す各種コマンドは、特に注記がない限り、Windows（PowerShell）とMacOS/Linuxの両方に対応している。注記がある場合はそれに従うこと。

（1）Auth0のセットアップ

Auth0（https://auth0.com/jp）にアクセスしてAuth0のアカウントを作成し、テナントを開設する。ドメイン名・リージョンは問わない。なお（2024年3月現在）東京リージョンを選ぶことができる。

次に、公式のQuick Startを参考に、Auth0 Management APIのセットアップを行う（「Configure the Provider」の項の前まで）。

https://github.com/auth0/terraform-provider-auth0/blob/main/docs/guides/quickstart.md

注意：PLATEAU Editorなどへのログインに失敗する原因になるため、今この時点ではまだ自身のユーザーをAuth0のテナント内に作成しないこと。

後のステップで必要となる情報は以下のとおりなので確認しておくこと。

- ドメイン
- クライアントID
- クライアントシークレット

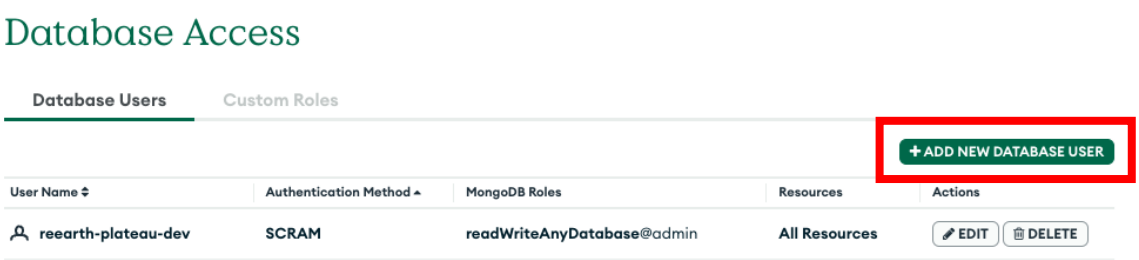
後で使用するので以下のコマンドを実行して、クライアントシークレットを変数に出力しておく。

```
export AUTH0_CLIENT_SECRET="<クライアントシークレット>"
```

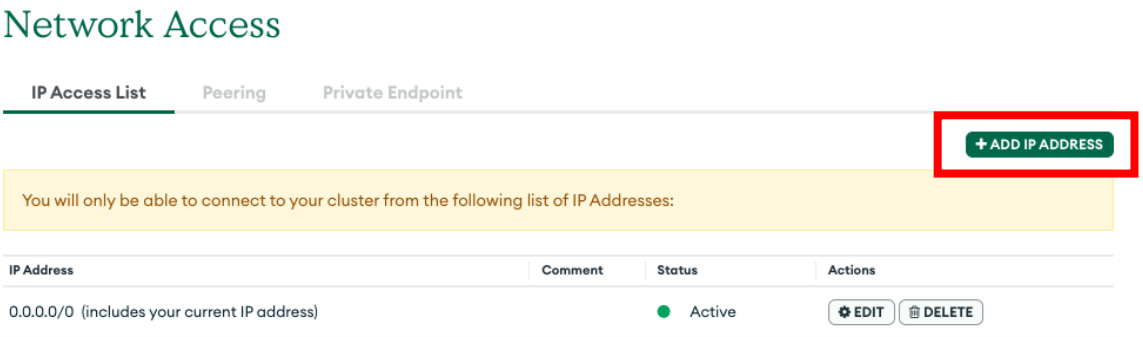
（2）MongoDB Atlas のセットアップ

MongoDB Atlas （https://www.mongodb.com/ja-jp/atlas） にアクセスしてアカウントを作成し、プロジェクト及びクラスタを作成する。なお有料のクラスタを作成する場合は、クレジットカードの登録が必要である。名前やリージョンなどは自由。

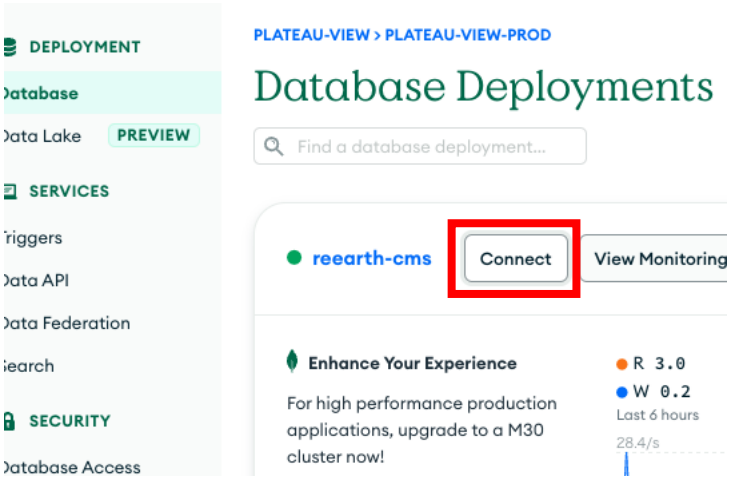
プロジェクトの「Database Access」の設定を確認し、データベースのユーザーが作成されていることを必ず確認する。作成されていない場合は、「ADD NEW DATABASE USER」ボタンを押下して、任意のパスワードなどの認証方式でユーザーを作成する。そのユーザーに対して、少なくとも読み書き（readWrite）の権限を許可するロール（Read and write to any database・Atlas Adminなど）を設定する必要がある。



次に、プロジェクトの「Network Access」の設定を確認し、全てのIPアドレスからの接続が許可されていることを必ず確認する。許可されていない場合は、「ADD IP ADDRESS」ボタンを押下し、Access List Entryで「0.0.0.0/0」と入力してエントリーを追加する。



クラスタを作成後、クラスタ名の右にある「Connect」ボタンを押下し、「Connect your application」からデータベース接続URLを取得する。なおデータベース接続URLは慎重に扱い、他人には共有しないこと。



後で使用するので以下のコマンドを実行して、データベース接続URLを変数に出力しておく。URL中の <password> を先ほど作成したユーザーのパスワードに置き換えることを忘れないよう注意すること。

```
export REEARTH_DB="<データベース接続URL>"
```

3.1.3 Google Cloud Platform向けセットアップ

次に、GCP（Google Cloud Platform）を利用して環境構築する手順を記載する。AWS（Amazon Web Services）を利用する手順は、3.1.4を参照すること。

（1）事前準備

Google Cloud Platform （https://console.cloud.google.com/?hl=JA） にアクセスし、Googleアカウントでログインし、プロジェクトを作成する。プロジェクトにはあらかじめクレジットカードを登録しひも付けておくことが必要である。
後で使用するので以下のコマンドを実行して、プロジェクトIDを変数に出力しておく。

```
export PROJECT_ID="<GCPプロジェクトID>"
```

ドメイン・プレフィックス名の決定

今回構築するシステムで使用したいドメインを変数に出力しておく。後のステップでドメインのDNS設定を変更するため、あらかじめドメインの確保を済ませておくこと。

例：plateauview.example.com

```
export DOMAIN="<ドメイン>"
```

次に今回構築するシステムで使用するプレフィックス名を決め、変数に出力しておく。

プレフィックス名とは、GCP内にさまざまなリソースを作成する時の名称の先頭に付加される任意の文字列である。20文字以内で半角英数ハイフンが使用可能。例えばステージング環境と本番環境で変えるなどの用途がある。管理者にしか見えないので自由に決めて良く、GCPプロジェクトIDと同じにしても問題ない。

例：plateauview-test

```
export SERVICE_PREFIX="<プレフィックス名>"
```

（2）gcloud コマンド

以下の説明に従ってgcloudコマンドのインストールを行う。

https://cloud.google.com/sdk/docs/install?hl=ja

インストール後、ターミナル上でgcloudコマンドを使用してGoogleアカウントでログインを行い、初期設定を済ませる。

```
gcloud auth login --update-adc
```

ブラウザが開くのでGoogleアカウントでログインする。 ログインが完了したあとに、以下のコマンドを実行してプロジェクトを設定する。

```
gcloud config set project <プロジェクトID>
```

（3）Terraform コマンド

以下の手順に従い、Terraformコマンドをインストールする。

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

（4）Terraform変数ファイルの用意

最初に、terraform.tfvars.exampleをコピーする。ここでは、terraform.tfvarsと命名したが、拡張子がtfvarsであれば何でも構わない。

```
cp terraform.tfvars.example terraform.tfvars
```

（５）GCPプロジェクトおよびGCSバケットの作成

GCPコンソールからGCPプロジェクトを作成する。 その後に、Terraformのバックエンドに使用するために、GCSバケットを作成する。作成したバケットのストレージクラスおよびロケーションをgoogle_storage_bucket.tfに設定する。

以下の例では、ストレージクラスSTANDARDおよびロケーションASIAに設定している。

```
resource "google_storage_bucket" "terraform" {
  location      = "ASIA"
  name          = var.gcs_bucket
  storage_class = "<STORAGE_CLASS>"
  storage_class = "STANDARD"
}
```

また、作成したバケットの名前をterraform.tfのbackendのbucketに設定する。

```
terraform {
  backend "gcs" {
  bucket = "<作成したバケット名>"
   }
  required_providers {
  ...
}
```

そして、作成したGCSバケットを取り込む。

```
# 初回一回のみ
$ terraform init

# APIの有効化
$ terraform import google_storage_bucket.terraform <バケット名>
```

（6）Terraform変数変数の設定

これまで構築してきたGCP、MongoDBおよびAuth0などの情報をterraform.tfvarsに設定する。

（7）GCP APIの有効化

ホスティングを行う前に、以下のAPIを有効化する。

```
# APIの有効化
$ terraform apply --target google_project_service.project
```

実行の承認を求められるため、yesを入力する（以降のterraform applyの実行でも同様にする）

（8）Cloud DNSマネージドゾーンの作成およびドメイン解決の委譲

以下のコマンドでCloud DNSマネージドゾーンを作成する。

```
terraform apply --target google_dns_managed_zone.zone
```

GCPコンソール上で、作成されたリソースを確認することができる。マネージドゾーン名を取得し、以下のコマンドを実行してNSレコードを取得する。

```
gcloud dns record-sets list --zone <マネージドゾーン名> --format='value（nameServers）' --flatten 'nameServers'
```

出力されたNSレコードを、ドメインのレジストラで、ドメインのネームサーバーとして設定する。設定方法は各レジストラによって異なるため、レジストラのドキュメントを参照すること。

（9）Terraformの実行

再度、すべてのリソースを作成するために以下のコマンドを実行する。

```
terraform apply
```

実行が成功すると、以下のような出力が表示される。

```
$ terraform apply
...
plateauview_cms_url = "*"
plateauview_cms_webhook_secret = <sensitive>
plateauview_cms_webhook_url = "*"
plateauview_geo_url = "*"
plateauview_reearth_url = "*"
plateauview_sdk_token = <sensitive>
plateauview_sidebar_token = <sensitive>
plateauview_sidecar_url = "*"
plateauview_tiles_url = "*"
```

これらの出力は、あとでログインする時に使用する。なお、もう一度表示したいときはterraform outputコマンドで表示できる。また、sensitiveと表示されているものは、マスクされており、以下のようなコマンドで実際の値を確認すること。

```
terraform output <確認したいOutput>
```

変数	説明
plateauview_cms_url	PLATEAU CMSのURL。
plateauview_cms_webhook_secret	後述の「CMS インテグレーション設定」で使用。
plateauview_cms_webhook_url	後述の「CMS インテグレーション設定」で使用。
plateauview_geo_url	タイルなどを変換・処理するサーバーのURL。
plateauview_reearth_url	PLATEAU EditorのURL。
plateauview_sdk_token	PLATEAU SDK for Unity/Unreal用のトークン。SDKのUIで設定する。
plateauview_sidebar_token	ビューワのサイドバー用のAPIトークン。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_sidecar_url	サイドカーサーバーのURL。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_tiles_url	タイル配信サーバーのURL

（１０）DNS・ロードバランサ・証明書のデプロイ完了の確認

実際にcurlコマンドなどでリクエストを送って、デプロイが完了していることを確認する。

```
curl https://api.${DOMAIN}/ping
```

先ほど作成したAuth0テナントにユーザーを作成する。 その後に、届くメールでメールアドレスを認証するか、メールアドレス認証のステータスをアカウント詳細画面からVerifiedにする。必ず上記ステップでデプロイが完了していることを確認してから、Auth0のユーザーを作成する。先に作成した場合、正常にEditorやCMSにログインできなくなる。

（１１）CMSインテグレーション設定

Terraformのの plateauview_cms_url のURL（https://reearth.${DOMAIN}）からPLATEAU CMSにログインする。ログイン後、ワークスペース・Myインテグレーションを作成する。

次に、インテグレーション内に以下のとおり Webhook を作成する。作成後、有効化を忘れないこと。

- URL: terraform outputsのplateauview_cms_webhook_url
- シークレット: terraform outputsのplateauview_cms_webhook_secret
- イベント: 全てのチェックボックスにチェックを入れる。

作成後、作成したワークスペースに作成したインテグレーションを追加し、オーナー権限に変更する。

先ほど作成したインテグレーションの詳細画面でインテグレーショントークンをコピーし、以下の ${REEARTH_PLATEAUVIEW_CMS_TOKEN} に貼り付けて以下のコマンドを実行する。

```
echo -n "${REEARTH_PLATEAUVIEW_CMS_TOKEN}" | gcloud secrets versions add reearth-cms-REEARTH_PLATEAUVIEW_CMS_TOKEN --data-file=-
```

環境変数の変更を適用するため、もう一度 Cloud Run をデプロイする。

```
gcloud run deploy plateauview-api ¥
  --image eukarya/plateauview2-sidecar:latest ¥
  --region asia-northeast1 ¥
  --platform managed ¥
  --quiet
```

（１２）完了

以下のアプリケーションにログインし、正常に使用できることを確認する。ここの ${DOMAIN} はドメインである。

- Editor: Terraformのoutputsの plateauview_reearth_url の値（https://reearth.${DOMAIN}）
- CMS: Terraformのoutputsの plateauview_cms_url の値（https://cms.${DOMAIN}）

サーバー構成は以上で完了である。

3.1.4 Amazon Web Services向けセットアップ

次に、AWS（Amazon Web Services）を利用する場合のセットアップ手順を記載する。GCPを利用して構築する場合は、3.1.3及び次章の3.1.5を参照すること。

（1）事前準備

このマニュアルに従ってシステムを構築するためには、マニュアルの以下のツールが必要である。事前にインストールしておくこと。

- aws-cli/2.13.15: 検証済み
- Terraform/v1.7.4: 検証済み

（2）Terraform変数ファイルの用意

最初に、terraform.tfvars.exampleをコピーする。ここでは、terraform.tfvarsと命名したが、拡張子がtfvarsであれば何でも構わない。

```
cp terraform.tfvars.example terraform.tfvars
```

（3）aws-cliのセットアップ

aws-cli を使用して、AWS環境にアクセスできるようする。aws-cli のセットアップは利用している環境に合わせること。

以下のcomandを実行し、出力される情報から作成予定のAWSアカウントかどうか確認する。

```
aws sts get-caller-identity
```

```
{
    "UserId": "xxxxxxxxxxxxxxx:example@example.com",
    "Account": "00123456789",
    "Arn": "arn:aws:sts::00123456789:assumed-
role/**************/example@example.com"
}
```

（4）s3バケットの作成

TerraformのバックエンドでS3使用するためのS3バケットを作成する。

```
##初期セットアップ
$　terraform init

$　terraform apply -target=module.tfstate
...
tfstate_bucket_name="${作成されたバケット名}"
```

その後、Terraformバックエンドの設定をを行う。 terraform.tfのbackendのbucketに設定する。

```
terraform {
 backend "s3" {
 bucket = "${作成されたバケット名}"
 key = "terraform.tfstate"
 region = "使用しているリージョン"
 }

 required_providers {
 ...
}
```

その後、もう一度terraform initを行い、terraform.tfstateをS3にアップロードする。

```
$ terraform init
```

（5）Terraform変数の設定

これまで構築してきたAWS、MongoDBなどの情報をterraform.tfvarsに設定する。

（6）Route53 パブリックゾーンの作成およびゾーンの委譲

以下のコマンドを実行し、Route53パブリックゾーンを作成する。

```
terraform apply --target aws_route53_zone.public_zone
```

マネージドゾーン名を取得し、以下のコマンドを実行してNSレコードを取得する。

```
$ aws route53 list-resource-record-sets --hosted-zone-id
/hostedzone/Z01251093SKX99FOVRAZN --query "ResourceRecordSets[?Type ==
'NS'].ResourceRecords[*].Value"
```

出力されたNSレコードを、ドメインのレジストラで、ドメインのネームサーバーとして設定する。設定方法は各レジストラによって異なるため、レジストラのドキュメントを参照すること。

（7）ECRのセットアップ

以下のコマンドを実行し、ECRを作成する。

```
terraform apply -target module.reearth_ecr -target module.reearth_cmsecr
```

ECRを作成後、dockerhubより各イメージをpullし、ECRにpushする。

```
export AWS_ACCOUNT_ID="AWSアカウントID"
export AWS_REGION="作成対象のリージョン"
#ECRへログイン
$ aws ecr get-login-password  | docker login --username AWS --password-stdin
${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
#dockerhubからイメージを取得
$ docker pull eukarya/plateauview2-reearth:latest
$ docker tag eukarya/plateauview2-reearth:latest
${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/reearth-api
$ docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/reearth-api
$ docker pull eukarya/plateauview-geo:latest
$ docker tag eukarya/plateauview-geo:latest
${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/plateauview-api
$ docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/plateauview-
api
$ docker pull eukarya/plateauview-geo:latest
$ docker tag eukarya/plateauview-
geo:latest${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/plateauview-geo
$ docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/plateauview-
geo
$ docker pul eukarya/plateauview2-reearth-cms:latest
$ docker tag eukarya/plateauview2-reearth-cms:latest
${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/reearth-cms
$ docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/reearth-cms
```

（8）関連リソースの作成

以下のコマンドを実行し、関連リソースを作成する。

```
terraform apply -target module.reearth -target module.reearth_cms module.cognito
```

（9）ドメインのセットアップ

以下のコマンドを実行し、残りのサービスを作成する。

```
terraform apply -target module.reearth_custom_domain -target
module.reearth_cms_custom_domain
```

再度、すべてのリソースを作成するために以下のコマンドを実行する。

```
terraform apply
```

実行が成功すると、以下のような出力が表示される。

```
$ terraform apply
...
plateauview_cms_url = "*"
plateauview_cms_webhook_secret = <sensitive>
plateauview_cms_webhook_url = "*"
plateauview_geo_url = "*"
plateauview_reearth_url = "*"
plateauview_sdk_token = <sensitive>
plateauview_sidebar_token = <sensitive>
plateauview_sidecar_url = "*"
plateauview_tiles_url = "*"
```

これらの出力は、あとでログインするときに使用する。なお、もう一度表示したいときはterraform outputコマンドで表示することができる。また、sensitiveと表示されているものは、マスクされており、以下のようなコマンドで実際の値を確認すること。

```
terraform output <確認したいOutput>
```

表　変数一覧

変数	説明
plateauview_cms_url	PLATEAU CMSのURL。
plateauview_cms_webhook_secret	下記「CMS インテグレーション設定」で使用。
plateauview_cms_webhook_url	下記「CMS インテグレーション設定」で使用。
plateauview_geo_url	タイルなどを変換・処理するサーバーのURL。
plateauview_reearth_url	PLATEAU EditorのURL。
plateauview_sdk_token	PLATEAU SDK for Unity/Unreal用のトークン。SDKのUIで設定する。
plateauview_sidebar_token	ビューワのサイドバー用のAPIトークン。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_sidecar_url	サイドカーサーバーのURL。エディタ上でサイドバーウィジェットの設定から設定する。
plateauview_tiles_url	タイル配信サーバーのURL。

（１０）DNS・ロードバランサ・証明書のデプロイ完了の確認

実際にcurlコマンドなどでリクエストを送って、デプロイが完了していることを確認する。

```
curl https://api.${DOMAIN}/ping
```

（１１）CMSインテグレーション設定

Terraformのの plateauview_cms_url のURL（https://reearth.${DOMAIN}）からPLATEAU CMSにログインする。ログイン後、ワークスペース・Myインテグレーションを作成する。

次に、インテグレーション内に以下のとおり Webhook を作成する。作成後、有効化を忘れないこと。

- URL: terraform outputsのplateauview_cms_webhook_url
- シークレット: terraform outputsのplateauview_cms_webhook_secret
- イベント: 全てのチェックボックスにチェックを入れる。

作成後、作成したワークスペースに作成したインテグレーションを追加し、オーナー権限に変更する。

先ほど作成したインテグレーションの詳細画面でインテグレーショントークンをコピーし、以下の ${REEARTH_PLATEAUVIEW_CMS_TOKEN} に貼り付けて以下のコマンドを実行する。

```
aws ssm put-parameter --name "/reearth-cms/REEARTH_PLATEAUVIEW_CMS_TOKEN" -type "String" --value "${REEARTH_PLATEAUVIEW_CMS_TOKEN}"
```

