

Entrega 4
Reproducible Geospatial Analysis Workflow



Pontificia Universidad
JAVERIANA
Colombia

Juan David Castillo Laverde
Juan Pablo Dávila Martínez
Eugenia Victoria Dayoub Barito
Luis Fernando Lee Rodríguez

Pontificia Universidad Javeriana
Facultad de Ingeniería
Administración de Bases de Datos
Bogotá D.C.
Noviembre 2025

1. Introducción

El presente informe documenta el desarrollo del Reproducible Geospatial Analysis Workflow correspondiente a la Entrega 4 del proyecto DBA. El objetivo consiste en diseñar e implementar un pipeline geoespacial completamente reproducible que permita:

- Filtrar footprints de edificaciones provenientes de Google Open Buildings y Microsoft Building Footprints usando los límites de los municipios PDET.
- Ejecutar consultas espaciales eficientes mediante índices 2dsphere en MongoDB.
- Calcular el número total de edificaciones y el área total de cubiertas dentro de cada municipio PDET.
- Exportar resultados a archivos CSV para análisis geoespacial y estadístico posterior.
- Asegurar trazabilidad, portabilidad y reproducibilidad en cualquier entorno mediante Docker y scripts estandarizados.

La contenerización con Docker y el uso de scripts automatizados garantizan que cualquier miembro del equipo pueda obtener resultados idénticos y verificables, independientemente del sistema operativo utilizado.

```
[root@lflee-beelink DBA_ProyectoFinal]# tree -L 2 data/
data/
├── cargar_google_footprints.py
├── cargar_microsoft_footprints.py
├── cargar_municipios.py
├── Dockerfile
├── eda_footprints.py
├── google1.gpkg
├── municipios_pdet_filtrados.geojson
├── municipios_pdet_filtrados.qmd
├── MunicipiosPDET.xlsx
├── run_etl.sh
├── samples
│   ├── google_buildings.geojson
│   ├── sample_google1.geojson
│   ├── sample_google1.qmd
│   ├── sample_microsoft.geojson
│   ├── sample_microsoft.qmd
│   └── sample_microsoft_small.geojson
├── scripts
│   ├── convert_csv_to_geojson.py
│   ├── convert_geojsonl_to_geojson.py
│   ├── create_mgn_municipios_pdet.py
│   ├── download_google.sh
│   ├── download_mgn.ps1
│   ├── download_mgn.sh
│   ├── download_microsoft.sh
│   ├── extract_features.py
│   └── fix_invalid_geometries.py
└── setup.sh

3 directories, 26 files
[root@lflee-beelink DBA_ProyectoFinal]#
```

Figura 1. Vista general de los archivos Google/Microsoft o una carpeta del proyecto mostrando la estructura base.

2. Metodología

El workflow se fundamenta en cuatro pilares: automatización, contenerización, scriptabilidad y verificación.

2.1. Automatización

Se desarrolló un script maestro:

pdet pipeline.sh

Este archivo ejecuta de forma secuencial:

- Validación de conexión a MongoDB.
- Verificación y creación de índices geoespaciales.
- Ejecución del filtrado PDET.
- Cálculo de conteos y áreas totales.
- Exportación de un archivo consolidado en CSV.
- Registro de logs durante todo el proceso.

```

etl-loader-script | ✓ Insertados: 1,805,000
etl-loader-script | ✓ Insertados: 1,810,000
etl-loader-script | ✓ Insertados: 1,815,000
etl-loader-script | ✓ Insertados: 1,820,000
etl-loader-script | ✓ Insertados: 1,825,000
etl-loader-script | Procesados: 12,790,000 | En PDET: 1,825,078 | Fuera: 10,964,322
etl-loader-script | Procesados: 12,800,000 | En PDET: 1,827,084 | Fuera: 10,972,916
etl-loader-script | ✓ Insertados: 1,830,000
etl-loader-script | ✓ Insertados: 1,835,000
etl-loader-script | ✓ Insertados: 1,840,000
etl-loader-script | ✓ Insertados: 1,845,000
etl-loader-script | ✓ Insertados: 1,850,000
etl-loader-script | Procesados: 13,190,000 | En PDET: 1,854,638 | Fuera: 11,335,362
etl-loader-script | ✓ Insertados: 1,855,000
etl-loader-script | ✓ Insertados: 1,860,000
etl-loader-script | ✓ Insertados: 1,865,000
etl-loader-script | ✓ Insertados: 1,870,000
etl-loader-script | ✓ Insertados: 1,875,000
etl-loader-script | Procesados: 13,650,000 | En PDET: 1,876,167 | Fuera: 11,773,833
etl-loader-script | Procesados: 13,680,000 | En PDET: 1,877,065 | Fuera: 11,802,395
etl-loader-script | Procesados: 13,690,000 | En PDET: 1,878,085 | Fuera: 11,811,915
etl-loader-script | ✓ Insertados: 1,880,000
etl-loader-script | ✓ Insertados: 1,885,000
etl-loader-script | Procesados: 13,860,000 | En PDET: 1,885,972 | Fuera: 11,974,028
etl-loader-script | Procesados: 13,900,000 | En PDET: 1,887,848 | Fuera: 12,012,152
etl-loader-script | ✓ Insertados: 1,890,000
mongo-proyecto-upme | [{"_id":{"$_date":"2025-11-17T04:05:05.109-08:00"},"s":{"t":{"c":"WTCHKPT","id":"22430","ctx":"Checkpointi
ntr"},"s":{"wiredTiger message","attr":{"message":{"ts_sec":"1763352305","ts_usec":"1099906","thread":"1:0x7f5630bcf0c0"},"session
name":"WT_SESSION_checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","log_id":"1000000","category_id":"7","verbose_level":"INF
O","verbose_level_id":"0","msg":"saving checkpoint snapshot min: 187611, snapshot max: 187611 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write key: 18250}}}]
mongo-proyecto-upme | [{"_id":{"$_date":"2025-11-17T04:06:05.131-08:00"},"s":{"t":{"c":"WTCHKPT","id":"22430","ctx":"Checkpointi
ntr"},"s":{"wiredTiger message","attr":{"message":{"ts_sec":"1763352365","ts_usec":"131566","thread":"1:0x7f5630bcf0c0"},"session
name":"WT_SESSION_checkpoint","category":"WT_VERB_CHECKPOINT_PROGRESS","log_id":"1000000","category_id":"7","verbose_level":"INF
O","verbose_level_id":"0","msg":"saving checkpoint snapshot min: 187624, snapshot max: 187624 snapshot count: 0, oldest timesta
mp: (0, 0), meta checkpoint timestamp: (0, 0) base write key: 18250}}}]
mongo-proyecto-upme | [{"_id":{"$_date":"2025-11-17T04:06:24.807-08:00"},"s":{"t":{"c":"QUERY","id":"2658100","ctx":"conn34
"},"s":{"hinted index could not provide a bounded scan,rsvecting to whole index scan"},"attr":{"hint":{"$hint":{"$id":"1"},"l

```

Figura 2. La terminal ejecutando pdet pipeline.sh.

2.2. Contenerización

Todo el proceso se ejecuta sobre Docker, utilizando el contenedor:

mongo-upme

Ventajas:

- Reproducibilidad entre Windows/macOS/Linux.
- Aislamiento del entorno geoespacial.

- No requiere instalar Python, bibliotecas geoespaciales o mongosh en el equipo del usuario.

```
3 directories, 28 files
[root@flee-beelink DBA_ProjectoFinal]# docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS        PORTS                    NAMES
9ee05756a8bf   dba_projectofinal-etl-loader        "bash -c 'dos2unix ./..."            8 hours ago   Up 8 hours   0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp   etl-loader-script
a79481bac192   mongo:latest                        "docker-entrypoint.s..."            8 hours ago   Up 8 hours   0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp   mongo-projecto-upme
```

Figura 3. Salida de docker ps con el contenedor mongo-upme activo.

