Kaufmann, J. (2023). Digitaler Baueingabeprozess [Digitales Bild]. DALL-E von OpenAI.

Digitale Baueingabe

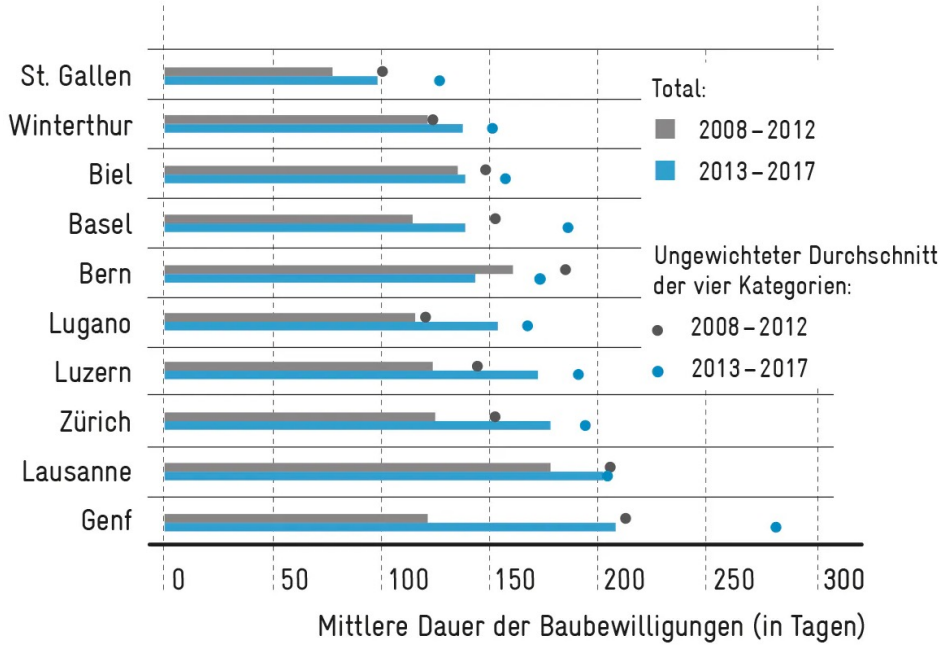
Automatisierter Vorprüfungsprozess

MEP Modul Digital Construction Programmierung 1 – (TA.BA_DC_SCRIPT_MM.H2301)

Inhalte

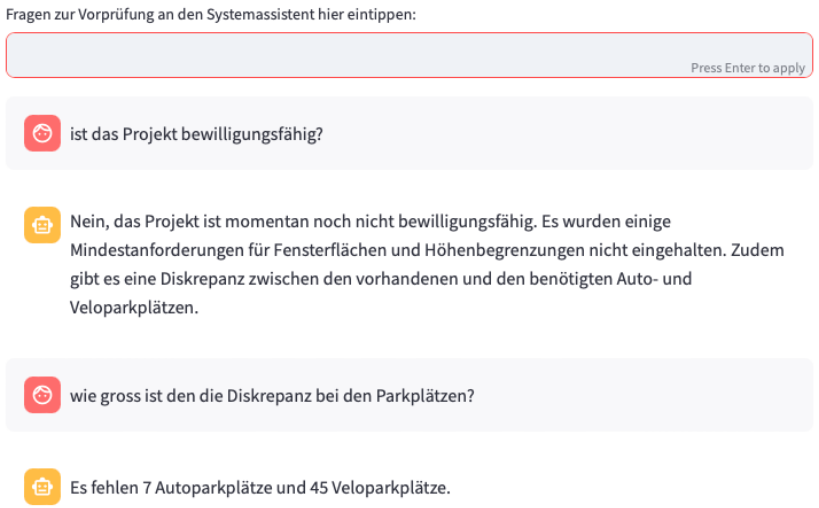
A) Absichten
B) Programmlogik
C) Prototyp
D) Technisches Highlight

Jan Kaufmann



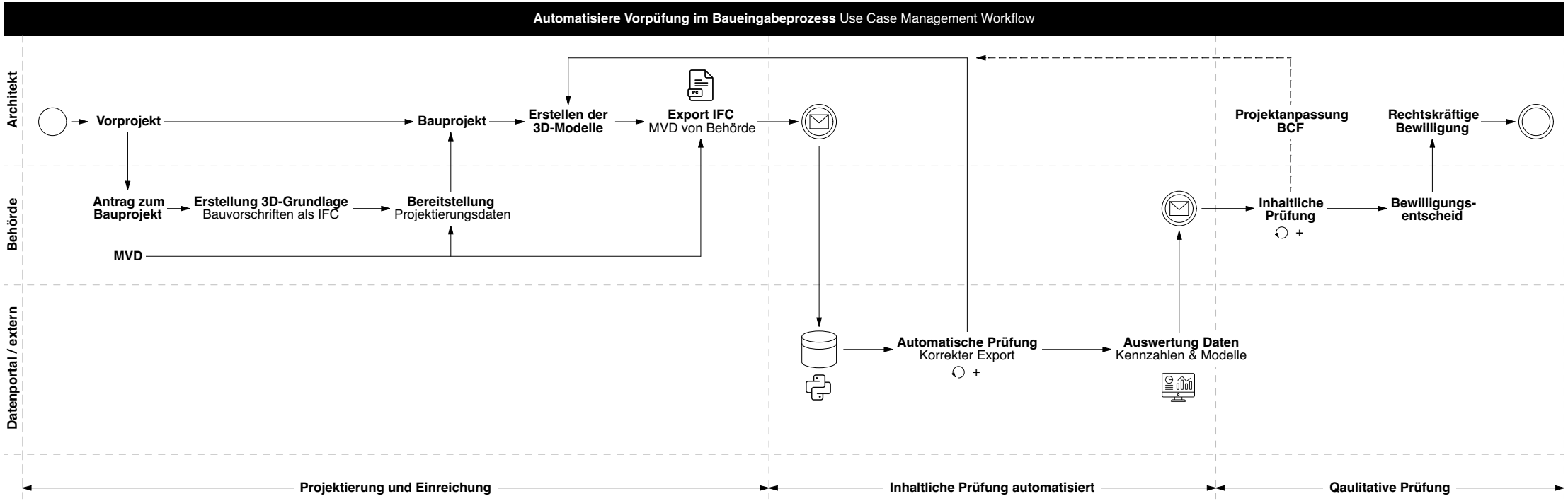
- *Formulare können **unvollständig** eingereicht sein*
- *Dokumente sind **alleinstehend** und schwer untereinander zu vernetzen*
- *Kennzahlen können **fehlerhaft** ermittelt sein und müssen kontrolliert werden (Vorprüfen)*
- *Projekte werden immer **komplexer***