環境変数の変更を適用するため、もう一度App Runnerをデプロイする。

```
## reearth-cmsのServiceArnを取得
export REEARTH_CMS_SERVICE_ARN=`aws apprunner list-services --region us-east-1 --query "ServiceSummaryList[?ServiceName == 'reearth-cms-server'].ServiceArn"`

## AppRunnerの更新
aws apprunner  start-deployment --service-arn ${REEARTH_CMS_SERVICE_ARN}
```

（１２）ユーザー作成

Cogito上で新規にユーザーを作成する。 AWSのマネジメントコンソールより Cognito->ユーザープール->ユーザー->ユーザーを作成を選択し、ユーザーを作成する。その後、作例された情報を元に、以下のパラメーターを埋め、実行する

```
curl -H 'Content-Type: application/json' https://reearth.${DOMAIN}/api/signup -d @- << EOF
{
  "sub": "YOUR AUTH0 ACCOUNT ID",
  "email": "YOUR AUTH0 ACCOUNT EMAIL",
  "username": "YOUR AUTH0 ACCOUNT NAME",
}
```

（１３）完了

以下のアプリケーションにログインし、正常に使用できることを確認する。ここの ${DOMAIN} はドメインである。

- Editor: Terraformのoutputsの plateauview_reearth_url の値（https://reearth.${DOMAIN}）
- CMS: Terraformのoutputsの plateauview_cms_url の値（https://cms.${DOMAIN}）

サーバー構成は以上で完了である。

3.1.5 PLATEAU CMSの動作確認

以下のURLにログインし、ログイン画面が表示されることを確認する。<ドメイン>は先ほど決めたドメインに置き換えること。HTTPではなくHTTPSであることに注意。

https://cms.<ドメイン>

例：ドメインが plateau-test.example.com の場合、https://cms.plateau-test.example.com

PLATEAU CMSへのログインを行い、ログイン後PLATEAU CMSの画面へ無事遷移すれば、PLATEAU CMSは正常に動作している。

ログインできない場合は、各種デプロイ完了後にAuth0にユーザーを作成済みであること、ユーザーのメールアドレスの認証が済んでいることを確認する。なお、ログインしてもすぐにログイン画面に戻される場合は、Auth0にユーザーは存在するが、PLATEAU Editor及びPLATEAU CMSのデータベース内にユーザーを作成する処理に失敗していることが考えられる。

次に、PLATEAU CMS連携機能を担うサイドカーサーバーが正しく動作しているか確認する。以下のURLにアクセスし、 `***pong***` と表示されたら成功である。

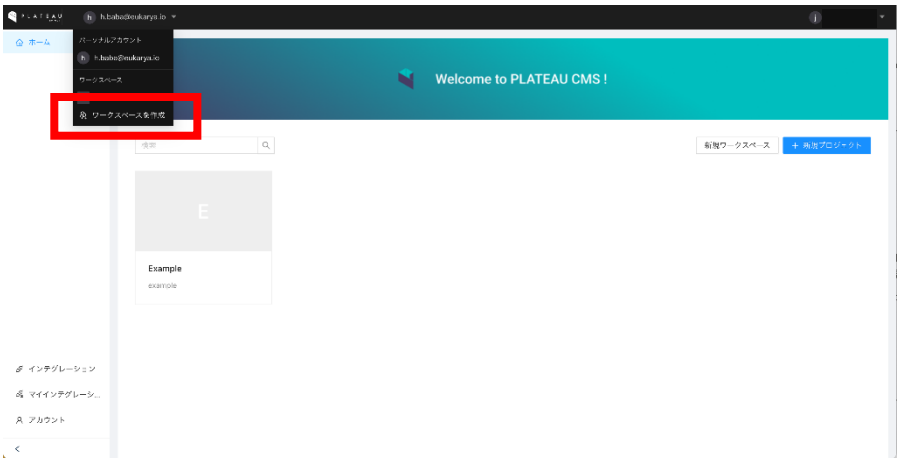
https://api.<ドメイン>/ping

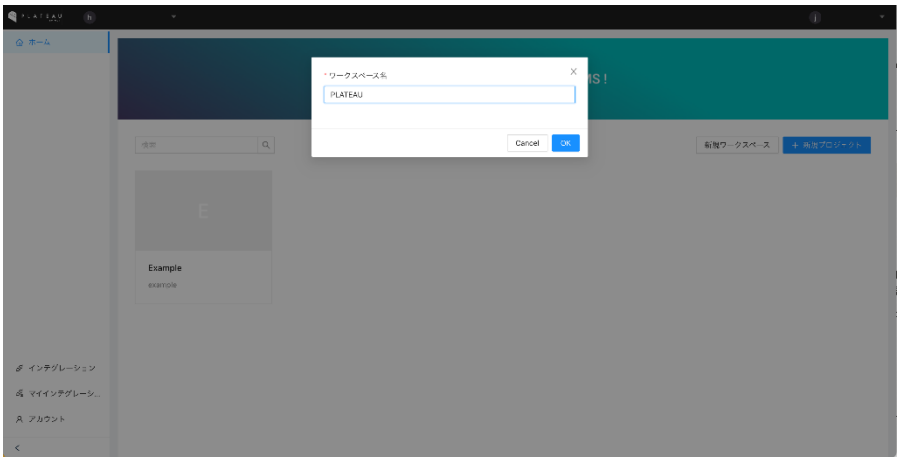
3.1.6 PLATEAU CMSのセットアップ

PLATEAU CMS内でのプロジェクトの初期設定及びインテグレーションの設定に移る。

（1）ワークスペースの作成

- 個人ワークスペースでは他のユーザーを招待できないため、新たにワークスペースを作成する。
- PLATEAU CMSにログインし、ホーム画面に遷移
- 新規ワークスペース作成ボタンを押下
  - ヘッダー > ワークスペースを作成
  - ホーム > 新規ワークスペース
  - ワークスペース名を入力



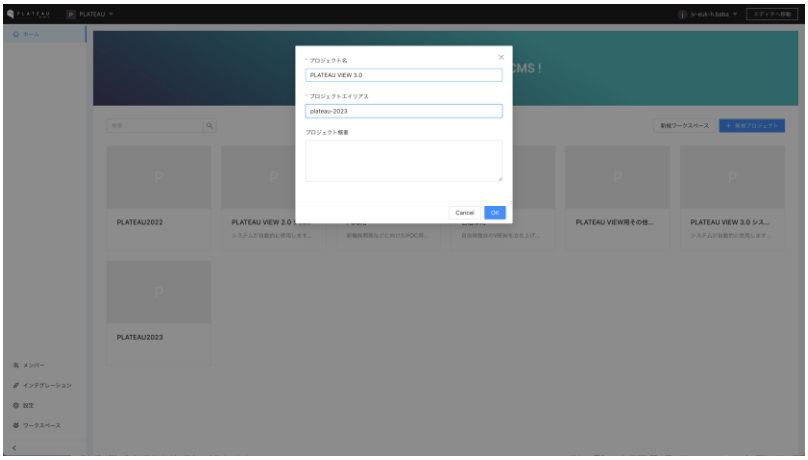


（2）プロジェクトの作成

2つのプロジェクトを作成する。

1つ目は、PLATEAU関連データセットを管理するためのプロジェクト（プロジェクトエイリアス：plateau-2023）であり、もう1つ目は、インテグレーションが自動的に動作する際に内部的に情報を保存するためのプロジェクト（プロジェクトエイリアス：plateauview）。

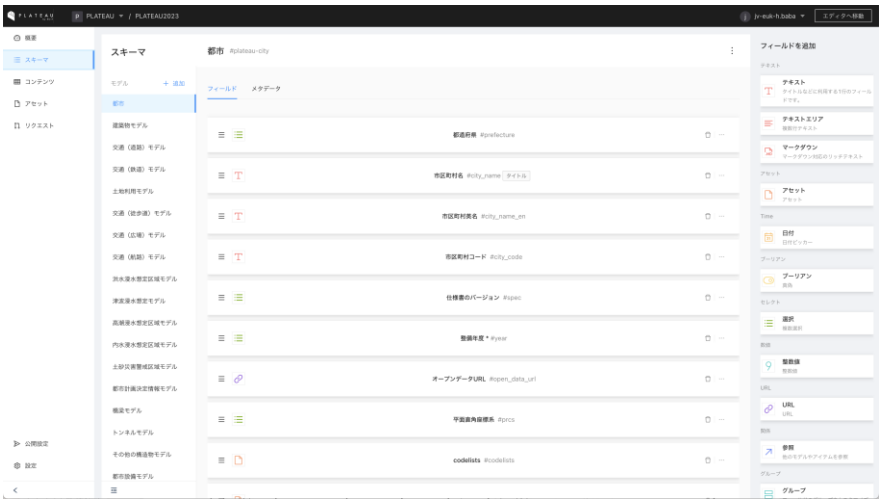
なお、プロジェクトのエイリアスとは、PLATEAU CMSから公開されるAPIの別名で、公開APIのURLの中で使われる。必須項目であり、5文字以上32文字以下の半角英数字と一部記号が使用可能（_又は-）。他プロジェクトが同じエイリアスを使うとエラーとなる。



（3）モデルの作成

1つ目の3D都市モデルデータ用プロジェクトに、以下のとおり都市と地物のスキーマを作成する。同一の都市を複数登録するとEditor/Viewer で確認できないので注意すること。

- 都市 （キー：plateau-city）
- データスキーマ
- メタデータスキーマ



以下にそれぞれのモデルに設定すべきスキーマを一覧化する。記載方法は以下のとおり。

- 名前：フィールドの表示名で、利用上理解しやすい名前を設定する。
- グループフィールド：フィールドをグループ化する際の項目で、利用する場合にはグループフィールドのキーを記載する。
- キー：システムがフィールドを一意に判別するためのキーである。同一モデル内で一意である必要がある。英数字１文字以上で設定をする。
- 型：フィールドのデータ型を設定する。「[]」の表記があるものは、複数値設定できる方を表す。
- 備考：フィールドに関する備考。

表　都市: データスキーマ一覧（1/2）

名前	グループフィールド	キー	型	備考
都道府県		prefecture	Select	選択肢は「東京都」など
市区町村名		city_name	Text	「横浜市」「茂原市」など
市区町村英名		city_name_en	Text	「kyoto-shi」など
市区町村コード		city_code	Text	
仕様書のバージョン		spec	Select	選択肢は「第3.2版」「第2.3版」「第3.0版」「第3.1版」「第3.3版」など
整備年度		year	Select	選択肢は「2023」「2022」「2021 」「2020」など
オープンデータURL		open_data_url	URL	
平面直角座標系		prcs	Select	選択肢は「1」「2」など
codelists		codelists	Asset	Zip形式でコードリストのフォルダーをアップロード
schemas		schemas	Asset	Zip形式でschemasフォルダーをアップロード
metadata		metadata	Asset	Zip形式でmetadataフォルダーをアップロード
specification		specification	Asset	Zip形式でspecificationフォルダーをアップロード
misc		misc	Asset	
建築物モデル		bldg	Reference（bldg）	
交通（道路）モデル		tran	Reference（tran）	
交通（鉄道）モデル		rwy	Reference（rwy）	
交通（徒歩道）モデル		trk	Reference（trk）	
交通（広場）モデル		squr	Reference（squr）	
交通（航路）モデル		wwy	Reference（wwy）	

表　都市: データスキーマ一覧（2/2）

名前	グループフィールド	キー	型	備考
土地利用モデル		luse	Reference（luse）	
洪水浸水想定区域モデル		fld	Reference（fld）	
津波浸水想定モデル		tnm	Reference（tnm）	
高潮浸水想定区域モデル		htd	Reference（htd）	
内水浸水想定区域モデル		ifld	Reference（ifld）	
土砂災害警戒区域モデル		lsld	Reference（lsld）	
都市計画決定情報モデル		urf	Reference（urf）	
橋梁モデル		brid	Reference（brid）	
トンネルモデル		tun	Reference（tun）	
その他の構造物モデル		cons	Reference（cons）	
都市設備モデル		frn	Reference（frn）	
地下埋設物モデル		unf	Reference（frn）	
地下街モデル		ubld	Reference（ubld）	
植生モデル		veg	Reference（veg）	
地形モデル		dem	Reference（dem）	
水部モデル		wtr	Reference（wtr）	
区域モデル		area	Reference（area）	
ユースケース		generic	Reference（area）	One-way reference で、それぞれ登録。1つの都市に対して複数のユースケースが登録されるため。
関連データセット		related	Reference（related）	
G空間情報センターデータ目録		geospatialjp-index	Reference（geospatialjp-index）	

表　都市: メタデータスキーマ一覧（1/2）

名前	グループフィールド	キー	型	備考
G空間情報センター登録用データ		geospatialjp-data	Reference（geospatialjp-data）	
PLATEAUデータステータス		plateau_data_status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
都市モデル公開		city_public	Boolean	
SDK公開		sdk_public	Boolean	
建築物モデル公開		bldg_public	Boolean	
交通（道路）モデル公開		tran_public	Boolean	
交通（鉄道）モデル公開		rwy_public	Boolean	
交通（徒歩道）モデル公開		trk_public	Boolean	
交通（広場）モデル公開		squr_public	Boolean	
交通（航路）モデル公開		wwy_public	Boolean	
土地利用モデル公開		luse_public	Boolean	
洪水浸水想定区域モデル公開		fld_public	Boolean	
津波浸水想定モデル公開		tnm_public	Boolean	
高潮浸水想定区域モデル公開		htd_public	Boolean	
内水浸水想定区域モデル公開		ifld_public	Boolean	
土砂災害警戒区域モデル公開		lsld_public	Boolean	
都市計画決定情報モデル公開		urf_public	Boolean	
橋梁モデル公開		brid_public	Boolean	
トンネルモデル公開		tun_public	Boolean	
その他の構造物モデル公開		cons_public	Boolean	

表　都市: メタデータスキーマ一覧（2/2）

名前	グループフィールド	キー	型	備考
都市設備モデル公開		frn_public	Boolean	
地下街モデル公開		ubld_public	Boolean	
地下埋設物モデル公開		unf_public		
植生モデル公開		veg_public	Boolean	
地形モデル公開		dem_public	Boolean	
水部モデル公開		wtr_public	Boolean	
区域モデル公開		area_public	Boolean	
汎用都市オブジェクトモデル公開		gen_public	Boolean	
関連データセット公開		related_public	Boolean	
G空間情報センターの公開準備		geospatialjp_prepare	Boolean	
G空間情報センターに公開		geospatialjp_publish	Boolean	

モデル：建築物モデル（キー：plateau-bldg）

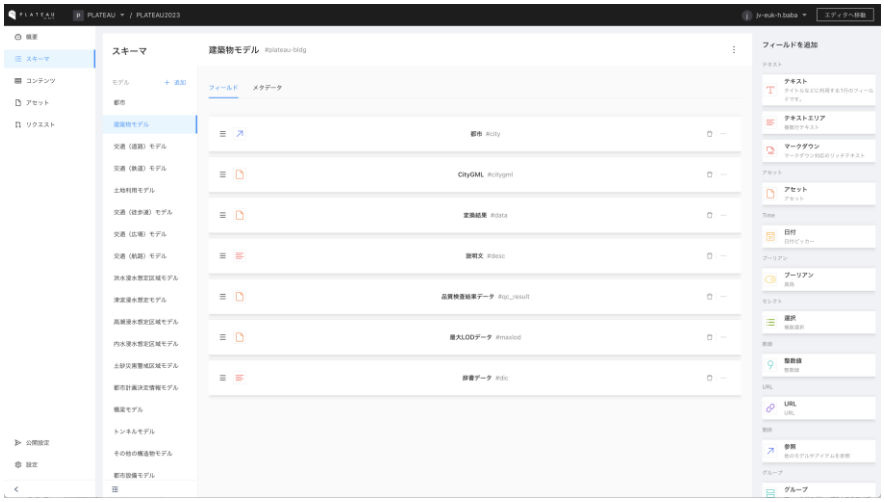


表　建築物モデル: データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

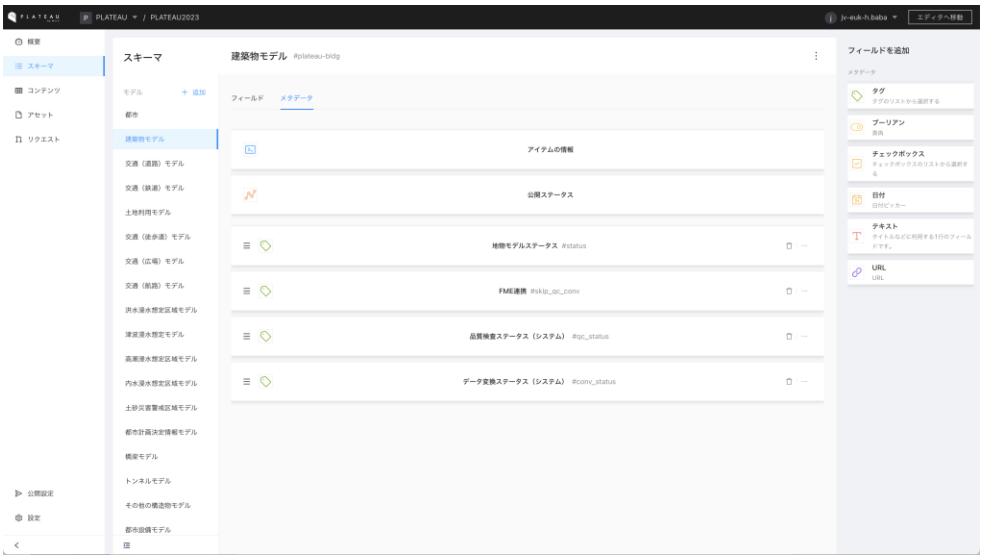


表　建築物モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：交通（道路）モデル（キー：plateau-tran）

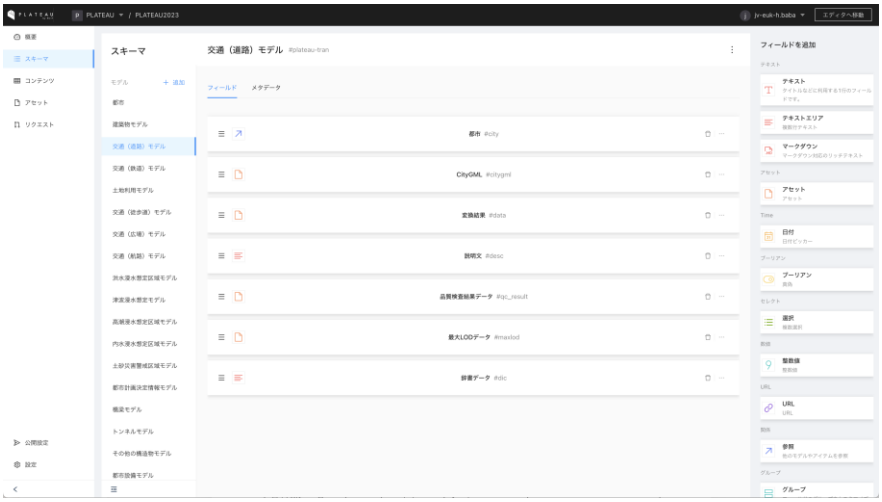


表　交通（道路）モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

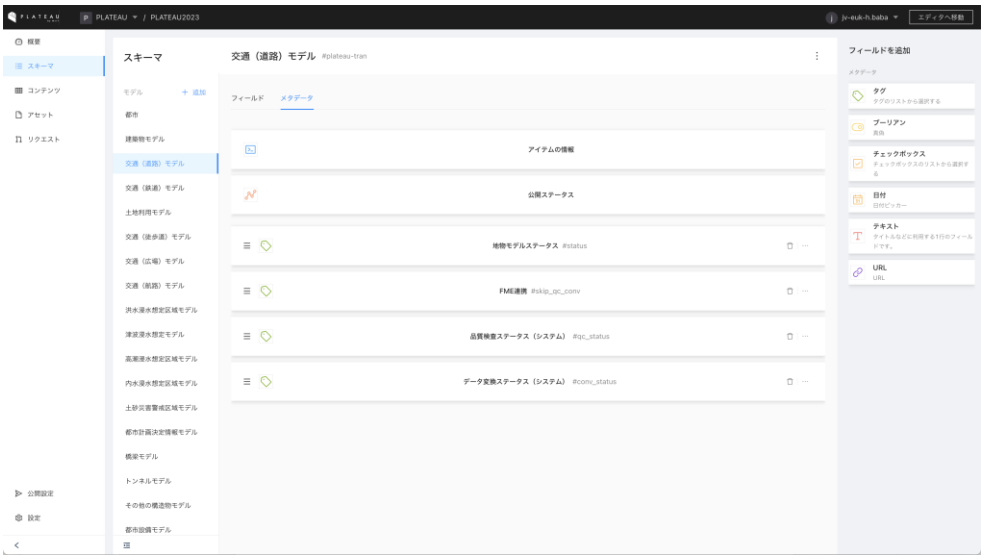


表　交通（道路）モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：交通（鉄道）モデル（キー：plateau-rwy）

