Paolo Melchiorre
@pauloxnet

Maps *with* GeoDjango PostGIS *and* Leaflet

Paolo Melchiorre @pauloxnet

- Computer Science Engineer
- Python Developer *since* 2006
- PostgreSQL user *(not a DBA)*
- Django Developer *since* 2011
- Remote Worker *since* 2015
- Senior Developer *at* 20tab

www.20tab.com

- Rome based with remote workers
- Meetup and conferences
- Agile and Lean methodologies
- Growth marketing approach
- Software development
- Python, Django, React JS, PostgreSQL

Goal

Find a simple way

to integrate a web map

in a Django project.


Basic map
Use case
PostGIS
GeoDjango
Leaflet JS
Leaflet

Web map

- Map delivered by GIS
- Static and Dynamic
- Interactive and view only
- Raster or Vector tiles
- Spatial databases
- Javascript library

GeoDjango

- django.contrib.gis
- Geographic framework
- Spatial Field Types
- Spatial ORM queries
- Admin geometry fields
- 4 database backends

PostGIS

- Best GeoDjango backend
- PostgreSQL extension
- Integrated spatial data
- Spatial data types
- Spatial indexing
- Spatial functions

Leaflet

- JavaScript library for maps
- Free Software
- Desktop & Mobile friendly
- Light (< 40 KB of gizp JS)
- Well documented
- Simple, performing, usable

Basic map example

Making queries

```
from django.db import models

class Entry(models.Model):
    headline = models.CharField(max_length=255)
    body_text = models.TextField()

    def __str__(self):
        return self.headline
```

Making queries - SQL

```
BEGIN;
--
-- Create model Entry
--
CREATE TABLE "blog_entry" (
    "id" serial NOT NULL PRIMARY KEY,
    "headline" varchar(255) NOT NULL,
    "body_text" text NOT NULL
);
COMMIT;
```

Settings

```
INSTALLED_APPS = [
    # …
    'django.contrib.gis',
]

DATABASES = {'default': {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    # …
}}
```

Migrations

```
from django.contrib.postgres import operations
from django.db import migrations

class Migration(migrations.Migration):
    dependencies = [('blog', '0001_initial')]
    operations = [
        operations.CreateExtension('postgis')
    ]
```

Migrations - SQL

```
BEGIN;
--
-- Creates extension postgis
--
CREATE EXTENSION IF NOT EXISTS "postgis";
COMMIT;
```

Point field

```
from django.contrib.gis.db.models import PointField
from django.db import models

class Entry(models.Model):
    # …
    point = PointField()

    @property
    def lat_lon(self):
        return list(getattr(self.point, 'coords', [])[::-1])
```


Point field - SQL

```
BEGIN;
--
-- Add field point to entry
--
ALTER TABLE "blog_entry"
  ADD COLUMN "point" geometry(POINT,4326) NOT NULL;
CREATE INDEX "blog_entry_point_id"
  ON "blog_entry" USING GIST ("point");
COMMIT;
```

