Hochschule für Technik Stuttgart
University of Applied Sciences
Photogrammetry and Geoinformatics

Master of Photogrammetry and Geoinformatics
ASSIGNMENT

Creating the evaluation map for redevelopment district by capturing data software and 3D modelling visualization

GIS STUDIO

GROUP MEMBER:

Faizi, Meena 20015075

Valere Mbanzi Iradukunda 20015257

DATE: 24.06.2024

SEMESTER: 2/2024

PROFESSORS: Prof. Volker Coors & Prof. Hamidreza Ostadabbas

Contents

TABLE OF FIGURES

TABLES

INTRODUCTION

1.1. Study Background

Accurate cadastral information is important for effective urban planning and redevelopment. This project focuses on enhancing the cadastral data for Stuttgart's city center, particularly in areas where existing data is outdated or incomplete. The primary aim is to extract, update, and manage official cadastral data from German sources using advanced data management and visualization techniques. By doing so, we aim to create accurate, detailed, and interactive 3D models that provide comprehensive insights into the urban environment, facilitating better decision-making and planning.

The core objectives of this project include:

a. Extracting and managing official cadastral data from German sources i.e. ALKIS.
b. Updating field data using specialized software such as QField and QGIS.
c. Creating 2D maps for enhanced visualization and analysis.
d. Assigning captured values to the 3D models to improve understanding and support informed decision-making.

This project is essential due to the need for accurate and up-to-date cadastral information in urban planning. The German cadastral system, encompassing ALKIS (Authoritative Real Estate Cadastre Information System), ATKIS (Authoritative Topographic-Cartographic Information System), and AFIS (Authoritative Control Point Information System), is fundamental to land management and urban planning processes. Familiarity with these systems and their specialized terminology is crucial for working with cadastral data effectively.

Additionally, the preference for 3D modeling in urban planning is a significant factor driving this project. 3D models offer a detailed and visually engaging representation of urban areas, making it easier to visualize complex spatial relationships and analyze urban dynamics. By integrating updated cadastral data into these 3D models, it provides planners and decision-makers with a powerful tool for evaluating and improving urban environments.

This project is not only about addressing the need for updated cadastral information but also about demonstrating the value of advanced data management and visualization techniques in urban planning.

1.1. Study Area

The study area for this project is located in Stuttgart, focusing on several key streets that are central to the city's redevelopment efforts. These areas include:

- Friedrichstraße, 70174
- Königstraße, 70173

- Arnulf-Klett-Platz, 70173
- Kronenstraße, 70174
- Geschwister-Scholl-Straße, 70174
- Lautenschlagerstraße, 70173

These locations encompasses a mix of commercial, residential, and public spaces, providing a comprehensive view of the area's redevelopment needs and opportunities. The buildings highlighted in black in Figure 1 shows the area of study, chosen based on the availability of ALKIS cadastral and 3D model data.

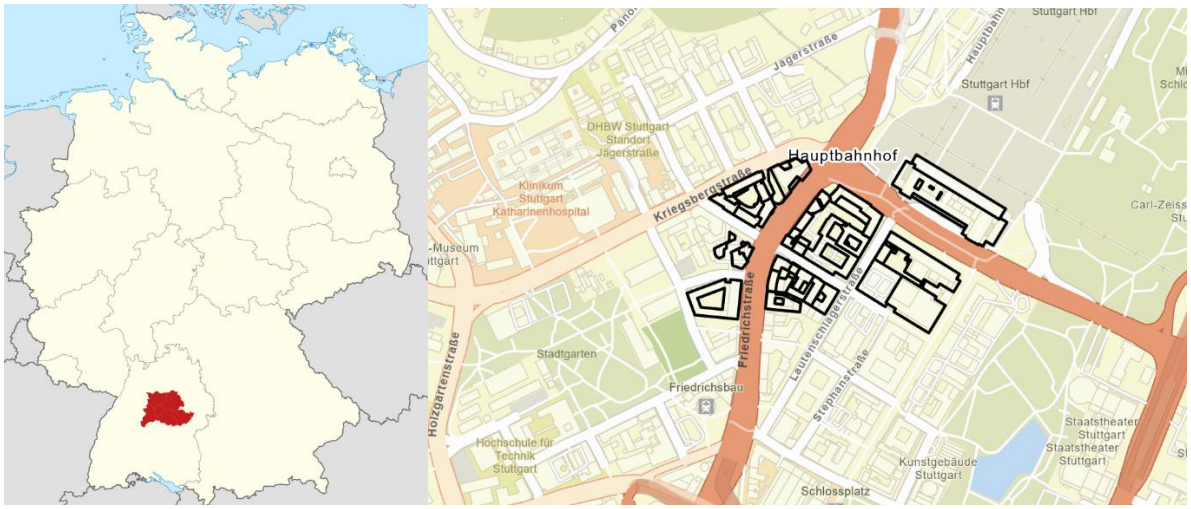


Figure 1. Map of Germany and Stuttgart city centre (Germany map image source: Wikipedia)

2. METHODOLOGY

This project employed a structured methodology to ensure the efficient extraction, updating, and visualization of cadastral data for Stuttgart's city centre. The process was divided into several key phases, each utilizing specialized tools and techniques to achieve the project objectives.

2.1. Data Extraction and Standardization

a) Data Sources: The project began with the extraction of cadastral data from official German sources such as ALKIS, ATKIS, and AFIS. These systems provide authoritative and comprehensive information on real estate, topography, and control points, respectively.
b) Data Standardization: Extracted data was standardized to ensure consistency and compatibility. This involved cleaning the data, verifying its accuracy, and transforming it into a uniform format suitable for further analysis and visualization.

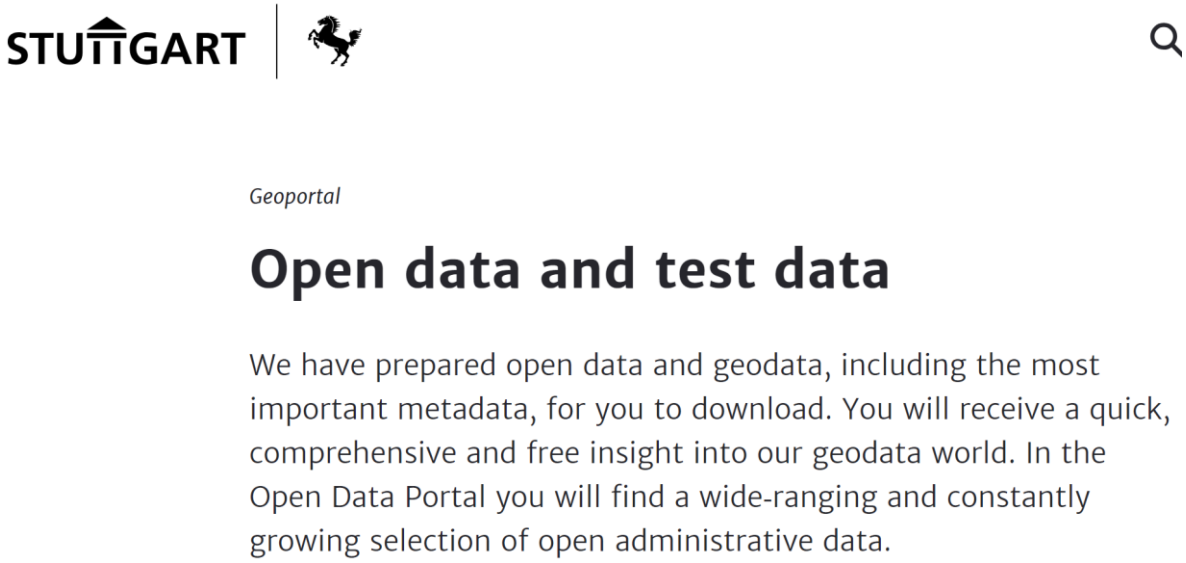


Figure 2. The website where ALKIS cadastral and 3D model were extracted [1]

2.2. Studying the Attribute Table

To understand the German terms used in the attribute table of ALKIS cadastral data, a structured approach was employed. The methodology involved the following steps:

a) The attribute table of the ALKIS cadastral data was thoroughly examined to identify all instances of German terms requiring interpretation.
b) The interpretation and understanding of these terms were facilitated by referencing the following authoritative sources:

- *Basiswissen ALKIS / ETRS 89* [2]: This foundational resource provided essential knowledge on ALKIS and ETRS 89, aiding in the comprehension of technical terms.

[1] Accessed from the ALKIS website: https://www.stuttgart.de/leben/bauen/geoportal/open-data-und-testdaten.php
[2] Niedersächsische Vermessungs- und Katasterverwaltung. (2010). *Basiswissen ALKIS / ETRS 89*.

- *ALKIS® Objektartenkatalog NRW 7.1*[3]: This catalog delineated the types of objects and their associated attributes within the ALKIS system, offering detailed definitions crucial for accurate interpretation.
- Ostadabbas, Hamidreza (die STEG): Insights from Ostadabbas, particularly relevant to ALKIS terminology and application, were consulted to enrich understanding.

2.3. Data Capture

The data capture and update of the attribute table were performed using two different methods. The first method involved personal observations during fieldwork, with data recorded in QField. The second method consisted of gathering relevant information for the attribute table by observing data in Google Earth, Google Maps, OpenStreetMap, and other internet sources.

a) Field Data Collection: QField, a mobile data collection app, was used to capture real-time spatial information in the field. This tool allowed for efficient and precise data collection, directly syncing with the main database to ensure up-to-date information Figure 3. Two buildings were the main focus during field day data collection Figure 4

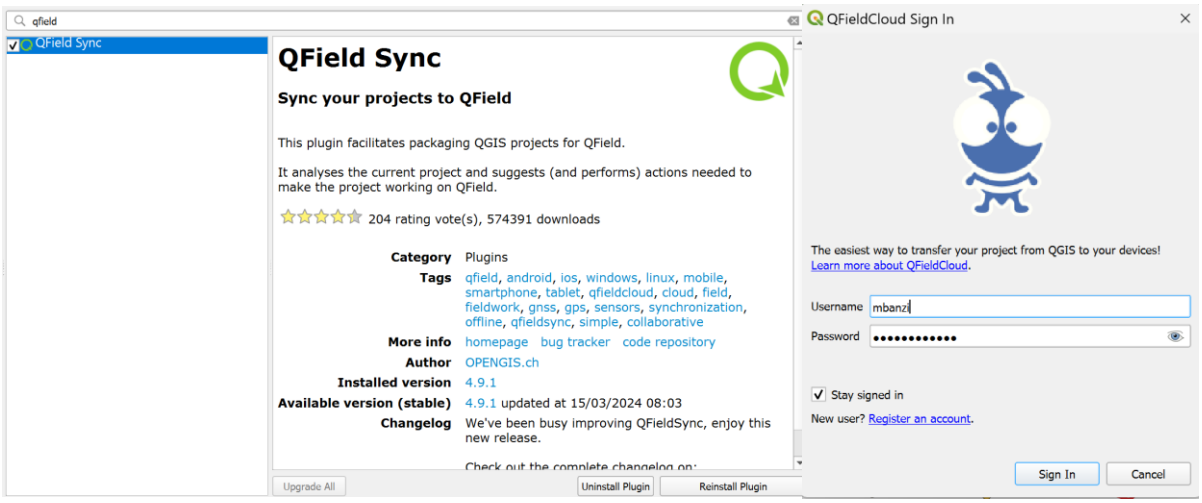


Figure 3. Setting up QField application for data collection

[3] *ALKIS®Objektartenkatalog NRW 7.1 ALKIS-OK NRW 7.1 basierend auf dem AFIS-ALKIS-ATKIS®-Anwendungsschema 7.1.0*. (2021).