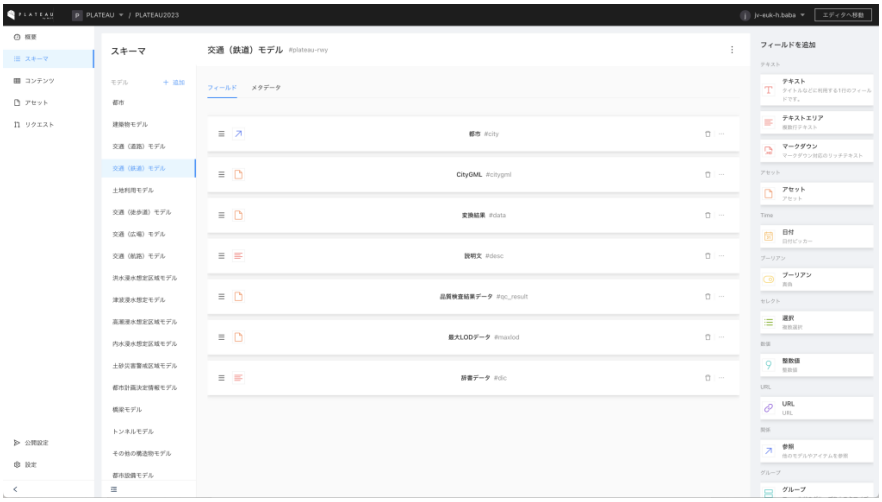


表　交通（鉄道）モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

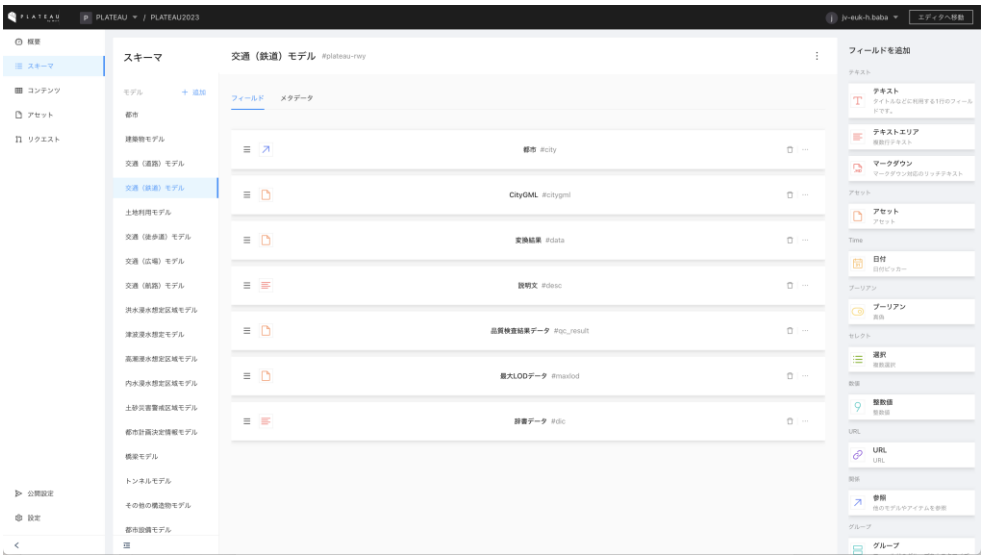


表　交通（鉄道）モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：交通（徒歩道）モデル（キー：plateau-trk）

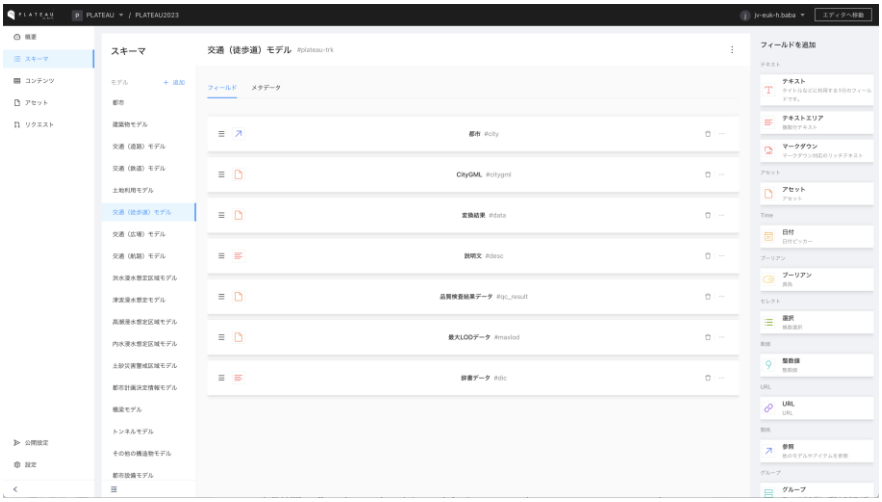


表　交通（徒歩道）モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

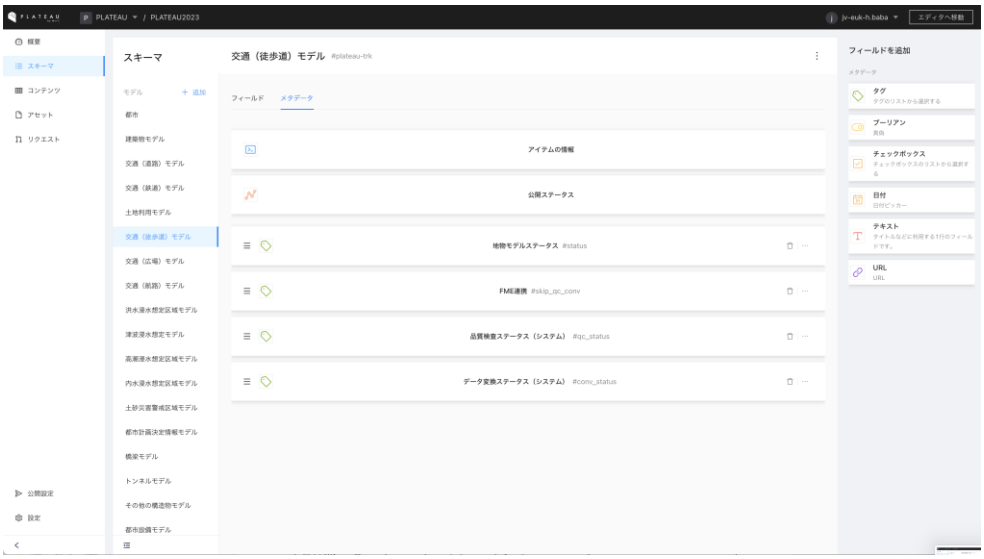


表　交通（徒歩道）モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：交通（広場）モデル（キー：plateau-squr）

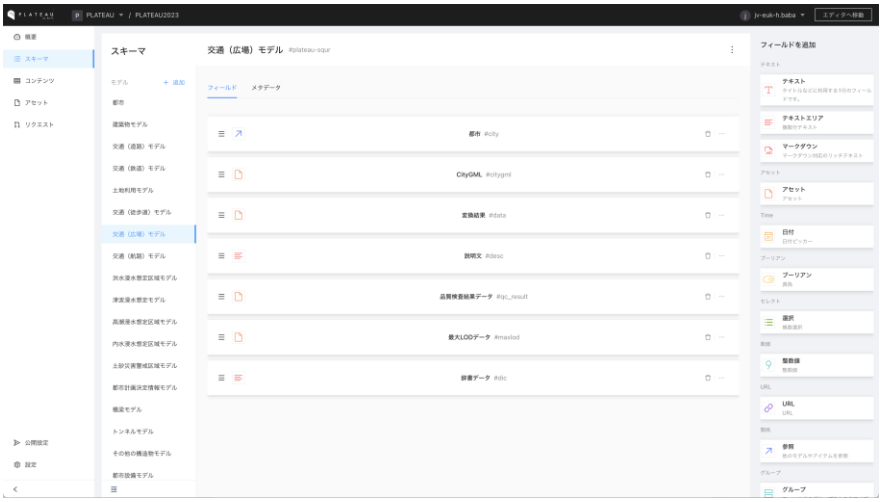


表　交通（広場）モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

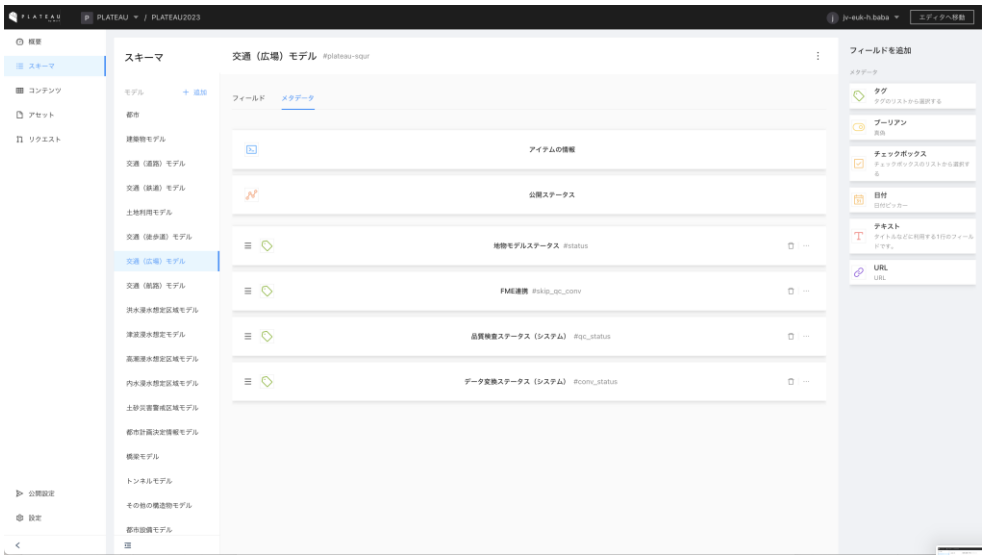


表　交通（広場）モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：交通（航路）モデル（キー：plateau-wwy）

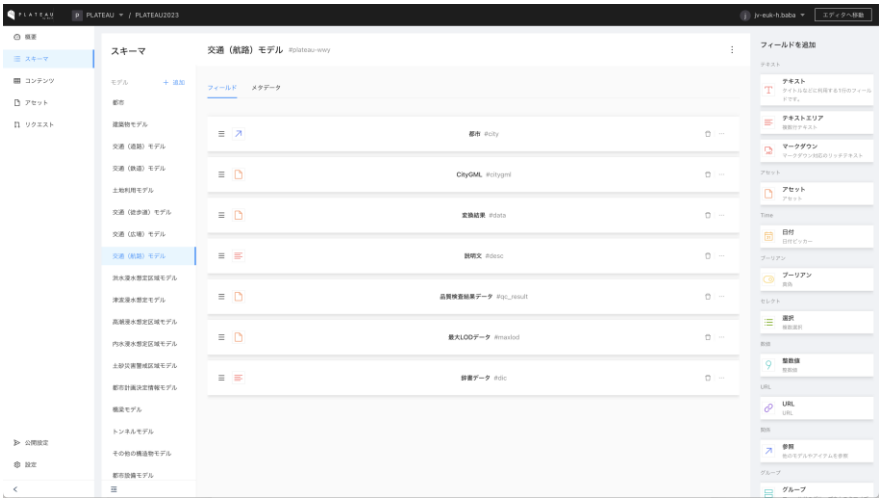


表　交通（航路）モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

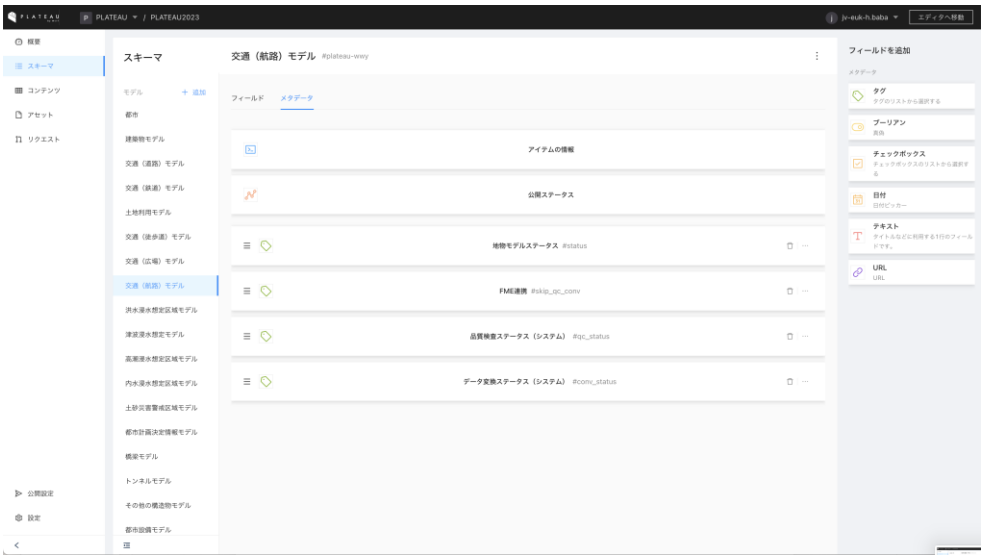


表　交通（航路）モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：土地利用モデル（キー：plateau-luse）

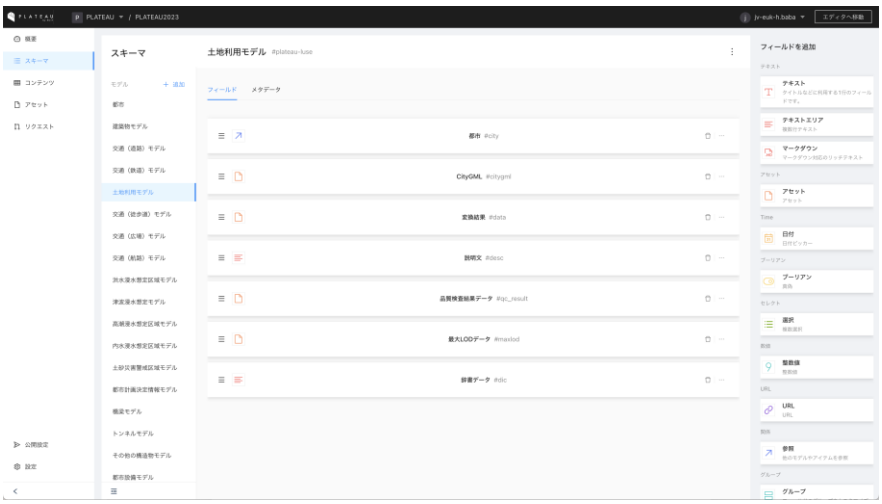


表　土地利用モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

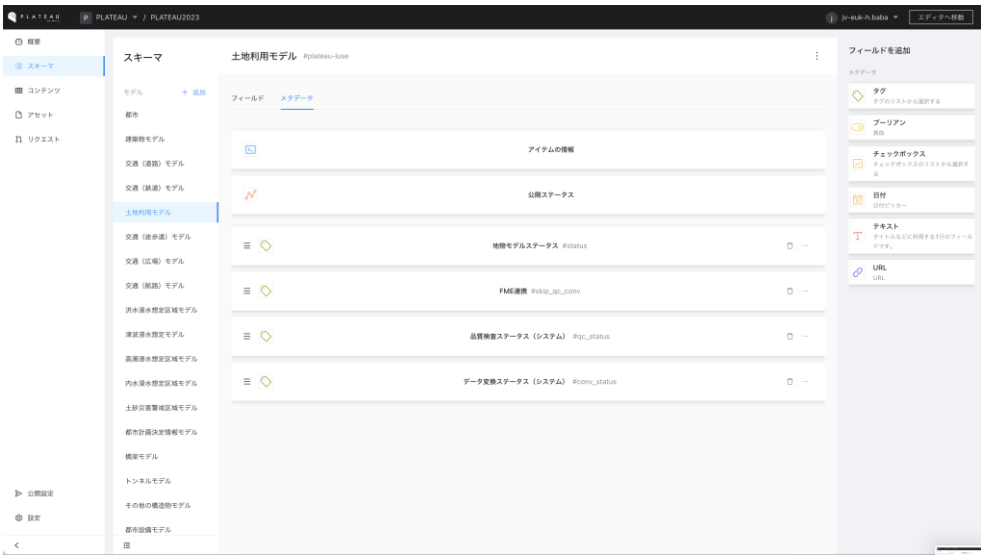


表　土地利用モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：洪水浸水想定区域モデル（キー：plateau-fld）

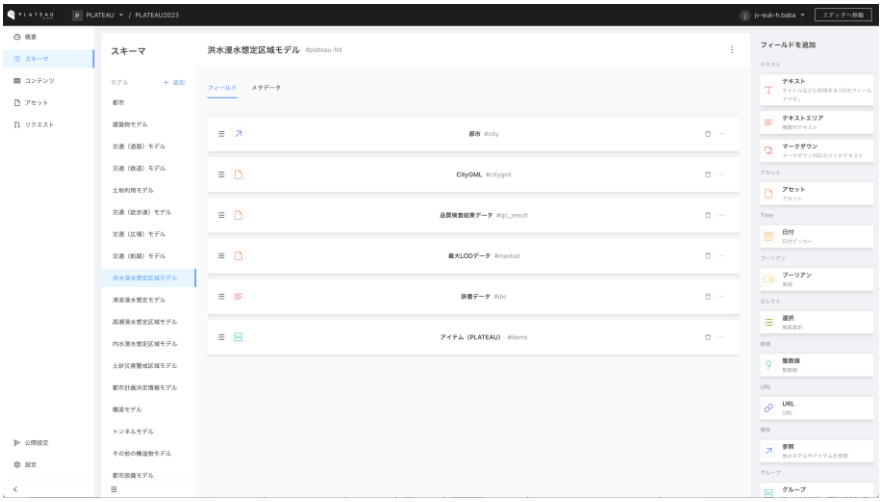


表　洪水浸水想定区域モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	
アイテム（PLATEAU）		items	Group[]	
変換結果	plateau-item-plateau	data	Asset[]	
説明文	plateau-item-plateau	desc	TextArea	

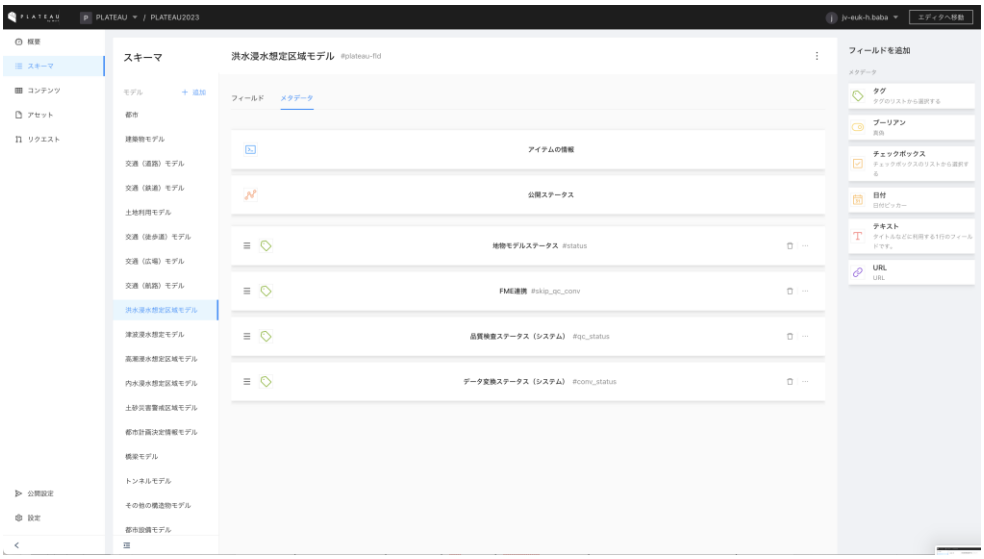


表　洪水浸水想定区域モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：津波浸水想定区域モデル（キー：plateau-tnm）

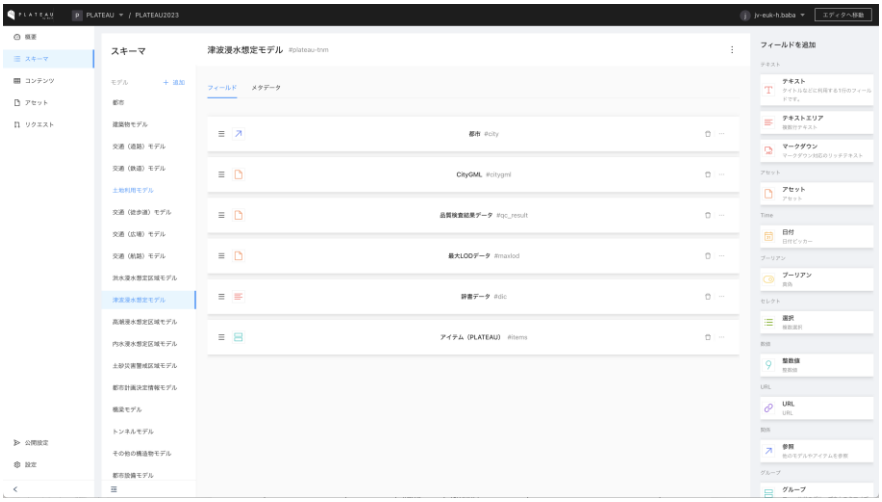


表　津波浸水想定区域モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	
アイテム（PLATEAU）		items	Group[]	
変換結果	plateau-item-plateau	data	Asset[]	
説明文	plateau-item-plateau	desc	TextArea	