Rühli, L., Schnell, F., & Müller-Jentsch, D. (2019, September 19). Baubewilligungen dauern immer länger. Avenir Suisse. https://www.avenir-suisse.ch/baubewilligungen-dauern-immer-laenger/



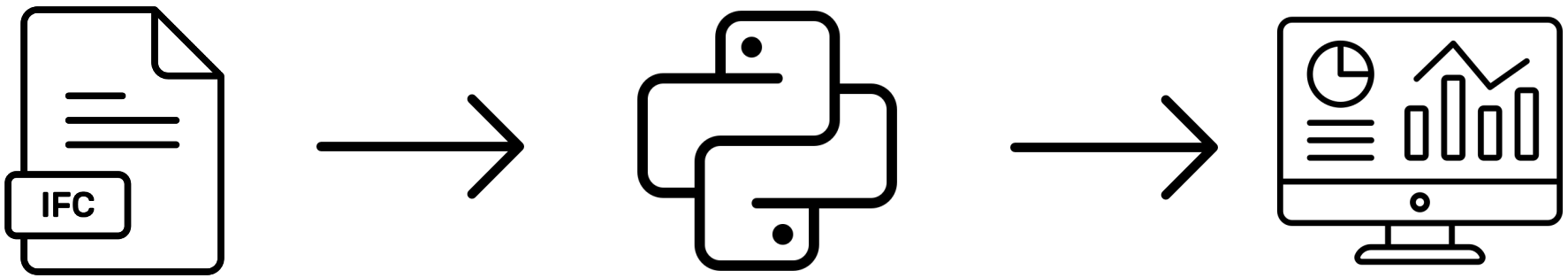
«Sobald wir den Moment erreichen, an dem künstliche Intelligenz Daten mit einer Präzision analysieren kann, die der menschlichen Fähigkeit gleichkommt, haben wir die grundlegenden Herausforderungen gemeistert.»

Kaufmann, J. (2024). Screenshot des Prüfportals [Screenshot der App]. Abgerufen von https://baueingabe-vorpruefung.streamlit.app

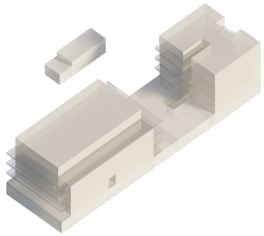


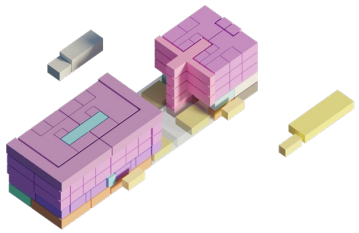
Kaufmann, J. (2023, 5. Dezember). Automatisiere Vorprüfung im Baueingabeprozess Use Case Management Workflow [Diagramm]. Eigene Erstellung.

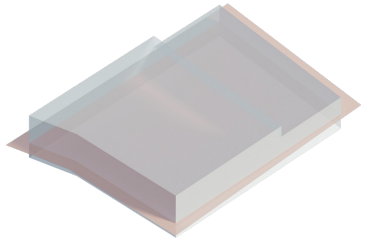
Mit dem Prozess sollen die administrativen Aufwände zur Prüfung gesenkt werden



Digitale Baueingabe IFC4 ***Digitale Vorprüfung*** ***Vorprüfungsdashboard***







Elementplan (Modellierungsrichtlinie)

_VOM (IFC4)

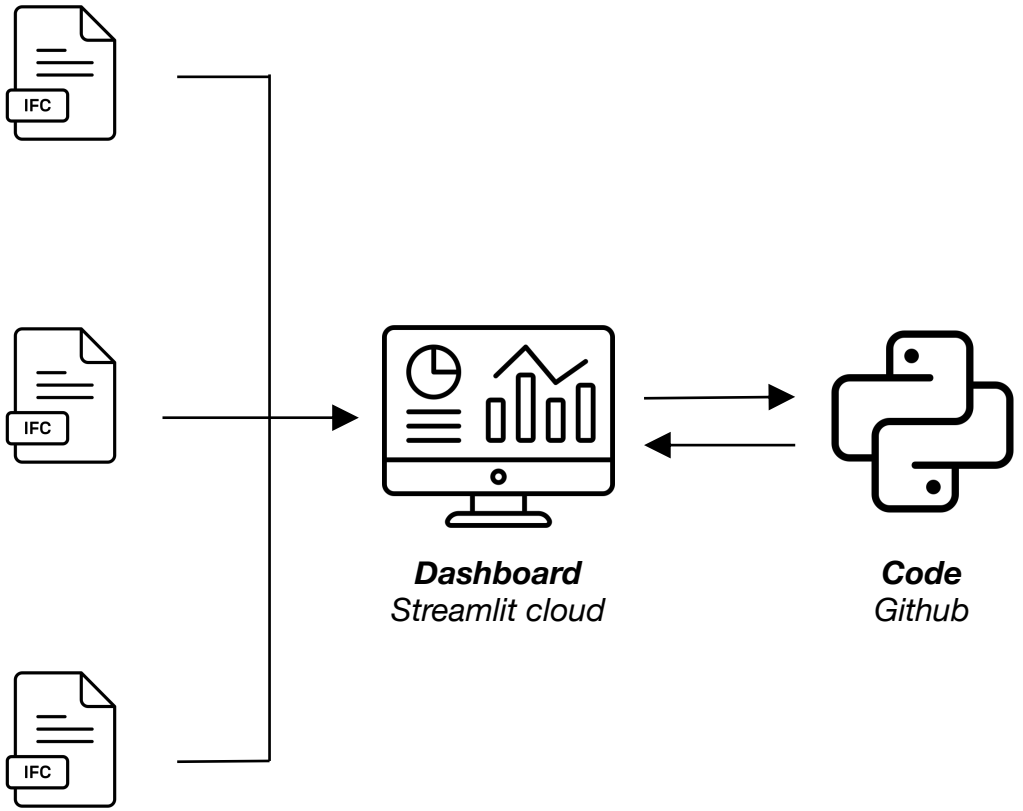
Pset	*_Baueingabe*	
	Gebäude_ID	A, B, C, …
	Geschossfläche	True / False
Pset	*Qto_SpaceBaseQuantites*	
	NetFloorArea	

_NUM (IFC4)