Admin

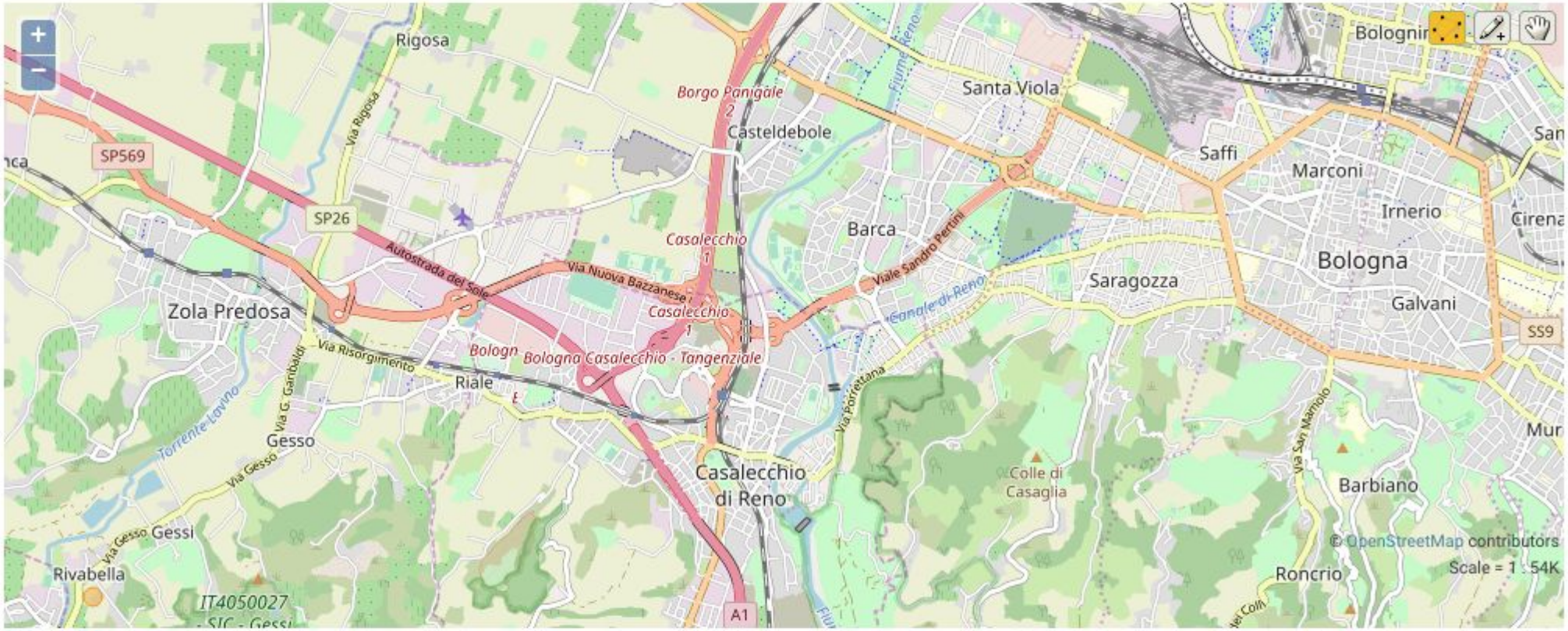
```
from django.contrib import admin
from django.contrib.gis.admin import OSMGeoAdmin
from .models import Entry

@admin.register(Entry)
class EntryAdmin(OSMGeoAdmin):
    default_lon = 1263000
    default_lat = 5542000
    default_zoom = 12
    # …
```


Change Entry

HISTORY

Point:



Delete all Features

Delete

Save and add another

Save and continue editing

SAVE

Views and urls

```
from django.urls import path
from django.views.generic import ListView
from .models import Entry

class EntryList(ListView):
    queryset = Entry.objects.filter(point__isnull=False)

urlpatterns = [
    path('map/', EntryList.as_view()),
]
```

Views and urls - SQL

```
SELECT "blog_entry"."id",
       "blog_entry"."headline",
       "blog_entry"."body_text",
       "blog_entry"."point"::bytea
FROM "blog_entry"
WHERE "blog_entry"."point" IS NOT NULL
```