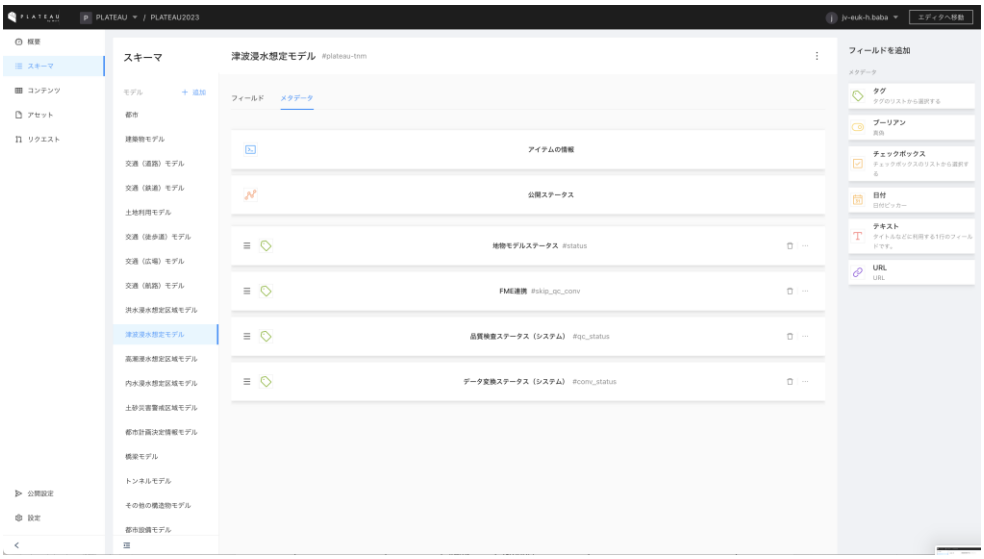


表　津波浸水想定区域モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：高潮浸水想定区域モデル（キー：plateau-htd）

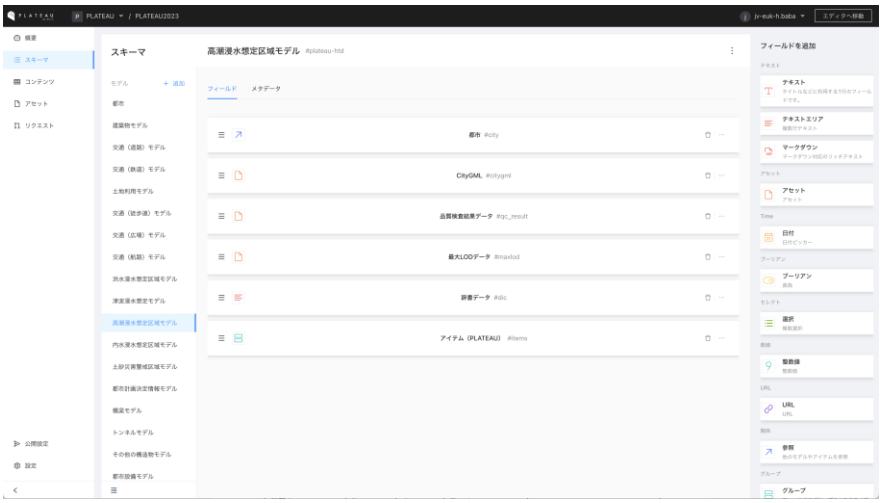


表　高潮浸水想定区域モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	
アイテム（PLATEAU）		items	Group[]	
変換結果	plateau-item-plateau	data	Asset[]	
説明文	plateau-item-plateau	desc	TextArea	

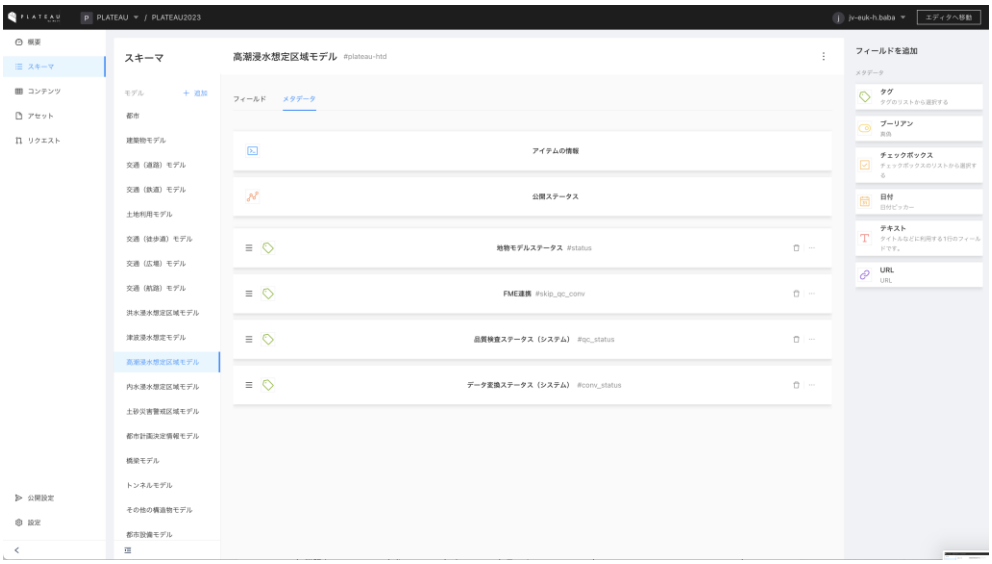


表　高潮浸水想定区域モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：内水浸水想定区域モデル（キー：plateau-ifld）

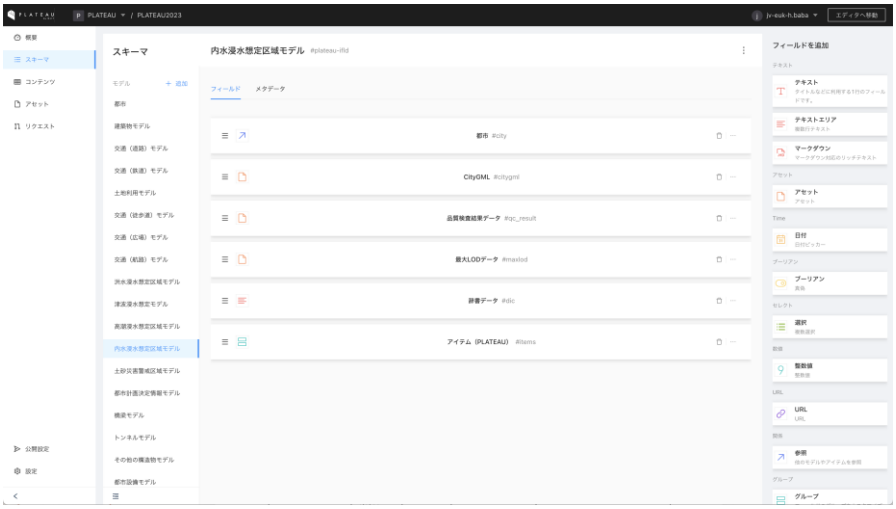


表　内水浸水想定区域モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	
アイテム（PLATEAU）		items	Group[]	
変換結果	plateau-item-plateau	data	Asset[]	
説明文	plateau-item-plateau	desc	TextArea	

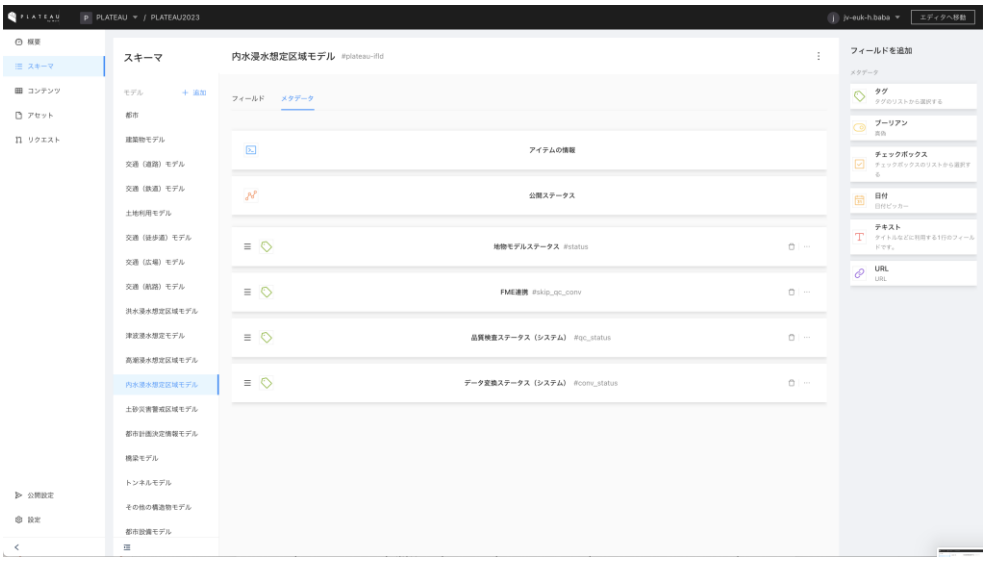


表　内水浸水想定区域モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：土砂災害警戒区域モデル（キー：plateau-lsld）

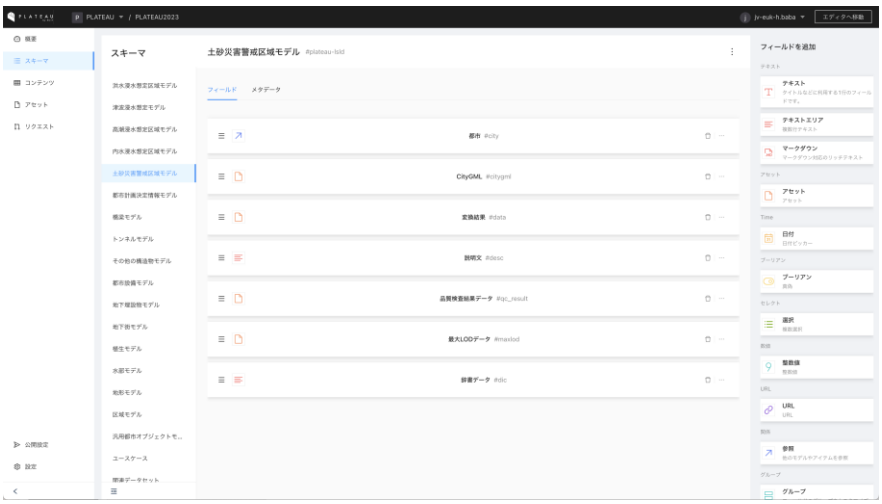


表　土砂災害警戒区域モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

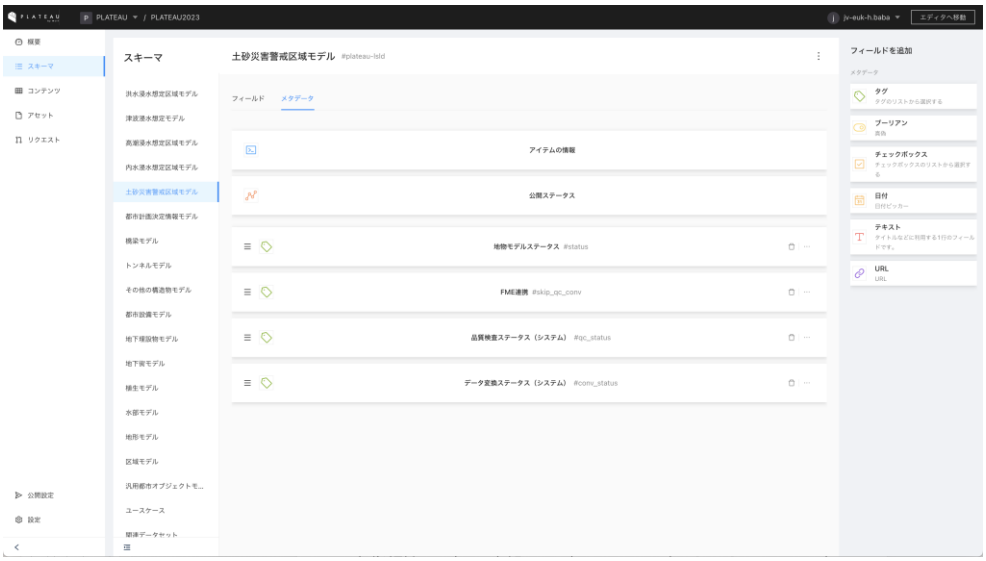


表　土砂災害警戒区域モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：都市計画決定情報モデル（キー：plateau-urf）

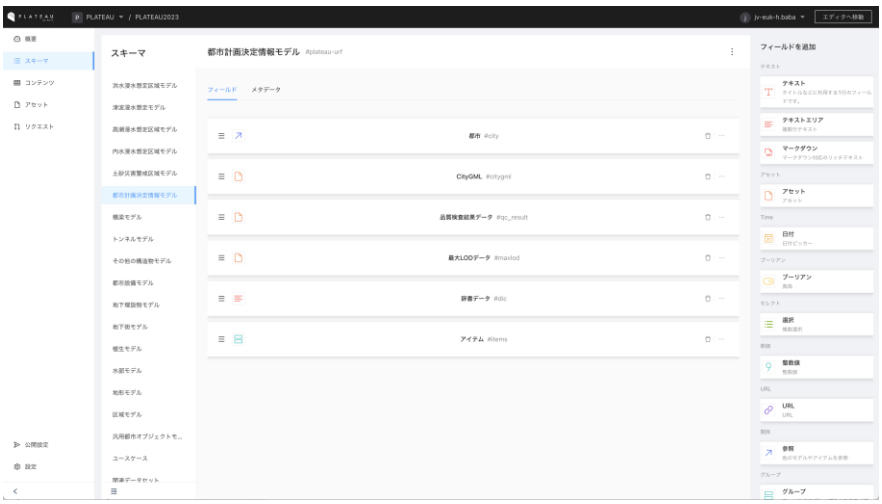


表　都市計画決定情報モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

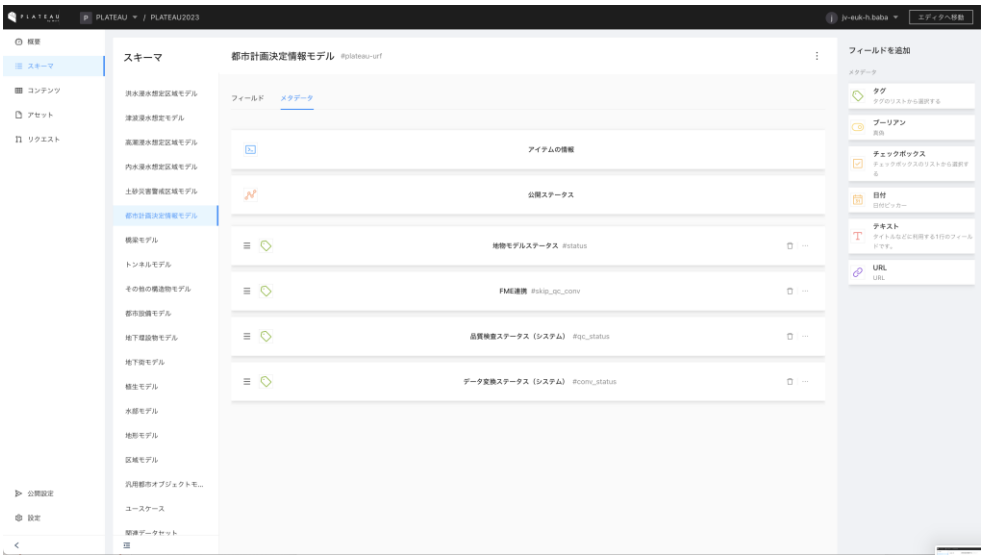


表　都市計画決定情報モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：橋梁モデル（キー：plateau-brid）

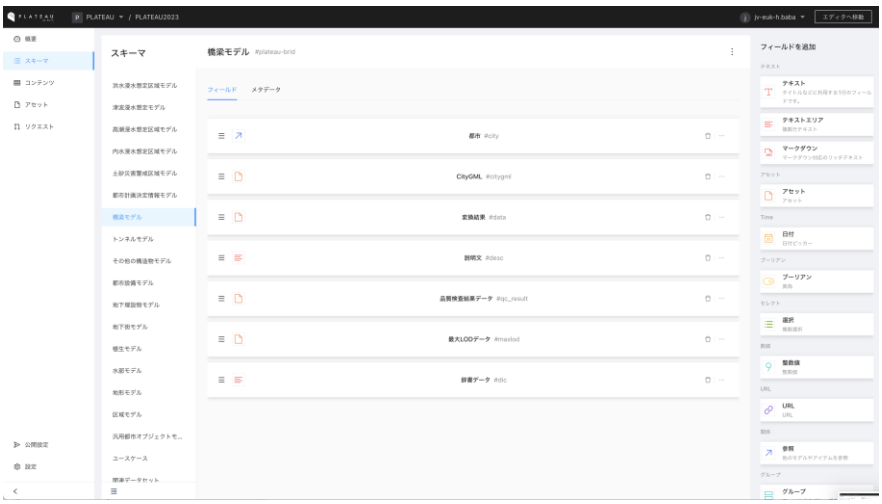


表　橋梁モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

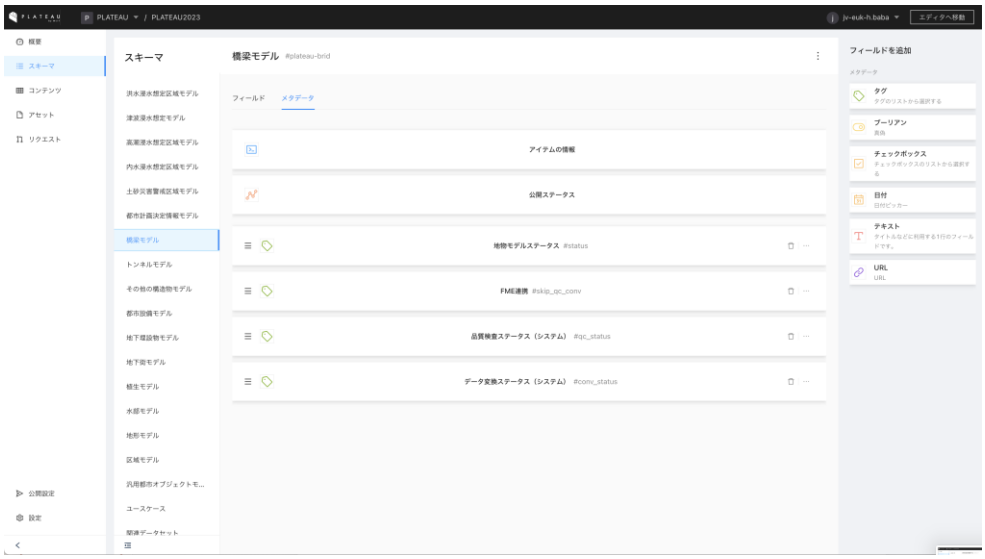


表　橋梁モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：トンネルモデル（キー：plateau-tun）