Pset	*_Baueingabe*	
	Gebäude_ID	A, B, C, …
	Einheit_ID	101, 102, 103, …
	Nutzungstyp	WOH, ERS, RED, TEC, LOG, …
	Wohnungstyp	1.5 , 2.5, 3.5, 4.5, 5.5
	Zimmer	True / False
	Fensteranteil	
Pset	*Qto_SpaceBaseQuantites*	
	NetFloorArea	
	Height	

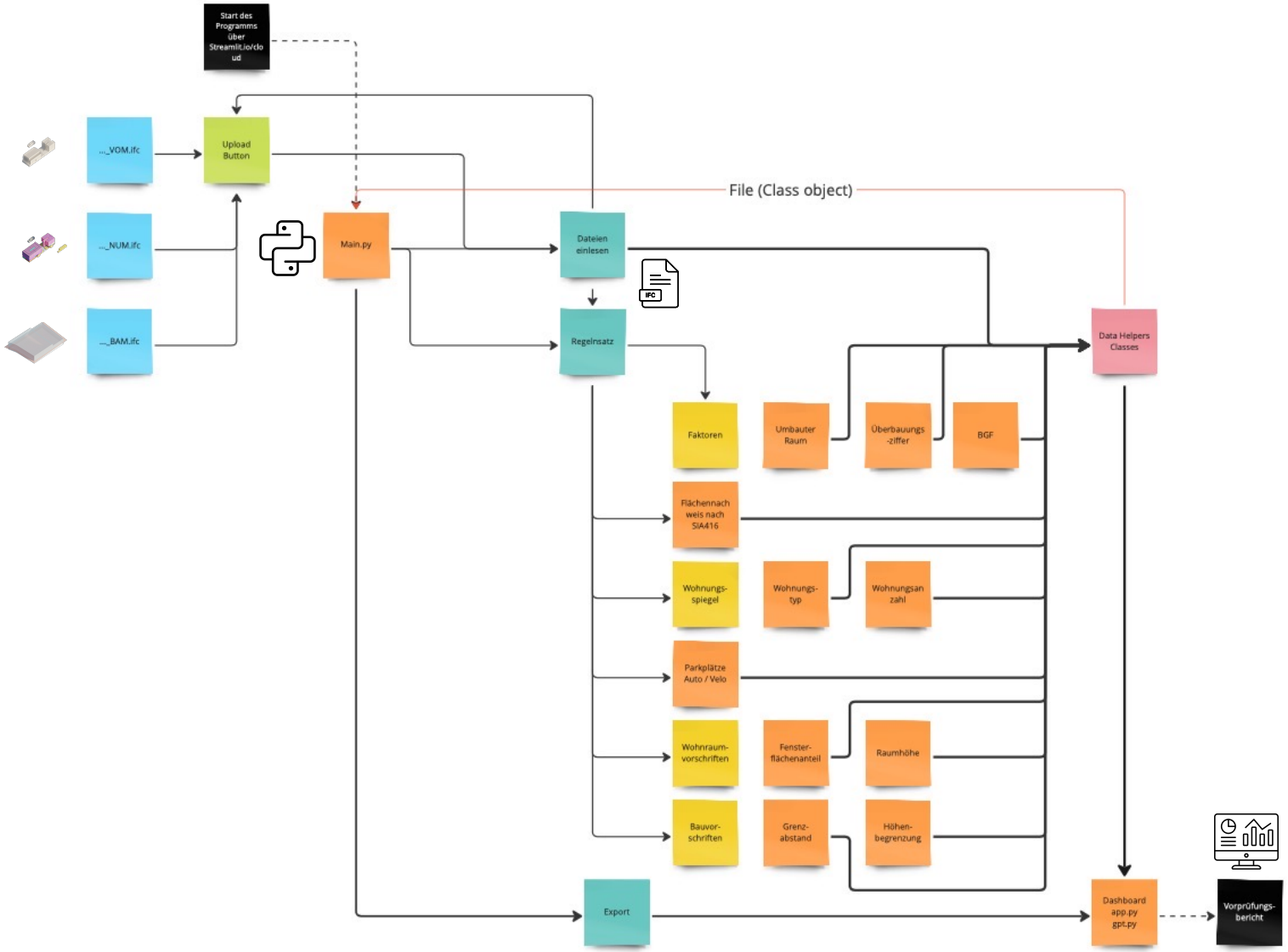
_BAM (IFC4)

Pset	*_Baueingabe*	
	Baurecht	HB, BL, GF
Pset	*Qto_SpaceBaseQuantites*	
	NetFloorArea	



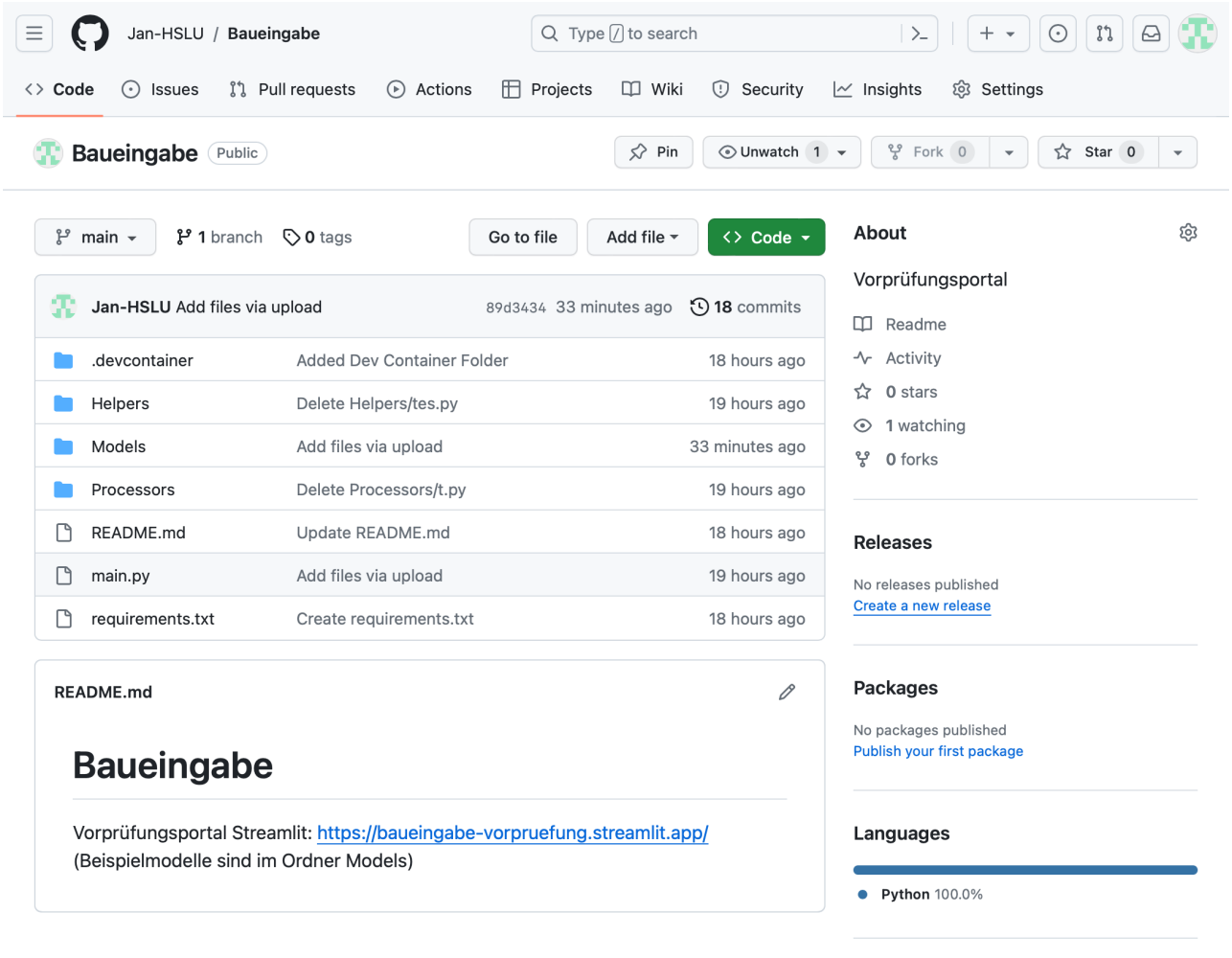
Kaufmann, J. (2023). Modelle zur digitalen Baueingabe [Digitales Bild]. Eigene Erstellung.