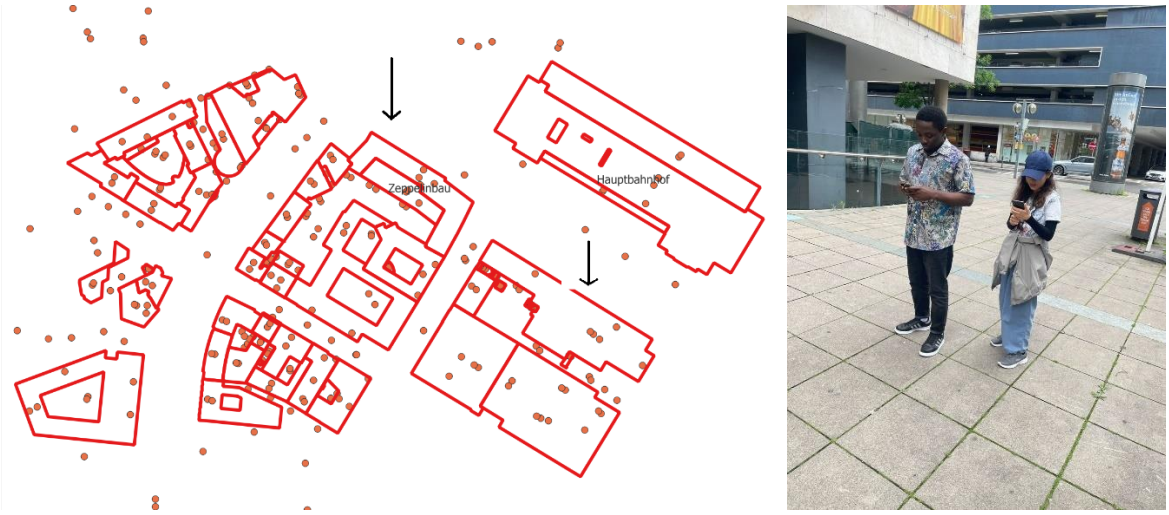
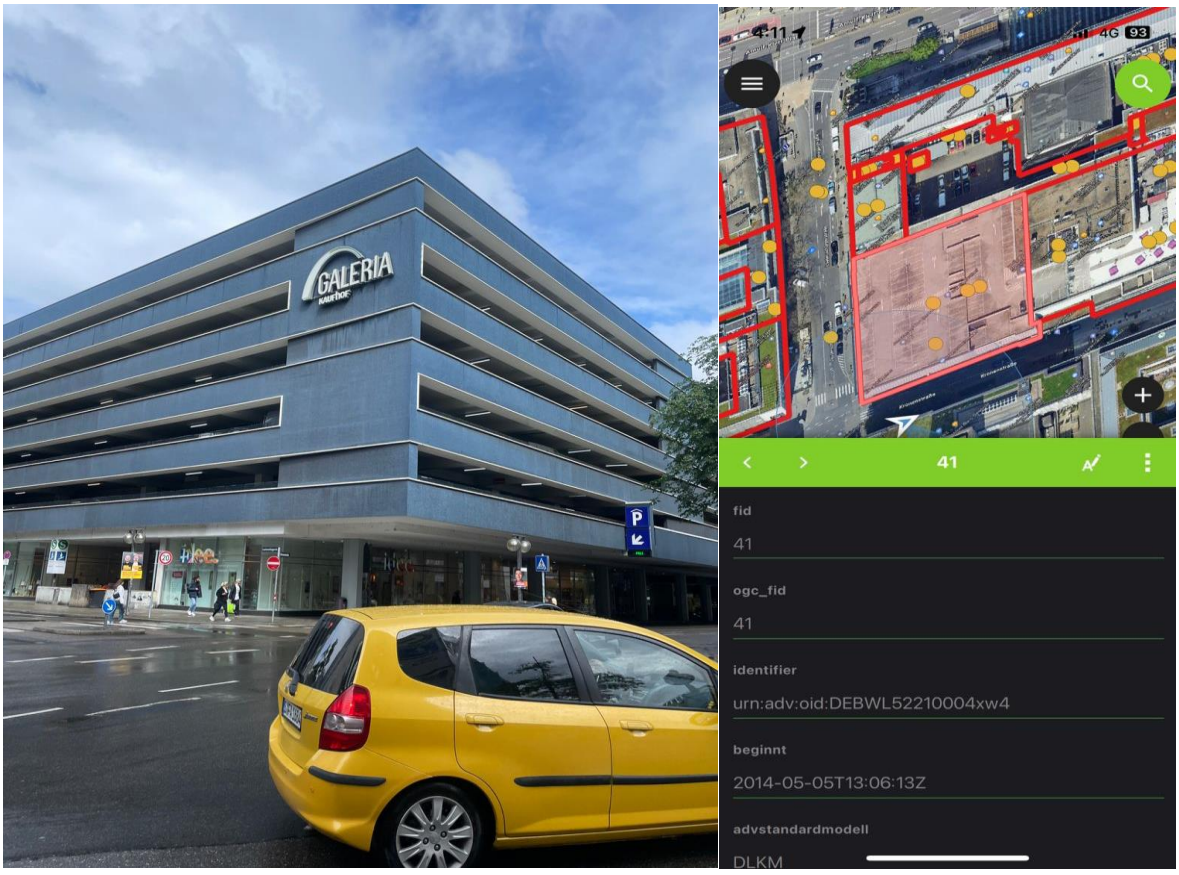


Figure 4. Field day data capturing. The two buildings were considered, and two devices, both logged into QField with the same account



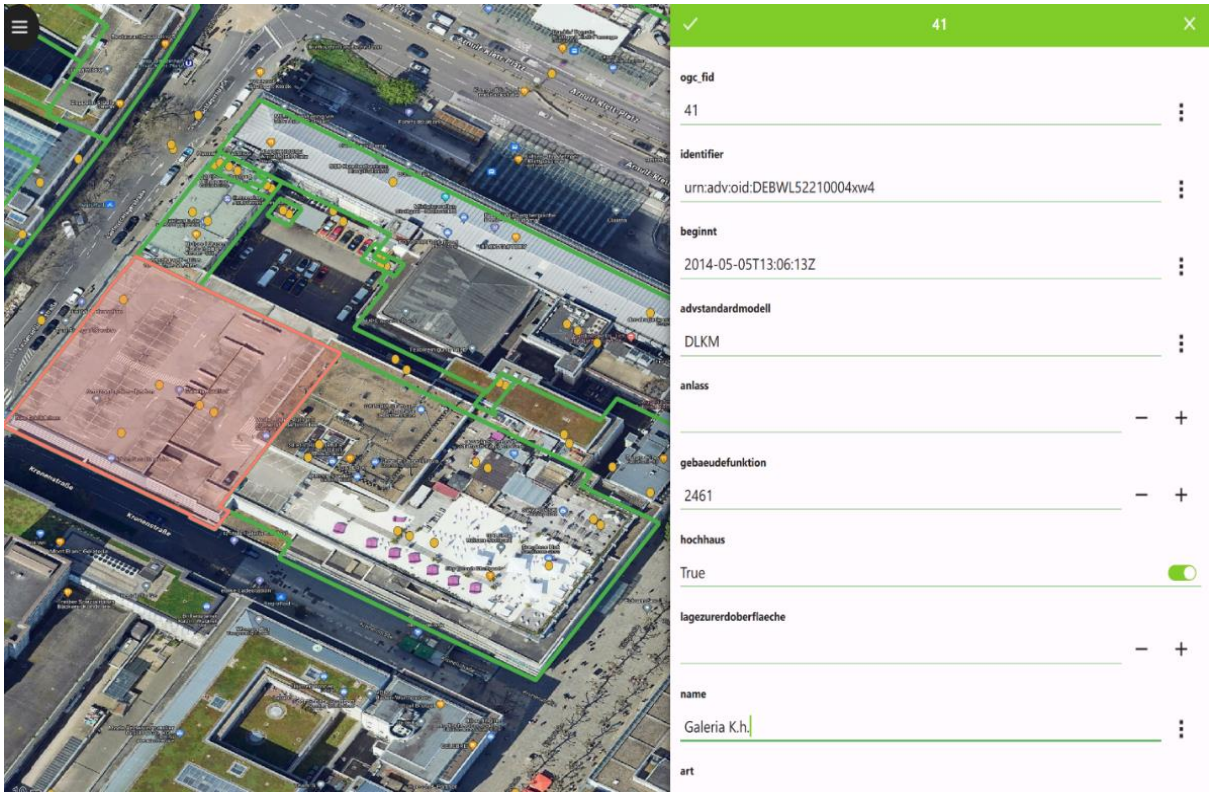


Figure 5. Example of updating building polygon name in QField

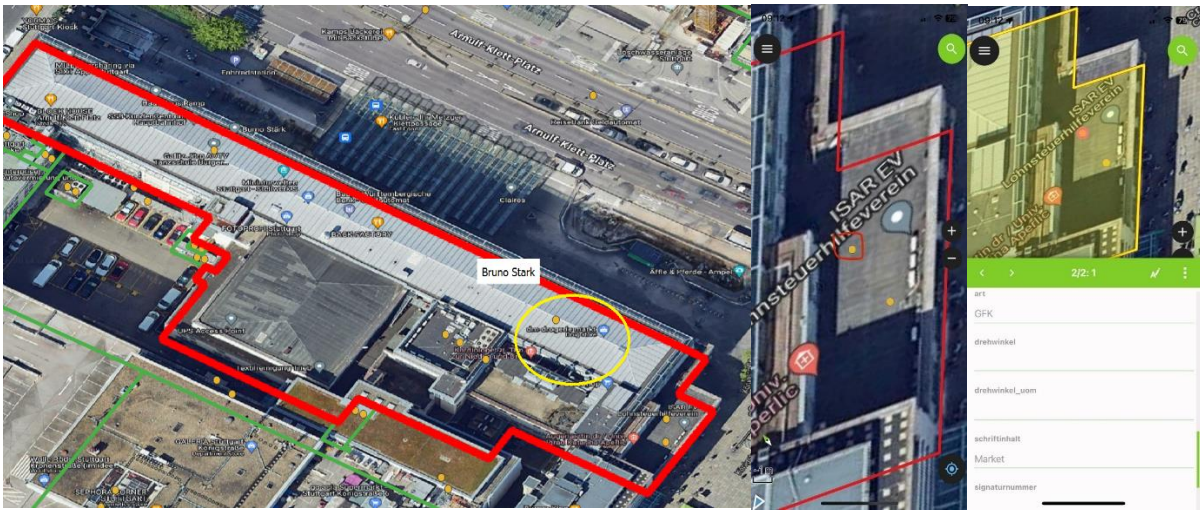


Figure 6. Example of adding points in the layer of the information that did not exists in the specific building polygon prior to this.