2.3. Scriptabilidad

El análisis se realiza exclusivamente mediante scripts:

- cargar_municipios.py
- cargar_google_footprints.py
- cargar_microsoft_footprints.py
- exportar_pdet_google.py
- exportar_pdet_microsoft.py
- run_pdet_pipeline.sh

Esta estructura modular permite reproducir, modificar o extender el workflow sin afectar la coherencia del proceso.

```

11 def cargar_datos_moneda_PDET en memoria
12 print("Cargando MONEDA_PDET en memoria...")
13 print("-----")
14
15 try:
16     monedas_pdet = loadpdet.collection.find({}). {
17         'codigo_moneda': 1,
18         'moneda_moneda': 1,
19         'moneda_moneda': 1,
20         'moneda_moneda': 1,
21         'moneda_moneda': 1
22     }
23 }
24
25 if not monedas_pdet:
26     print("Error: No hay monedas PDET en la base de datos.")
27     print("Ejecute primero: python3 /app/scripts/inserta_moneda_moneda.py")
28     print("-----")
29     exit(1)
30
31 print("Voy a cargar los [len(monedas_pdet)] monedas PDET")
32
33 # Generar geometria y shapely para luego cargar
34 monedas_shapely = []
35 for moneda in monedas_pdet:
36     try:
37         geom = shape(moneda['geometria'])
38         monedas_shapely.append(
39             {'codigo_moneda': moneda['codigo_moneda'],
40              'moneda_moneda': moneda['moneda_moneda'],
41              'geom': geom})
42     except Exception as e:
43         print("Error al cargar monedas [moneda['codigo_moneda']]: {e}")
44
45 print("Voy a [len(monedas_shapely)] geometrias preparadas para luego cargar")
46
47 except Exception as e:
48     print("Error al cargar monedas PDET: {e}")
49     print("-----")
50     exit(1)
51
52 # 3. Limpiar colección destino
53 collection.delete_many({})
54 print("Voy a limpiar la colección destino")
55
56 # 4. Insertar archivos
57 if not os.path.exists(MONEDA_PATH):
58     print("Voy a crear el directorio de archivos [MONEDA_PATH]")
59     os.makedirs(MONEDA_PATH)
60
61 # 5. Generar geometria
62 for moneda in monedas_shapely:
63     try:
64         geom = moneda['geom']
65         geom.save(MONEDA_PATH + moneda['codigo_moneda'] + ".shp")
66     except Exception as e:
67         print("Error al guardar geometria: {e}")
68
69 # 6. Guardar en base de datos
70 print("Voy a guardar en la base de datos las monedas PDET")
71
72 # 7. Guardar en base de datos
73 print("Voy a guardar en la base de datos las monedas PDET")
74
75 # 8. Guardar en base de datos
76 print("Voy a guardar en la base de datos las monedas PDET")
77
78 # 9. Guardar en base de datos
79 print("Voy a guardar en la base de datos las monedas PDET")
80
81 # 10. Guardar en base de datos
82 print("Voy a guardar en la base de datos las monedas PDET")
83
84 # 11. Guardar en base de datos
85 print("Voy a guardar en la base de datos las monedas PDET")
86
87 # 12. Guardar en base de datos
88 print("Voy a guardar en la base de datos las monedas PDET")
89
90 # 13. Guardar en base de datos
91 print("Voy a guardar en la base de datos las monedas PDET")
92
93 # 14. Guardar en base de datos
94 print("Voy a guardar en la base de datos las monedas PDET")
95
96 # 15. Guardar en base de datos
97 print("Voy a guardar en la base de datos las monedas PDET")
98
99 # 16. Guardar en base de datos
100 print("Voy a guardar en la base de datos las monedas PDET")
101
102 # 17. Guardar en base de datos
103 print("Voy a guardar en la base de datos las monedas PDET")
104
105 # 18. Guardar en base de datos
106 print("Voy a guardar en la base de datos las monedas PDET")
107
108 # 19. Guardar en base de datos
109 print("Voy a guardar en la base de datos las monedas PDET")
110
111 # 20. Guardar en base de datos
112 print("Voy a guardar en la base de datos las monedas PDET")
113
114 # 21. Guardar en base de datos