Das Python-Programm I



Kaufmann, J. (2023, 5. Dezember). Programmlogik [Diagramm]. Eigene Erstellung.

https://github.com/Jan-HSLU/Baueingabe.git

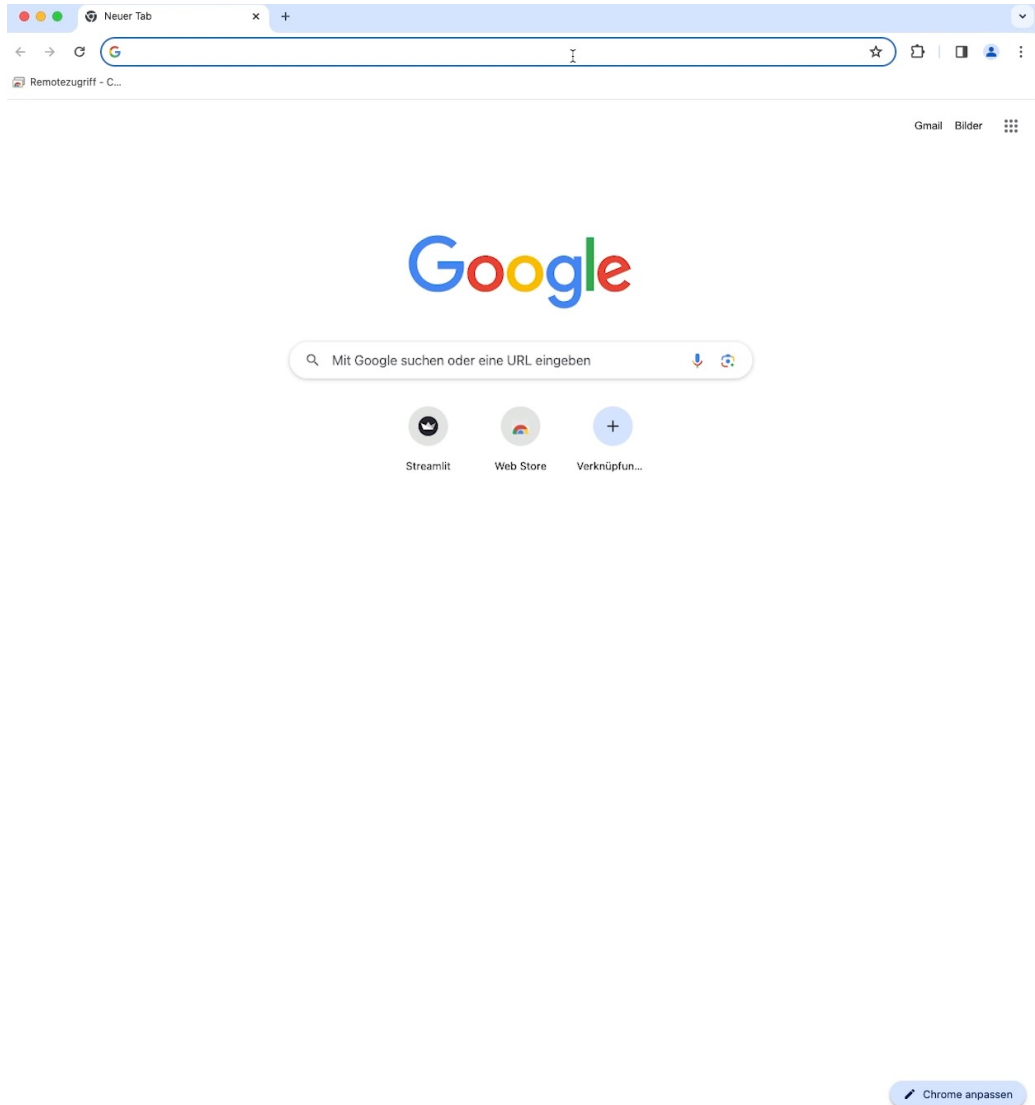


Requirements: streamlit, numpy, Pandas, plotly, Shapely, ifcopenshell, openai

Kaufmann, J. (2023, 6. Dezember). Baueingabe [Code und Dokumentation]. GitHub. Abgerufen von https://github.com/Jan-HSLU/Baueingabe/tree/main

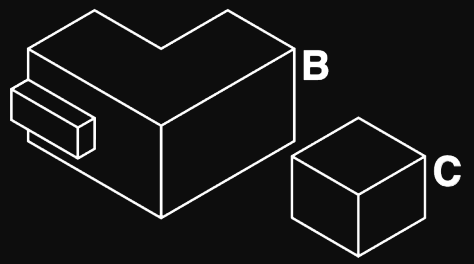
Digitale Baueingabe
Automatisierter Vorprüfungsprozess

https://baueingabe-vorpruefung.streamlit.app



Kaufmann, J. (2024). Showcase [Video]. Abgerufen von https://baueingabe-vorpruefung.streamlit.app