Template

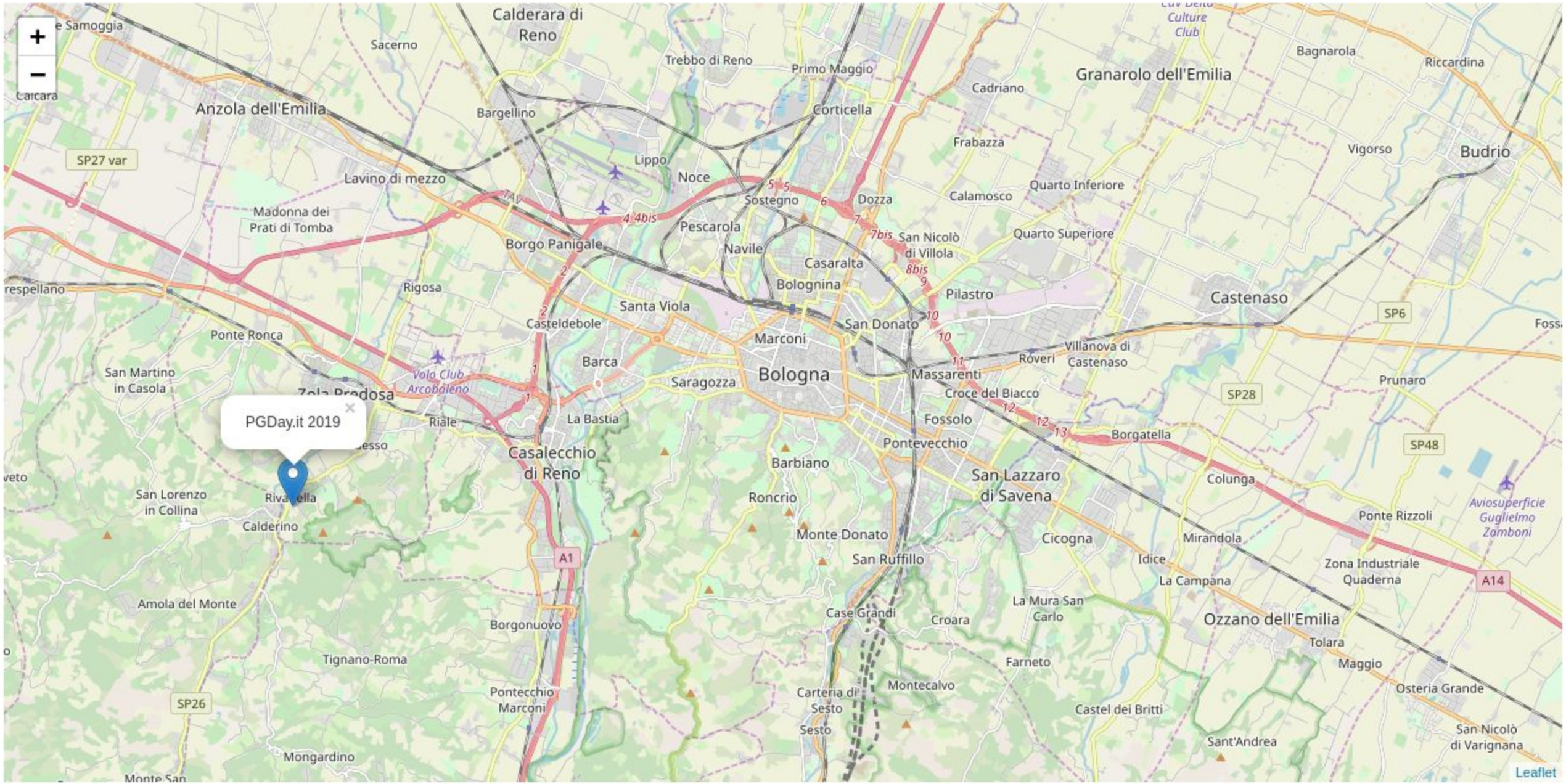
```
<html><head>
  <link rel="stylesheet"
        href="//unpkg.com/leaflet/dist/leaflet.css"/>
  <script src="//unpkg.com/leaflet/dist/leaflet.js"></script>
</head>

<body><h1>PGDay.IT 2019 Venue</h1>
  <div id="m" style="width: 1920px; height: 1080px;"></div>
  <!-- add javascript here -->
</body></html>
```

Javascript

```
<script type="text/javascript">
  var m = L.map('m').setView([44.49, 11.34], 12); # Bologna
  L.tileLayer('//{s}.tile.osm.org/{z}/{x}/{y}.png').addTo(m);
  {% for e in object_list %}
    L.marker({{e.lat_lon}}).addTo(m);
  {% endfor %}
</script>
```