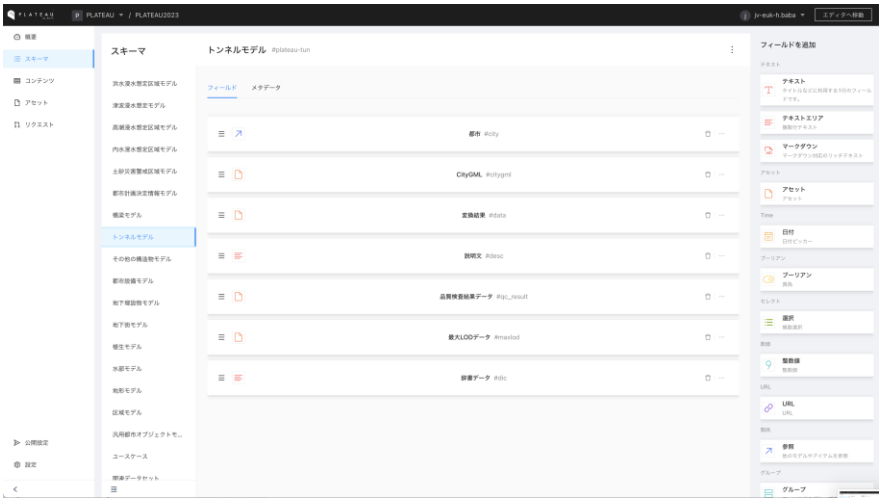


表　トンネルモデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

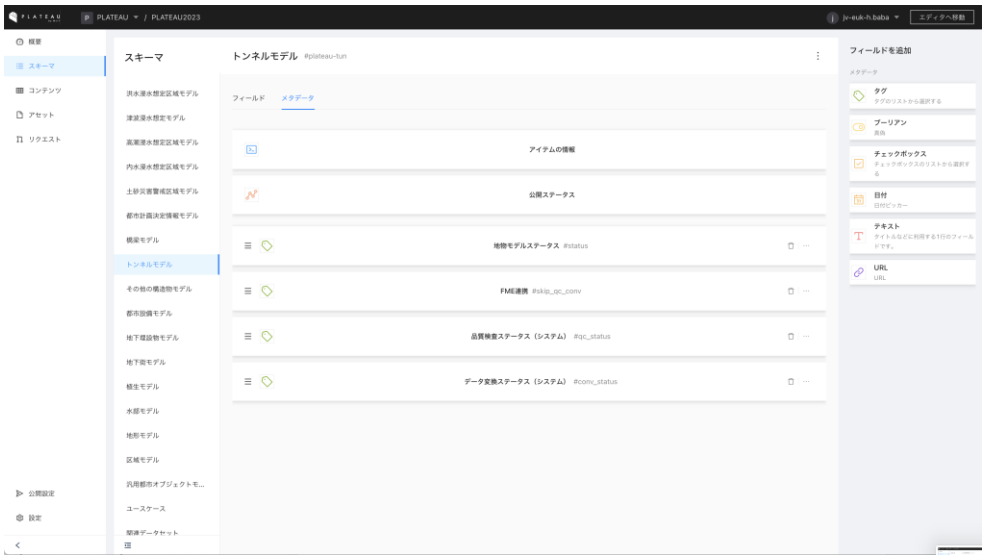


表　トンネルモデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：その他の構造物モデル（キー：plateau-cons）

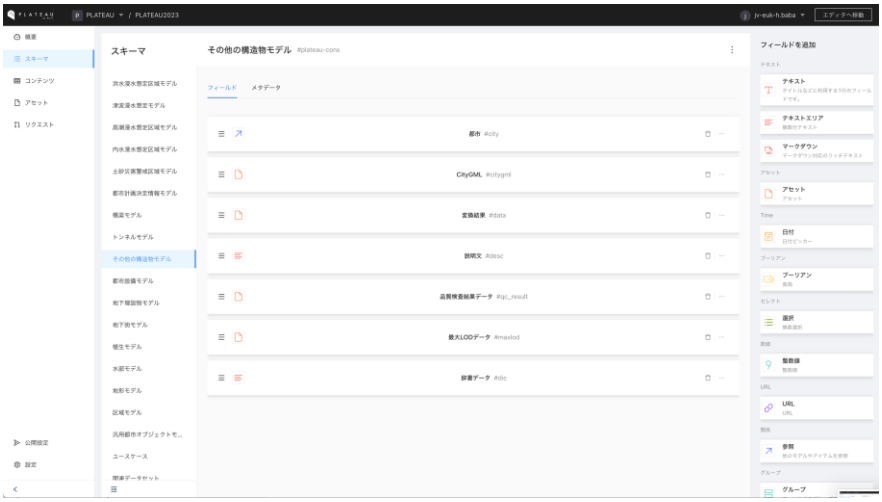


表　その他の構造物モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

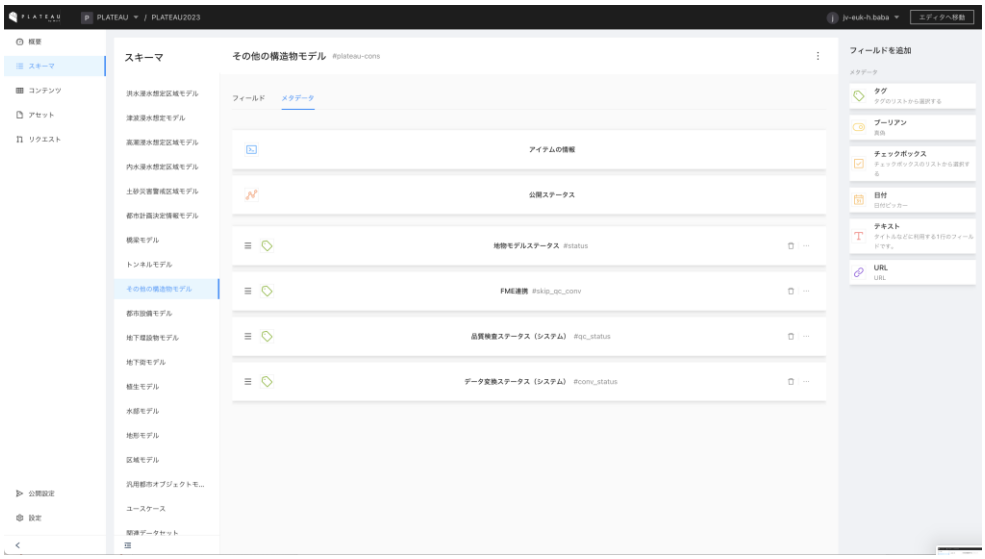


表　その他の構造物モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：都市設備モデル（キー：plateau-frn）

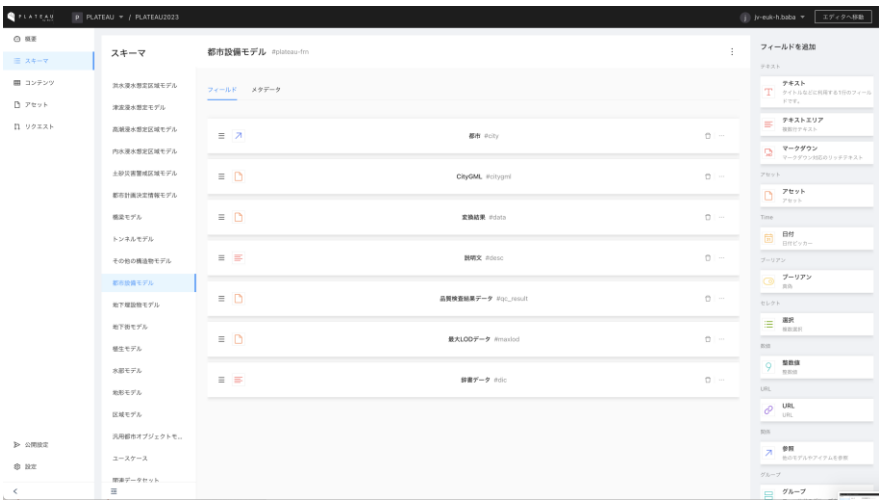


表　都市設備モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

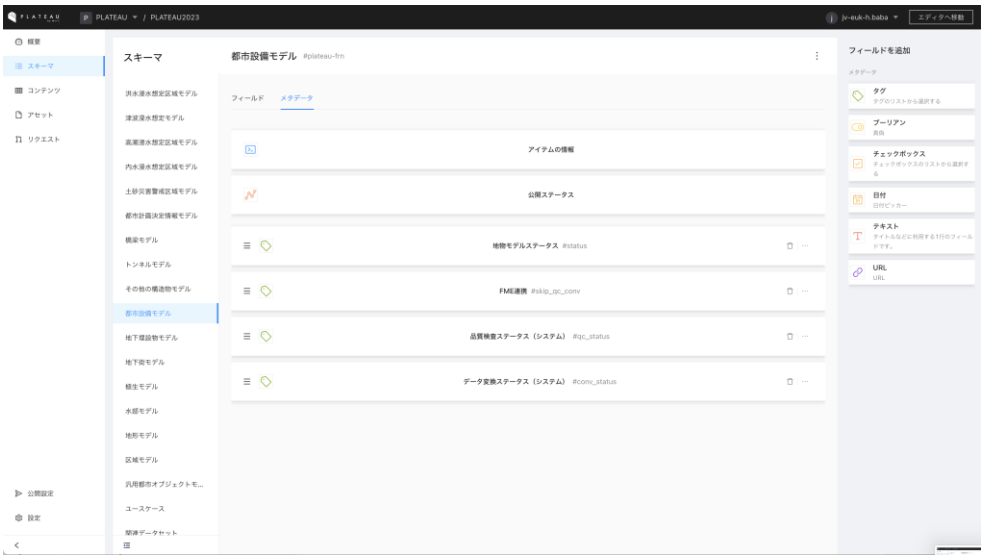


表　都市設備モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：地下埋設物モデル（キー：plateau-unf）

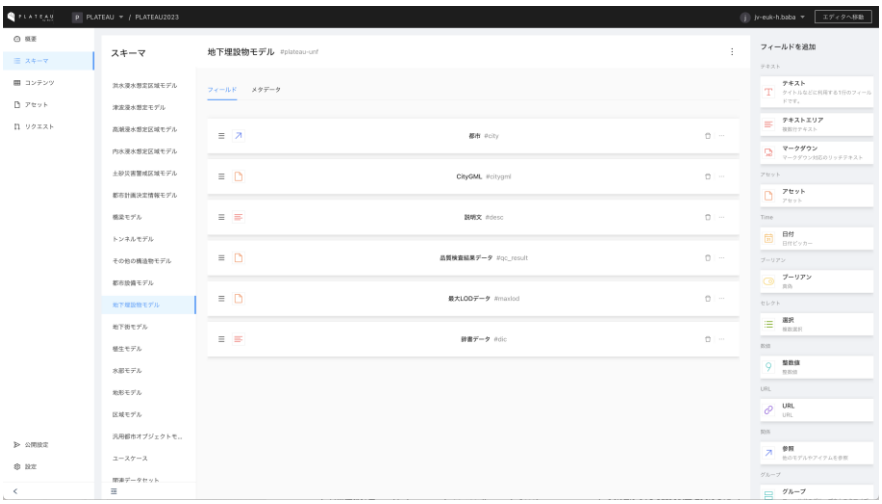


表　地下埋設物モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

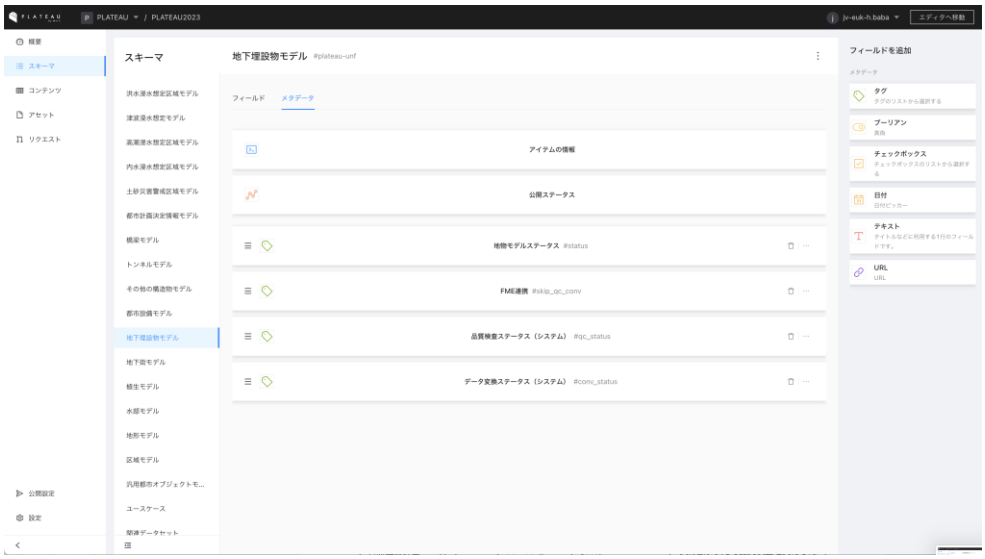


表　地下埋設物モデル：メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：地下街モデル（キー：plateau-ubld）

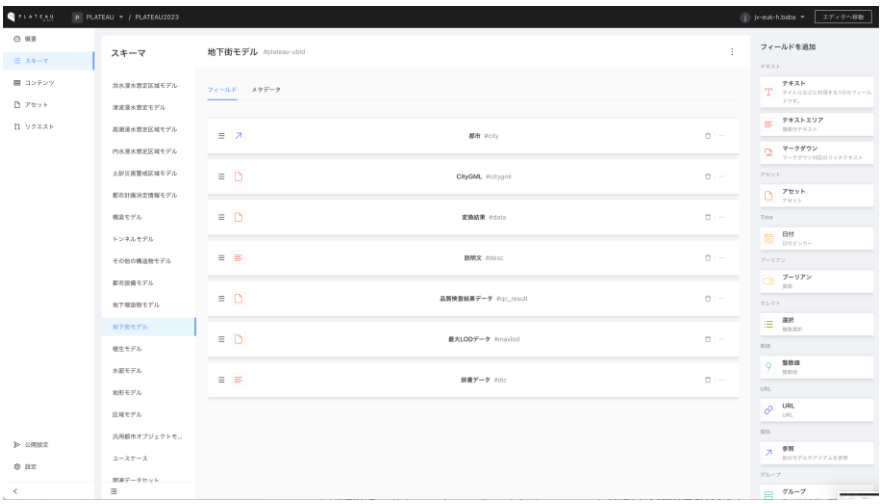


表　地下街モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

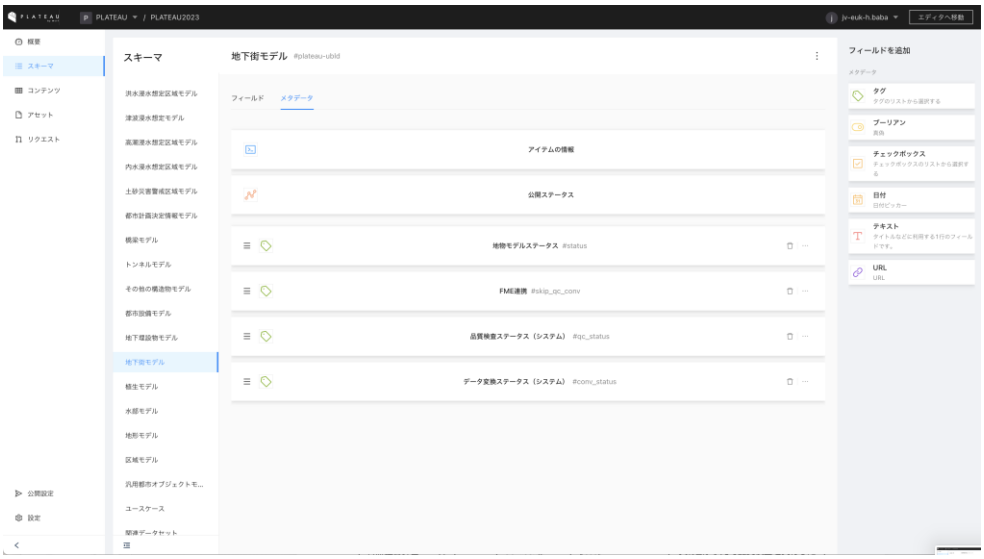


表　地下街モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：植生モデル（キー：plateau-veg）