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

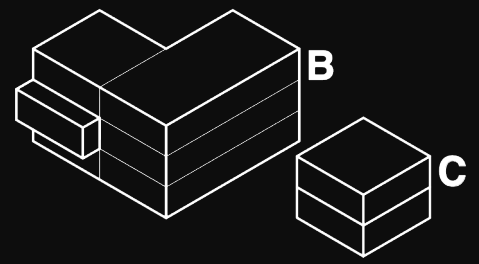



```
[#1734=IfcSpace('36nqMxXmb0yPlToLjfpQgx',#12,'02',$,$,#1674,#1729,'Geschossfläche',.ELEMENT.,.INTERNAL.,$)]
...
```

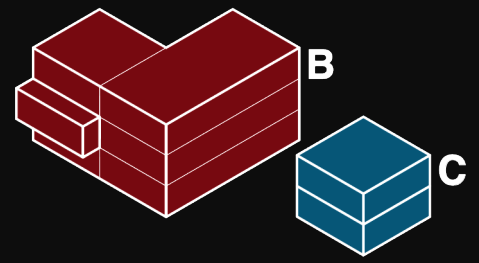

```
{'B': <POLYGON Z ((17.5 3 0, 5.8 3 0, 5.8 6.85 0, 5.8 20 0, 31.2 20 0, 31.2 6.85 0...>,
 'C': <POLYGON Z ((46.6 37.7 0, 46.6 33.1 0, 36.3 33.1 0, 36.3 33.1 0, 32.5 33.1 0...>}
```

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

Spaces sind vorgängig gefiltert (UG / AGF) – Zuweisung nach Keys im Dict.



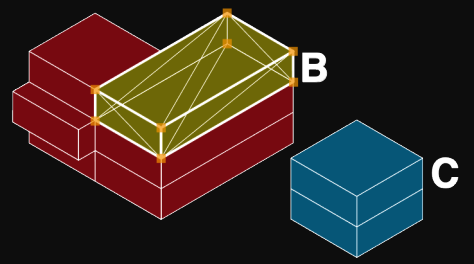

```
[#1734=IfcSpace('36nqMxXmb0yPlToLjfpQgx',#12,'02',$,$,#1674,#1729,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#3858=IfcSpace('0QJIwlFNr8MwB0RwyYwfct',#12,'01',$,$,#3798,#3853,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#3050=IfcSpace('0da_dzlaH42BK7ShQF8LKG',#12,'01',$,$,#2990,#3045,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#1234=IfcSpace('1F0mTaprf8_hlKWEPzjy1_',#12,'01',$,$,#1150,#1229,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#2242=IfcSpace('33Y6uzoFb1wucrccUD0sAU',#12,'01',$,$,#2182,#2237,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#1634=IfcSpace('0iLhR$Wbj5rQUW348I8Jw_',#12,'01',$,$,#1574,#1629,'Geschossfläche',.ELEMENT.,.INTERNAL.,$)]
```




```
{'B':
[#3858=IfcSpace('0QJIwlFNr8MwB0RwyYwfct',#12,'01',$,$,#3798,#3853,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#3050=IfcSpace('0da_dzlaH42BK7ShQF8LKG',#12,'01',$,$,#2990,#3045,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#1234=IfcSpace('1F0mTaprf8_hlKWEPzjy1_',#12,'01',$,$,#1150,#1229,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#2242=IfcSpace('33Y6uzoFb1wucrccUD0sAU',#12,'01',$,$,#2182,#2237,'Geschossfläche',.ELEMENT.,.INTERNAL.,$)],
'C':
[#1734=IfcSpace('36nqMxXmb0yPlToLjfpQgx',#12,'02',$,$,#1674,#1729,'Geschossfläche',.ELEMENT.,.INTERNAL.,$),
#1634=IfcSpace('0iLhR$Wbj5rQUW348I8Jw_',#12,'01',$,$,#1574,#1629,'Geschossfläche',.ELEMENT.,.INTERNAL.,$)]}
```

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

Space (Value) mit Vertices ersetzen -> eine Liste mit X, Y, Z Werten

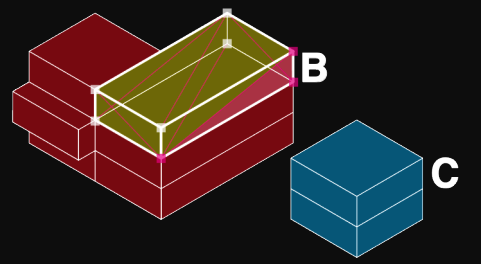


{'B': [[array([5.8, 20. , 9.]), array([5.8, 20. , 12.]), array([5.8 , 6.85, 9.]), array([5.8 , 6.85, 12.]), array([31.23365918, 20.   , 9.   ]), array([31.23365918, 6.85   , 9.   ]), array([31.23365918, 6.85   , 12.   ]), array([31.23365918, 20.   , 12.   ])],
[array([5.8, 20. , 6.]), array([5.8, 20. , 9.]), array([5.8, 3. , 6.]), array([5.8, 3. , 9.]),
array([31.23365918, 20.   , 6.   ]), array([31.23365918, 3.   , 6.   ]),
array([31.23365918, 3.   , 9.   ]), array([31.23365918, 20.   , 9.   ])], [array([5.8, 20. , 0.]), array([5.8, 20. , 3.]), array([5.8, 3. , 0.]), array([5.8, 3. , 3.]),
array([31.23365918, 20.   , 0.   ]), array([31.23365918, 3.   , 0.   ]),
array([19.56682959, 3.   , 0.   ]), array([19.56682959, 9.85   , 0.   ]),
array([17.46682959, 9.85   , 0.   ]), array([17.46682959, 3.   , 0.   ]),
array([17.46682959, 3.   , 3.   ]), array([31.23365918, 20.   , 3.   ]),
array([17.46682959, 9.85   , 3.   ]), array([19.56682959, 9.85   , 3.   ]),
array([19.56682959, 3.   , 3.   ]), array([31.23365918, 3.   , 3.   ])], [array([5.8, 20. , 3.]), array([5.8, 20. , 6.]), array([5.8, 3. , 3.]), array([5.8, 3. , 6.]),
array([31.23365918, 20.   , 3.   ]), array([31.23365918, 3.   , 3.   ]),
array([31.23365918, 3.   , 6.   ]), array([31.23365918, 20.   , 6.   ])]], 'C':
[[array([32.52938834, 37.68197839, 0.   ]), array([32.52938834, 37.68197839, 3.1   ]),
array([32.52938834, 33.13197839, 0.   ]), array([32.52938834, 33.13197839, 3.1   ]),
array([36.27938834, 37.68197839, 0.   ]), array([36.27938834, 33.13197839, 0.   ]),
array([36.27938834, 33.13197839, 3.1   ]), array([36.27938834, 37.68197839, 3.1   ])],
[array([46.63347749, 33.13197839, 0.   ]), array([46.63347749, 33.13197839, 4.3   ]),
array([46.63347749, 37.68197839, 0.   ]), array([46.63347749, 37.68197839, 4.3   ]),
array([36.27938834, 33.13197839, 0.   ]), array([36.27938834, 37.68197839, 0.   ]),
array([36.27938834, 37.68197839, 4.3   ]), array([36.27938834, 33.13197839, 4.3   ])]]}

