T-rex, a vector tile server for your own data

@PirminKalberer
Sourcepole, Zurich, Switzerland
www.sourcepole.com

SOURCEPOLE
Linux & Open Source Solutions

Vector Tiles

Demo

- **Vector tile demo**
  - https://www.mapbox.com/maps/

Demo

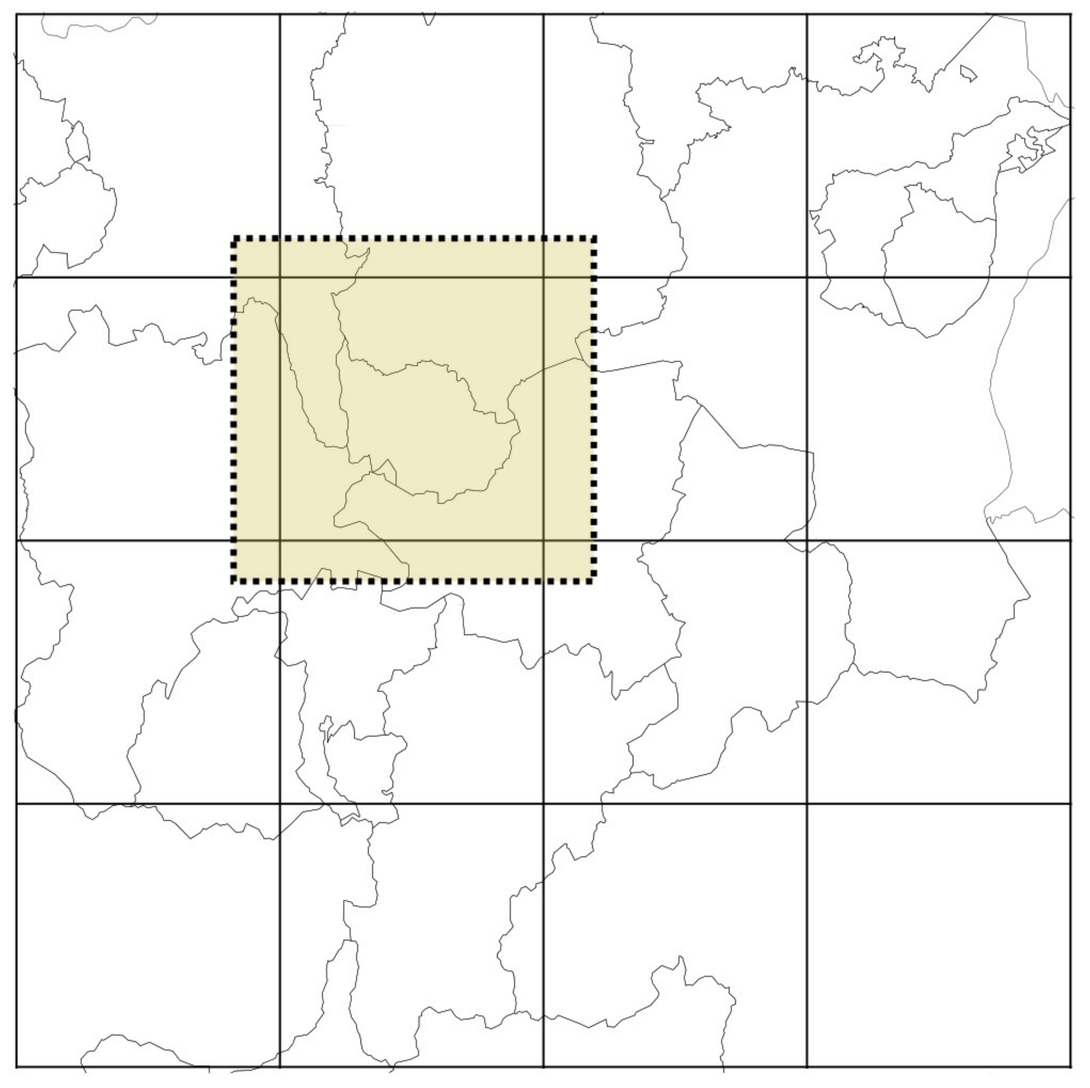
- **Vector tile demo**
  - https://www.mapbox.com/maps/

Mapbox Vector Tiles

https://github.com/mapbox/vector-tile-spec

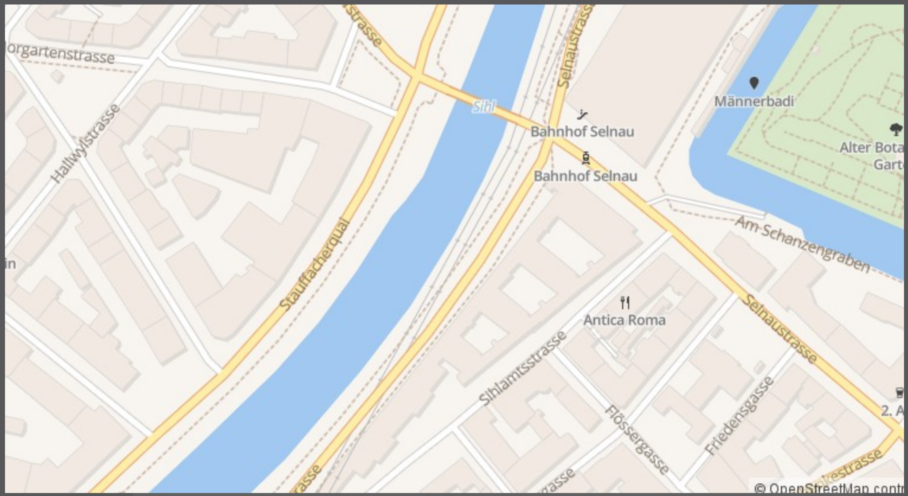
- Protocol buffer format (PBF, binary, Streamable)
- Geometry in screen pixel coordinates (Integers, ZigZag encoded)
- Multipoint/Multiline/Multipolygon
- Non-spatial attributes (optional Feature-ID)
- Multiple layers per tile

