



Clase 04: Pipeline de Desarrollo Geoespacial

Arquitectura y mejores prácticas para soluciones geoespaciales

Profesor: Francisco Parra O.

2 de septiembre de 2025

USACH - Ingeniería Civil en Informática

Agenda

Introducción: ¿Qué es un Pipeline Geoespacial?

Arquitectura de un Pipeline Geoespacial

Adquisición de Datos Geoespaciales

Almacenamiento y Gestión de Datos

Procesamiento y Análisis

APIs y Servicios

Visualización y Dashboards

Introducción: ¿Qué es un Pipeline Geoespacial?

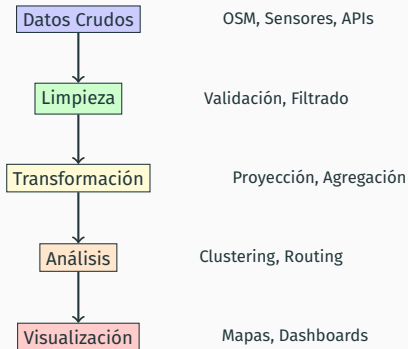
¿Qué es un Pipeline Geoespacial?

Definición

Un **pipeline geoespacial** es una secuencia automatizada de procesos que transforman datos geográficos crudos en información accionable.

Características clave:

- 🗄 Manejo de grandes volúmenes
- 🔄 Procesamiento continuo
- ✅ Validación automática
- 📈 Escalabilidad



¿Por qué necesitamos pipelines?

Problemas sin pipeline:

- ✗ Procesos manuales repetitivos
- ✗ Errores humanos frecuentes
- ✗ Difícil reproducibilidad
- ✗ Escalabilidad limitada
- ✗ Falta de trazabilidad

Beneficios con pipeline:

- ✓ Automatización completa
- ✓ Consistencia garantizada
- ✓ Reproducibilidad total
- ✓ Escalabilidad horizontal
- ✓ Auditoría y monitoreo

⚠ Un pipeline bien diseñado puede reducir el tiempo de procesamiento de días a minutos

Arquitectura de un Pipeline Geoespacial

Principios SOLID en Pipelines Geoespaciales

S - Single Responsibility

- Cada módulo una tarea
- Geocoder solo geocodifica
- Router solo calcula rutas

O - Open/Closed

- Extensible para nuevas fuentes
- Cerrado para modificaciones core

L - Liskov Substitution


- Interfaces consistentes
- Proveedores intercambiables

I - Interface Segregation

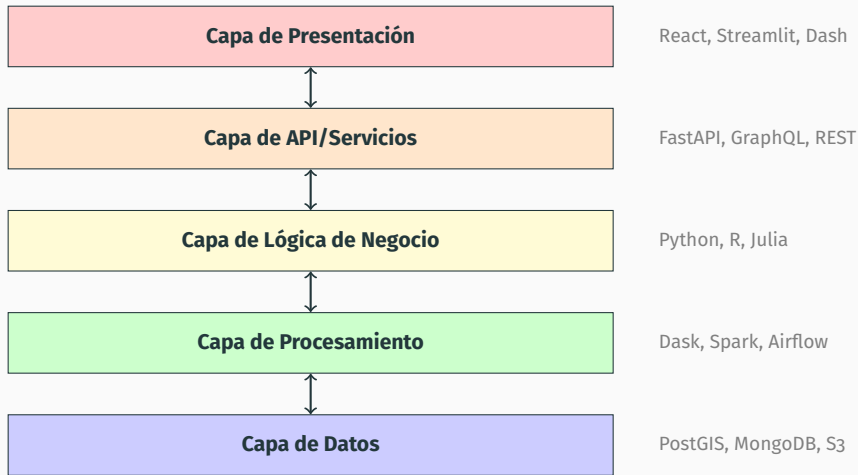
- APIs específicas por dominio
- No forzar dependencias innecesarias