b) Data Update and Collection from Online Sources: Online sources such as OpenStreetMap and Google Maps were used to confirm existing data and gather additional information as needed. Some information, such as house numbers on buildings, was difficult to confirm in the field. These details were verified using OpenStreetMap, which clearly displays house numbers and other relevant information. The update is again made through the QField app Figure 8.

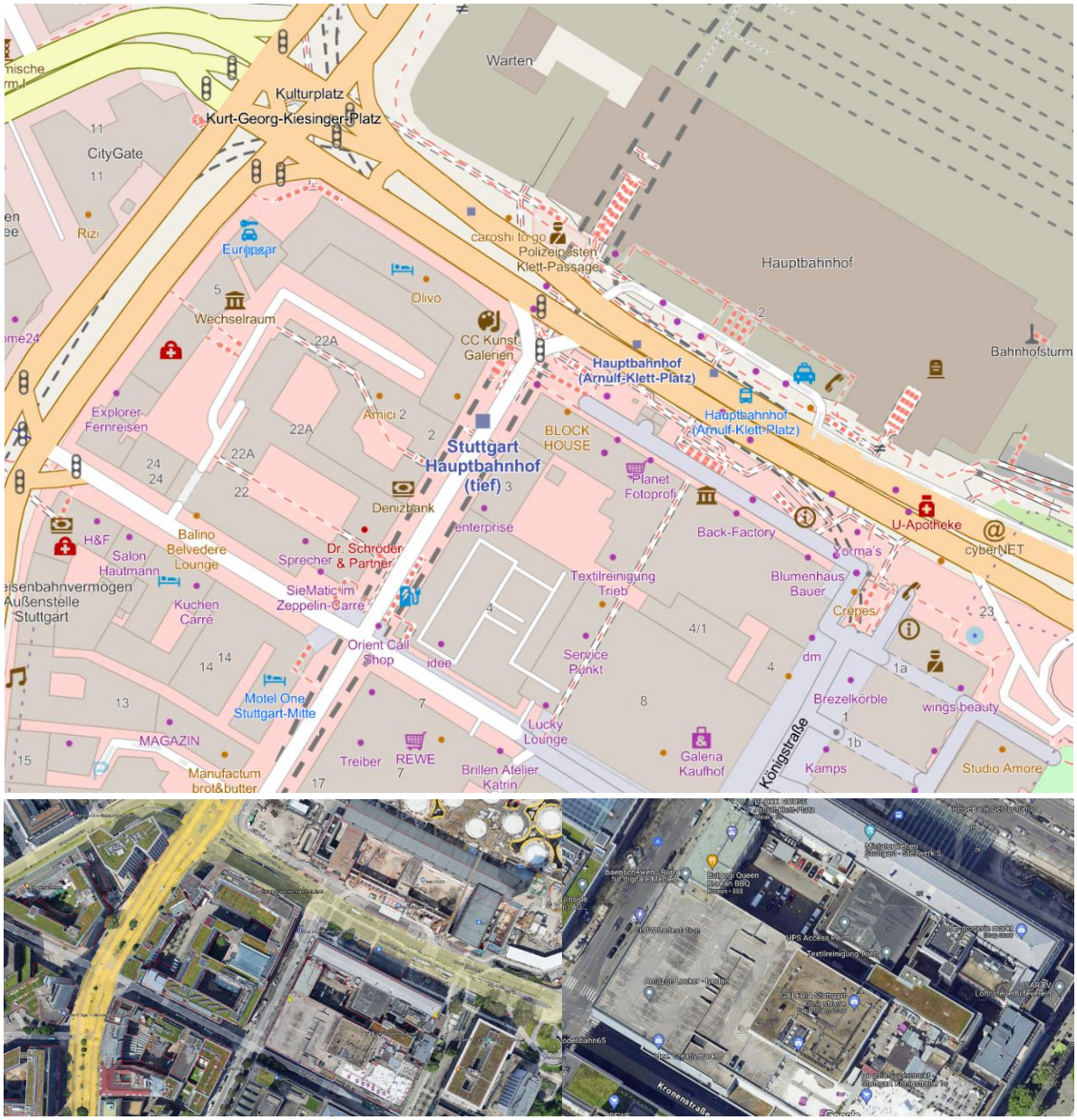
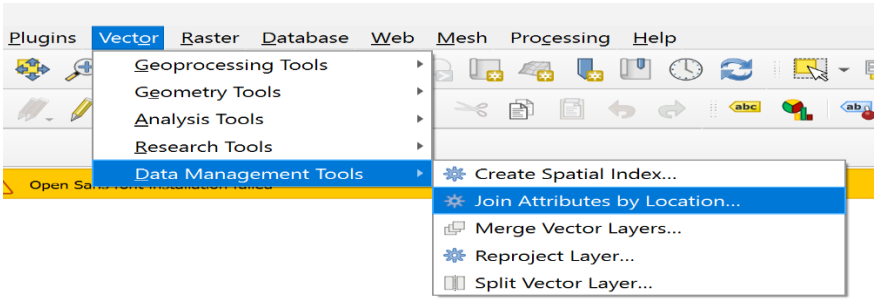


Figure 7. Visible information in Open Street map, Google Earth, and Google Maps

Figure 8. Example of confirming the already existing data in open street map and QField. HNR refers to house number and schriftinhalt 3 , house number 3 as confirmed in the open street map as well.

2.4. Spatial join of the attribute tables

Following the data capture and update of the attribute tables, the attribute table of the points is joined with that of the building polygons, including the relevant fields. This step prepares the joint tables for incorporation into a 3D model that will contain information from both the point table and the building polygon table.



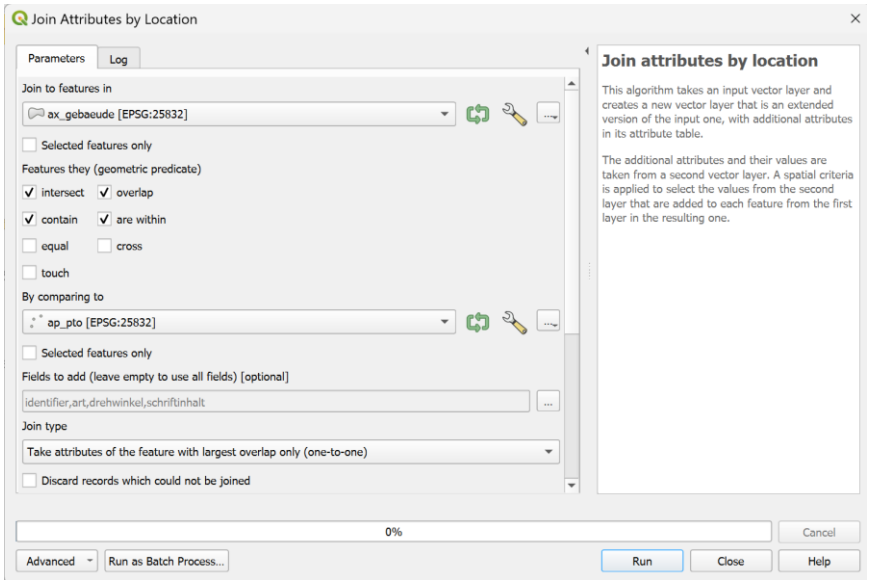


Figure 9. Join attribute by location tool in QGIS is used to join two attribute tables that do not have any common attribute however are spatially related to each other.

2.5. Data Storage and Synchronization

Data Synchronization: Captured field data was synchronized with the main database stored in PostgreSQL. This robust and scalable data storage solution ensured that all updates were accurately recorded and easily accessible for analysis. The setup process began with the creation of a PostgreSQL account. This involved installing PostgreSQL on a server and configuring it for remote access. After installation, a new user account with administrative privileges was created using the `CREATE USER` command, followed by the `CREATE DATABASE` command to establish a new database dedicated to the project.

The PostgreSQL database was then configured to support spatial data types and operations by enabling the PostGIS extension. PostGIS extends PostgreSQL to support geographic objects, allowing for the storage and query of spatial data. This was achieved by executing the `CREATE EXTENSION postgis;` command within the newly created database.

Once the database was set up and configured, the standardized cadastral data was imported. This process involved using tools like `pgAdmin` or command-line utilities such as `psql` to load the data into the database tables.