Mapbox Vector Tiles



Vector tile size

- **OSM data set:**
  - Boston: 24 MB
  - USA: 7.2 GB
  - Planet: 55 GB



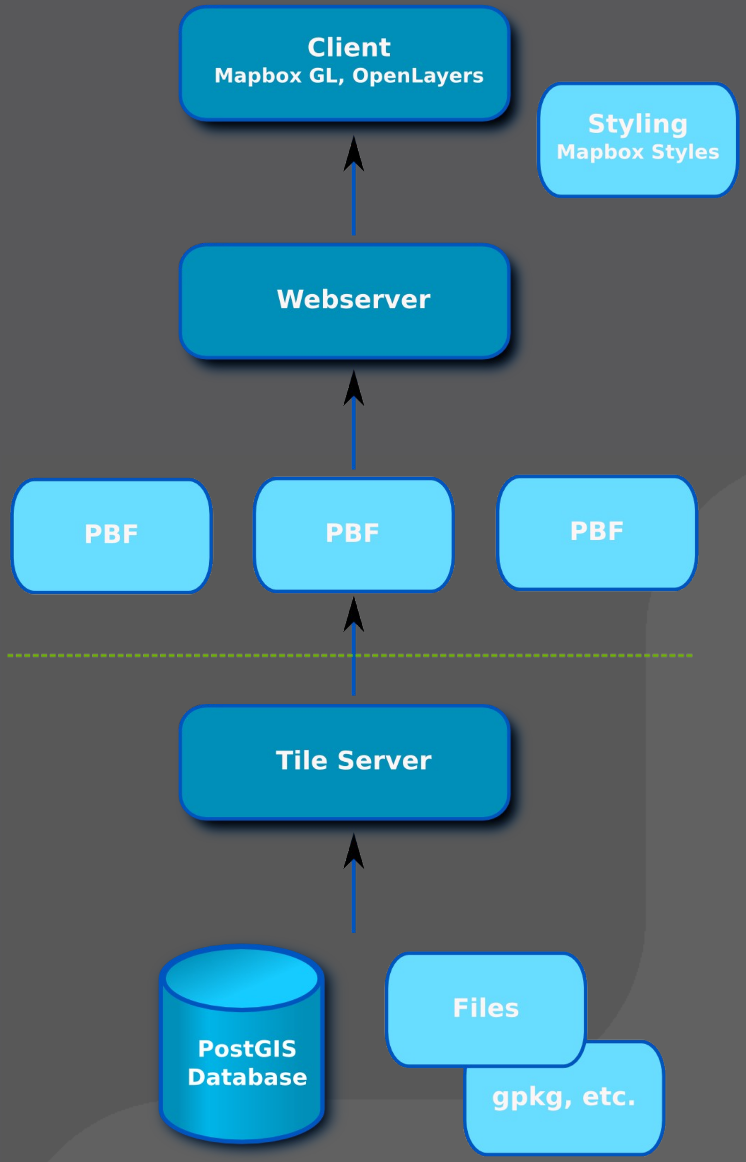
- **Offline maps!**
- **Download & Build-Tools:**
  - http://osm2vectortiles.org/
  - https://openmaptiles.org/

WMS -> WMTS -> Vector tiles

- WMS
  - No tiling problems (labels, etc.)
  - Printing
- WMTS
  - Scalability
  - Caching (server and client)
- Vector Tiles
  - Scalability
  - Caching (server and client)
  - Interactivity
  - Flexible styling (client-side rendering)
  - Hi-DPI

Creating vector tiles

Vector tile stack for custom data



Styling / viewer

- **Mapbox Styles (JSON)**
  - Viewer:
    - Mapbox GL JS
    - OpenLayers 3/4
  - Style Editor (OSS)
    - Maputnik
- **Mapzen Tangram Styles (YAML)**
  - Viewer:
    - Tangram
  - Style Editor (OSS)
    - Tangram Play

Vector tile creation

- **Read geodata within tiles borders**
- **Clip geometries**
- **Simplify geometries**
  - Polygons: e.g. SnapToGrid
  - Lines: e.g. Douglas-Peucker
  - Points: clustering
- **Generate label points**
- **Deliver MVT (Protobuf) format**
- **Serve live or seed cache (parallelization!)**

Vector tile stack for custom data (PG)

- node-mapnik (Kartotherian, tessera)
- Tilezen tileserver
- Tegola
- t-rex
- GeoServer
- PostGIS ST_AsMVT

https://github.com/mapbox/awesome-vector-tiles

t-rex

t-rex

- Multiple datasources (PostGIS + GDAL/OGR)
- Auto-detection of layers in PostGIS database
- Built-in viewers for data display and inspection
- Tile generation command with simple parallelization
- Automatic reprojection to grid CRS
- Support for custom tile grids
- Single executable

Rust

- New programming language from Mozilla
- Next Firefox rendering engine
- Systems programming (like C, C++)
- Zero-cost abstractions
- Guaranteed memory safety
- https://www.rust-lang.org/

Workflow with t-rex (1)