115 print("Voy a guardar en la base de datos las monedas PDET")
116
117 # 22. Guardar en base de datos
118 print("Voy a guardar en la base de datos las monedas PDET")
119
120 # 23. Guardar en base de datos
121 print("Voy a guardar en la base de datos las monedas PDET")
122
123 # 24. Guardar en base de datos
124 print("Voy a guardar en la base de datos las monedas PDET")
125
126 # 25. Guardar en base de datos
127 print("Voy a guardar en la base de datos las monedas PDET")
128
129 # 26. Guardar en base de datos
130 print("Voy a guardar en la base de datos las monedas PDET")
131
132 # 27. Guardar en base de datos
133 print("Voy a guardar en la base de datos las monedas PDET")
134
135 # 28. Guardar en base de datos
136 print("Voy a guardar en la base de datos las monedas PDET")
137
138 # 29. Guardar en base de datos
139 print("Voy a guardar en la base de datos las monedas PDET")
140
141 # 30. Guardar en base de datos
142 print("Voy a guardar en la base de datos las monedas PDET")
143
144 # 31. Guardar en base de datos
145 print("Voy a guardar en la base de datos las monedas PDET")
146
147 # 32. Guardar en base de datos
148 print("Voy a guardar en la base de datos las monedas PDET")
149
150 # 33. Guardar en base de datos
151 print("Voy a guardar en la base de datos las monedas PDET")
152
153 # 34. Guardar en base de datos
154 print("Voy a guardar en la base de datos las monedas PDET")
155
156 # 35. Guardar en base de datos
157 print("Voy a guardar en la base de datos las monedas PDET")
158
159 # 36. Guardar en base de datos
160 print("Voy a guardar en la base de datos las monedas PDET")
161
162 # 37. Guardar en base de datos
163 print("Voy a guardar en la base de datos las monedas PDET")
164
165 # 38. Guardar en base de datos
166 print("Voy a guardar en la base de datos las monedas PDET")
167
168 # 39. Guardar en base de datos
169 print("Voy a guardar en la base de datos las monedas PDET")
170
171 # 40. Guardar en base de datos
172 print("Voy a guardar en la base de datos las monedas PDET")
173
174 # 41. Guardar en base de datos
175 print("Voy a guardar en la base de datos las monedas PDET")
176
177 # 42. Guardar en base de datos
178 print("Voy a guardar en la base de datos las monedas PDET")
179
180 # 43. Guardar en base de datos
181 print("Voy a guardar en la base de datos las monedas PDET")
182
183 # 44. Guardar en base de datos
184 print("Voy a guardar en la base de datos las monedas PDET")
185
186 # 45. Guardar en base de datos
187 print("Voy a guardar en la base de datos las monedas PDET")
188
189 # 46. Guardar en base de datos
190 print("Voy a guardar en la base de datos las monedas PDET")
191
192 # 47. Guardar en base de datos
193 print("Voy a guardar en la base de datos las monedas PDET")
194
195 # 48. Guardar en base de datos
196 print("Voy a guardar en la base de datos las monedas PDET")
197
198 # 49. Guardar en base de datos
199 print("Voy a guardar en la base de datos las monedas PDET")
200
201 # 50. Guardar en base de datos
202 print("Voy a guardar en la base de datos las monedas PDET")
203
204 # 51. Guardar en base de datos
205 print("Voy a guardar en la base de datos las monedas PDET")
206
207 # 52. Guardar en base de datos
208 print("Voy a guardar en la base de datos las monedas PDET")
209
210 # 53. Guardar en base de datos
211 print("Voy a guardar en la base de datos las monedas PDET")
212
213 # 54. Guardar en base de datos
214 print("Voy a guardar en la base de datos las monedas PDET")
215
216 # 55. Guardar en base de datos
217 print("Voy a guardar en la base de datos las monedas PDET")
218
219 # 56. Guardar en base de datos
220 print("Voy a guardar en la base de datos las monedas PDET")
221
222 # 57. Guardar en base de datos