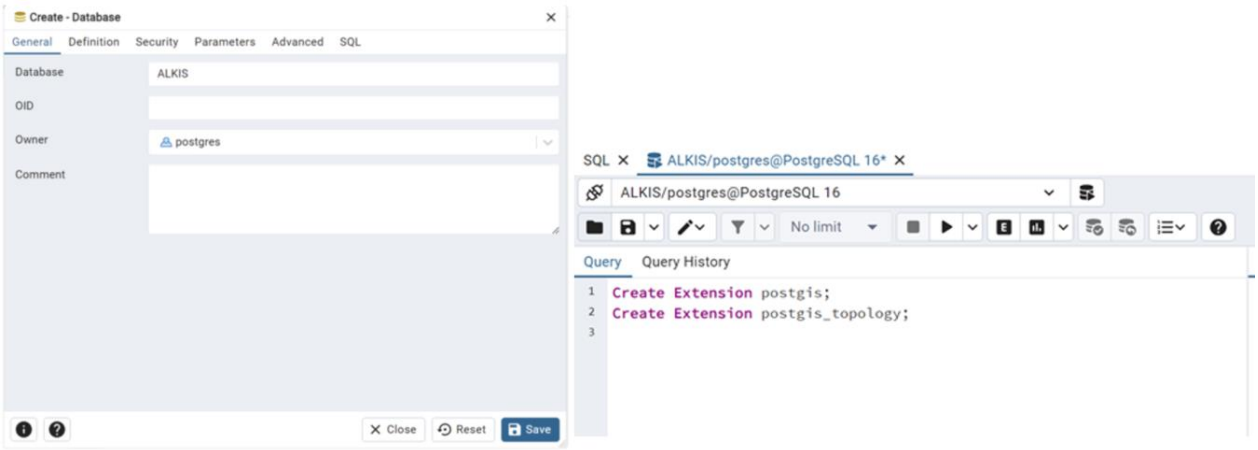


Figure 10: Database and Extensions Creation in PostgreSQL

The Next step is to connect the geodatabase to QGIS. The main objective of connecting our geodatabase to QGIS was that to make sure each time we add any information to the building or points attribute table in Qfield it will automatically be updated in the database in postgreSQL. The process involves creating a Postigis connection in QGIS and naming it ALKIS and for database connection we wrote the name of the database that we already created in our PostgreSQL (ALKIS). Host was localhost as everything was using same pc or same IP address as for port we used 5432 which is default port for postgreSQL.

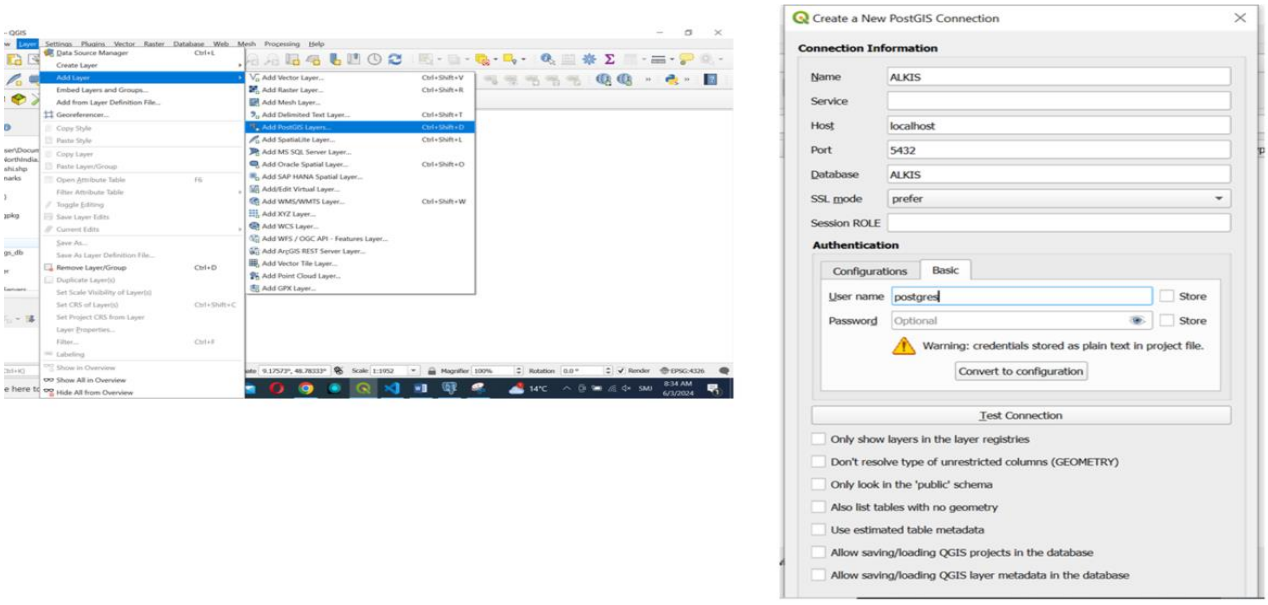


Figure 11: Creation of postgis connection in QGIS

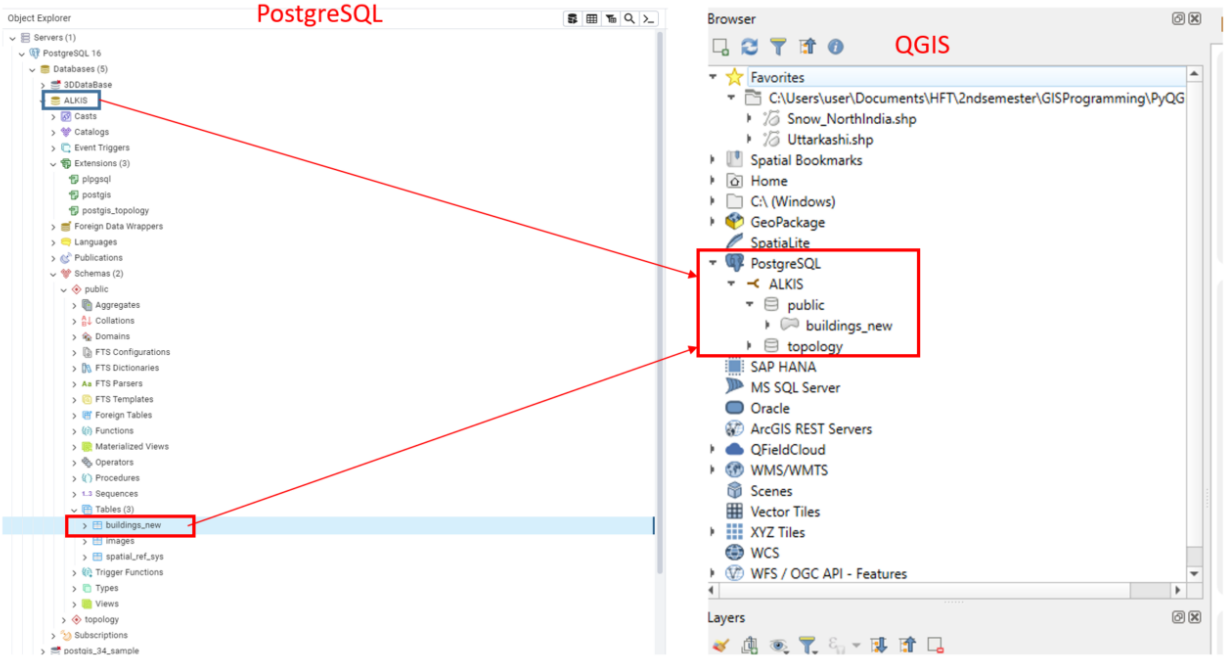


Figure 12: PostgreSQL together with QGIS interfaces after connection

2.6. 2D Spatial Analysis and Visualization

a) 2D Mapping: QGIS, an open-source Geographic Information System, was employed for 2D spatial analysis and visualization. QGIS offers powerful tools for mapping, allowing for the creation of detailed and informative 2D maps.

b) Building Function Visualization**:** The 2D visualization focused on the functional attributes of buildings, providing a clear representation of different building types and their uses within the study area. This helped in understanding the spatial distribution and functional zoning of the area.

2.7. 3D Modeling and Integration

There are few steps we had to follow in order to prepare the data for 3D visualization. The following will explain these steps

a) Data conversation

As we downloaded our 3D data as gml, it was very import to convert them into 3D Cesium data which can be supported by Cesium Ion. To do that we used FME software

Process:

- Importing CityGML data into FME: As seen in Figure 13, we used Add reader tool to import our CityGML into FME

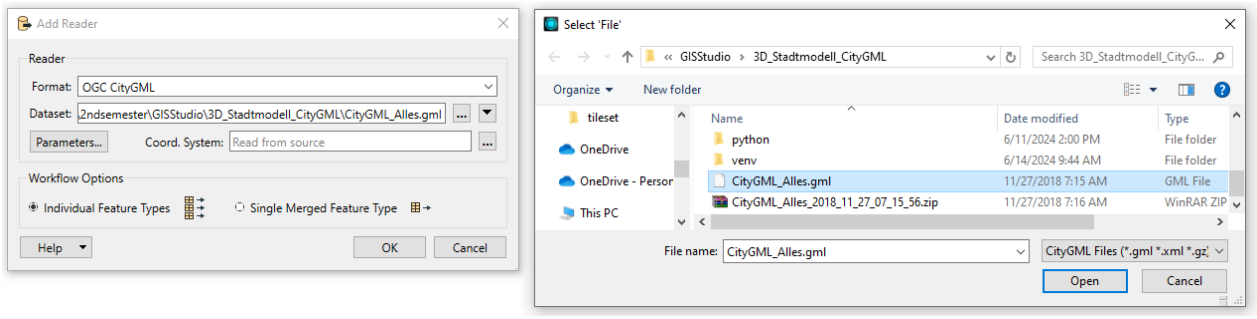


Figure 13: Importing CityGML data into FME

- Data conversion: As seen in Figure 14, we used Add writer tool to create a 3D Cesium tileset transformer.