- **Installation:**
  - http://t-rex.tileserver.ch/

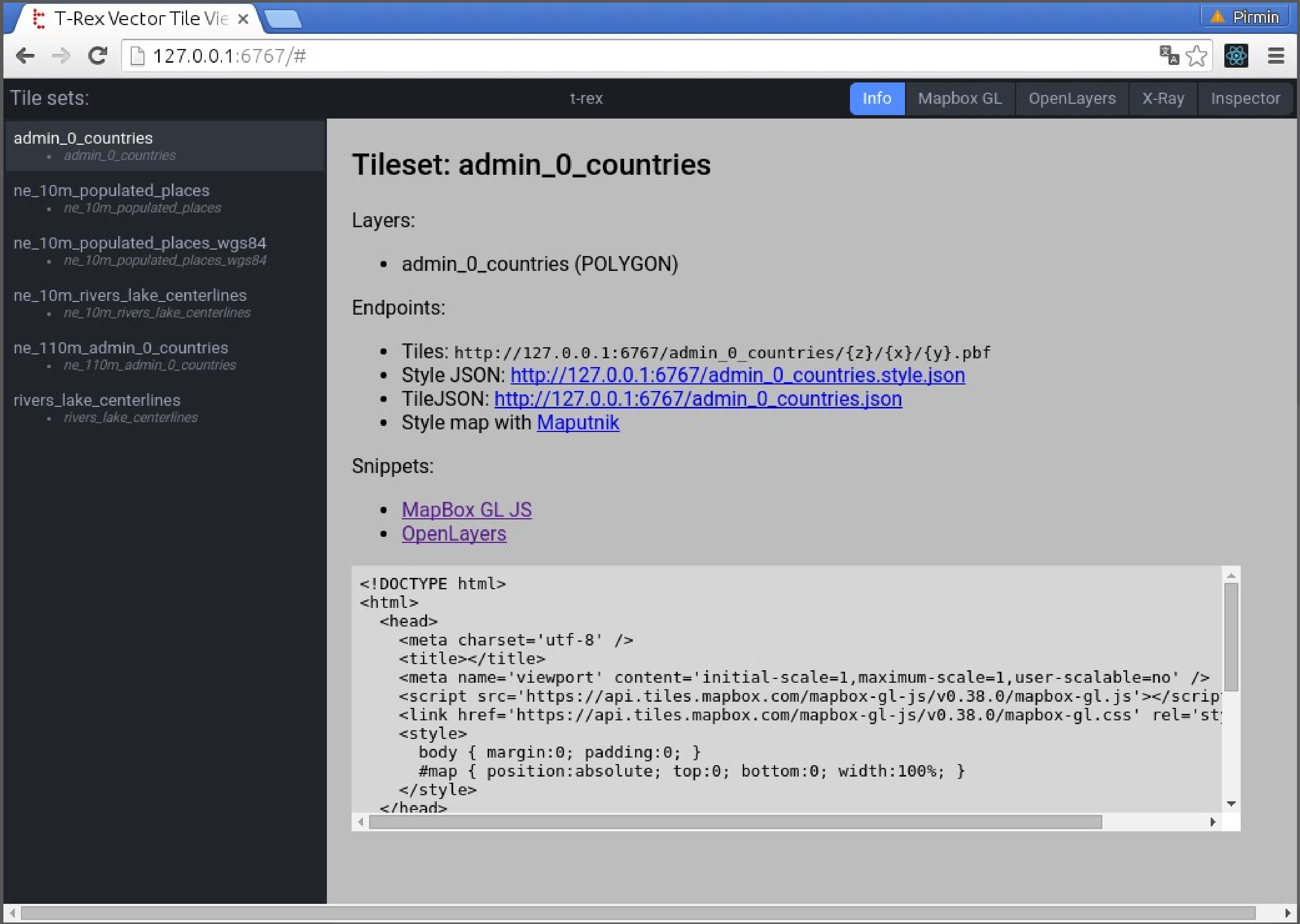
- **Start server:**

```
t_rex serve --dbconn postgresql://user@host/database

or

t_rex serve --datasource <file_or_gdal_ds>
```

OGR/GDAL examples (t-rex >0.8)

- natural_earth.gpkg
- ne_110m_coastline.shp
- placemarks.kml
- route.gpx
- osm.pbf
- https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_week.geojson
- spreadsheet.vrt (Spreadsheet with lat/lon columns)
- dm01avch24d.itf, inspire.gml, OCI, ...

Built-in viewer



Tile sets:

t-rex

- admin_0_countries
  - admin_0_countries
- ne_10m_populated_places
  - ne_10m_populated_places
- ne_10m_populated_places_wgs84
  - ne_10m_populated_places_wgs84
- ne_10m_rivers_lake_centerlines
  - ne_10m_rivers_lake_centerlines
- ne_110m_admin_0_countries
  - ne_110m_admin_0_countries
- rivers_lake_centerlines
  - rivers_lake_centerlines

Tileset: admin_0_countries

Layers:

- admin_0_countries (POLYGON)

Endpoints:

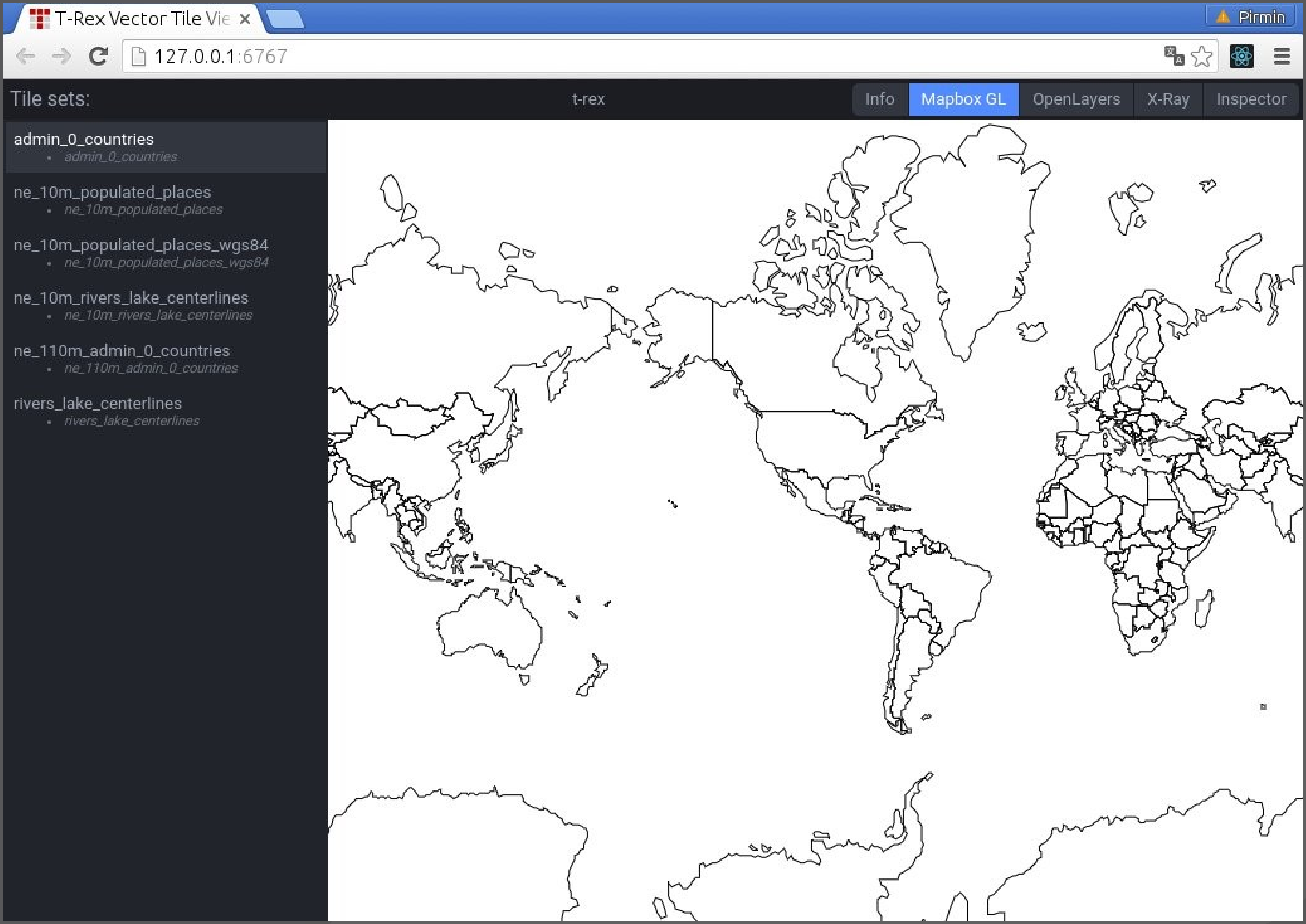
- Tiles: `http://127.0.0.1:6767/admin_0_countries/{z}/{x}/{y}.pbf`
- Style JSON: http://127.0.0.1:6767/admin_0_countries.style.json
- TileJSON: http://127.0.0.1:6767/admin_0_countries.json
- Style map with Maputnik

Snippets:

- MapBox GL JS
- OpenLayers

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='utf-8' />
    <title></title>
    <meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
    <script src='https://api.tiles.mapbox.com/mapbox-gl-js/v0.38.0/mapbox-gl.js'></scrip
    <link href='https://api.tiles.mapbox.com/mapbox-gl-js/v0.38.0/mapbox-gl.css' rel='st
    <style>
      body { margin:0; padding:0; }
      #map { position:absolute; top:0; bottom:0; width:100%; }
    </style>
  </head>
```

Mapbox GL JSON viewer

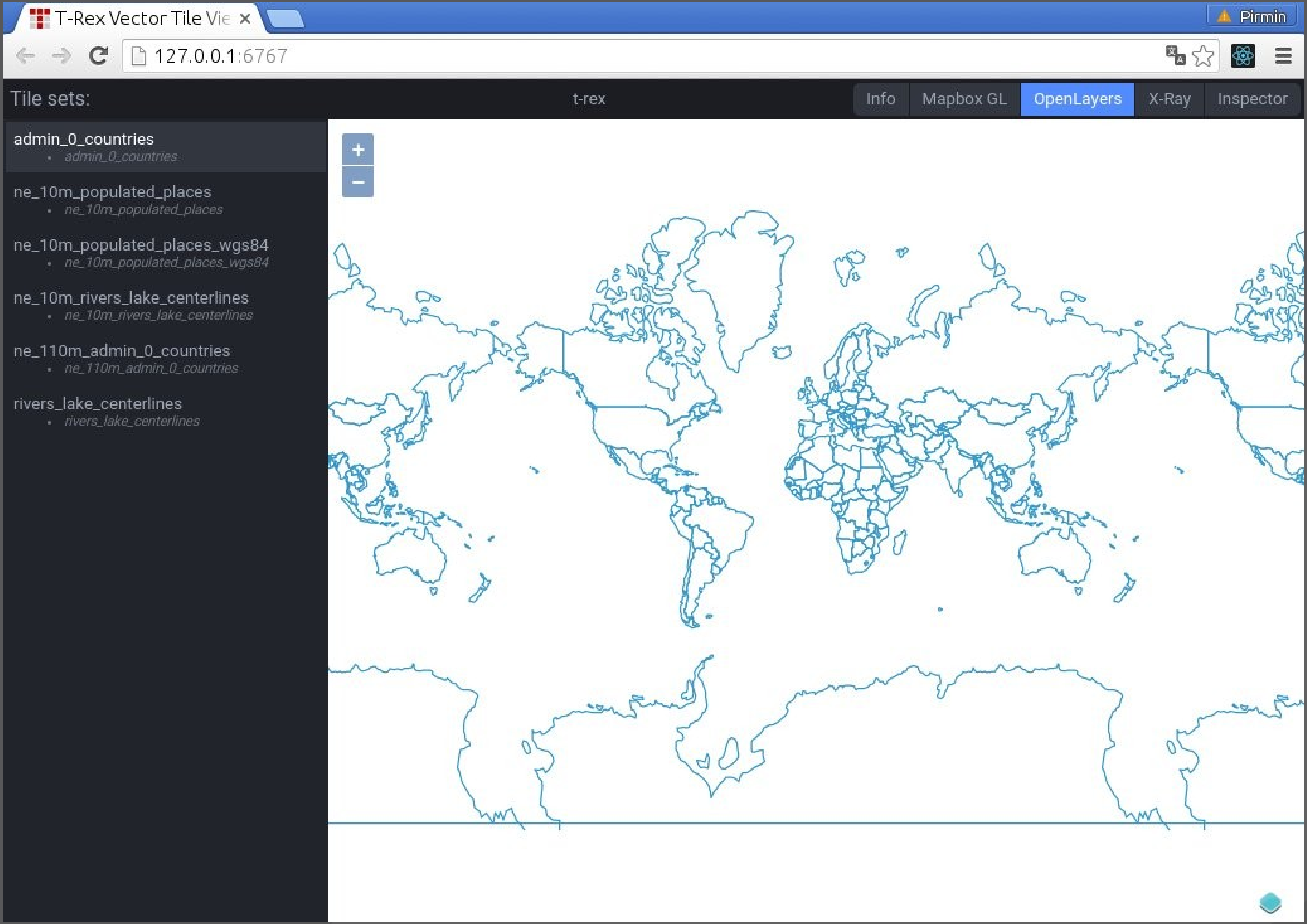


Tile sets:

t-rex

- admin_0_countries
  - admin_0_countries
- ne_10m_populated_places
  - ne_10m_populated_places