…

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

Triangulierte Faces – Index des Vertex im Raum als Indexnummer

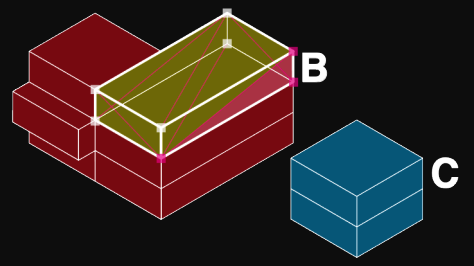


{'B': [[(2, 1, 0), (2, 3, 1), (4, 2, 0), (4, 5, 2), (3, 2, 5), (3, 5, 6), (7, 1, 3), (7, 3, 6), (0, 7, 4), (0, 1, 7), (6, 5, 4), (6, 4, 7)], [(2, 1, 0), (2, 3, 1), (4, 2, 0), (4, 5, 2), (3, 2, 5), (3, 5, 6), (6, 7, 1), (6, 1, 3), (0, 7, 4), (0, 1, 7), (6, 5, 4), (6, 4, 7)], [(2, 1, 0), (2, 3, 1), (8, 2, 0), (8, 9, 2), (5, 6, 7), (4, 7, 8), (4, 5, 7), (4, 8, 0), (3, 2, 9), (3, 9, 10), (12, 1, 3), (11, 12, 13), (11, 1, 12), (10, 12, 3), (15, 13, 14), (15, 11, 13), (0, 11, 4), (6, 1, 11), (15, 5, 4), (15, 4, 11), (6, 5, 15), (14, 6, 15), (7, 6, 14), (7, 14, 13), (12, 3, 7), (12, 7, 13), (9, 8, 12), (9, 12, 10)], [(2, 1, 0), (2, 3, 1), (4, 2, 0), (4, 5, 2), (3, 2, 5), (3, 5, 6), (6, 7, 1), (6, 1, 3), (0, 7, 4), (0, 1, 7), (5, 4, 7), (5, 7, 6)], [(2, 1, 0), (2, 3, 1), (4, 2, 0), (4, 5, 2), (3, 2, 5), (3, 5, 6), (7, 1, 3), (6, 7, 3), (0, 7, 4), (0, 1, 7), (6, 5, 4), (6, 4, 7)]]}

{'B': [[array([5.8, 20. , 9.]), array([5.8, 20. , 12.]), array([5.8 , 6.85, 9.]), array([5.8 , 6.85, 12.]), array([31.23365918, 20.   , 9.   ]), array([31.23365918, 6.85   , 9.   ]), array([31.23365918, 6.85   , 12.   ]), array([31.23365918, 20.   , 12.   ])],
[array([5.8, 20. , 6.]), array([5.8, 20. , 9.]), array([5.8, 3. , 6.]), array([5.8, 3. , 9.]),
array([31.23365918, 20.   , 6.   ]), array([31.23365918, 3.   , 6.   ]),
array([31.23365918, 3.   , 9.   ]), array([31.23365918, 20.   , 9.   ])], [array([5.8, 20. , 0.]), array([5.8, 20. , 3.]), array([5.8, 3. , 0.]), array([5.8, 3. , 3.]),
array([31.23365918, 20.   , 0.   ]), array([31.23365918, 3.   , 0.   ]),
array([19.56682959, 3.   , 0.   ]), array([19.56682959, 9.85   , 0.   ]),
array([17.46682959, 9.85   , 0.   ]), array([17.46682959, 3.   , 0.   ]),
array([17.46682959, 3.   , 3.   ]), array([31.23365918, 20.   , 3.   ]),
array([17.46682959, 9.85   , 3.   ]), array([19.56682959, 9.85   , 3.   ]),
array([19.56682959, 3.   , 3.   ])], [array([5.8, 20. , 3.]), array([5.8, 20. , 6.]), array([5.8, 3. , 3.]), array([5.8, 3. , 6.]),
array([31.23365918, 20.   , 3.   ]), array([31.23365918, 3.   , 3.   ]),
array([31.23365918, 3.   , 6.   ]), array([31.23365918, 20.   , 6.   ])]], 'C':
[[array([32.52938834, 37.68197839, 0.   ]), array([32.52938834, 37.68197839, 3.1   ]),
array([32.52938834, 33.13197839, 0.   ]), array([32.52938834, 33.13197839, 3.1   ]),
array([36.27938834, 37.68197839, 0.   ]), array([36.27938834, 33.13197839, 0.   ]),
array([36.27938834, 33.13197839, 3.1   ]), array([36.27938834, 37.68197839, 3.1   ])],
[array([46.63347749, 33.13197839, 0.   ]), array([46.63347749, 33.13197839, 4.3   ]),
array([46.63347749, 37.68197839, 0.   ]), array([46.63347749, 37.68197839, 4.3   ]),
array([36.27938834, 33.13197839, 0.   ]), array([36.27938834, 37.68197839, 0.   ]),
array([36.27938834, 37.68197839, 4.3   ]), array([36.27938834, 33.13197839, 4.3   ])]]}