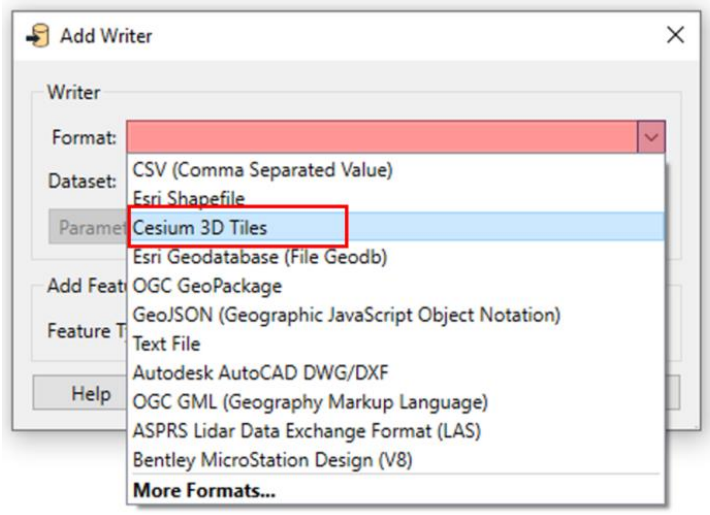


Figure 14: Add Writer with Cesium 3D Tiles as format

- Connecting imported cityGML to 3D cesium transformer: Next step was to connect our imported data with our transformer. As seen in Figure 15 for parameter we used 3D object as geometry and for attributes we only chose the ones that was imported for our project

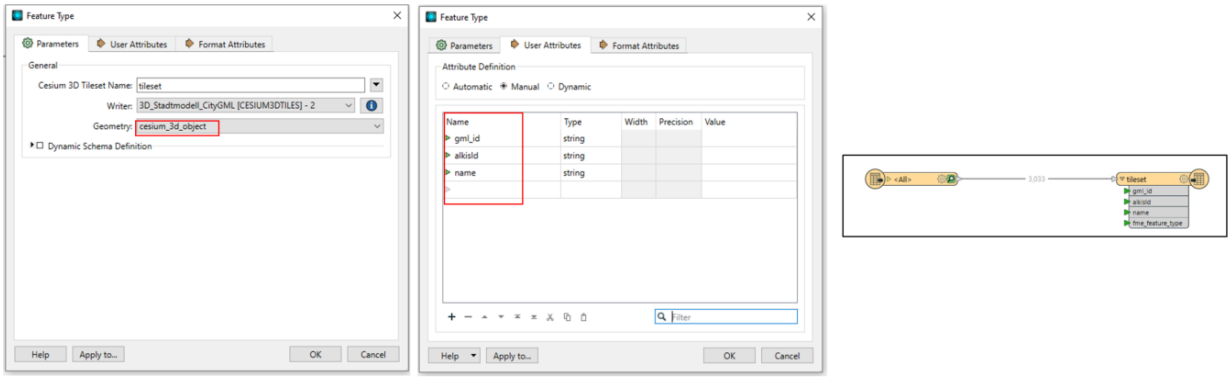


Figure 15: Parameters and Attributes

c) 3D Modeling**:** Cesium, a platform for 3D geospatial data visualization, was used to create detailed and interactive 3D models. This platform enabled the integration of updated cadastral data into a 3D environment, enhancing the visualization of the redevelopment area.

d) Attribute Integration: Relevant captured details from the 2D attribute table were assigned to the 3D models. This integration allowed for a more comprehensive understanding of each building, including its attributes and spatial context.

3. RESULT AND DISCUSSION

The results of the various phases of the project are discussed below, highlighting both the successes and the challenges encountered.

3.1. Data Extraction and Standardization

Two types of data is extracted from ALKIS sources:

a) XML file which consists of cadastral information Building polygons and points Figure 16.

b) GML file which consists of 3D model of the buildings limited to certain areas of Stuttgart Figure 17.

Figure 16. building polygon and points with their attribute tables showing the 2D coverage

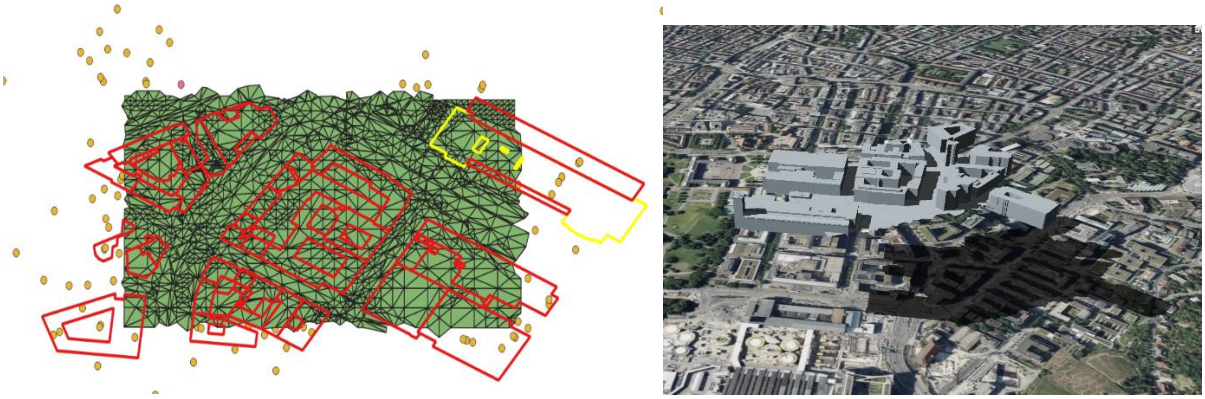


Figure 17. 3D model coverage in QGIS and Cesium

The data was then cleaned and transformed into a consistent format. Two of the attribute tables that we are dealing with is the building polygon or AX_Gebaeude and points or AP_PTO.

a) Building Polygon (AX_Gebaeude)

This layer consists of polygon data representing the footprints of buildings. The attributes that we are dealing with in this task are name and gebaeudefunktion.

Value	Gebaeudefunktion (German)	Gebaeudefunktion (English)
1123	Wohn-und Geschäftsgebäude	Residential and commercial building
2020	Bürogebäude	Office building
2050	Geschäftsgebäude	Commercial building
2071	Hotel, Motel, Pension	Hotel, motel, guesthouse
2081	Gaststätte, Restaurant	Inn, restaurant
2112	Betriebsgebäude	Industrial building
2461	Parkhaus	Parking garage
2463	Garage	Garage

2465	Tiefgarage	Underground parking garage
3010	Verwaltungsgebäude	Administrative building
3090	Empfangsgebäude	Reception building

Table 1. attribute present in the building polygon, describing the function of the building with the values. The meaning to each value is taken from ALKIS® Objektartenkatalog NRW 7.1[4] source

b) Point layer (AP_PTO)

This layer consists of point data representing various cadastral points related to the buildings, streets and land parcels. The key columns or attributes that we are dealing with in points layer are art and schriftinhalt. Art refers to the type or category of a cadastral entity. Schriftinhalt provides textual information associated with a cadastral entity.

Abbreviations	Description
GFK	Bulding functionality
HNR	Haus Number
ZAE_NEN	zaehler nenner
ATP	archaeologischerTyp
BEZ	Bezeichnung (Designation)
FKT	Function describes what purpose the building serves.
Fliessgewaesser	Flowing water
Friedhof	Cemetery
Halde_LGT	Stockpile
NAM	Name is the designation associated with a 'stockpile' or its proper name
OFL	Position relative to ground surface' is the indication of the relative position of 'Storage tank, storage structure' to the earth's surface.
PKN	Point identifier' is an ordinal feature assigned by the cadastral authority
Platz	Name' is the proper name of 'place'.
Strasse	Street name

Table 2. description of the german alkis data terms (source: Professor Ostadabbas, Hamidreza)

Nummer	Beschreibung	Bezeichnung ALKIS
1	Wohnhaus	Whs
2	Wohn- und Wirtschaftsgebäude	WWg
3	Wohn- und Verwaltungsgebäude	WVwg
4	Wohn- und Geschäftsgebäude	WGhs
5	Wohn- und Bürogebäude	WBüro
6	Wohn- und Betriebsgebäude	WBtrg
7	Bürogebäude	Büro
8	Verwaltungsgebäude	Vwg
9	Geschäftshaus	Ghs
10	Rathaus	Rathaus
11	Post	Post
12	Gaststätte/Restaurant	Gast
13	Kinderkrippe/Kindergarten/Kindertagesstätte	Kiga

[4] *ALKIS®Objektartenkatalog NRW 7.1 ALKIS-OK NRW 7.1 basierend auf dem AFIS-ALKIS-ATKIS®-Anwendungsschema 7.1.0*. (2021).

14	Allgemeinbildende Schule	Schule
15	Hochschule	Hochsch
16	Forschungsinstitut	Forsch
17	Bibliothek, Bücherei	Bibl
18	Sporthalle/Turnhalle	Sporth
19	Veranstaltungsgebäude	Veranst
20	Hallenbad	Hbad
21	Gemeindehaus	Gdehs
22	Gebäude für Sportzwecke	Sportg
23	Freizeitheim/Vereinsheim/Dorfhaus/ Gemeinschaftshaus/Bürgerhaus	Frzhm
24	Krankenhaus	Krhs
25	Seniorenheim	Senhm
26	Sanatorium	Sanat
27	Gebäude für Kurbetrieb	Kurg
28	Wohnheim	Heim
29	Hotel/Motel/Pension	Hotel
30	Jugendherberge	JH
31	Museum	Museum
32	Feuerwehr	Fwg
33	Polizei	Polizei
34	Gericht	Gericht
35	Justizvollzugsanstalt	JVA
36	Wirtschaftsgebäude	Wirtg
37	Empfangsgebäude	Empfg
38	Schloss	Schloss
39	Kiosk	Kiosk
40	Freizeit- und Vergnügungsstätte	Vergnst
41	Werkstatt	Wkst
42	Fabrik	Fabr
43	Betriebsgebäude	Btrg
44	Gewächshaus	Gewhs
45	Kirche	Kirche
46	Kapelle	Kapelle
47	Synagoge	Synagoge
48	Gebäude für religiöse Zwecke	Relg
49	Friedhofsgebäude	Fhfg
50	Stall	Stall
51	Schuppen	Schu
52	Scheune und Stall	Scheu-St
53	Scheune	Scheu
54	Gebäude für Vorratshaltung	Lagg
55	Parkplatz	PPL
56	Garage	Gar
57	Parkhaus	Phs
58	Tiefgarage	Tgar
59	Friedhof	FHF

60	Bahnverkehr	BVK
61	Gebäude zur Energieversorgung	Energie
62	Umformer	Ust
63	Burg/Festung	Burg

Table 3. Schriftinhalt attribute term description (Source: professor Ostadabbas, Hamidreza)

By extracting, cleaning, and standardizing this data, the project ensured a robust foundation for subsequent 2D and 3D visualization and analysis.

3.2. Captured Data and 2D Data Visualization

There are around 20 new points added to the point layers and only few fields were updated in the name section of the polygon attribute table. All of the information of the point layer is joint with the building polygon layer. This is needed when incorporating the joint layer into 3D model.

art	hwin	winke	schriftinhalt
GFK	N...	N...	Krhs
GFK	N...	N...	Gast
GFK	N...	N...	Büro
GFK	N...	N...	Gast
GFK	N...	N...	Gast
GFK	N...	N...	Gast
GFK	N...	N...	Sportg
GFK	N...	N...	Gast
GFK	N...	N...	ghs
HNR	N...	N...	1
HNR	N...	N...	8
GFK	N...	N...	Phs (Parkhaus)
GFK	N...	N...	Market
GFK	N...	N...	Market

Figure 18. Example of some of the new points that were added to the layer

When it comes to 2D visualization of the cadastral data, the visualization is made based on the gebaeudefunktion of the buildings. Added are street names as well.