D - Dependency Inversion

- Dependier de abstracciones
- Inyección de dependencias
- Configuración externa

 Aplicar SOLID reduce acoplamiento y mejora mantenibilidad

Arquitectura en Capas



Principio de Separación de Responsabilidades

Patrones de Diseño para Pipelines

≡ ETL/ELT

- Extract
- Transform/Load
- Load/Transform

Ideal para: Batch processing

⚡ Event-Driven

- Triggers
- Webhooks
- Message Queues

Ideal para: Real-time

🔗 Microservicios

- Servicios independientes
- API Gateway
- Service Mesh





Ideal para: Escalabilidad

Regla de oro: Elige el patrón según tu caso de uso, no por moda tecnológica




Adquisición de Datos Geoespaciales

Fuentes de Datos Geoespaciales

Fuentes Abiertas:

-  OpenStreetMap
-  Sentinel Hub
-  Datos gubernamentales
-  APIs meteorológicas

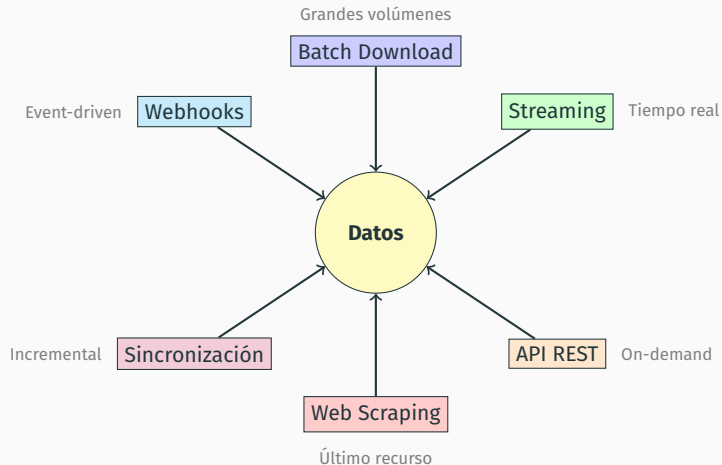
Fuentes Comerciales:

- Google Maps API
-  Mapbox
-  HERE Technologies
-  Planet Labs

Consideraciones Clave

- **Licencias:** Revisa restricciones
- **Calidad:** Valida precisión
- **Actualización:** Frecuencia de cambios
- **Cobertura:** Área geográfica
- **Formato:** Vector vs Raster

Estrategias de Adquisición



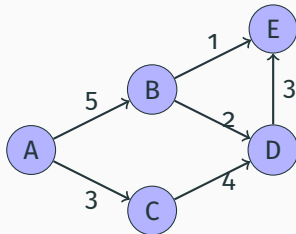
Teoría de Grafos en Redes Viales

Modelado como Grafo

- **Nodos:** Intersecciones
- **Aristas:** Segmentos de calle
- **Pesos:** Distancia, tiempo, costo
- **Dirección:** Sentido del tráfico

Métricas de Red:

- **Centralidad:** Importancia del nodo
- **Conectividad:** Robustez de la red
- **Clustering:** Agrupación local
- **Shortest Path:** Ruta óptima



Grafo dirigido con pesos

Algoritmos Clave

Dijkstra, A*, Bellman-Ford, Floyd-Warshall

¿Qué es OSMnx?


Framework que abstrae la complejidad de trabajar con datos de OpenStreetMap para análisis de redes urbanas

Problemas que resuelve:

- Descarga eficiente de datos OSM
- Limpieza de topología
- Simplificación de intersecciones
- Proyección automática
- Cálculo de métricas

Flujo conceptual:

1. **Definir** área de estudio
2. **Filtrar** tipo de red
3. **Construir** grafo topológico
4. **Analizar** propiedades
5. **Visualizar** resultados

 OSMnx maneja automáticamente la complejidad geométrica y topológica






Almacenamiento y Gestión de Datos

PostGIS: El Corazón del Pipeline

¿Qué es PostGIS?


