WebGL to render vector data
&
Cloud Optimized Geotiff

Guadaltel
el valor de la diferencia

Índice de contenido

1. Principales objetivos
2. Arquitectura definida
3. Conclusiones

Principales objetivos

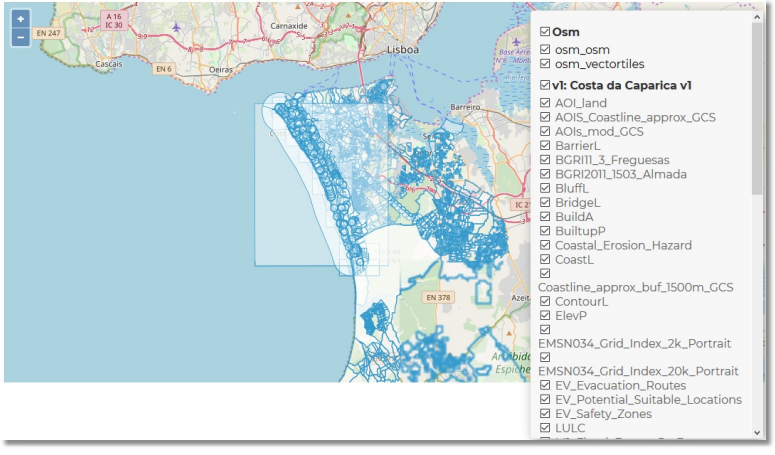
- Consumo de datos vectoriales
  - Consumo de datos masivo: > 500.000 features
  - Incorporación de datos de OSM
  - Requisitos tecnológicos
    - Infraestructura backend mínima
      - Sin base de datos
      - Sin servidor de mapas
    - Consumo de datos estáticos por lotes
  - Uso de tiles vectoriales .pbf

Principales objetivos

- Consumo de datos raster
  - Origen de imágenes satelitales en formato .tiff
  - Requisitos tecnológicos
    - Infraestructura backend mínima
      - Sin servidor de mapas
      - Publicación dinámica de datos
  - Uso de Cloud Optimized Geotiff

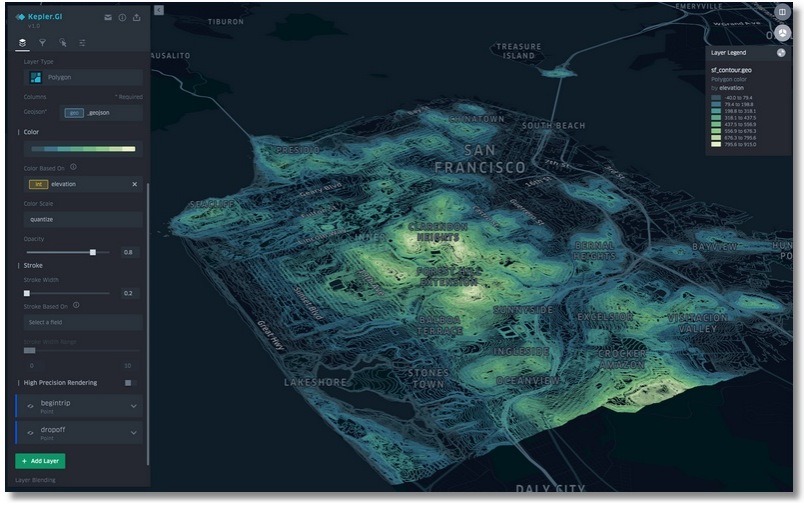
Arquitectura definida

- Tecnología del visualizador
  - OpenLayers
    - Renderizado de .pbf soportado
    - Buen rendimiento
    - No hace uso de WebGL para datos vectoriales
    - Problemas de rendimiento para gran volumen de features



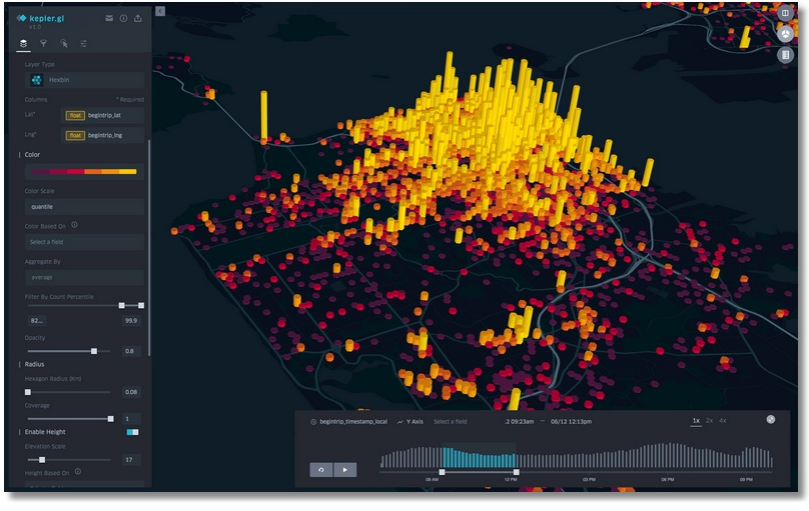
Arquitectura definida

- Tecnología del visualizador
  - Kepler.gl
    - React.js sobre deck.gl
    - Funcionalidades desarrolladas
    - Uso de WebGL para datos vectoriales
    - Gestiona gran volumen de datos
    - Orientado a la carga de .json