PGDay.IT 2019 Venue



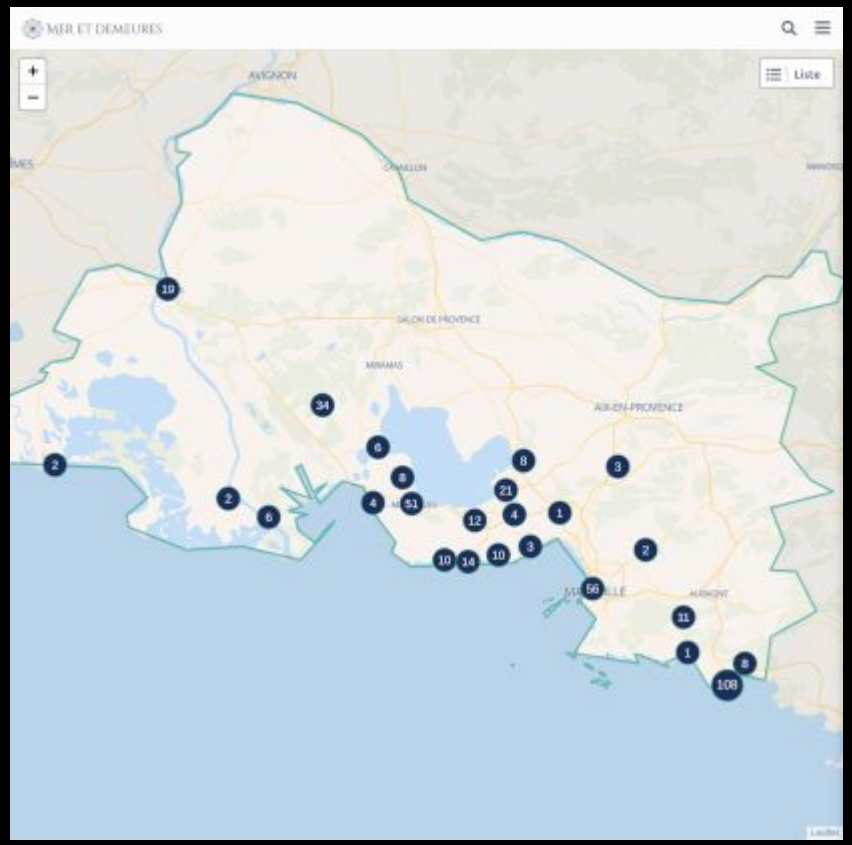
Use case

- Coastal properties
- Active since 2014
- 8 Languages
- ~ 100k active advertisements
- ~ 40 Countries
- 6 Continents

Version 1.0

- Django 1.6
- Python 2.7
- PostgreSQL 9.3
- Text Spatial Fields
- Leaflet 1.0
- Static/View-only map

Version 2.0



- Django 2.2 / GeoDjango
- Python 3.6
- PostgreSQL 10
- PostGIS 2.4 / Spatial data
- Leaflet 1.5
- Dynamic/Interactive map

Models

```
from django.db import models
from django.contrib.gis.db.models import (
    MultiPolygonField, PointField
)

class City(models.Model):
    borders = MultiPolygonField()

class Ad(models.Model):
    location = PointField()
```

City - SQL

```
BEGIN;
--
-- Create model City
--
CREATE TABLE "blog_city" (
    "id" serial NOT NULL PRIMARY KEY,
    "borders" geometry(MULTIPOLYGON,4326) NOT NULL
);
CREATE INDEX "blog_city_borders_id"
  ON "blog_city" USING GIST ("borders");
COMMIT;
```

Ad - SQL

```
BEGIN;
--
-- Create model Ad
--
CREATE TABLE "blog_ad" (
    "id" serial NOT NULL PRIMARY KEY,
    "location" geometry(POINT,4326) NOT NULL
);
CREATE INDEX "blog_ad_location_id"
  ON "blog_ad" USING GIST ("location");
COMMIT;
```

RESTful API

```
$ pip install djangorestframework      # RESTful API
$ pip install djangorestframework-gis # Geographic add-on
$ pip install django-filter            # Filtering support
```

```
INSTALLED_APPS = [
    # …
    'django.contrib.gis',
    'rest_framework',
    'rest_framework_gis',
    'django_filters',
]
```


Serializer

```
from rest_framework_gis.serializers import (
    GeoFeatureModelSerializer
)
from .models import Ad

class AdSerializer(GeoFeatureModelSerializer):
    class Meta:
        model = Ad
        geo_field = 'location'
        fields = ('id',)
```

Views

```
from rest_framework.viewsets import ReadOnlyModelViewSet
from rest_framework_gis.filters import InBBoxFilter
from .models import Ad
from .serializers import AdSerializer

class AdViewSet(ReadOnlyModelViewSet):
    bbox_filter_field = 'location'
    filter_backends = (InBBoxFilter,)
    queryset = Ad.objects.filter(location__isnull=False)
    serializer_class = AdSerializer
```

Views - SQL

```
SELECT "blog_ad"."id", "blog_ad"."location"::bytea
FROM "blog_ad"
WHERE (
    "blog_ad"."location" IS NOT NULL AND
    "blog_ad"."location" @ ST_MakeEnvelope(
        5, 35, 20, 45, 4326
    )
)
```

Urls

```
from rest_framework.routers import DefaultRouter
from .views import AdViewSet


router = DefaultRouter()
router.register(r'markers', AdViewSet, basename='marker')
urlpatterns = router.urls
```

GeoJSON

```
{"type": "FeatureCollection", "features": [{
    "id": 1,
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [11.203305, 44.467230]
    },
    "properties": {}
}]}
```