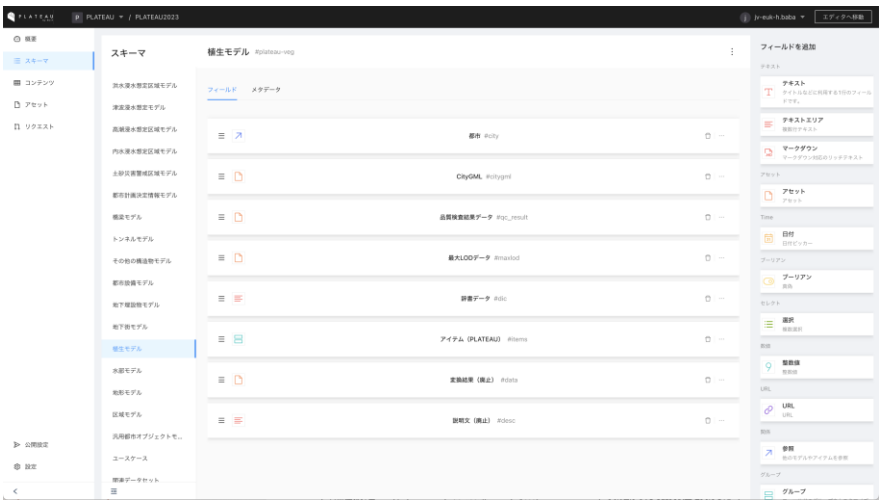


表　植生モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

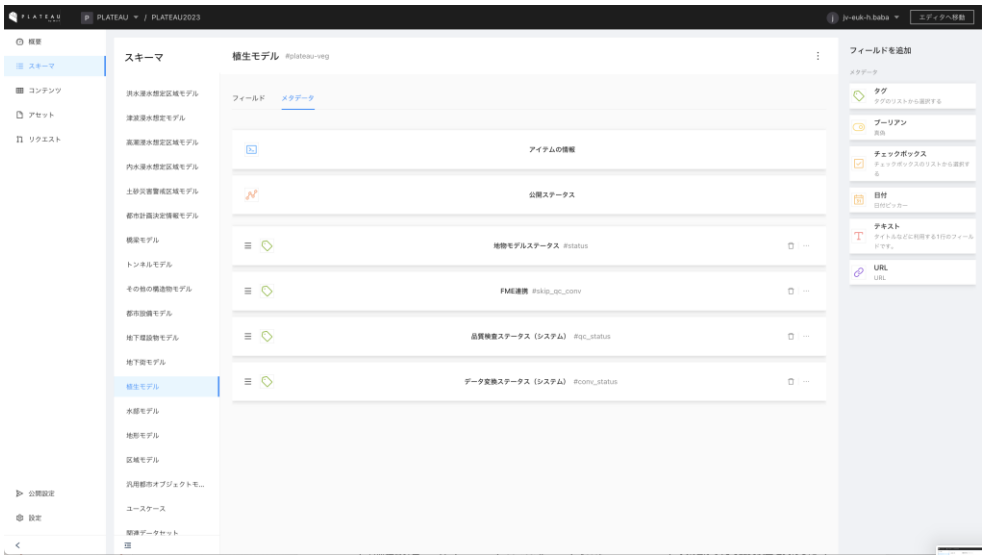


表　植生モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：地形モデル（キー：plateau-dem）

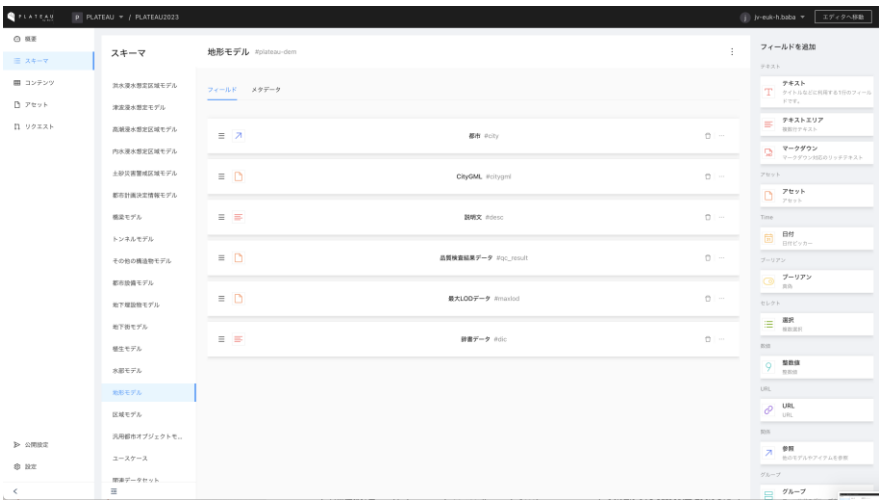


表　地形モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

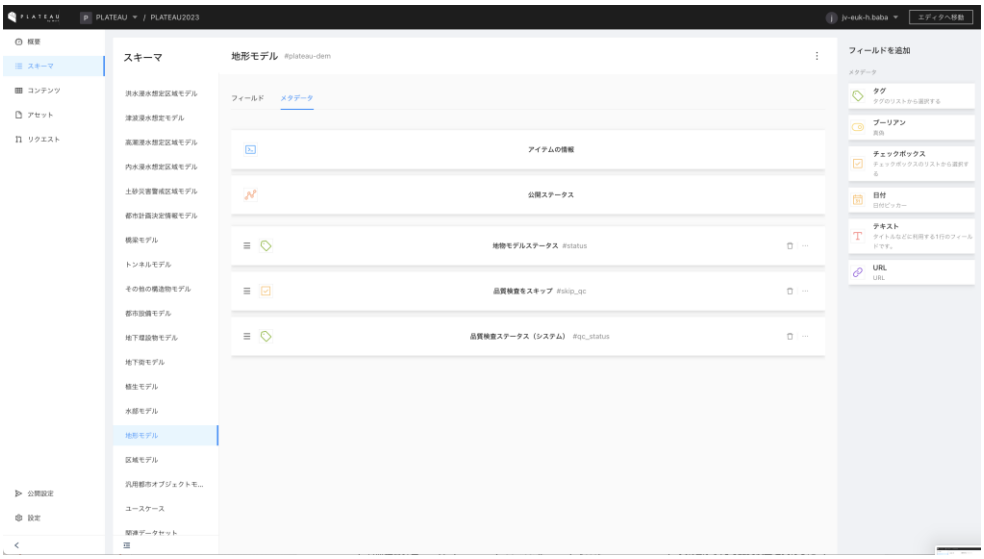


表　地形モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：水部モデル（キー：plateau-wtr）

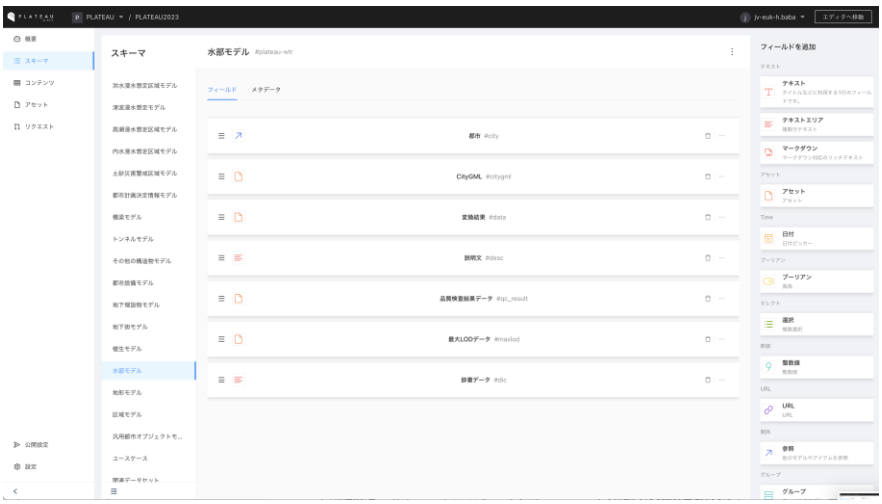


表　水部モデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

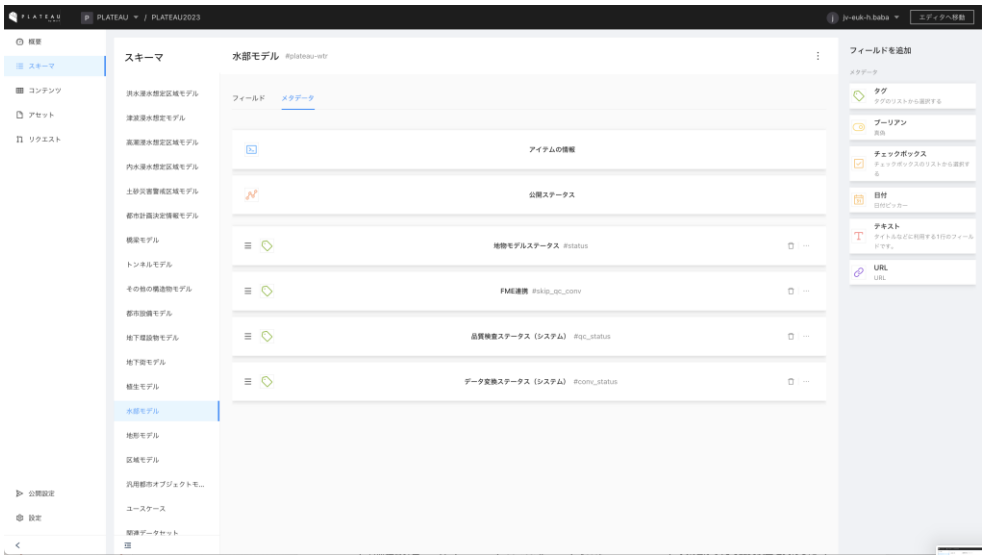


表　水部モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：区域モデル（キー：plateau-area）

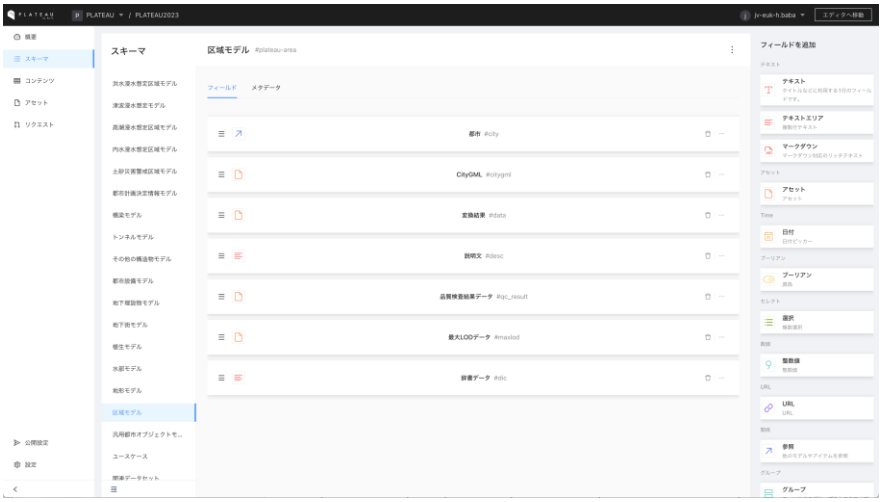


表　areaモデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

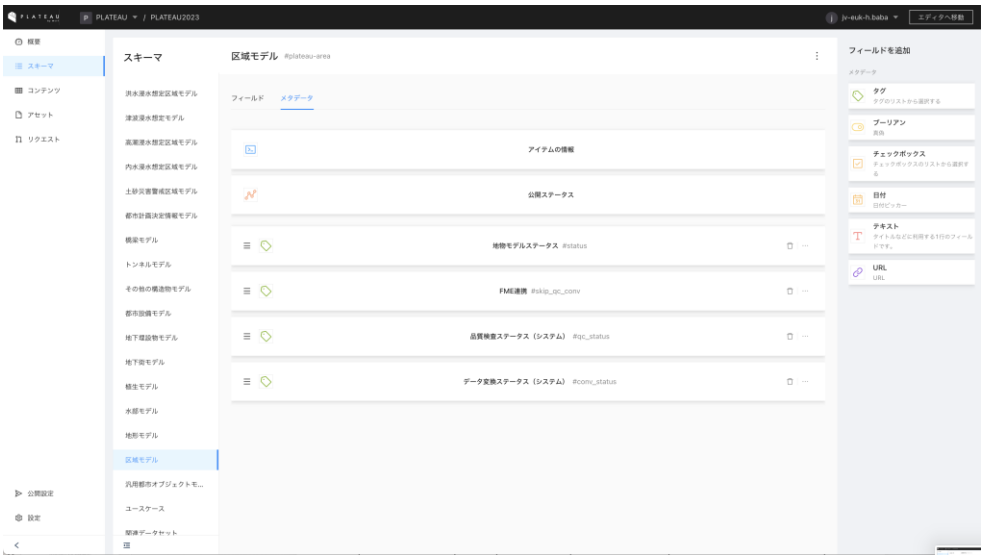


表　区域モデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：汎用都市オブジェクトモデル（キー：plateau-gen）

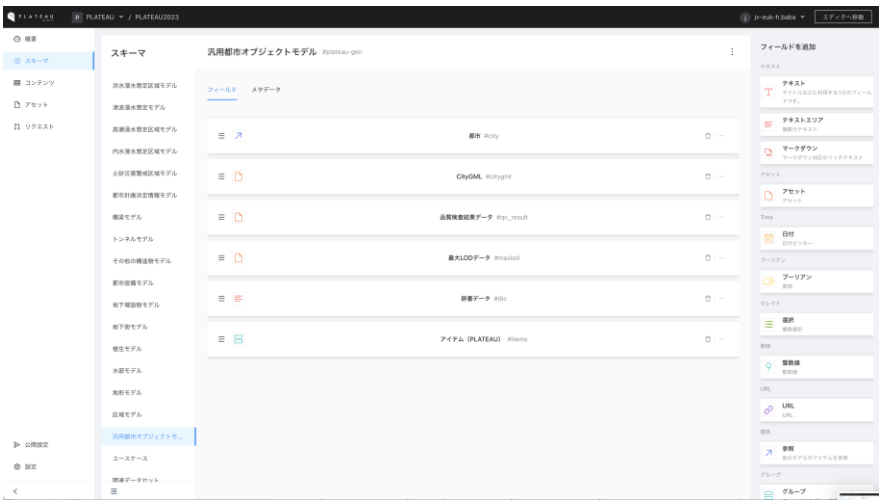


表　汎用都市オブジェクトモデル：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
CityGML		citygml	Asset	
変換結果		data	Asset[]	
説明文		desc	TextArea	
品質検査結果データ		qc_result	Asset	事業者が手元で品質検査を実行したことを担保するため
最大LODデータ		maxlod	Asset	
辞書データ		dic	Textarea	

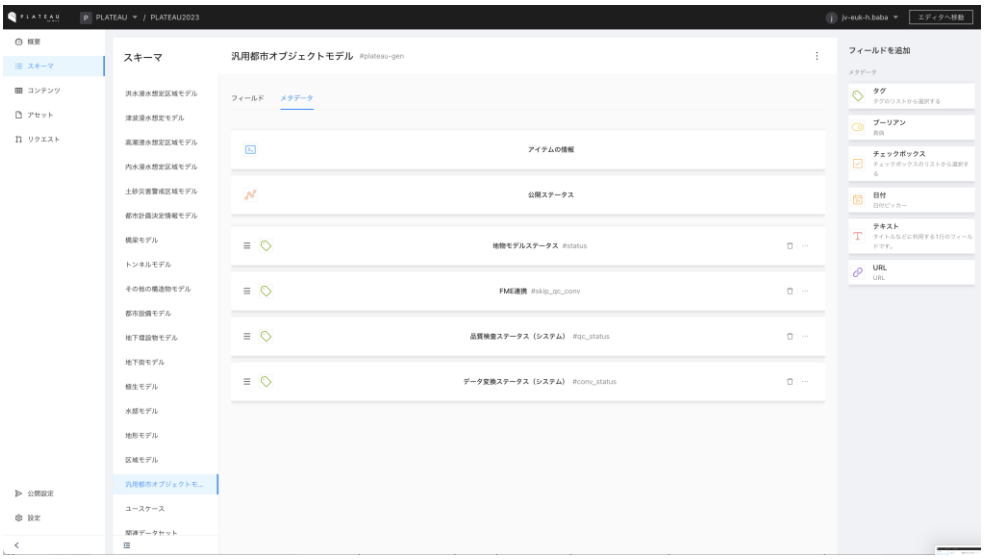


表　汎用都市オブジェクトモデル: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
地物モデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
FME連携		skip_qc_conv	Select	選択肢は「品質検査・変換を実行」「品質検査のみスキップ」「変換のみスキップ」「品質検査・変換をスキップ」デフォルト値は「品質検査・変換を実行」
品質検査ステータス（システム）		qc_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
データ変換ステータス（システム）		conv_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：ユースケース（キー：plateau-generic）

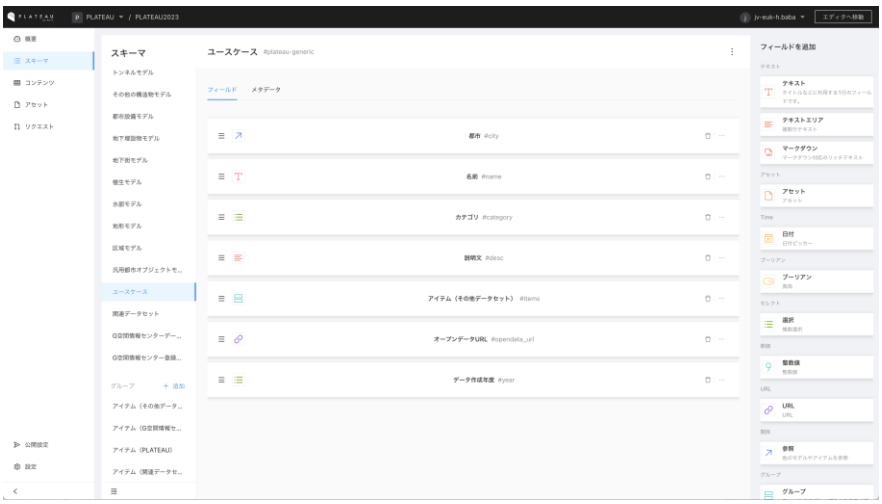


表　ユースケース：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（plateau-city）	
名前		name	Text	
カテゴリ		category	Select	選択肢は「ユースケース」「サンプルデータ」
説明文		desc	TextArea	
アイテム（その他データセット）		items	Group[]	
名前	plateau-item-generic	item_name	Text	
データ	plateau-item-generic	data	Asset	
データURL	plateau-item-generic	url	URL	
データフォーマット	plateau-item-generic	format	Select	選択肢は「3D Tiles」「MVT」「CSV」など
レイヤー名	plateau-item-generic	layer	Text	
オープンデータURL		opendata_url	URL	
データ作成年度		year	Select	「2023」

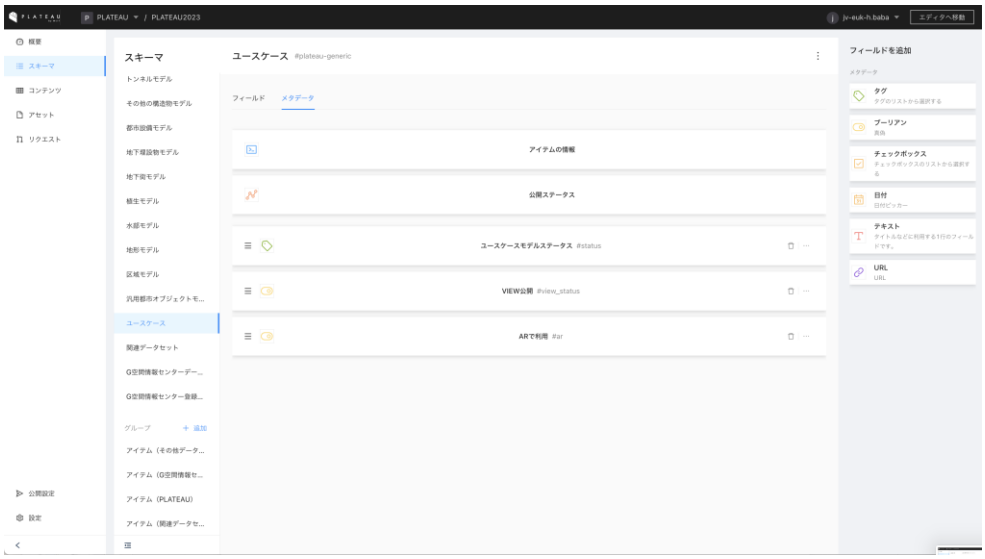


表　ユースケース: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
ユースケースモデルステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
VIEW公開		view_status	Bool	
ARで利用		ar	Bool	default False

モデル：関連データセット（キー：plateau-related）

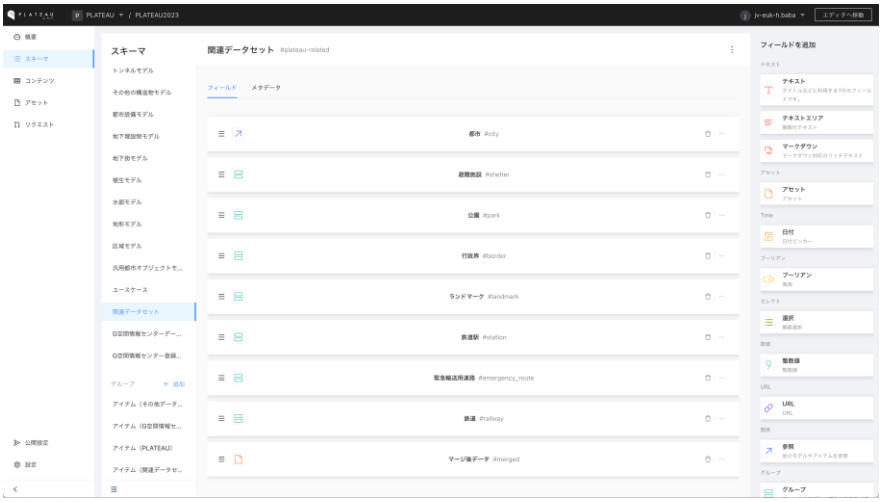


表　関連データセット：データスキーマ一覧（1/2）

名前	グループフィールド	キー	型	備考
都市		city	Reference （plateau-city）	
避難施設		shelter	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
公園		park	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	

表　関連データセット：データスキーマ一覧（2/2）

名前	グループ フィールド	キー	型	備考
行政界		border	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
ランドマーク		landmark	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
鉄道駅		station	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
緊急輸送用道路		emergency_route	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
鉄道		railway	Group	
アイテム（関連データセット）		plateau-item-related	Group[]	
GeoJSON	plateau-item-related	asset	Asset[]	
CZML（変換後）	plateau-item-related	conv	Asset[]	
説明	plateau-item-related	description	TextArea	
マージ後データ		merged	Asset	

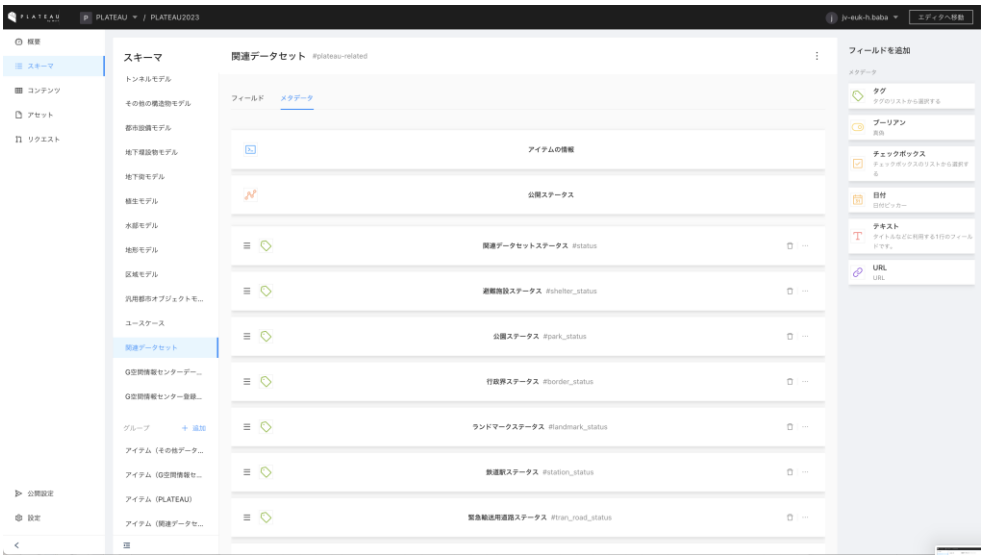


表　ユースケース: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
関連データセットステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」
避難施設ステータス		shelter_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
公園ステータス		park_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
行政界ステータス		border_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
ランドマークステータス		landmark_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
鉄道駅ステータス		station_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
緊急輸送用道路ステータス		tran_road_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
鉄道ステータス		train_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
マージステータス（システム）		merge_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

モデル：G空間情報センターデータ目録（キー：plateau-geospatialjp-index）

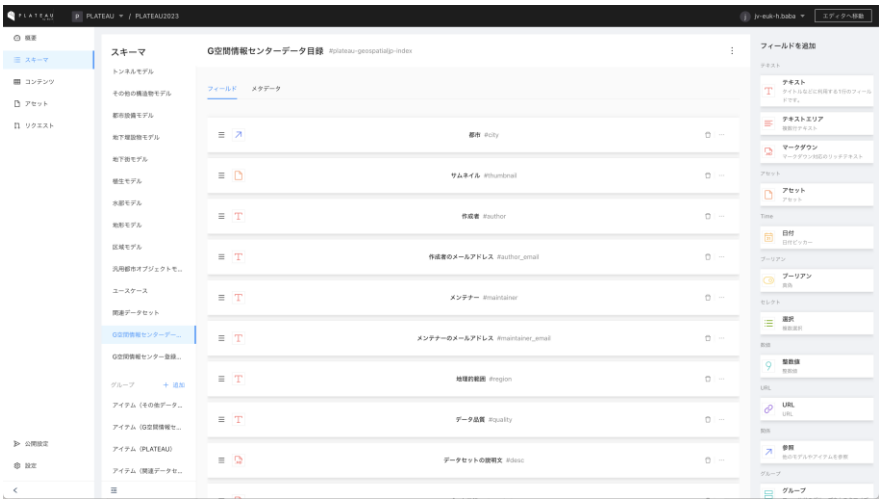


表　G空間情報センターデータ目録: データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（city）	
サムネイル		thumbnail	Asset	
作成者		region	Text	
作成者のメールアドレス		desc	Markdown	
メンテナー		maintainer	Text	
メンテナーのメールアドレス		maintainer_email	Text	
地理的範囲		region	Text	例：製品仕様書による（地図情報レベル2500相当）
データ品質		quality	Text	
データアセットの説明文		desc	Markdown	
データ目録		desc_index	Markdown	自動生成される目録を使用しない場合は、ここに目録リソースの説明文を記入する。
データ目録用ファイル		index_data	Asset	2022年度以前のG空間情報センターのデータセットに目録を掲載する際に使用する。2023年度以降は自動生成されるため使用する必要はない。
CityGMLリソースの説明文		desc_citygml	Markdown	
3D Tiles,MVTリソースの説明文		desc_plateau	Markdown	
関連データセットの説明文		desc_related	Markdown	
ユースケースデータ		generic	Group	
アイテム（G空間情報センター）		#plateau-item-geospatialjp	Group[]	
種類		type	Select	選択肢は「オルソ写真データ」「ユースケースデータ」