Extensión espacial de PostgreSQL que añade soporte para objetos geográficos, permitiendo consultas espaciales SQL.

Ventajas clave:

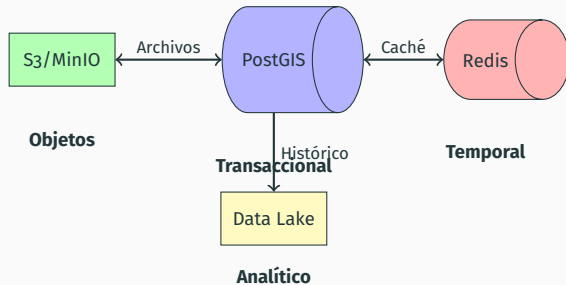
-  ACID compliance
-  Índices espaciales (R-tree)
-  +3000 funciones espaciales
-  Seguridad empresarial
-  Multi-usuario

Operaciones fundamentales:

- **ST_Contains:** ¿A contiene B?
- **ST_Distance:** Distancia entre objetos
- **ST_Buffer:** Área de influencia
- **ST_Intersection:** Intersección
- **ST_Union:** Unión de geometrías
- **ST_DWithin:** Objetos cercanos

 PostGIS es el estándar de facto para bases de datos espaciales

Arquitectura de Almacenamiento



Estrategia Híbrida

Combina diferentes tecnologías según el tipo de dato y patrón de acceso

Procesamiento y Análisis

Paradigmas de Procesamiento Geoespacial

Análisis histórico

Tracking GPS

Alertas tráfico

Batch Grandes volúmenes Latencia alta

Stream Datos continuos Baja latencia

Micro-batch Híbrido Balance

Apache Spark PostGIS bulk ops

Kafka Streams Apache Flink

Spark Streaming Storm Trident

Criterio de Selección

Volumen × Velocidad × Variedad = Paradigma adecuado

Técnicas de Procesamiento Espacial

Geocoding

- Dirección → Coordenadas
- Nominatim
- Google Geocoding API
- Pelias

Caso de uso: Localizar clientes

Clustering Espacial


- DBSCAN
- K-means espacial
- Hierarchical clustering
- OPTICS

Caso de uso: Zonas comerciales

Análisis de Redes

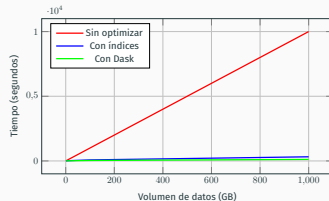
- Shortest path
- Isócronas
- Service areas
- Network flow

Caso de uso: Rutas óptimas

 **Tip:** Siempre valida tus resultados con visualización

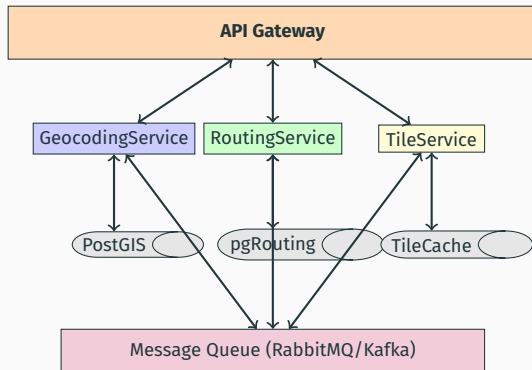
Estrategias de Optimización:

- **Índices espaciales:** R-tree, Quadtree
- **Particionamiento:** Por región, tiempo
- **Materialización:** Vistas precalculadas
- **Simplificación:** Douglas-Peucker
- **Paralelización:** Dask, Ray



APIs y Servicios

Arquitectura de Microservicios Geoespaciales



Ventajas

Escalabilidad independiente • Despliegue granular • Tolerancia a fallos • Tecnología heterogénea

Principios REST para Geo-APIs

- **Recursos claros:** /api/v1/places, /api/v1/routes
- **Filtros espaciales:** ?within=polygon&distance=1000
- **Formatos estándar:** GeoJSON, WKT, KML
- **Paginación:** Especialmente importante con geometrías