MER ET DEMEURES

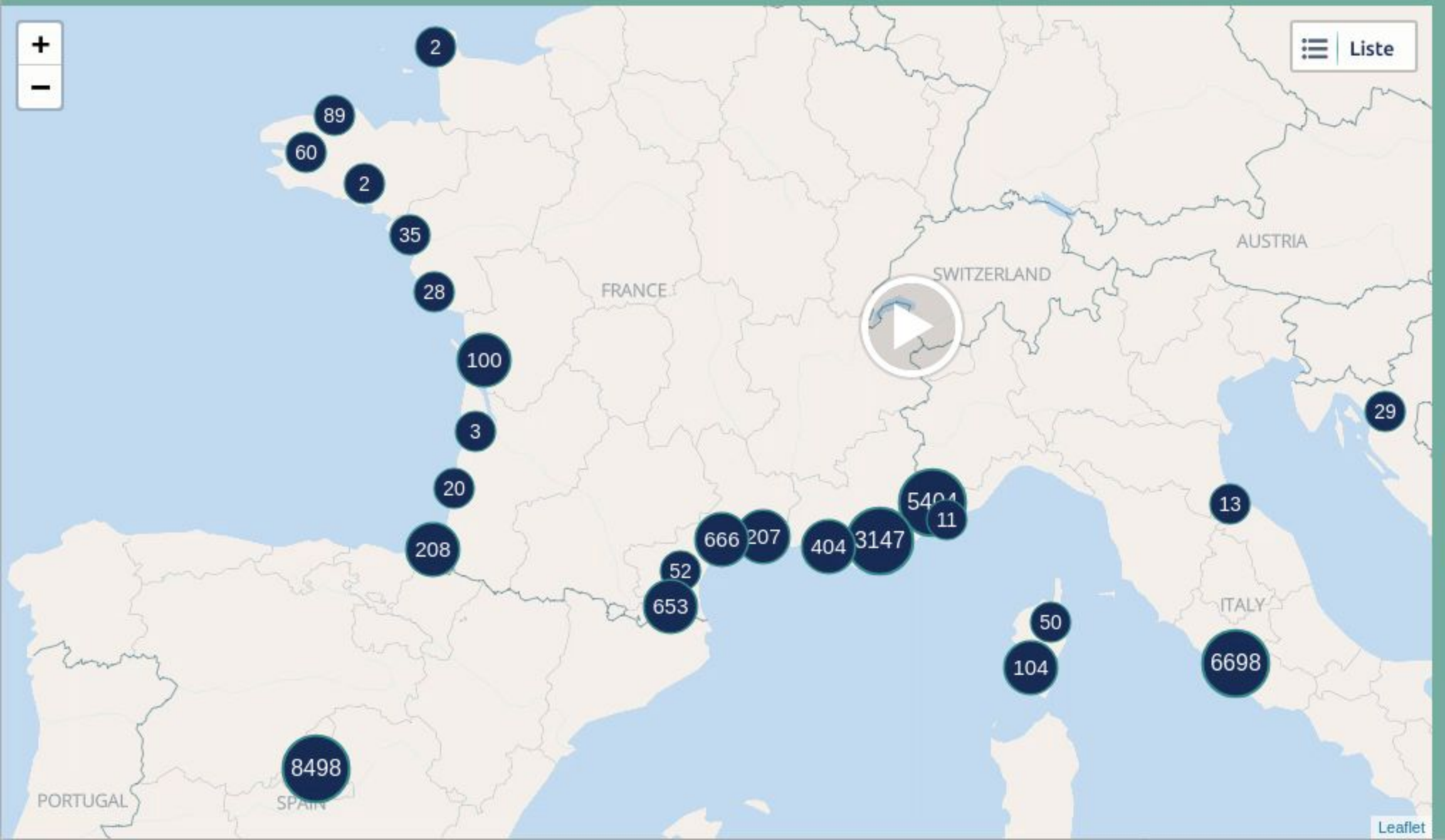
Place an ad Login

Sale | France | Type de Bien | Plus de Critères +

11235 résultats pour:
Sale properties - France

Trier par...

Liste



€520,000

Apartment, Royan

2 1 70 m²

€290,000

Apartment, Cannes

35 m²

villa for sale in le-lavandou var

Conclusion

- Out-of-the-box features
- Spatial & Relational queries
- Django/PostgreSQL
- Backend clusterization
- Administrative levels
- Dynamic spatial entity

Resources

- docs.djangoproject.com/en/
- github.com/django/django
- postgis.net/docs/
- github.com/postgis/postgis
- leafletjs.com/reference.html
- github.com/leaflet/leaflet

Acknowledgments

Mer et Demeures
meretdemeures.com

Twentytab
info@20tab.com

Info

- www.paulox.net/talks
- Slides
- Code samples
- Resource URLs
- Questions and comments
- License (CC BY-SA)

Thank you!

Paolo Melchiorre

www.paulox.net

@pauloxnet