223 print("Voy a guardar en la base de datos las monedas PDET")
224
225 # 58. Guardar en base de datos
226 print("Voy a guardar en la base de datos las monedas PDET")
227
228 # 59. Guardar en base de datos
229 print("Voy a guardar en la base de datos las monedas PDET")
230
231 # 60. Guardar en base de datos
232 print("Voy a guardar en la base de datos las monedas PDET")
233
234 # 61. Guardar en base de datos
235 print("Voy a guardar en la base de datos las monedas PDET")
236
237 # 62. Guardar en base de datos
238 print("Voy a guardar en la base de datos las monedas PDET")
239
240 # 63. Guardar en base de datos
241 print("Voy a guardar en la base de datos las monedas PDET")
242
243 # 64. Guardar en base de datos
244 print("Voy a guardar en la base de datos las monedas PDET")
245
246 # 65. Guardar en base de datos
247 print("Voy a guardar en la base de datos las monedas PDET")
248
249 # 66. Guardar en base de datos
250 print("Voy a guardar en la base de datos las monedas PDET")
251
252 # 67. Guardar en base de datos
253 print("Voy a guardar en la base de datos las monedas PDET")
254
255 # 68. Guardar en base de datos
256 print("Voy a guardar en la base de datos las monedas PDET")
257
258 # 69. Guardar en base de datos
259 print("Voy a guardar en la base de datos las monedas PDET")
260
261 # 70. Guardar en base de datos
262 print("Voy a guardar en la base de datos las monedas PDET")
263
264 # 71. Guardar en base de datos
265 print("Voy a guardar en la base de datos las monedas PDET")
266
267 # 72. Guardar en base de datos
268 print("Voy a guardar en la base de datos las monedas PDET")
269
270 # 73. Guardar en base de datos
271 print("Voy a guardar en la base de datos las monedas PDET")
272
273 # 74. Guardar en base de datos
274 print("Voy a guardar en la base de datos las monedas PDET")
275
276 # 75. Guardar en base de datos
277 print("Voy a guardar en la base de datos las monedas PDET")
278
279 # 76. Guardar en base de datos
280 print("Voy a guardar en la base de datos las monedas PDET")
281
282 # 77. Guardar en base de datos
283 print("Voy a guardar en la base de datos las monedas PDET")
284
285 # 78. Guardar en base de datos
286 print("Voy a guardar en la base de datos las monedas PDET")
287
288 # 79. Guardar en base de datos
289 print("Voy a guardar en la base de datos las monedas PDET")
290
291 # 80. Guardar en base de datos
292 print("Voy a guardar en la base de datos las monedas PDET")
293
294 # 81. Guardar en base de datos
295 print("Voy a guardar en la base de datos las monedas PDET")
296
297 # 82. Guardar en base de datos
298 print("Voy a guardar en la base de datos las monedas PDET")
299
300 # 83. Guardar en base de datos
301 print("Voy a guardar en la base de datos las monedas PDET")
302
303 # 84. Guardar en base de datos
304 print("Voy a guardar en la base de datos las monedas PDET")
305
306 # 85. Guardar en base de datos
307 print("Voy a guardar en la base de datos las monedas PDET")
308
309 # 86. Guardar en base de datos
310 print("Voy a guardar en la base de datos las monedas PDET")
311
312 # 87. Guardar en base de datos
313 print("Voy a guardar en la base de datos las monedas PDET")
314
315 # 88. Guardar en base de datos
316 print("Voy a guardar en la base de datos las monedas PDET")
317
318 # 89. Guardar en base de datos
319 print("Voy a guardar en la base de datos las monedas PDET")
320
321 # 90. Guardar en base de datos
322 print("Voy a guardar en la base de datos las monedas PDET")
323
324 # 91. Guardar en base de datos
325 print("Voy a guardar en la base de datos las monedas PDET")
326
327 # 92. Guardar en base de datos
328 print("Voy a guardar en la base de datos las monedas PDET")
329
330 # 93. Guardar en base
```