Endpoints típicos:

- GET /pois?type=hospital
- POST /geocode
- GET /route?from=A&to=B
- GET /isochrone?point=x,y
- POST /spatial-query

Consideraciones:

- Rate limiting
- Caché de resultados
- Compresión gzip
- CORS headers
- API keys/OAuth

Open Geospatial Consortium


Organización que desarrolla estándares abiertos para datos y servicios geoespaciales

Servicios Web OGC:

- **WMS:** Web Map Service
- **WFS:** Web Feature Service
- **WCS:** Web Coverage Service
- **WPS:** Web Processing Service
- **CSW:** Catalog Service

Formatos OGC:

- **GML:** Geography Markup Language
- **KML:** Keyhole Markup Language
- **GeoPackage:** SQLite espacial
- **CityGML:** Modelos 3D urbanos

 Usar estándares OGC garantiza interoperabilidad entre sistemas

Visualización y Dashboards

Estrategias de Visualización

Mapas Estáticos

Matplotlib, QGIS

Mapas Interactivos

Folium, Leaflet

Dashboards

Streamlit, Dash

Visualización 3D

Cesium, Deck.gl

Realidad Aumentada

ARCore, ARKit

Tiempo Real

WebSockets, SSE

Principio de Visualización

La mejor visualización es aquella que comunica la información de manera clara y permite tomar decisiones informadas

Mejores Prácticas de Visualización

Do's: ✓

- Usa proyecciones apropiadas
- Incluye leyendas claras
- Optimiza para el dispositivo
- Permite interacción
- Muestra contexto
- Usa colores accesibles

Don'ts: ✗

- Sobrecargar con información
- Ignorar la escala
- Usar proyecciones incorrectas
- Olvidar la fuente de datos
- Abusar de efectos 3D
- Ignorar el rendimiento

¡ Regla 5-segundo: El usuario debe entender el mapa en 5 segundos

Deployment y DevOps

Estrategias de Escalamiento

Escalamiento Vertical:

- ↑ Más CPU/RAM
- 🖥️ Servidor más potente
- 💰 Costo exponencial
- ⚠️ Límite físico

Cuándo usar:

- PostGIS principal
- Cálculos complejos
- Datos correlacionados

Escalamiento Horizontal:

- ↔ Más nodos
- 📄 Replicación
- 📈 Costo lineal
- ∞ Sin límite teórico

Cuándo usar:

- APIs stateless
- Procesamiento paralelo
- Cache distribuido

💡 **Patrón híbrido:** Escala vertical para BD, horizontal para servicios

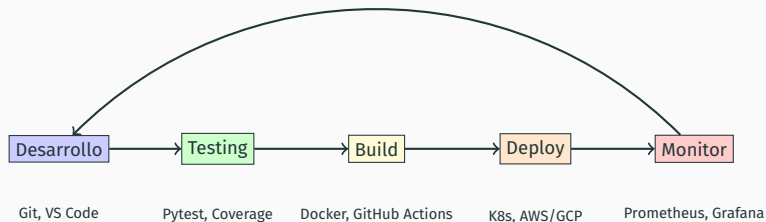
¿Por qué Docker para Geo?

- **Reproducibilidad:** Mismo ambiente en desarrollo y producción
- **Dependencias complejas:** GDAL, GEOS, PROJ fácilmente instaladas
- **Escalabilidad:** Kubernetes para orquestación
- **Aislamiento:** Cada servicio en su contenedor

Stack típico con Docker Compose:

- PostGIS database
- Redis cache
- API backend (FastAPI)
- Frontend (React/Vue)
- Nginx proxy
- GeoServer
- Jupyter notebooks
- pgAdmin
- Grafana monitoring
- Elasticsearch logs