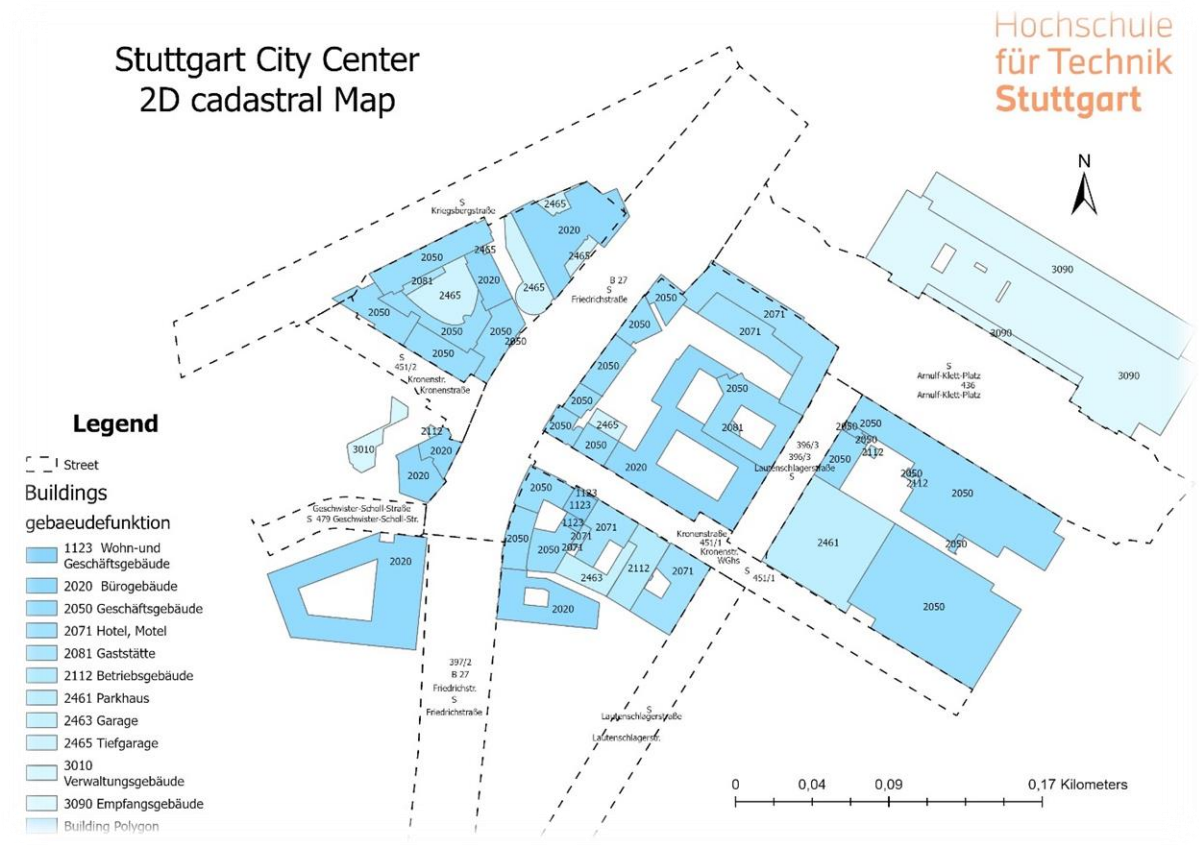


Figure 19. 2D visualization of the ALKIS data

3.3. 3D Modeling and Integration

a) 3D Visualisation in Cesium Ion

To view the 3D cesium tileset in cesium ion, the codes were referenced similar to the ones found on the Cesium platform (Cesium, n.d.). Once the 3D model was visible, the issue of a floating building was encountered as seen in Figure 20. To resolve the 'floating building' issue we used the same platform (Cesium, n.d.), the option of 3D riles Adjust Height was utilized as seen in Figure 21, there the sample code together with the code we had previously was used to resolve the height issue. Figure 22 shows the final results where the terrain issue are resolved.

Figure 20: Buildings Floating

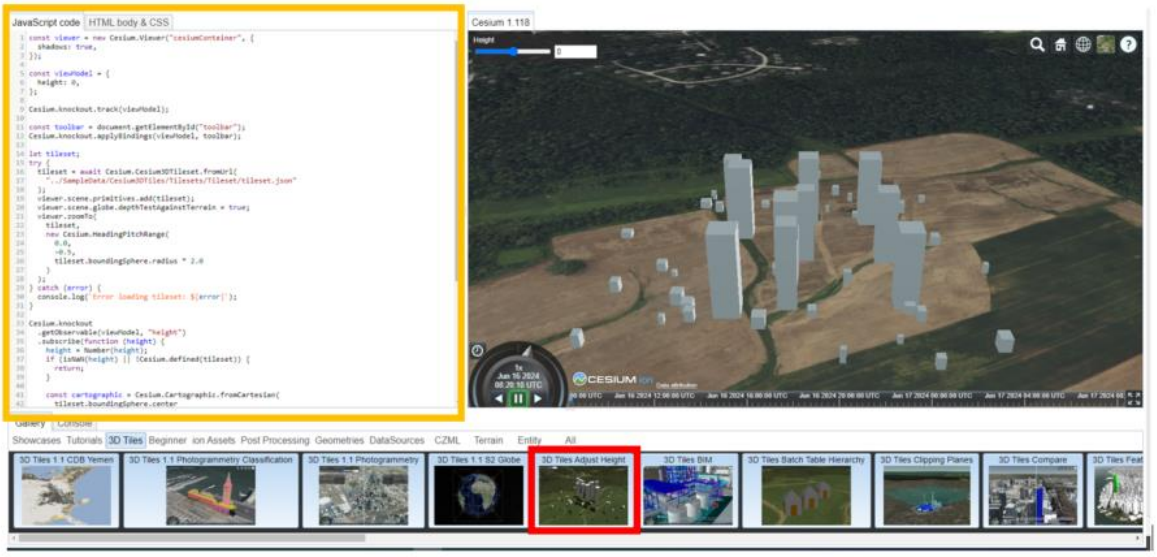


Figure 21: 3D tiles Adjust Height

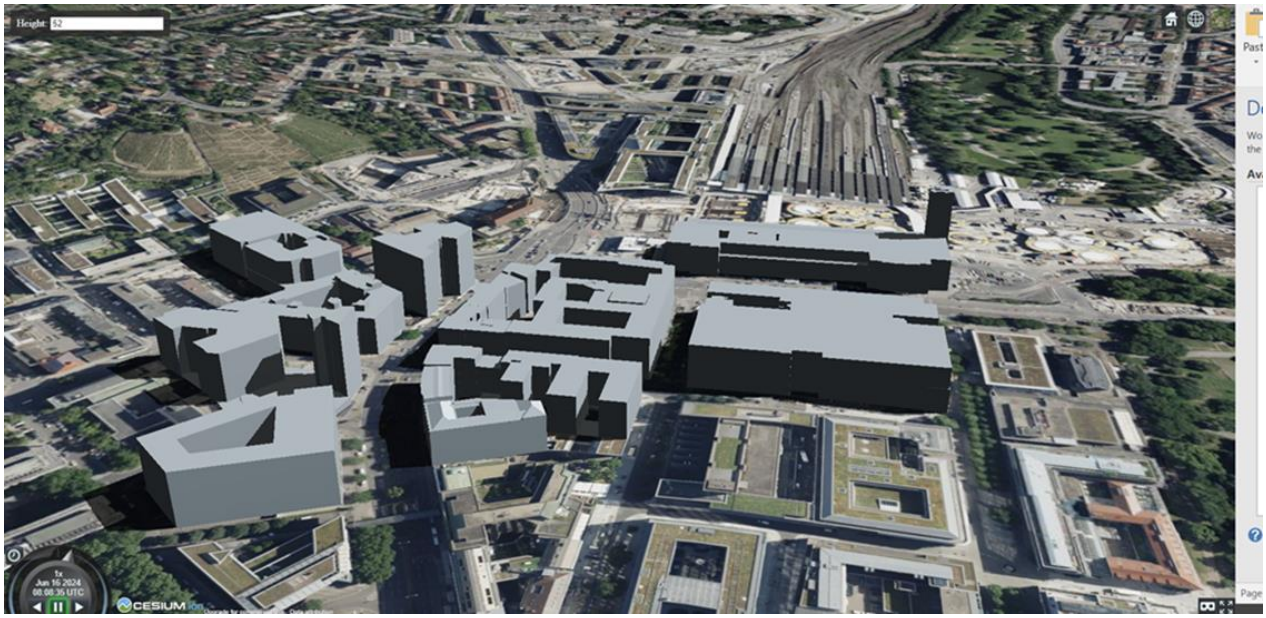


Figure 22: Terrain issue resolved

b) CONNECTING 3D and 2D.

For there to be a connection between the 3D cesium tileset data and the ALKIS data from postgreSQL, there should be a common identifier between the two. In the case of this project, there is no common identifier in the attribute table. To resolve that issue we had to use the Join Attributes by location tool in QGIS. To do so, the 3D cesium tileset was first transformed into 2D Esri Shapefile in FME.

Process:

As seen in Figure 23, we added a new writer with Esri Shapefile as the format. We then connected it with the already imported CityGML data in FME. For parameters, the Shapefile polygon was chosen, and for the attribute, only gml_id was selected, as we wanted gml_id to be used as the common identifier with the 2D data.

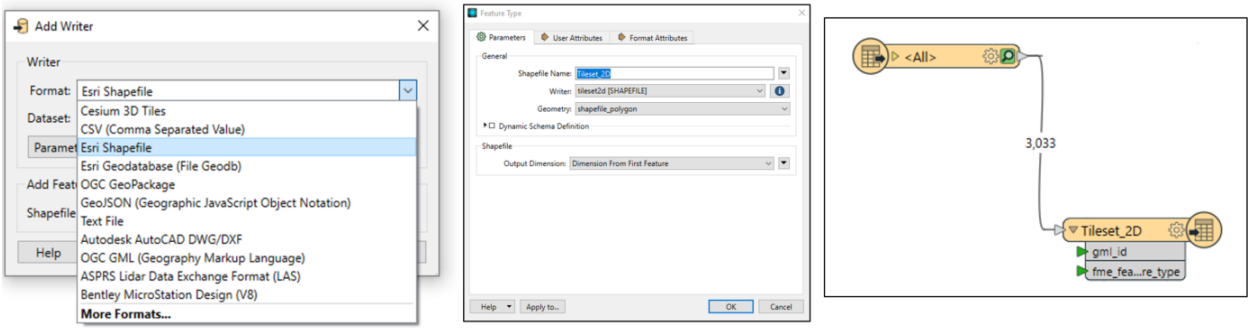


Figure 23: 2D Esri shapefile

Join attributes by location in QGIS: At this point we had to use the join attributes by location so that our 2D data would have a common identifier. In Figure 24, we can see that now after joining that our 2D alkis data now has a gml_id column.