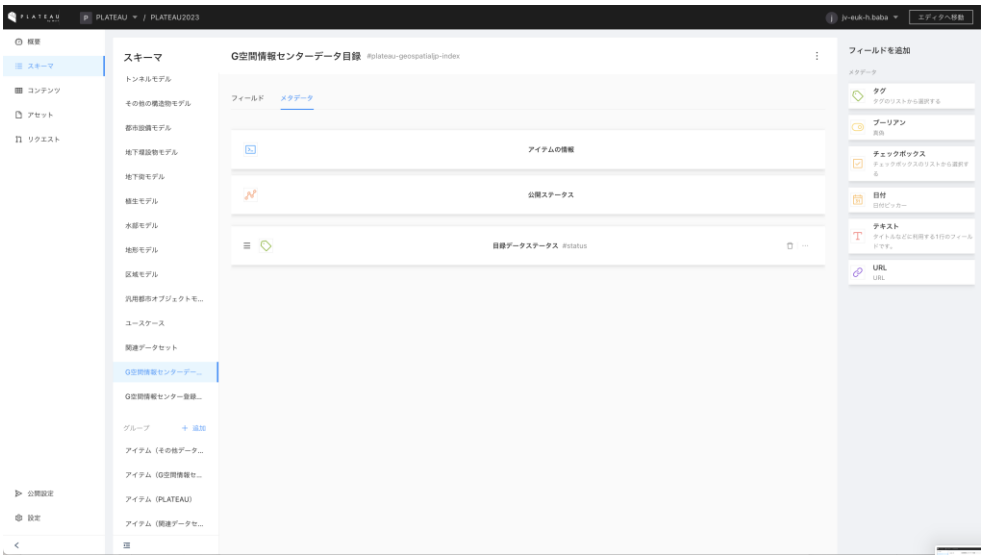


表　G空間情報センターデータ目録: メタデータスキーマ一覧

名前	グループフィールド	キー	型	備考
目録データステータス		status	Select	選択肢は「登録未着手」「新規登録中」「対象外」「登録済み」「確認可能」デフォルト値は「登録未着手」

モデル： G空間情報センター登録用データ （キー：plateau-geospatialjp-data）

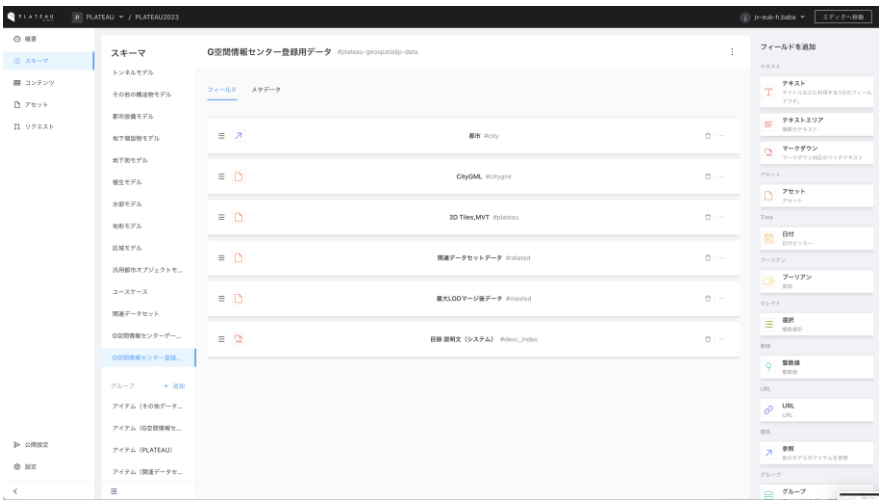


表　G空間情報センター登録用データ：データスキーマ一覧

名前	グループフィールド	キー	型	備考
都市		city	Reference（plateau-city）	
CityGML		citygml	Asset	
3D Tiles.MVT		plateau	Asset	
関連データセット		related	Asset	
最大LODマージ後データ		maxlod	Asset	
目録 説明文（システム）		desc_index	Asset	
都市		city	Reference（plateau-city）	
CityGML		citygml	Asset	

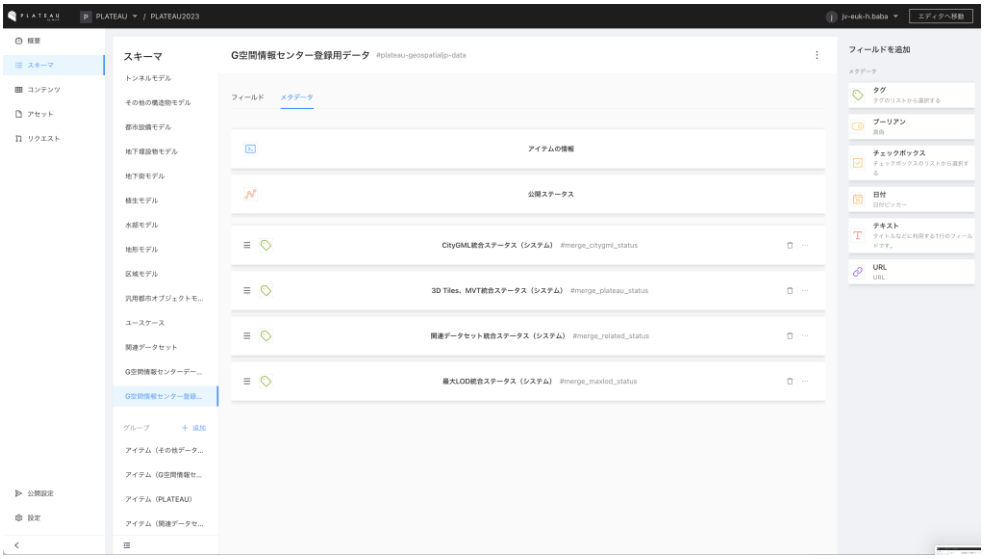


表　G空間情報センター登録用データ: メタデータスキーマ一覧

名前	グループ フィールド	キー	型	備考
CityGML統合ステータス（システム）		merge_citygml_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
3D Tiles、MVT統合ステータス（システム）		merge_plateau_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
関連データセット統合ステータス（システム）		merge_related_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」
最大LOD統合ステータス（システム）		merge_maxlod_status	Select	選択肢は「未実行」「実行中」「エラー」「成功」 デフォルト値は「未実行」

（4）システム用プロジェクトの設定

次に、2つ目のPLATEAU VIEW 3.0システム用プロジェクトを作成し、同様に以下のとおりにモデルを4つ作成する。

名前は自由につけてよいが、連携機能の動作に必要なためキーは必ず指定の文字列を設定すること。なお必須フィールドは存在しない。

1. itemasset
- キー: itemasset
- フィールド

名前	フィールドタイプ
item	テキスト
asset	アセット

2. share
- キー: share
- フィールド

名前	フィールドタイプ
data	テキストエリア

3. sidebar-data
- キー: sidebar-data
- フィールド

名前	フィールドタイプ
data	テキストエリア

4. sidebar-template
- キー: sidebar-template
- フィールド

名前	フィールドタイプ
data	テキストエリア

5. sidebar-template
- キー: sidebar-features
- フィールド

名前	フィールドタイプ
name	テキストエリア
Spec	Reference（plateau-spec）
Name_en	テキストエリア

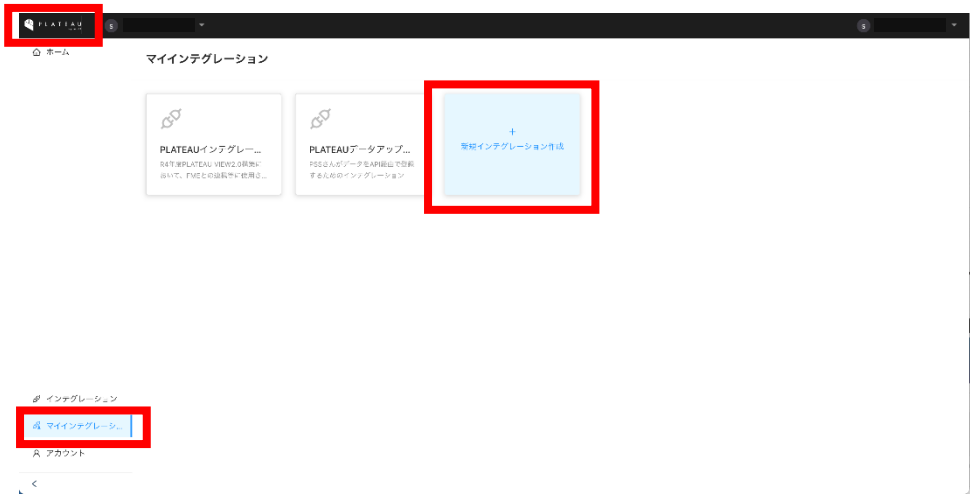
6. sidebar-template
- キー: sidebar-template
- フィールド

名前	フィールドタイプ
majar_version	テキストエリア
Minor_versions	テキストエリア

（５）インテグレーションの作成

インテグレーションとは、PLATEAU CMSと外部アプリケーションとの連携を行うAPIなどの仕組みである。PLATEAU CMSのワークスペースにインストールすることで、PLATEAU CMSの管理APIを利用してデータの取得や変更を行ったり、任意のイベント発生時に外部アプリケーションと連携を行ったりすることができる。

左上のPLATEAU CMSのロゴを押下してトップページ（プロジェクト選択画面）に戻り、左下の「マイインテグレーション」から、インテグレーションを作成する。



インテグレーション作成後、以下のような設定を行う。

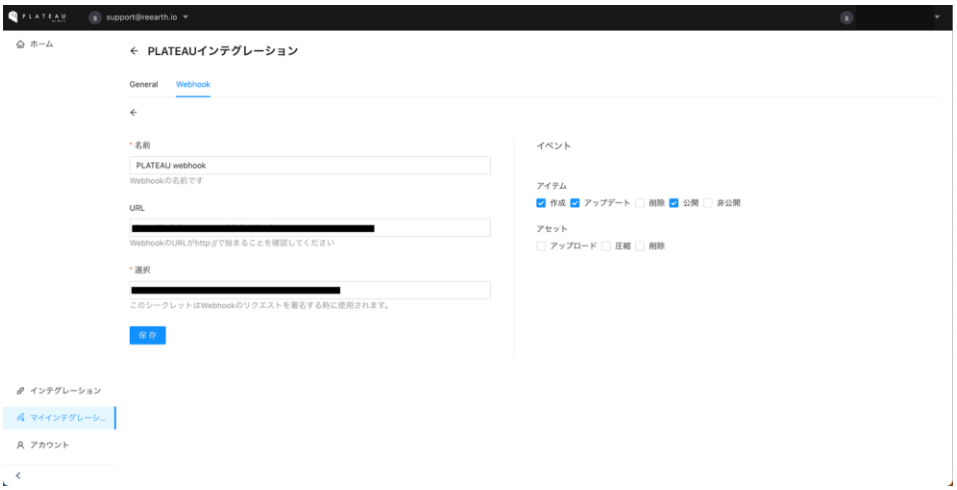
•Webhook：以下のとおりWebhookを設定する。
注意：作成後Webhookの有効化を忘れないこと。

•名前：任意

•URL：イベント発行時のリクエスト先URL。サーバーのURLを入力する。terraform outputsの「plateauview_cms_webhook_url」の値をそのまま指定する。

•シークレット：Webhookのリクエストが正当なリクエストかどうかをクライアント側で検証するためのシークレット。terraform outputsの「plateauview_cms_webhook_secret」の値をそのまま指定する。

•イベント：Webhookが発行されるイベント。全て選択する。



（6）インテグレーションの招待

該当プロジェクトに遷移後、インテグレーションタブで、先ほど作成したインテグレーションを連携する。その後、インテグレーションの権限を「オーナー」に変更する。

最後に、サイドカーサーバーにインテグレーションのトークンを設定する。
先ほど作成したインテグレーションの詳細画面で「インテグレーショントークン」をコピーし、以下の `***${REEARTH_PLATEAUVIEW_CMS_TOKEN}***` の部分に貼り付けてコマンドを実行する。

```
echo -n "${REEARTH_PLATEAUVIEW_CMS_TOKEN}" | gcloud secrets versions add reearth-cms-REEARTH_PLATEAUVIEW_CMS_TOKEN --data-file=-
```

環境変数の変更を適用するため、もう一度サイドカーサーバーをCloud Runにデプロイする。

```
gcloud run deploy plateauview-api ¥
  --image eukarya/plateauview2-sidecar:latest ¥
  --region asia-northeast1 ¥
  --platform managed ¥
  --quiet
```

3.1.7 PLATEAU Editorの動作確認

以下のURLにログインし、ログイン画面が表示されることを確認する。
https://reearth.<ドメイン>

ログインを行い、ログイン後Editorのダッシュボード画面に遷移すれば、構築は完了である。
ログインできない場合は、各種デプロイ完了後にAuth0にユーザーを作成済みであること、ユーザーのメールアドレスの認証が済んでいることを確認する。なお、ログインしてもすぐにログイン画面に戻される場合は、Auth0にユーザーは存在するが、Re:Earth及びPLATEAU CMSのデータベース内にユーザーを作成する処理に失敗していることが考えられる。

次にPLATEAU VIEWの構築に移る。PLATEAU VIEWの作成・公開方法については3.4を参照。

3.1.8 参考：Terraformの実行（2回目以降）

後からtfvarsの内容を変更する場合、変更後に再度Terraformを実行し、インフラに最新の設定を反映する必要がある。
まずtfvarsの編集を行い、次に以下のコマンドを実行し、差分を確認して問題がなければyesと入力して実行を行い、インフラに設定を反映させる（tfvarsファイルを別名で作成し場合は適宜ファイル名を変更）。なお、事前に変数AUTH0_CLIENT_SECRETの設定が必要となる。

```
terraform apply -var-file=env/example.tfvars
```

3.2 FME Flow

PLATEAU VIEW 3.0におけるFME Flowの構築では、Safe Software Inc.のFME Flowを稼働させるサーバー環境とFME Flowのライセンスの取得が必要となる。ここでは、FME Flow稼働のための推奨環境について解説する。

3.2.1 環境の準備

FME FlowはWindowsあるいはLinux上で動作するアプリケーションである。動作確認されているOSはWindows（11、10、Server2022、Server2019、Server2016、Server2012R2）、Linux（Ubuntu 20.04 LTS、Ubuntu 18.04 LTS、Debian 11、Debian 10、Red Hat Enterprise Linux 8（要EPEL8リポジトリ）、 Rocky Linux 8（要EPEL8リポジトリ）、Oracle Linux 8（要 EPEL8リポジトリ）、Red Hat Enterprise Linux 7 （要EPEL7リポジトリ）、CentOS 7 （要EPEL7リポジトリ）、Oracle Linux 7 （要EPEL7リポジトリ））となる。最新の情報はSafe社HPのドキュメントを参照されたい。

インストールにはPentium4、あるいはAMD Opteron以降のCPU、8GB以上のRAM、20GB以上のディスクスペースが必要である。ただし、PLATEAU VIEWで使用する3D都市モデルの変換の実行には、64GB以上のRAMと500GB以上のディスクスペースの確保が望ましい。

3.2.2 構築の手順

以下にWindows及びLinuxへのインストール手順を示す（より詳細な設定についてはSafe社のドキュメントを参照されたい）。

（1）Windows

Safe社のFME Downloadsページからインストーラを入手し、管理者権限で実行する。実行に必要な権限はインストールするディレクトリへの書き込み権限、マシンへのLog on as a serviceの権限である。インストールはインストーラのダイアログに従って、下記のとおり実行する。

- Choose Setup Type : Expressを選択
- Destination Folder : アプリケーション、リポジトリとリソースのディレクトリを指定（デフォルトで実行）
- FME Flow Hostname : サーバーのHostnameを指定
- Web Application Server Port : 80番、あるいは8080番が推奨
- Database User : インストール時にPostgreSQL データベースに作成されるFME Flow
- Databaseののユーザー名とパスワードを設定

（2）Linux

Safe社のFME Downloadsページからインストーラを入手し、以下のコマンドを実行する。

（インストーラと同ディレクトリでroot userとして実行）

```
chmod +x fme-server-b18205-linux-x64~ubuntu.16.04.run
   ./fme-server-b18205-linux-x64~ubuntu.16.04.run
```

プロンプトで指定する内容はWindowsの項を参照されたい。

3.2.3 FME Flowの設定

2.2.2でFME Flowを稼働させるサーバーを用意し、FME Flow をインストールしてライセンシング（ライセンスをインストールするための操作）完了後、FME Flow の稼働環境が整う。

稼働環境が整ったのち、FME Flow ウェブインターフェースに管理者（admin）権限のユーザーでログインし、以下の手順でプロジェクトファイルのインポートおよび各種設定を行う。

※FME Flow のウェブインターフェースの具体的な操作方法については、使用するバージョンの FME Flow 公式ドキュメントを参照すること。

※以下の説明において「メニュー」とは、FME Flow のウェブインターフェース上のメインメニューを指す。

（1）プロジェクトファイルのインポート

メニュー：Projects > Manage Projects を選択して Projects 画面に移動する。

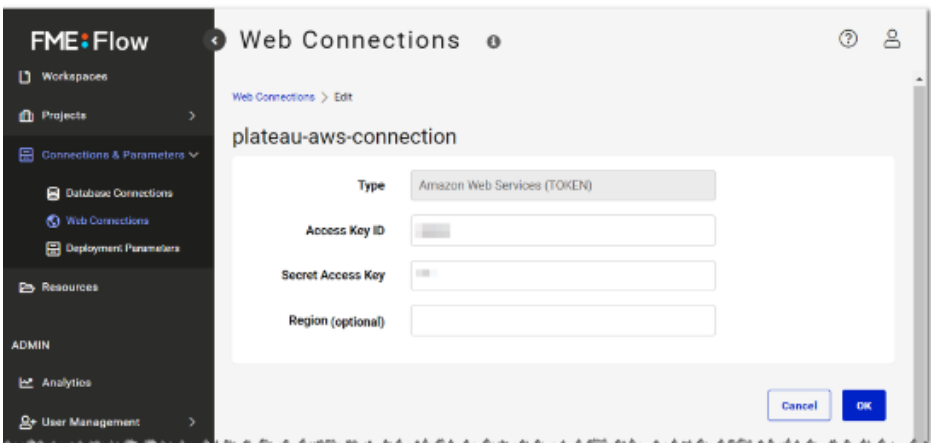
Projects 画面上部の [Import] ボタンを押下して Import Project 画面に移動し、前述のプロジェクトファイル（*.fsproject）を選択してアップロードする。

プロジェクトが展開、設定されるまでそのまま待機する（数分かかることがある）。

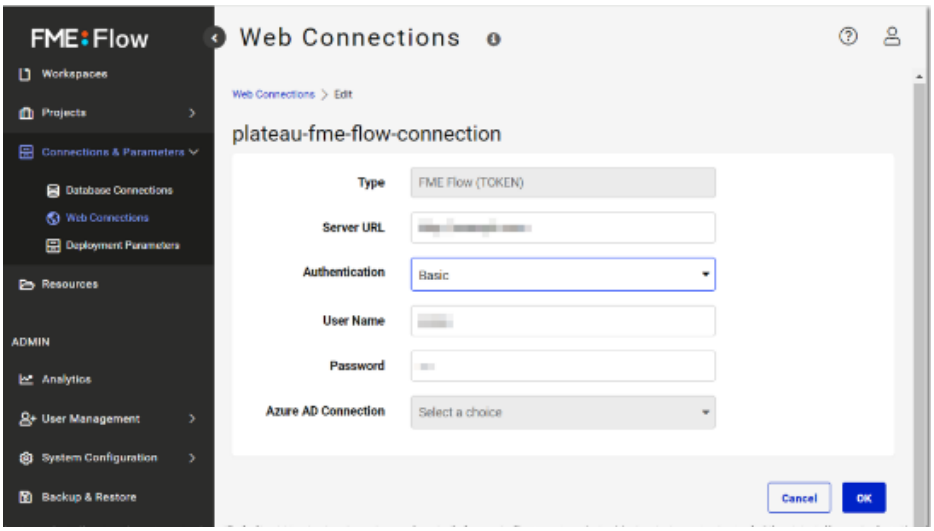
（2）接続設定（Web Connections）

メニュー：Connections & Parameters > Web Connections を選択して Web Connections 画面に移動し、次のふたつの接続パラメーターを使用環境にあわせて設定する。（接続名を押下するとパラメーター設定画面が開くので、画面上に表示される各フィールドに入力する）