CI/CD para Pipelines Geoespaciales

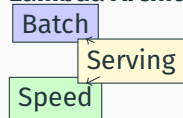


⚠ Automatiza todo lo que puedas, especialmente validación de datos espaciales

Casos de Uso Reales

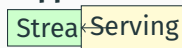
Patrones Arquitectónicos Comunes

Lambda Architecture



Uso: Análisis histórico + real-time

Kappa Architecture



Uso: Solo streaming, reprocesar si necesario

Event Sourcing



Uso: Auditoría completa, time-travel

⚠ Elige el patrón según requisitos de latencia y consistencia

Caso 1: Sistema de Routing Urbano

Problema:

- Optimizar rutas de delivery
- 1000+ pedidos diarios
- Ventanas de tiempo
- Tráfico en tiempo real

Solución:

- OSM para red vial
- pgRouting para cálculo
- Redis para caché
- WebSocket para actualizaciones

Pipeline implementado:

1. Geocoding de direcciones
2. Clustering por zonas
3. Cálculo de rutas óptimas
4. Asignación a conductores
5. Tracking en tiempo real
6. Analytics post-delivery

Resultado

30 % reducción en tiempo de entrega

Caso 2: Análisis de Mercado Inmobiliario

Objetivo: Predecir precios de propiedades basado en ubicación y amenidades cercanas

Datos utilizados:

- Propiedades históricas
- POIs (colegios, metro, parques)
- Demografía por zona
- Calidad del aire
- Ruido ambiental

Técnicas aplicadas:

- Buffer analysis (500m, 1km)
- Spatial join con amenidades
- Kriging para interpolación
- Random Forest con features espaciales
- Validación cruzada espacial

Precisión

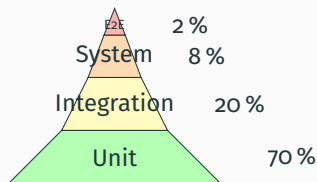
$R^2 = 0.87$ en predicción de precios

Testing y Calidad

Estrategias de Testing para Pipelines Geoespaciales

Niveles de Testing:

- **Unit Tests:** Funciones individuales
- **Integration:** Componentes conectados
- **System:** Pipeline completo
- **Acceptance:** Requisitos de negocio



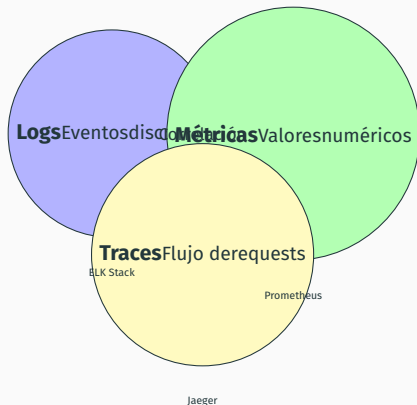
Tests Específicos Geo:

- Validación de geometrías
- Proyecciones correctas
- Topología consistente
- Precisión espacial

Cobertura Objetivo

Mínimo 80 % para código crítico

Monitoreo y Observabilidad



Métricas Clave para Geo-Pipelines

- **Latencia:** Tiempo de procesamiento por geometría

- **Throughput:** Features procesadas/segundo

Mejores Prácticas y Antipatrones

Mejores Prácticas

✓ Datos:

- Validar geometrías siempre
- Mantener CRS consistente
- Documentar fuentes
- Versionar cambios
- Implementar data lineage

✓ Procesamiento:

- Usar índices espaciales
- Cachear resultados costosos
- Paralelizar cuando sea posible
- Monitorear performance
- Implementar circuit breakers

✓ Arquitectura:

- Separar responsabilidades
- Usar colas para async
- Implementar retry logic
- Logs estructurados
- Health checks

✓ Seguridad:

- Sanitizar inputs espaciales
- Rate limiting en APIs
- Encriptar datos sensibles
- Auditar accesos
- Backup regular

Antipatrones Comunes

✘ No hagas esto:

- Ignorar proyecciones
- Procesar todo en memoria
- Hardcodear coordenadas
- Olvidar validación
- Mezclar CRS
- SQL injection con WKT