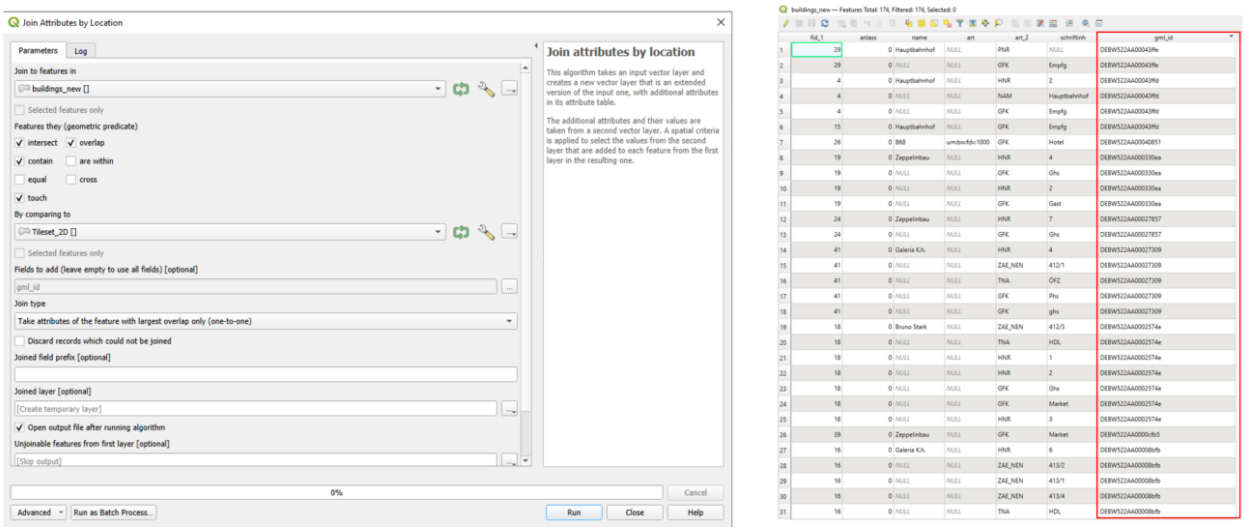


Figure 24: gml_id as common identifier

Connecting postgreSQL Database and Cesium: To do that we had to create a backend file that fetched data from postgresQL and in Figure 25, we can see a sample of the codes that we used for the connection.

```python
from flask import Flask, jsonify, send_from_directory
from flask_cors import CORS
import psycopg2

app = Flask(__name__)
CORS(app)

# Database connection configuration
db_config = {
    'host': 'localhost',
    'database': 'ALKIS',
    'user': 'postgres',
    'password': 'postgres',
    'port': 5432
}

def get_db_connection():
    conn = psycopg2.connect(**db_config)
    return conn
```

Figure 25: PostgreSQL and Cesium connection

- Adding images to the buildings: To do so we had to first add a new column and to our attribute table and gave it the name ***image_url.*** Then we gave the name to our images based on the gml_ids of the buildings they presents. In Figure 25, we can see the code that we used to update our image_url where each building was assigned to it's corresponding image based on the gml_id

ALKIS/postgres@PostgreSQL 16*

ALKIS/postgres@PostgreSQL 16

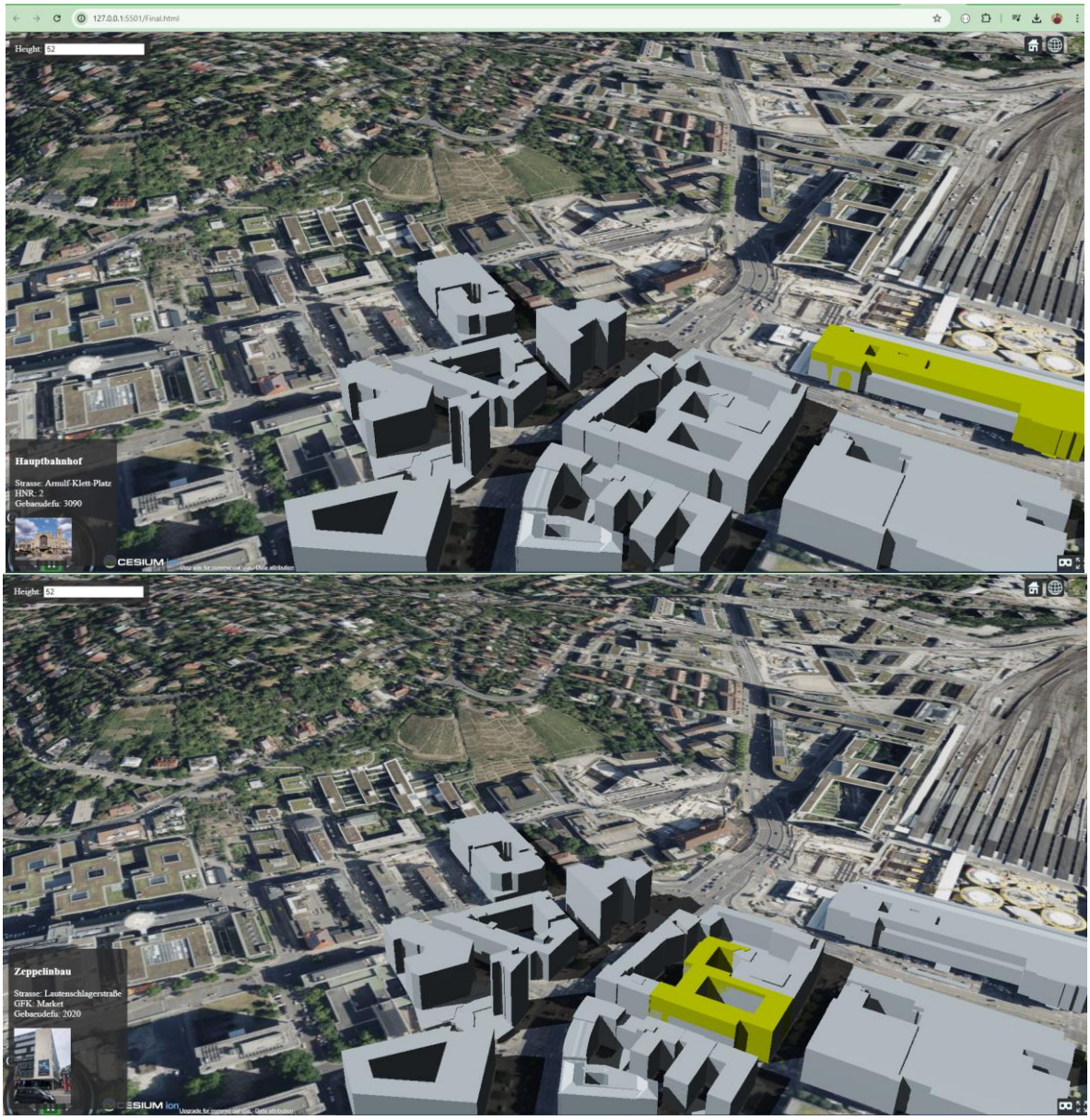
No limit

Query | Query History

```
1  UPDATE buildings_new SET image_url = 'C:\static\images\DEBW522AA0002574e.png' WHERE gml_id = 'DEBW522AA0002574e';
2
3
```

Figure 26: Updating our image_url attribute table

Following images are the final result



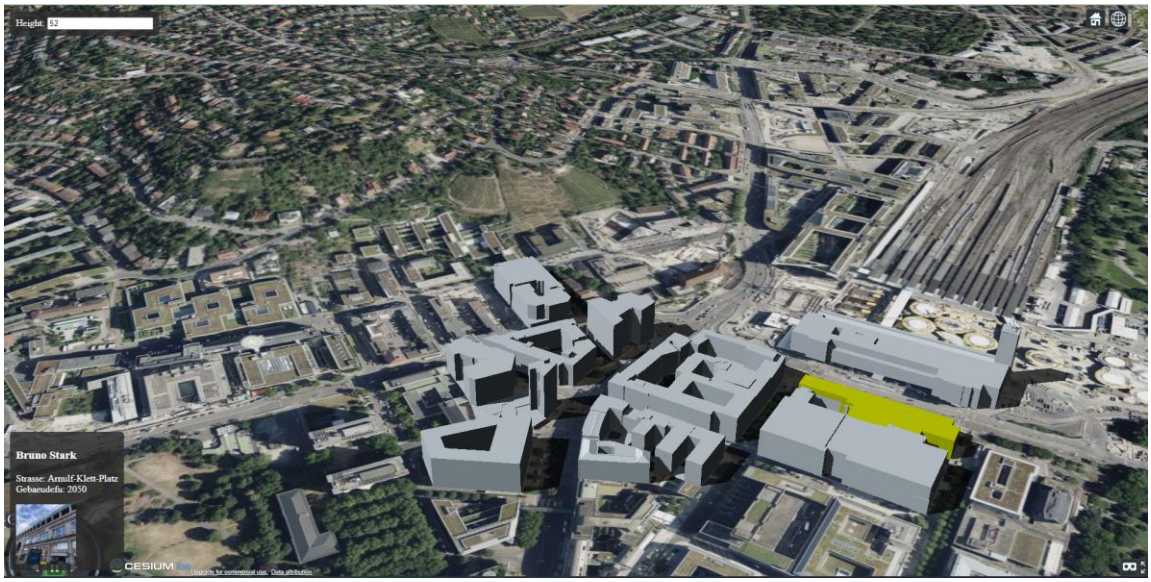


Figure 27: Final Results of the 3D map

In the final results, click on each of the individual building shows ALKIS information and as well as the corresponding images

Summary of key findings:

- **Floating Buildings Issue:** Initially, the 3D models had issues with floating buildings, where some structures were not correctly aligned with the ground. This problem was identified and resolved, ensuring accurate and realistic representation in the 3D environment. Correcting these issues was crucial for maintaining the integrity and usability of the 3D models.
- **Attribute Integration:** Relevant attributes from the 2D maps were successfully integrated into the 3D models. This integration allowed for detailed building information to be accessed directly within the 3D environment. Users could click on buildings in the 3D model to retrieve information such as building function, height, and other attributes, significantly enhancing the utility of the 3D models for analysis and decision-making.
- **Image Integration Challenges:** Initially, integrating images into the 3D models presented challenges, such as the same image appearing for all polygons or images not displaying at all. To resolve this, we added a new column named image_url to the attribute table. Each image was named based on the gml_ids of the buildings they represent. A script then updated the image_url column, ensuring that each building was correctly linked to its corresponding image. This solution successfully integrated images into the 3D models, enhancing their richness and providing more detailed visual context

4. CONCLUSION

The project successfully updated the attribute table for ALKIS data, implementing changes smoothly without errors. One minor challenge arose in finding information for the name column in the polygon attribute table. A 2D visualization based on building functions was created seamlessly, and data storage was established without complications.