- plateau-aws-connection：データ変換結果等の格納用として使用する AWS S3 バケットに接続するのに必要なパラメーター（AWS Acces Key ID, Secret Access Key）を設定する。バケット名は後述の自動処理の設定において指定する。



- plateau-fme-flow-connection：データ変換処理の一部では FME Flow が自分自身に接続して処理をリクエストするものがあり、それに必要なパラメーター（Server URL, User Name, Password）を設定する。



（3）エンジン設定（Engine Management）

メニュー：Engine Manegement 以下の各メニューで、エンジン数（同時実行プロセス数）の設定やキューへのエンジンの割当などを行う。以下は簡易な設定の例であり、FME Flow の使い方に習熟したら、FME Flow の公式ドキュメントに従ってさらに高度で効率的な設定を行っても良い。

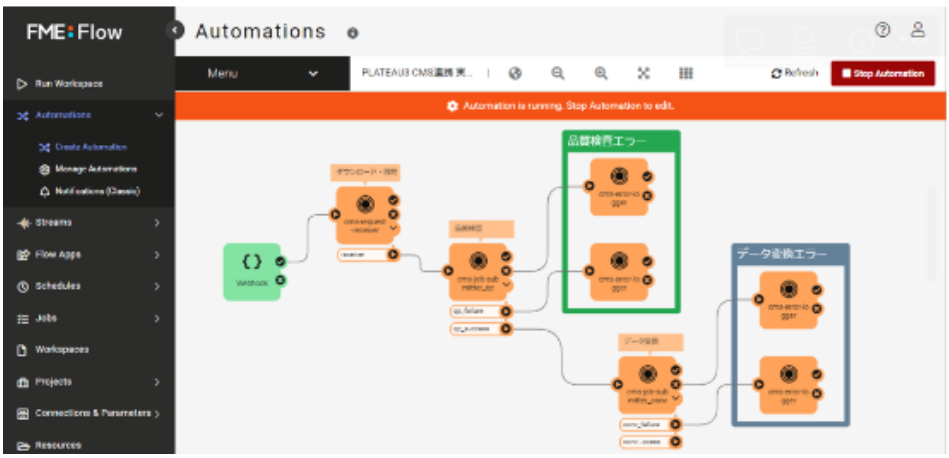
- Engines：エンジン数（同時実行可能なプロセス数）を設定する。Standard Engines（上限：FME Flow ライセンス数）と Dynamic Engines（上限なし）のエンジン数の合計は、次に掲げるキューの数 x 2以上とするのが望ましい。
- Queues：次の5つのキューを作成する。a ～ d のキューの名称は任意で良い。
    - CMSからのリクエスト受付用（対応するリポジトリ：plateau2023-cms）
    - 品質検査用（同：PLATEAU 2023 品質検査）
    - データ変換用（同：PLATEAU 2023 可視化用データ変換）
    - ユーティリティ用（同：plateau-utilities）
    - データ変換における並列処理専用（キューの名称：Parallel Processing）
- Job Routing Rules：キュー a ～ d に対応するリポジトリを割り当てる。各ルールの名称は任意で良い。
- Engine Assignment Rules：キュー a ～ e にエンジンを割り当てる。各キューの間で重複せずに、それぞれ2個以上のエンジンを割り当てるのが望ましい。

（4）自動処理の設定と起動（Automations）

メニュー：Automations > Manage Automations で Automations 画面に移動し、「PLATEAU3 CMS連携 実証環境構築用」を押下してその編集画面を開き、以下の操作、設定を行う。

- 編集画面上の左端にある Webhook トリガーを押下して詳細画面を開き、そこに表示されている Webhook URL をコピーする。このURLが、CMSからのリクエスト先となる。
- 画面上部の地球型アイコンを押下して自動処理パラメーター編集画面（Automation Paramters Editor）を開き、S3_BUCKET_NAME パラメーターにデータ変換結果等の格納用として使用するS3バケット名を設定（初期状態で dummny と設定されているのを上書き）し、[OK] で閉じる。
- 画面右上の [Start Automation] ボタンを押下して、自動処理を起動する。

以上の設定により、Webhook トリガーから取得したURLあてにCMSから品質検査やデータ変換のリクエストをポストすると、自動処理が開始される。下図に自動処理実行中の画面例を示す。



運用上の留意事項

CMSから品質検査やデータ変換のリクエストがあると、FME Flow は対象とするデータセット（zipアーカイブ）をCMSからダウンロードし、次のリソースフォルダー以下に展開してから処理を始める。

Resources > Data > plateau2023 > downloads

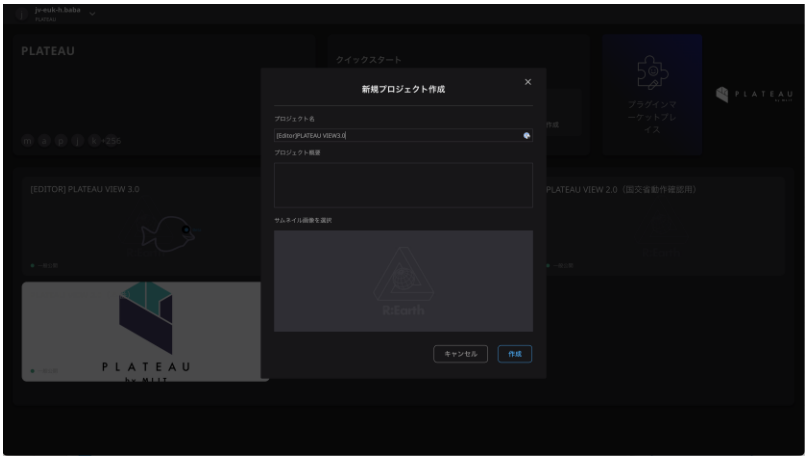
データ変換が正常に終了したとき、および、毎日の自動クリーンアップ処理で取得から1週間以上経過したデータファイルは自動的に削除されるが、処理の失敗等によって削除されずに残ることもあるので、定期的に上記リソースフォルダー内をチェックし、不要なデータが残っているときは手動で削除することが望ましい。

3.3 PLATEAU EditorとVIEWのセットアップ

3.3.1 プロジェクトの作成

PLATEAU Editor内でのプロジェクトの作成とセットアップを行う。2つのプロジェクトを作成する。

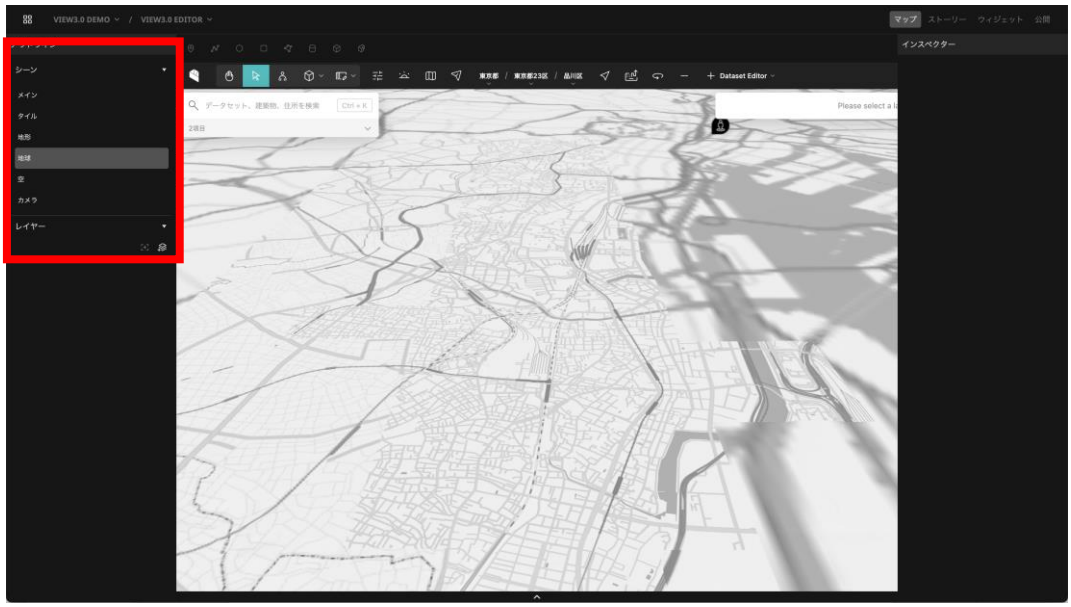
1つ目は、Editor用プロジェクト、もう一つはVIEW用プロジェクトである。以下のようにプロジェクトを新規作成する。プロジェクト名は任意である。



3.3.2 シーンの設定

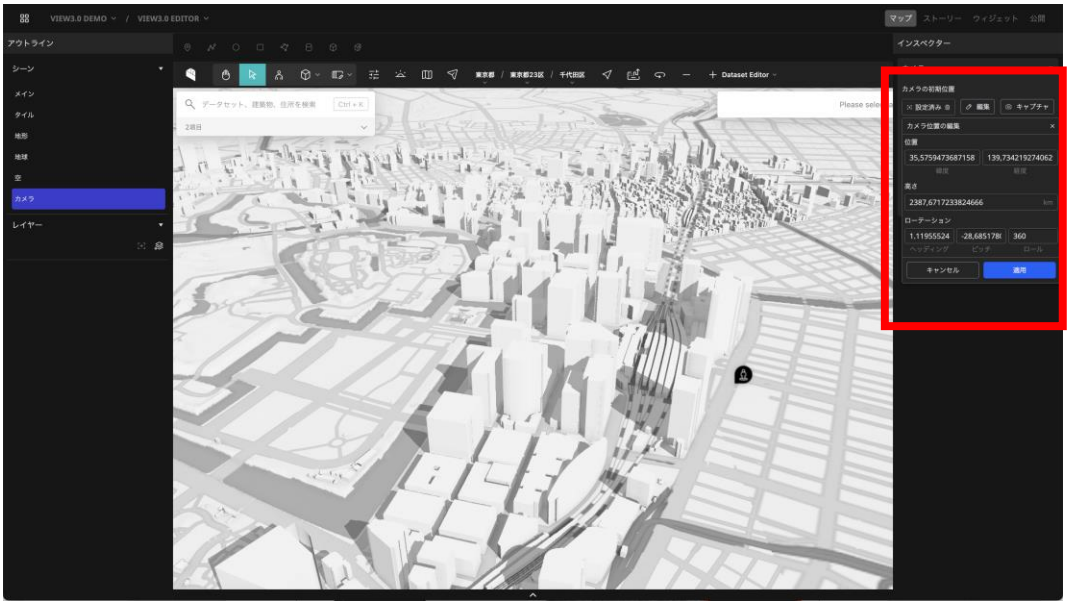
シーンでは、デジタルアースのベースマップやカメラの初期位置など、プロジェクト全体の設定等を行うことができる。またレイヤーやインフォボックスの配置などができる。

基本的な考え方として、画面左のパネル（アウトライン）または地球儀上で編集対象を選択し、画面右のパネルで設定を変更するという操作ができる。



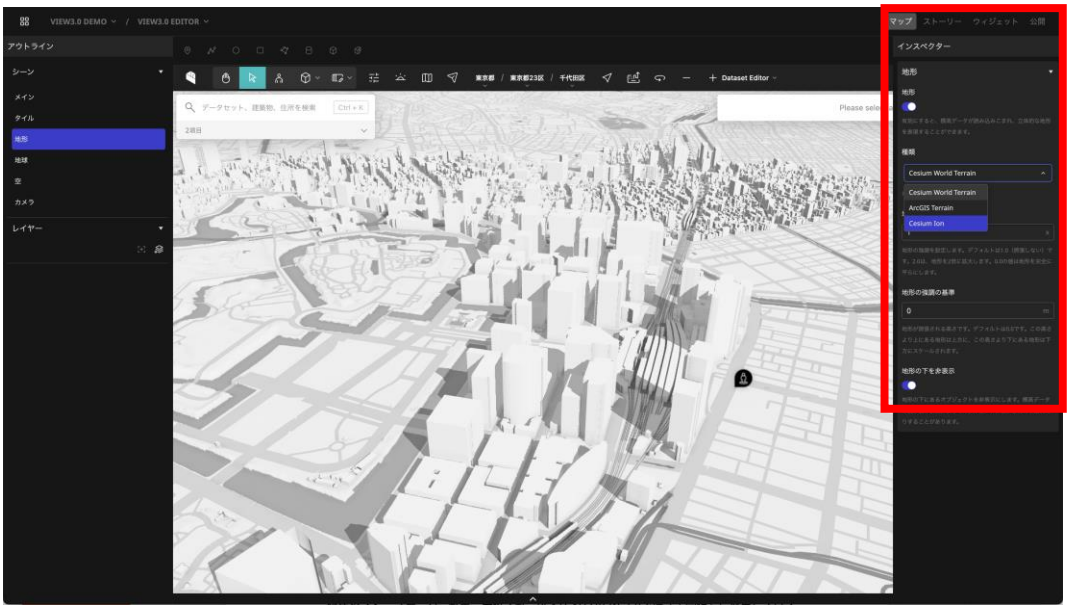
（1）初期カメラの設定

ここではPLATEAU VIEW 3.0におけるシーン全体の設定を行う。左サイドバー上部の「シーン」を選択すると、右パネルにシーンの設定が表示される。
デフォルトでは、ページロード時に北アメリカ全土が表示される設定になっている。
「カメラ初期位置の設定」では、ページロード後、最初に表示されるカメラの位置を設定できる。
「キャプチャ」を押下すると、その位置と画角がカメラ初期位置に設定される。



（2）タイル・地形の設定

タイルとは地表のベースマップの画像のことである。またTerrainとは地表の標高などに基づく三次元的形状のことである。デフォルトでは、デフォルト（Cesium）のタイルと、Cesium World Terrainを使用するようになっているため、PLATEAU独自のTerrainのURLを設定する。

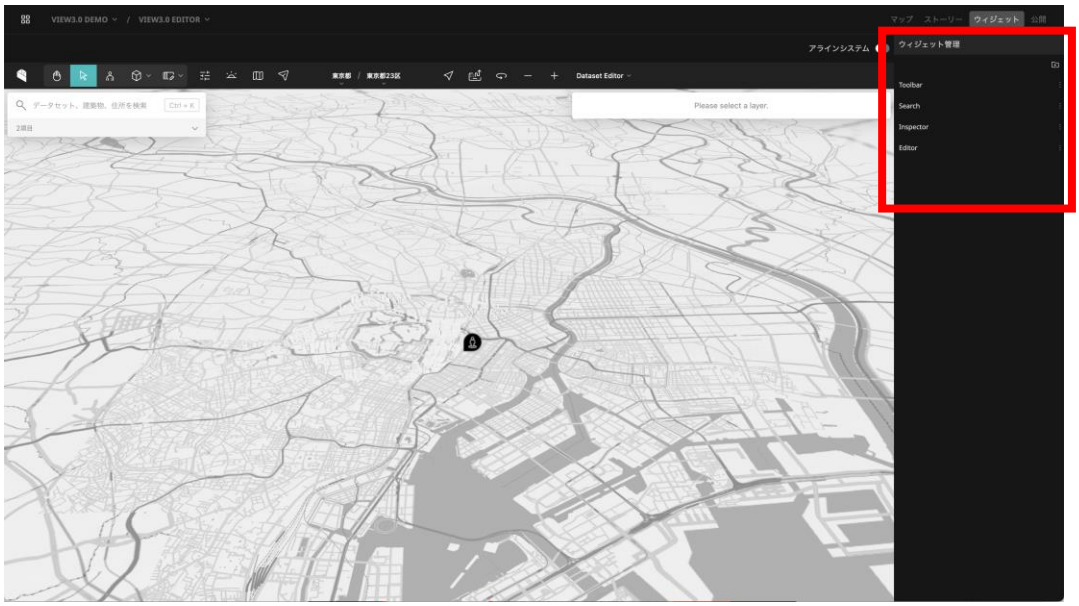


3.3.3 ウィジェットの設定

ウィジェットでは、ウインドウ上に配置される機能の設定が可能である。また、ウィジェット配置システムによって、それらのウィジェットの配置を自由に設定することができる。

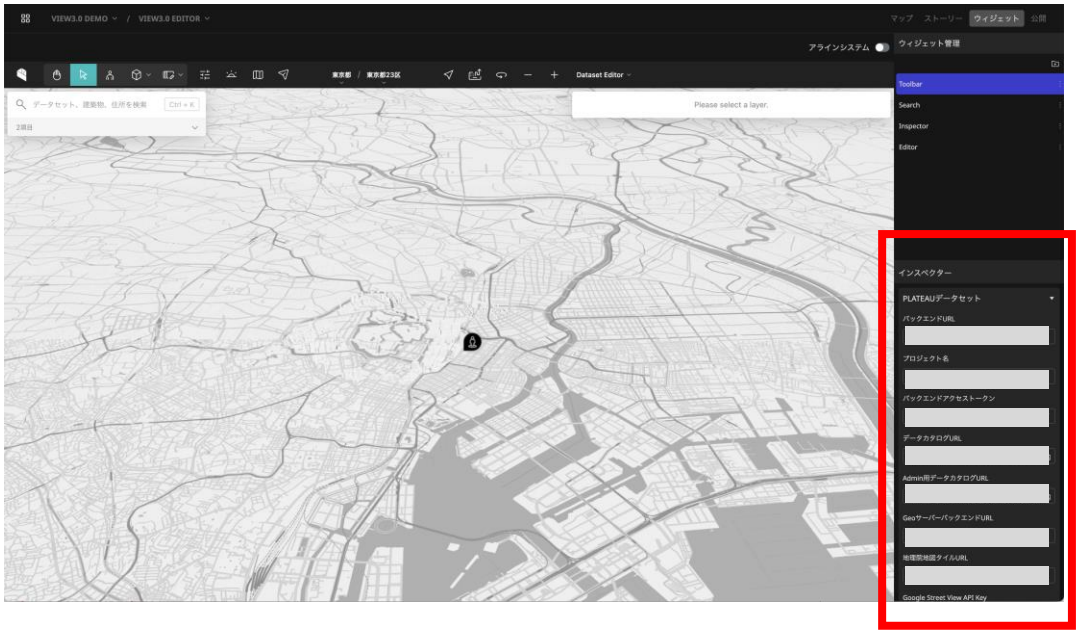
（1）ウィジェットの追加

左パネルより、ウィジェットをシーンへ追加する。



（2）ウィジェットの設定

それぞれのウィジェットは設定項目を有しており、さまざまな設定を行うことができる。左パネル上部でウィジェットを選択すると、右パネルに選択中のウィジェットの設定項目が表示される。



ウィジェット設定における詳細は以下のとおり。

- **ツールバー**
  - **バックエンド URL**: サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。
  - **プロジェクト名**: 共有機能などのデータを保存するための、PLATEAU CMS上のプロジェクトエイリアス名。「3.1.6 PLATEAU CMSのセットアップ」で作成したPLATEAU CMSのPLATEAU VIEW用プロジェクトのプロジェクトエイリアスを入力する。
  - **バックエンドアクセストークン**: PLATEAU Viewサーバーの認証に使用されるアクセストークン。3.1のTerraformの実行結果として得た「plateauview_sidebar_token」の値を入力する。
  - **データカタログUR**L: データカタログ用サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。通常、PLATEAU Backend Base URLに「datacatalog/graphql」を付与した値にとなる。
  - **Admin用データカタログUR**L: Editorで利用するデータカタログ用サイドカーサーバーのエンドポイント。3.1のTerraform実行結果として得た「plateauview_sidecar_url」の値を入力する。通常、PLATEAU Backend Base URLに「/datacatalog/admin/graphql」をつけた値になる。
  - **GeoサーバーバックエンドURL**：ジオコーディングや住所検索に利用するAPIサーバーへのエンドポイントを記載する。 3.1のTerraform実行結果として得た「plateauview_geoapi_url」の値を入力する。
  - **地理院地図タイルURL**：国土地理院が実験的に提供しているベクトルタイル形式の「地理院タイル」配信エンドポイントを入力する。
  - **Google Street View API Key**：Google Street Viewを利用するためのAPIキーを入力する。
  - **プロジェクトの公開URL**: 共有URL機能を利用した際のURL。3.1のTerraformの実行結果として得た「plateauview_reearth_url」にプロジェクトエイリアスをサブドメインとして付加したURL。
  - **フィードバックを非表示**：ご意見ご要望機能をOFFにするオプション設定。

PLATEAU VIEW構築マニュアル 第4.0版

PLATEAU VIEW Setup Manual

令和6年3月　発行

国土交通省 都市局

PLATEAU by MLIT