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

Index mit X, Y, Z Liste ersetzten -> Face = Liste mit benötigten Vertices



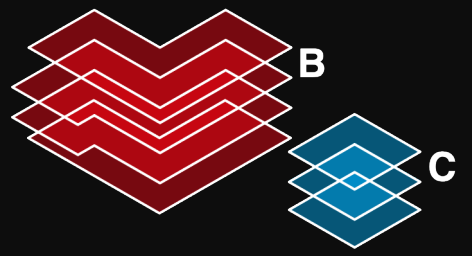
{'B': [[[array([5.8 , 6.85, 9.]), array([5.8, 20. , 12.]), array([5.8, 20. , 9.])], [array([5.8 , 6.85, 9.]), array([5.8 , 6.85, 12.]), array([5.8, 20. , 12.])], [array([31.23365918, 20. , 9.]), array([5.8 , 6.85, 9.]), array([5.8, 20. , 9.])], [array([31.23365918, 20. , 9.]), array([31.23365918, 6.85 , 9.]), array([5.8 , 6.85, 9.])], [array([5.8 , 6.85, 12.]), array([5.8 , 6.85, 9.]), array([31.23365918, 6.85 , 9.])], [array([5.8 , 6.85, 12.]), array([31.23365918, 6.85 , 9.]), array([31.23365918, 6.85 , 12.])], [array([31.23365918, 20. , 12.]), array([5.8, 20. , 12.]), array([5.8 , 6.85, 12.])], [array([31.23365918, 20. , 12.]), array([5.8 , 6.85, 12.]), array([31.23365918, 6.85 , 12.])], [array([5.8, 20. , 9.]), array([31.23365918, 20. , 12.]), array([31.23365918, 20. , 9.])], [array([5.8, 20. , 9.]), array([5.8, 20. , 12.]), array([31.23365918, 20. , 12.])], [array([31.23365918, 20. , 12.]), array([31.23365918, 6.85 , 12.]), array([31.23365918, 6.85 , 9.])], [array([31.23365918, 20. , 9.]), array([31.23365918, 6.85 , 12.]), array([31.23365918, 20. , 9.])], [array([31.23365918, 6.85 , 12.]), array([31.23365918, 20. , 12.])]], [[array([5.8, 3. , 6.]), array([5.8, 20. , 9.]), array([5.8, 20. , 6.])], [array([5.8, 3. , 6.]), array([5.8, 3. , 9.]), array([5.8, 20. , 9.])], [array([31.23365918, 20. , 6.]), array([5.8, 3. , 6.]), array([5.8, 20. , 6.])], [array([31.23365918, 20. , 6.]), array([31.23365918, 3. , 6.]), array([5.8, 3. , 6.])], [array([5.8, 3. , 9.]), array([5.8, 3. , 6.]), array([31.23365918, 3. , 6.])], [array([5.8, 3. , 9.]), array([31.23365918, 3. , 6.]), array([31.23365918, 3. , 9.])], [array([31.23365918, 3. , 9.]), array([31.23365918, 20. , 9.]), array([5.8, 20. , 9.])], [array([31.23365918, 3. , 9.]), array([5.8, 20. , 9.]), array([5.8, 3. , 9.])], [array([5.8, 20. , 6.]), array([31.23365918, 20. , 9.]), array([31.23365918, 20. , 6.])], [array([5.8, 20. , 6.]), array([5.8, 20. , 9.]), array([31.23365918, 20. , 9.])], [array([31.23365918, 3. , 9.]), array([31.23365918, 3. , 9.]), array([31.23365918, 3. , 6.])], [array([31.23365918, 20. , 6.])], [array([31.23365918, 3. , 9.]), array([31.23365918, 20. , 6.]), array([31.23365918, 20. , 9.])]], [[array([5.8, 3. , 0.]), array([5.8, 20. , 3.]), array([5.8, 20. , 0.])], [array([5.8, 3. , 0.]), array([5.8, 3. , 3.]), array([5.8, 20. , 3.])], ...
```
```

Digitale Baueingabe
Automatisierter Vorprüfungsprozess

Rule 3 - ÜZ

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

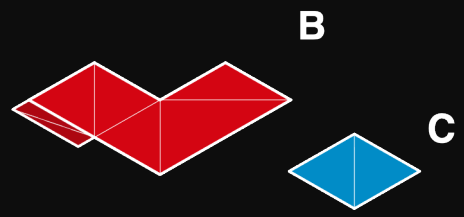
Faces mit nicht 3x gleichem Z-Wert werden aus dem Dictionary löschen




```
{'B': [[[array([5.8 , 6.85, 9.  ]), array([ 5.8, 20. , 12. ]), array([ 5.8, 20. ,  9. ])], [array([5.8 ,
6.85, 9.  ]), array([ 5.8 ,  6.85, 12.  ]), array([ 5.8, 20. , 12. ])], [array([31.23365918, 20.        ,
9.        ]), array([5.8 , 6.85, 9.  ]), array([ 5.8, 20. ,  9. ])], [array([31.23365918, 20.        ,  9.
]), array([31.23365918,  6.85      ,  9.        ]), array([5.8 , 6.85, 9.  ])], [array([ 5.8 ,  6.85, 12.
]), array([5.8 , 6.85, 9.  ]), array([31.23365918,  6.85      ,  9.        ])], [array([ 5.8 ,  6.85, 12.
]), array([31.23365918,  6.85      ,  9.        ]), array([31.23365918,  6.85      , 12.        ])],
[array([31.23365918, 20.        , 12.        ]), array([ 5.8, 20. , 12. ]), array([ 5.8 ,  6.85, 12.  ])],
[array([31.23365918, 20.        , 12.        ]), array([ 5.8 ,  6.85, 12.  ]), array([31.23365918,  6.85
, 12.        ])], [array([ 5.8, 20. ,  9. ]), array([31.23365918, 20.        , 12.        ]),
array([31.23365918, 20.        ,  9.        ])], [array([ 5.8, 20. ,  9. ]), array([ 5.8, 20. , 12. ]),
array([31.23365918, 20.        , 12.        ])], [array([31.23365918,  6.85      , 12.        ]),
array([31.23365918,  6.85      ,  9.        ]), array([31.23365918, 20.        ,  9.        ])],
[array([31.23365918,  6.85      , 12.        ]), array([31.23365918, 20.        ,  9.        ]),
array([31.23365918, 20.        , 12.        ])]], [[array([5.8, 3. , 6. ]), array([ 5.8, 20. ,  9. ]),
array([ 5.8, 20. ,  6. ])], [array([5.8, 3. , 6. ]), array([5.8, 3. , 9. ]), array([ 5.8, 20. ,  9. ])],
[array([31.23365918, 20.        ,  6.        ]), array([5.8, 3. , 6. ]), array([ 5.8, 20. ,  6. ])],
[array([31.23365918, 20.        ,  6.        ]), array([31.23365918,  3.        ,  6.        ]), array([5.8,
3. , 6. ])], [array([5.8, 3. , 9. ]), array([5.8, 3. , 6. ]), array([31.23365918,  3.        ,  6.
])], [array([5.8, 3. , 9. ]), array([31.23365918,  3.        ,  6.        ]), array([31.23365918,  3.
,  9.        ])], [array([31.23365918,  3.        ,  9.        ]), array([31.23365918, 20.        ,  9.
]), array([ 5.8, 20. ,  9. ])], [array([31.23365918,  3.        ,  9.        ]), array([ 5.8, 20. ,  9. ]),
array([5.8, 3. , 9. ])], [array([ 5.8, 20. ,  6. ]), array([31.23365918, 20.        ,  9.        ]),
array([31.23365918, 20.        ,  6.        ])], [array([ 5.8, 20. ,  6. ]), array([ 5.8, 20. ,  9. ]),
array([31.23365918, 20.        ,  9.        ])], [array([31.23365918,  3.        ,  9.        ]),
array([31.23365918,  3.        ,  6.        ]), array([31.23365918, 20.        ,  6.        ])],
[array([31.23365918,  3.        ,  9.        ]), array([31.23365918, 20.        ,  6.        ]),
array([31.23365918, 20.        ,  9.        ])]], [[array([5.8, 3. , 0. ]), array([ 5.8, 20. ,  3. ]),
array([ 5.8, 20. ,  0. ])], [array([5.8, 3. , 0. ]), array([5.8, 3. , 3. ]), array([ 5.8, 20. ,  3. ])], ...
```