Figura 4. Código de uno de los scripts: `ensure_indexes.js`, `run_pdet_filter.js` o los `export *.py`.

2.4.Verificación

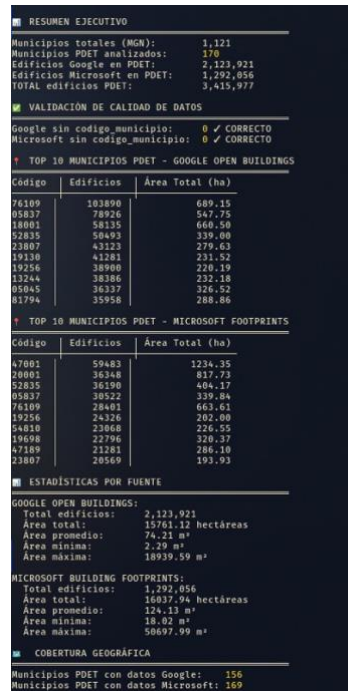
La ejecución genera:

logs pipeline.txt

Este archivo contiene:

- Timestamps del proceso.
- Validación de índices.
- Conteo de edificios procesados.
- Lectura y escritura de archivos.
- Mensajes de éxito o error.

Este mecanismo garantiza transparencia, trazabilidad y auditoría del pipeline.



```
■ RESUMEN EJECUTIVO
Municipios totales (MOR): 1,121
Municipios PDET analizados: 170
Edificios Google en PDET: 2,123,921
Edificios Microsoft en PDET: 1,292,056
TOTAL edificios PDET: 3,415,977

■ VALIDACIÓN DE CALIDAD DE DATOS
Google sin codigo_municipio: 0 ✓ CORRECTO
Microsoft sin codigo_municipio: 0 ✓ CORRECTO

↑ TOP 10 MUNICIPIOS PDET - GOOGLE OPEN BUILDINGS
Código | Edificios | Área Total (ha)
-----|-----|-----
06109 | 183898 | 689.15
05837 | 78926 | 547.75
18801 | 58135 | 660.58
52835 | 58493 | 339.08
23807 | 43123 | 279.63
19130 | 41281 | 231.52
19256 | 38908 | 220.19
13244 | 38386 | 232.18
05045 | 36337 | 326.52
01794 | 35958 | 288.86

↑ TOP 10 MUNICIPIOS PDET - MICROSOFT FOOTPRINTS
Código | Edificios | Área Total (ha)
-----|-----|-----
47001 | 59483 | 1234.35
20801 | 36348 | 817.73
52835 | 36198 | 404.17
05837 | 38522 | 339.84
06109 | 28401 | 663.61
19256 | 24326 | 202.08
54810 | 23968 | 226.55
19698 | 22796 | 326.37
47189 | 21281 | 286.18
23807 | 20569 | 193.93

■ ESTADÍSTICAS POR FUENTE
GOOGLE OPEN BUILDINGS:
Total edificios: 2,123,921
Área total: 15761.12 hectáreas
Área promedio: 74.21 m²
Área mínima: 2.29 m²
Área máxima: 18939.59 m²

MICROSOFT BUILDING FOOTPRINTS:
Total edificios: 1,292,056
Área total: 16837.94 hectáreas
Área promedio: 124.13 m²
Área mínima: 18.02 m²
Área máxima: 50697.99 m²

■ COBERTURA GEOGRÁFICA
Municipios PDET con datos Google: 156
Municipios PDET con datos Microsoft: 169
```