- ne_10m_populated_places_wgs84
  - ne_10m_populated_places_wgs84
- ne_10m_rivers_lake_centerlines
  - ne_10m_rivers_lake_centerlines
- ne_110m_admin_0_countries
  - ne_110m_admin_0_countries
- rivers_lake_centerlines
  - rivers_lake_centerlines

OpenLayers viewer



admin_0_countries
- admin_0_countries

ne_10m_populated_places
- ne_10m_populated_places

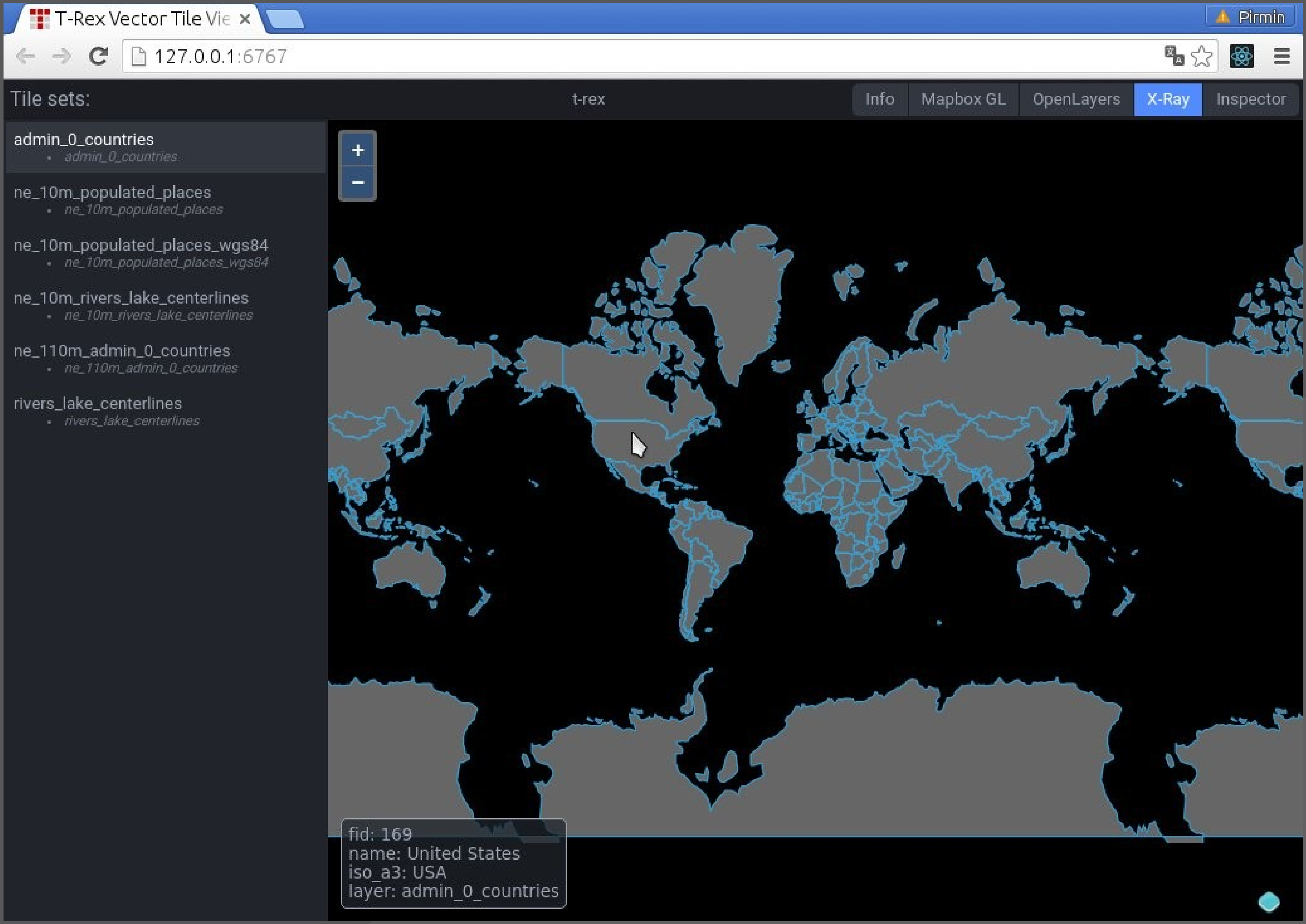
ne_10m_populated_places_wgs84
- ne_10m_populated_places_wgs84

ne_10m_rivers_lake_centerlines
- ne_10m_rivers_lake_centerlines

ne_110m_admin_0_countries
- ne_110m_admin_0_countries

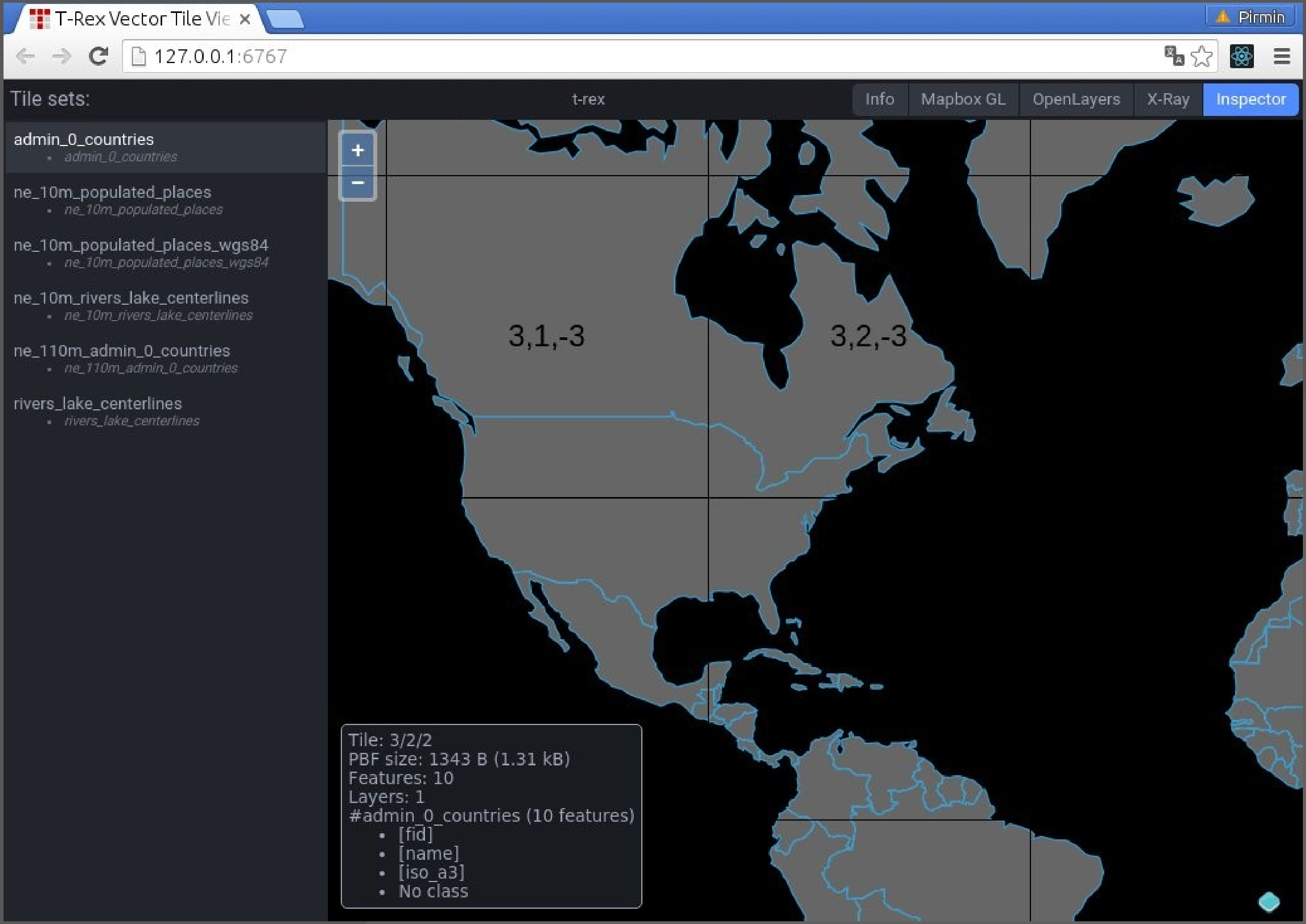
rivers_lake_centerlines
- rivers_lake_centerlines

X-Ray viewer