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

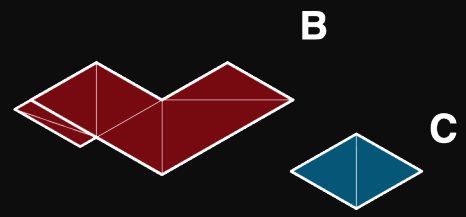
Z Wert auf 0 Setzen -> Schwarzplan



{'B': [[[31.2336591796, 20.0, 0], [5.8, 6.85, 0], [5.8, 20.0, 0]], [[31.2336591796, 20.0, 0], [31.2336591796, 6.85, 0], [5.8, 6.85, 0]], [[31.2336591796, 20.0, 0], [5.8, 20.0, 0], [5.8, 6.85, 0]], [[31.2336591796, 20.0, 0], [5.8, 6.85, 0], [31.2336591796, 6.85, 0]]], [[[31.2336591796, 20.0, 0], [5.8, 3.0, 0], [5.8, 20.0, 0]], [[31.2336591796, 20.0, 0], [31.2336591796, 3.0, 0], [5.8, 3.0, 0]], [[31.2336591796, 3.0, 0], [31.2336591796, 20.0, 0], [5.8, 20.0, 0]], [[31.2336591796, 3.0, 0], [5.8, 20.0, 0], [5.8, 3.0, 0]]], [[[17.4668295898, 9.85, 0], ...

```
[…]

#Vertex pro Space und Gebäude als koordinate xyz
    vertices_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_vertices = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            verts = shape.geometry.verts
            matrix = ifcopenshell.util.placement.get_local_placement(space.ObjectPlacement)
            transformed_verts = [np.dot(matrix, np.array(list(verts[i:i+3]) + [1]))[:3] for i in range(0, len(verts), 3)]
            building_vertices.append(transformed_verts)
        vertices_by_building_id[building_id] = building_vertices

    #Faces pro Space und Gebäude als Triangulierung
    faces_by_building_id = {}
    for building_id, spaces in spaces_by_building_id.items():
        building_faces = []
        for space in spaces:
            shape = ifcopenshell.geom.create_shape(settings, space)
            faces = shape.geometry.faces
            grouped_faces = [faces[i:i+3] for i in range(0, len(faces), 3)]
            building_faces.append(grouped_faces)
        faces_by_building_id[building_id] = building_faces

    #Übertragen der Vertex-Koordinaten an die Idexes bei den Faces
    faces_with_coords_by_building_id = {}
    for building_id in faces_by_building_id:
        building_faces_with_coords = []
        for space_idx, space_faces in enumerate(faces_by_building_id[building_id]):
            space_faces_with_coords = []
            for face in space_faces:
                face_coords = [vertices_by_building_id[building_id][space_idx][idx] for idx in face]
                space_faces_with_coords.append(face_coords)
            building_faces_with_coords.append(space_faces_with_coords)
        faces_with_coords_by_building_id[building_id] = building_faces_with_coords

    #Entfernen der nicht horizontalen Flächen
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        for space_idx, space_faces in enumerate(building_faces):
            horizontal_faces = [face for face in space_faces if all(v[2] == face[0][2] for v in face)]
            faces_with_coords_by_building_id[building_id][space_idx] = horizontal_faces

    #Elemente auf Z Achse 0 setzen
    faces_with_z_zero_by_building_id = {}
    for building_id, building_faces in faces_with_coords_by_building_id.items():
        building_faces_with_z_zero = []
        for space_faces in building_faces:
            space_faces_with_z_zero = []
            for face in space_faces:
                face_with_z_zero = [[v[0], v[1], 0] for v in face]
                space_faces_with_z_zero.append(face_with_z_zero)
            building_faces_with_z_zero.append(space_faces_with_z_zero)
        faces_with_z_zero_by_building_id[building_id] = building_faces_with_z_zero

    #Mergen der Faces
    unified_area_by_building_id = {}
    for building_id, building_faces in faces_with_z_zero_by_building_id.items():
        polygons = []
        for space_faces in building_faces:
            for face in space_faces:
                polygon = Polygon(face)
                polygons.append(polygon)

        unified_polygon = unary_union(polygons)
        unified_area_by_building_id[building_id] = unified_polygon

[…]
```

Vereinen der Faces zu einem Polygon mit Shapely




```
{'B': <POLYGON Z ((17.5 3 0, 5.8 3 0, 5.8 6.85 0, 5.8 20 0, 31.2 20 0, 31.2 6.85 0...>,
 'C': <POLYGON Z ((46.6 37.7 0, 46.6 33.1 0, 36.3 33.1 0, 36.3 33.1 0, 32.5 33.1 0...>}
```


```
{'B': <POLYGON Z ((17.5 3 0, 5.8 3 0, 5.8 6.85 0, 5.8 20 0, 31.2 20 0, 31.2 6.85 0...>,
 'C': <POLYGON Z ((46.6 37.7 0, 46.6 33.1 0, 36.3 33.1 0, 36.3 33.1 0, 32.5 33.1 0...>}
```

Fragen und Feedback

Digitale Baueingabe

Automatisierter Vorprüfungsprozess

Kaufmann, J. (2023). Digitaler Baueingabeprozess [Digitales Bild]. DALL-E von OpenAI.