Figura 5. Logs del pipeline con la información de Municipios en Google y Microsoft

3. Precisión de las Operaciones Espaciales

3.1. Índices 2dsphere

Para garantizar precisión y eficiencia, se definieron índices en:

```
municipalities_pdet
buildings_google
buildings_microsoft
```

Mediante:

```
db.collection.createIndex({ geometry: "2dsphere" });
```

Esto permite:

- Consultas \$geoWithin precisas.
- Intersección adecuada de polígonos.
- Procesamiento eficiente incluso con geometrías complejas.

```
dba_proyectofinal> db.buildings_google.getIndexes()
[
  { v: 2, key: { '_id': 1 }, name: '_id_',
    v: 2,
    key: { geometry: '2dsphere' },
    name: 'geometry_2dsphere',
    '2dsphereIndexVersion': 3
  },
  { v: 2,
    key: { centroid: '2dsphere' },
    name: 'centroid_2dsphere',
    '2dsphereIndexVersion': 3
  },
  { v: 2,
    key: { building_id: 1 },
    name: 'building_id_1',
    unique: true
  },
  { v: 2, key: { codigo_municipio: 1 }, name: 'codigo_municipio_1' },
  { v: 2, key: { area_m2: 1 }, name: 'area_m2_1' }
]
dba_proyectofinal>
... db.buildings_google.findOne(
...   codigo_municipio: "76109"
... ), {
...   building_id: 1,
...   codigo_municipio: 1,
...   area_m2: 1,
...   geometry: 1
... })
{
  _id: ObjectId('691a3d84d8f800ab0033afdb'),
  building_id: 'G-Bldg-00080566',
  codigo_municipio: '76109',
  geometry: {
    type: 'Polygon',
    coordinates: [
      [
        [ -77.0269895197531, 3.87758895246983 ],
        [ -77.0269844618465, 3.87765031420812 ],
        [ -77.0270387336682, 3.87765482746553 ],
        [ -77.0270437917704, 3.87759346564895 ],
        [ -77.0269895197531, 3.87758895246983 ]
      ]
    ]
  },
  area_m2: 37.18172351464114
}
dba_proyectofinal>
```

Figura 6. Índices 2dsphere creados

3.2. Filtrado espacial

El filtrado se realizó municipio por municipio usando:

```
{
  geometry: {
    $geoWithin: { $geometry: municipio.geometry }
  }
}
```

Este método es apropiado para esquemas con Polygon y MultiPolygon, asegurando que solo footprints dentro de los límites municipales sean incluidos.

4. Estructura del pipeline

4.1. Flujo general del proceso

- Levantar contenedor MongoDB.
- Ejecutar run_pdet_pipeline.sh.

- Crear índices geoespaciales.
- Filtrar footprints Google por municipio PDET.
- Filtrar footprints Microsoft por municipio PDET.
- Calcular área total (m²).
- Exportar CSV consolidado.
- Registrar logs del proceso.

4.2. Archivos generados

A diferencia de versiones anteriores del pipeline, la versión final genera un único archivo consolidado:

resultados_pdet_por_municipio.csv

Incluye:

- Código DANE del municipio
- Total de edificios Google
- Total de edificios Microsoft
- Área total Google (m²)
- Área total Microsoft (m²)

Además se generó:

- logs_pipeline.txt

5. Resultados

A continuación se presentan las tablas correspondientes al Top 10 de municipios PDET con mayor número de edificaciones según Google Open Buildings y Microsoft Building Footprints.

Estos valores se transformaron a áreas tanto en hectáreas como en metros cuadrados.