⚠ Consecuencias:

- Resultados incorrectos
- Out of memory
- Código no portable
- Datos corruptos
- Cálculos erróneos
- Vulnerabilidades

⚠ El 80 % de los errores en geoinformática se deben a problemas con proyecciones y validación de datos

Desafíos y Soluciones

Desafíos Comunes en Pipelines Geoespaciales

⚠ Desafío 1: Volumen de Datos

- Terabytes de imágenes satelitales
- Millones de puntos GPS

Solución: Particionamiento + Procesamiento paralelo

⚠ Desafío 2: Heterogeneidad

- Múltiples formatos
- Diferentes CRS

Solución: ETL robusto + Estandarización

⚠ Desafío 3: Tiempo Real

- Actualizaciones constantes
- Baja latencia requerida

Solución: Stream processing + Caché

⚠ Desafío 4: Calidad de Datos

- Geometrías inválidas
- Datos faltantes

Solución: Validación + Limpieza automática

⚠ El 60 % del tiempo en un proyecto geo se invierte en limpieza de datos

Matriz de Decisión Tecnológica

Criterio	PostGIS	MongoDB	Elasticsearch	BigQuery
Consultas espaciales	✓	●	✗	✓
Escalabilidad	●	✓	✓	✓
ACID	✓	●	✗	●
Costo	✓	✓	●	✗
Tiempo real	●	✓	✓	●
Análisis complejo	✓	✗	●	✓

Recomendación

No existe una solución única. Combina tecnologías según tus necesidades específicas.

Herramientas y Recursos

Stack Tecnológico Recomendado

Backend:

- Python 3.9+
- GeoPandas
- Shapely
- Rasterio
- PostGIS
- FastAPI

Frontend:

- Leaflet/OpenLayers
- Mapbox GL JS
- Deck.gl
- Folium
- Streamlit
- Dash





DevOps:

- Docker
- GitHub Actions
- Terraform
- Kubernetes
- Prometheus
- Grafana



Criterios de Selección

Elige herramientas basándote en: comunidad activa, documentación, licencia, y curva de aprendizaje

Documentación:

-  PostGIS in Action
-  geopandas.org
-  Awesome GIS
-  PyGIS tutorials
-  Coursera GIS

Comunidades:

-  OSGeo
-  [r/gis](https://www.reddit.com/r/gis)
-  GIS Stack Exchange
-  GeoPython

Datasets de práctica:

- Natural Earth Data
- OpenStreetMap extracts
- GADM boundaries
- NASA Earthdata
- Copernicus Open Access Hub

Herramientas online:

- geojson.io
- kepler.gl
- Overpass Turbo
- QGIS Cloud

Conclusiones

Resumen de la Clase

Aprendimos:

- ✓ Qué es un pipeline geoespacial
- ✓ Arquitectura en capas
- ✓ Fuentes de datos
- ✓ Almacenamiento con PostGIS
- ✓ Procesamiento escalable
- ✓ APIs y servicios
- ✓ Visualización efectiva
- ✓ Deployment con Docker

Conceptos clave:





- Automatización
- Reproducibilidad
- Escalabilidad
- Interoperabilidad
- Monitoreo
- Optimización
- Validación
- Documentación

Mensaje Final

Un buen pipeline geoespacial no es el que usa más tecnología, sino el que resuelve el problema de manera eficiente y mantenible

Próximos Pasos

Para consolidar lo aprendido:

1.  Completa los notebooks de práctica
2.  Experimenta con PostGIS localmente
3.  Construye tu primer pipeline completo
4.  Comparte tu proyecto con la comunidad

Proyecto sugerido:

Sistema de Análisis Urbano

Construye un pipeline que:

- Descargue datos de OSM de tu ciudad
- Identifique zonas comerciales (clustering)
- Calcule accesibilidad a transporte público

Genere un dashboard interactivo

¿Preguntas?

✉ fparra@usach.cl

🐙 github.com/fparrao

Material disponible en el repositorio del curso