```
fid: 169
name: United States
iso_a3: USA
layer: admin_0_countries
```

X-Ray viewer



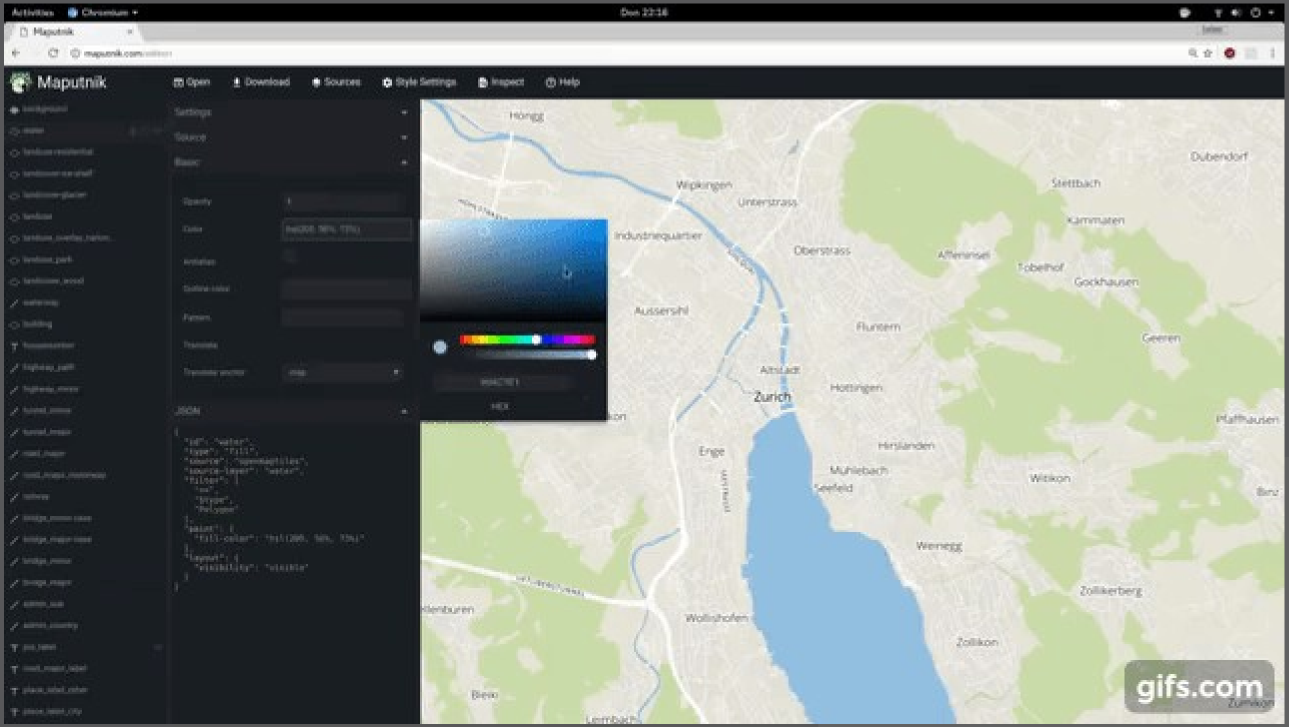
Tile sets:

t-rex

- admin_0_countries
  - admin_0_countries
- ne_10m_populated_places
  - ne_10m_populated_places
- ne_10m_populated_places_wgs84
  - ne_10m_populated_places_wgs84
- ne_10m_rivers_lake_centerlines
  - ne_10m_rivers_lake_centerlines
- ne_110m_admin_0_countries
  - ne_110m_admin_0_countries
- rivers_lake_centerlines
  - rivers_lake_centerlines