Tabla 1: Conteo y área por municipio (Google)

Código municipio	Edificios	Área total (ha)	Área total (m ²)
47001	103890	689.15	6891500
20001	78926	547.77	5477700
52835	58135	660.50	6605000
05837	50493	339.00	3390000
76109	43123	279.63	2796300

19256	41281	231.52	2315200
54810	38900	220.19	2201900
19698	38386	232.18	2321800
47189	36337	326.52	3265200
23807	35958	288.86	2888600

Tabla 2: Conteo y área por municipio (Microsoft)

Código municipio	Edificios	Área total (ha)	Área total (m²)
76109	59483	1234.35	12343500
05837	36348	817.73	8177300
18001	36190	404.17	4041700
52835	30522	339.84	3398400
23807	28401	663.61	6636100
19130	24326	202.00	2020000
19256	23068	226.55	2265500
13244	22796	320.37	3203700
05045	21281	286.10	2861000
81794	20569	193.93	1939300

6. Mapas de Resultados Geoespaciales

A continuación, se presentan los mapas generados en QGIS a partir de la unión entre el shapefile oficial del MGN y la tabla resultados_pdet_por_municipio. Estos mapas permiten visualizar de forma clara los municipios PDET para los cuales se obtuvieron resultados válidos desde cada fuente de datos.

6.1.Municipios PDET con datos de Google Open Buildings

En el caso de Google Open Buildings, se obtuvo información válida para 156 municipios PDET.

El siguiente mapa muestra únicamente aquellos municipios donde el pipeline produjo resultados de conteo y área total de edificaciones.

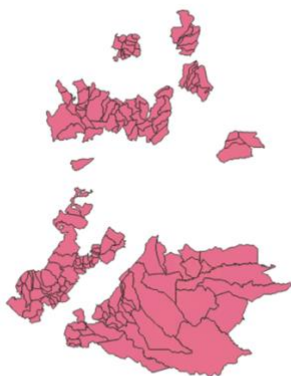


Figura 7. Municipios PDET procesados con Google Open Buildings

6.2. Municipios PDET con datos de Microsoft Building Footprints

Para Microsoft Building Footprints, se obtuvieron resultados válidos para 169 municipios PDET.

El siguiente mapa presenta los municipios donde se identificaron edificaciones mediante esta fuente.



Figura 8. Municipios PDET procesados con Microsoft Building Footprints

6.3. Comparación general

La cobertura espacial difiere entre fuentes:

- Google cubre 156 municipios.
- Microsoft cubre 169 municipios.

Microsoft presenta mayor continuidad en áreas urbanas, mientras que Google detecta mejor edificaciones dispersas.

7. Ejecución del Workflow

Para ejecutar completamente la Entrega 4, el usuario debe correr:

```
cd DBA_ProyectoFinal
docker-compose up -d mongo-upme
chmod +x entrega4/run_pdet_pipeline.sh
./entrega4/pdet_pipeline.sh
```

8. Conclusiones

La Entrega 4 consolidó un flujo de análisis geoespacial totalmente reproducible, que permite estimar de manera consistente y auditable el número de edificaciones y el área total de cubiertas dentro de los municipios PDET. La implementación mediante Docker garantiza independencia del entorno y evita inconsistencias entre sistemas operativos.

El uso de índices 2dsphere permitió lograr precisión en las operaciones espaciales realizadas, especialmente en filtros \$geoWithin y cálculos \$geoArea. Los archivos CSV generados constituyen la base para análisis comparativos entre Google y Microsoft, y satisfacen los requerimientos técnicos del proyecto.

Finalmente, la estructura modular del workflow facilita su actualización, depuración y ampliación, permitiendo que futuros análisis puedan incorporarse sin modificar la integridad del proceso. Este pipeline cumple los criterios de trazabilidad, reproducibilidad y transparencia exigidos en la entrega.