Arquitectura definida

- Tecnología del visualizador
  - Kepler.gl
    - .json cargado de una vez
    - Adaptaciones necesarias:
      - Carga de fuentes http://
      - Gestión de datasets por niveles de zoom
      - Gestión de tiles a cada nivel de zoom
      - Mergeo de información vectorial
    - Gestión de un gran número de llamadas producía una bajada en el rendimiento

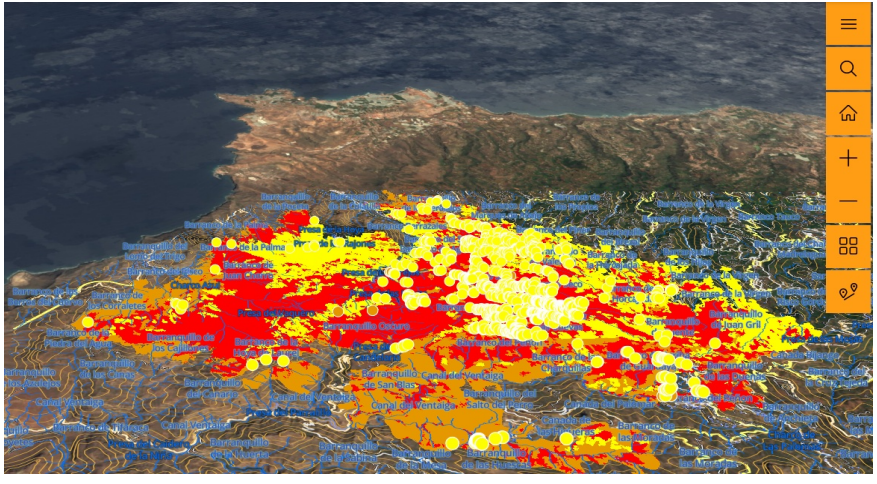


Arquitectura definida

- Tecnología del visualizador
  - deck.gl
    - React.js sobre Mapbox GL
    - Orientación a componentes
    - Diseño modular
    - Arquitectura rígida
    - Problemas en la interacción entre los componentes

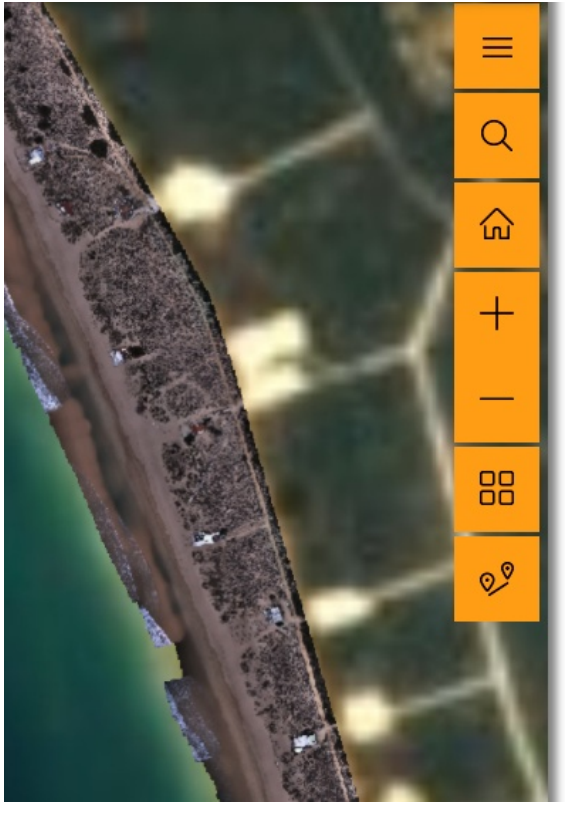
Arquitectura definida

- Tecnología del visualizador
  - Mapbox GL
    - Componentes react.js
    - Orientación a componentes
    - Diseño modular
    - Rendimiento óptimo
    - Unificación de .pbf
    - Estilos específicos
    - Renderizado en cliente



Arquitectura definida

- Tecnología del visualizador
  - geotiff.io vs rdnt.io
    - Consumo de imágenes en streaming
    - Reducción del ancho de banda
    - Acceso a la zona de interés sin acceder al global del fichero raster
    - Client vs. Proxy approach



Conclusiones

- Entorno productivo haciendo uso de arquitectura serverless (mostly)
  - WebGL para datos vectoriales
    - Mapbox GL
  - COG para datos raster
    - Geotiff.io
- Escalabilidad intrínseca
- Gestión de estilos en cliente

Gracias por su atención

jenriquesoriano@guadaltel.com