```
Tile: 3/2/2
PBF size: 1343 B (1.31 kB)
Features: 10
Layers: 1
#admin_0_countries (10 features)
    • [fid]
    • [name]
    • [iso_a3]
    • No class
```

Create styles with Maputnik



Opacity: 1

Color: hsl(205, 56%, 73%)

Translate anchor: map

```
{
  "id": "water",
  "type": "fill",
  "source": "openmaptiles",
  "source-layer": "water",
  "filter": [
    "==",
    "$type",
    "Polygon"
  ],
  "paint": {
    "fill-color": "hsl(205, 56%, 73%)"
  },
  "layout": {
    "visibility": "visible"
  }
}
```

- Kickstarter financed OSS editor by Lukas Martinelli
- Integrated in t-rex backend

Workflow with t-rex (3)

- Generate a configuration template:

```
t_rex genconfig --dbconn postgresql://user@host/database
```

- Run with configuration file:

```
t_rex serve --config myconfig.cfg
```

Workflow with t-rex (4)

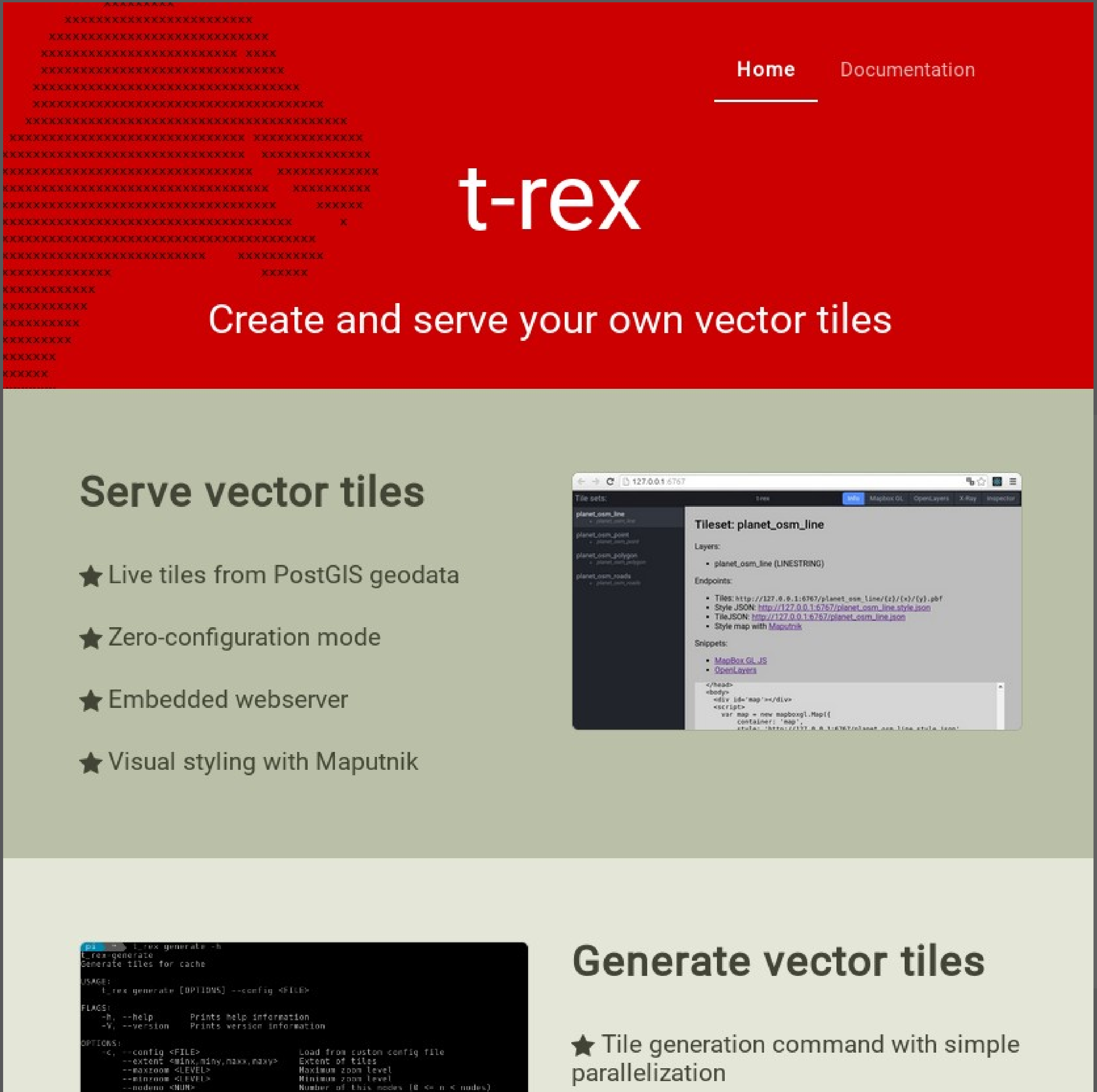
- **Generate tile cache:**

```
t_rex generate --config myconfig.cfg
```

- **Create MBTiles File:**

```
mb-util --image_format=pbf /tmp/mvtcache/ne ne.mbtiles
```

t-rex.tileserver.ch



t-rex

Create and serve your own vector tiles

Serve vector tiles

- ★ Live tiles from PostGIS geodata
- ★ Zero-configuration mode
- ★ Embedded webserver
- ★ Visual styling with Maputnik

Tileset: planet_osm_line

Layers:

- planet_osm_line (LINESTRING)

Endpoints:

- Tiles: `http://127.0.0.1:6767/planet_osm_line/{z}/{x}/{y}.pbf`
- Style JSON: http://127.0.0.1:6767/planet_osm_line.style.json
- TileJSON: http://127.0.0.1:6767/planet_osm_line.json
- Style map with Maputnik

Snippets:

- MapBox GL JS
- OpenLayers

```
</head>
<body>
  <div id='map'></div>
  <script>
    var map = new mapboxgl.Map({
        container: 'map',
        style: 'http://127.0.0.1:6767/planet_osm_line.style.json',
```

```
pi  ~  t_rex generate -h
t_rex-generate
Generate tiles for cache

USAGE:
    t_rex generate [OPTIONS] --config <FILE>

FLAGS:
    -h, --help       Prints help information
    -V, --version    Prints version information

OPTIONS:
    -c, --config <FILE>                       Load from custom config file
        --extent <minx,miny,maxx,maxy>        Extent of tiles
        --maxzoom <LEVEL>                     Maximum zoom level
        --minzoom <LEVEL>                     Minimum zoom level
        --nodeno <NUM>                        Number of this nodes (0 <= n < nodes)
```

Generate vector tiles

- ★ Tile generation command with simple parallelization

Roadmap

- Release of 0.8 with OGR/GDAL support
- Clipping & Simplification for OGR/GDAL layers
- More cache output formats (S3, etc.)
- Performance optimizations for big geometries
- More to come - driven by customer needs or contributions
- ST_AsMVT integration?

Questions? Thank you!

@PirminKalberer
@sourcepole