For the 3D model in Cesium, we resolved the issue of floating buildings and effectively assigned relevant captured details from the ALKIS attribute table to the 3D model. Although initially faced with challenges in integrating images into the 3D models, this was resolved by adding an image_url column to the attribute table and linking images based on gml_ids.

The project successfully enhanced our understanding and visualization of Stuttgart's city center redevelopment needs by creating detailed 2D and 3D models. This work demonstrated the importance of accurate, detailed, and interactive models in modern urban planning and cadastral management. The integration of advanced data management and visualization techniques highlighted the significance of 3D modeling in providing a comprehensive and visually appealing representation of urban redevelopment projects.

REFERENCES

- *ALKIS®Objektartenkatalog NRW 7.1 ALKIS-OK NRW 7.1 basierend auf dem AFIS-ALKIS-ATKIS®-Anwendungsschema 7.1.0*. (2021).
- Cesium. (n.d.). *Cesium Sandcastle*. Retrieved from 3D Tiles Visualisation: https://sandcastle.cesium.com/?src=3D%20Tiles%20Feature%20Styling.html
- Niedersächsische Vermessungs- und Katasterverwaltung. (2010). *Basiswissen ALKIS / ETRS 89*.
- https://de.m.wikipedia.org/wiki/Datei:Locator_map_Verband_Region_Stuttgart_in_Germany.svg (Germany Map)
- Accessed from the ALKIS website: https://www.stuttgart.de/leben/bauen/geoportal/open-data-und-testdaten.php

APPENDICES

Backend codes.

```
from flask import Flask, jsonify, send_from_directory
from flask_cors import CORS
import psycopg2

app = Flask(__name__)
CORS(app)

# Database connection configuration
db_config = {
    'host': 'localhost',
    'database': 'ALKIS',
    'user': 'postgres',
    'password': 'postgres',
    'port': 5432
}

def get_db_connection():
    conn = psycopg2.connect(**db_config)
    return conn

@app.route('/attributes', methods=['GET'])
def get_attributes():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('SELECT gml_id, name, Strasse, HNR, GFK, gebaeudefu,
image_url FROM buildings_new')
    rows = cursor.fetchall()
    cursor.close()
    conn.close()

    attributes = [{'gml_id': row[0], 'name': row[1], 'Strasse': row[2], 'HNR':
row[3], 'GFK': row[4], 'gebaeudefu': row[5], 'image_url': row[6]} for row in
rows]
    return jsonify(attributes)

@app.route('/images/<path:filename>', methods=['GET'])
def serve_image(filename):

    return send_from_directory('static/images', filename)

if __name__ == '__main__':
    app.run(debug=True)
```

Frontend codes.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <script
src="https://cesium.com/downloads/cesiumjs/releases/1.93/Build/Cesium/Cesium.j
s"></script>
    <link
href="https://cesium.com/downloads/cesiumjs/releases/1.93/Build/Cesium/Widgets
/widgets.css" rel="stylesheet">
    <style>
        html,
        body,
        #cesiumContainer {
            width: 100%;
            height: 100%;
            margin: 0;
            padding: 0;
            overflow: hidden;
        }

        #toolbar {
            position: absolute;
            top: 10px;
            left: 10px;
            background: rgba(42, 42, 42, 0.8);
            padding: 10px;
            border-radius: 5px;
            z-index: 1000;
        }

        #toolbar label {
            color: white;
        }

        #infoBox {
            position: absolute;
            bottom: 10px;
            left: 10px;
            background: rgba(42, 42, 42, 0.8);
            padding: 10px;
            border-radius: 5px;
            display: none;
            color: white;
        }

        #infoBox img {
```

```
            max-width: 100px;
            height: auto;
            display: block;
            margin-top: 10px;
        }
    </style>
</head>

<body>
    <div id="cesiumContainer"></div>
    <div id="toolbar">
        <label for="height">Height:</label>
        <input type="number" id="height" data-bind="value: height,
valueUpdate: 'input'">
    </div>
    <div id="infoBox">
        <h3 id="infoTitle"></h3>
        <p id="infoDescription"></p>
        <img id="infoImage" src="" alt="Building Image">
    </div>
    <script>
        Cesium.Ion.defaultAccessToken =
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJjYTgxYWNiMy1hYTA2LTQ5NTctYTg4
ZC03Y2U1NWRjODFiNjciLCJpZCI6MjE0ODc0LCJpYXQiOjE3MTU2MTgzMTR9.4stNA85SiPJzkuJcm
vBm8SKXNKOvyNr2aaP5tkGIs60';

        const viewer = new Cesium.Viewer('cesiumContainer', {
            terrainProvider: Cesium.createWorldTerrain(),
            baseLayerPicker: true,
            vrButton: true,
            geocoder: false,
            navigationHelpButton: false,
            selectionIndicator: false,
            shadows: true,
            timeline: false,
            sceneModePicker: true,
        });

        const defaultHeight = 52;
        const viewModel = {
            height: defaultHeight,
        };

        Cesium.knockout.track(viewModel);

        const toolbar = document.getElementById("toolbar");
        Cesium.knockout.applyBindings(viewModel, toolbar);
```

```
const tilesetUrl = "./tileset.json";

const tileset = new Cesium.Cesium3DTileset({
    url: tilesetUrl
});

viewer.scene.primitives.add(tileset);

let selectedFeature;
let originalColor = new Cesium.Color();

tileset.readyPromise.then(function () {
    console.log('Tileset loaded successfully');

    const cartographic = Cesium.Cartographic.fromCartesian(
        tileset.boundingSphere.center
    );
    const surface = Cesium.Cartesian3.fromRadians(
        cartographic.longitude,
        cartographic.latitude,
        0.0
    );
    const offset = Cesium.Cartesian3.fromRadians(
        cartographic.longitude,
        cartographic.latitude,
        defaultHeight
    );
    const translation = Cesium.Cartesian3.subtract(
        offset,
        surface,
        new Cesium.Cartesian3()
    );
    tileset.modelMatrix = Cesium.Matrix4.fromTranslation(translation);

    viewer.flyTo(tileset);

    fetch('http://localhost:5000/attributes')
        .then(response => response.json())
        .then(data => {
            console.log(data);
            tileset.tileLoad.addEventListener((tile) => {
                const content = tile.content;
                const featuresLength = content.featuresLength;

                for (let i = 0; i < featuresLength; i++) {
                    const feature = content.getFeature(i);
                    const gml_id = feature.getProperty('gml_id');
```

```
                            const attribute = data.find(item => item.gml_id
=== gml_id);

                            if (attribute) {
                                feature.setProperty('name', attribute.name);
                                feature.setProperty('Strasse',
attribute.Strasse);
                                feature.setProperty('HNR', attribute.HNR);
                                feature.setProperty('GFK', attribute.GFK);
                                feature.setProperty('gebaeudefu',
attribute.gebaeudefu);
                                feature.setProperty('image_url',
`http://localhost:5000/images/${attribute.gml_id}.png`);
                            }
                        }
                    });

                    viewer.screenSpaceEventHandler.setInputAction(function
onLeftClick(movement) {
                        const pickedFeature =
viewer.scene.pick(movement.position);
                        if (Cesium.defined(pickedFeature) &&
pickedFeature.tileset === tileset) {
                            if (Cesium.defined(selectedFeature)) {
                                selectedFeature.color = originalColor;
                            }
                            selectedFeature = pickedFeature;
                            Cesium.Color.clone(pickedFeature.color,
originalColor);
                            pickedFeature.color = Cesium.Color.YELLOW;

                            const name = pickedFeature.getProperty('name');
                            const Strasse =
pickedFeature.getProperty('Strasse');
                            const HNR = pickedFeature.getProperty('HNR');
                            const GFK = pickedFeature.getProperty('GFK');
                            const gebaeudefu =
pickedFeature.getProperty('gebaeudefu');
                            const imageUrl =
pickedFeature.getProperty('image_url');

                            document.getElementById('infoTitle').textContent =
name || 'No name available';
                            let description = '';

                            if (Strasse) description += `Strasse:
${Strasse}<br>`;
                            if (HNR) description += `HNR: ${HNR}<br>`;
```

```
                            if (GFK) description += `GFK: ${GFK}<br>`;
                            if (gebaeudefu) description += `Gebaeudefu:
${gebaeudefu}<br>`;

                            document.getElementById('infoDescription').innerHT
ML = description;
                            document.getElementById('infoImage').src =
imageUrl || '';
                            document.getElementById('infoBox').style.display =
'block';
                        } else {
                            if (Cesium.defined(selectedFeature)) {
                                selectedFeature.color = originalColor;
                                selectedFeature = undefined;
                            }
                            document.getElementById('infoBox').style.display =
'none';
                        }
                    }, Cesium.ScreenSpaceEventType.LEFT_CLICK);
                })
                .catch(error => console.error('Error fetching attribute
data:', error));
        }).otherwise(function (error) {
            console.log(`Error loading tileset: ${error}`);
        });

        Cesium.knockout.getObservable(viewModel, "height").subscribe(function
(height) {
            height = Number(height);
            if (isNaN(height)) {
                console.log('Invalid height value');
                return;
            }

            if (!Cesium.defined(tileset)) {
                console.log('Tileset not defined');
                return;
            }

            const cartographic = Cesium.Cartographic.fromCartesian(
                tileset.boundingSphere.center
            );
            const surface = Cesium.Cartesian3.fromRadians(
                cartographic.longitude,
                cartographic.latitude,
                0.0
            );
            const offset = Cesium.Cartesian3.fromRadians(
```

```
                cartographic.longitude,
                cartographic.latitude,
                height
            );
            const translation = Cesium.Cartesian3.subtract(
                offset,
                surface,
                new Cesium.Cartesian3()
            );
            tileset.modelMatrix = Cesium.Matrix4.fromTranslation(translation);
        });
    </script>
</body>

</html>
```