UNIVERSIDAD DE SANTIAGO DE CHILE
Facultad de Ingeniería
Departamento de Ingeniería Informática

Laboratorio Integrador
Análisis Geoespacial Completo de una Comuna Chilena

Curso: Desarrollo de Aplicaciones Geoinformáticas

Prof. Francisco Parra O.
francisco.parra.o@usach.cl

Fecha de entrega: 3 semanas desde la publicación

Índice

1. Introducción y Objetivos

1.1. Contexto

Este laboratorio integrador representa la culminación de las primeras 7 semanas del curso, donde aplicarán todos los conocimientos adquiridos en un proyecto geoespacial completo y realista. Trabajarán con una comuna chilena de su elección, desarrollando un análisis integral que combine tecnologías, métodos y herramientas aprendidas.

1.2. Objetivos de Aprendizaje

Al completar este laboratorio, serán capaces de:

1. **Integrar múltiples fuentes de datos geoespaciales** (vectoriales, raster, satelitales)
2. **Implementar un pipeline completo** desde la adquisición hasta la visualización
3. **Aplicar técnicas de análisis espacial avanzado** incluyendo geoestadística y ML
4. **Desarrollar una aplicación web interactiva** para presentar resultados
5. **Trabajar colaborativamente** usando control de versiones y buenas prácticas
6. **Documentar técnicamente** un proyecto geoespacial complejo

1.3. Modalidad de Trabajo

Trabajo en Parejas

- Formar grupos de **exactamente 2 personas**
- Ambos integrantes deben contribuir equitativamente (se revisará Git)
- División clara de responsabilidades pero integración conjunta
- Presentación oral conjunta del trabajo

2. Descripción del Proyecto

2.1. Visión General

Desarrollarán un **Sistema de Análisis Territorial Integral** para una comuna chilena, que incluya:

1. **Caracterización territorial completa** usando datos oficiales y satelitales
2. **Análisis de patrones espaciales** de variables socioeconómicas y ambientales
3. **Modelo predictivo** usando machine learning geoespacial
4. **Aplicación web interactiva** para exploración de resultados
5. **Documentación y reproducibilidad** completa del análisis

2.2. Selección de la Comuna

Criterios para elegir su comuna:

- **Disponibilidad de datos**: Verificar acceso a datos INE, municipales, etc.
- **Diversidad territorial**: Preferir comunas con variedad urbana/rural
- **Problemática interesante**: Identificar un desafío territorial real
- **Tamaño manejable**: Evitar comunas extremadamente grandes (ej: Putre) o densas (ej: Santiago Centro) para su primer análisis

Comunas sugeridas (pero no obligatorias):

- Región Metropolitana: La Florida, Maipú, Puente Alto, Quilicura
- Valparaíso: Viña del Mar, Quilpué, Villa Alemana
- Biobío: Talcahuano, Chiguayante, San Pedro de la Paz
- La Araucanía: Temuco, Padre Las Casas, Villarrica

3. Componentes Técnicos Requeridos

3.1. Parte 1: Preparación del Entorno (10 %)

Entregable 1: Ambiente de Desarrollo

- **Docker Compose** configurado con todos los servicios
- **PostGIS** con extensiones espaciales activadas
- **Jupyter Lab** con kernel geoespacial
- **Scripts de inicialización** automatizados
- **Documentación** de instalación paso a paso

3.1.1. Configuración Docker

Deben crear un `docker-compose.yml` que incluya:

```
version: '3.8'

services:
  postgis:
    image: postgis/postgis:15-3.3
    environment:
      POSTGRES_DB: geodatabase
      POSTGRES_USER: geouser
      POSTGRES_PASSWORD: geopass
    volumes:
```

```
      - postgres_data:/var/lib/postgresql/data
      - ./scripts/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "5432:5432"

  jupyter:
    build: ./docker/jupyter
    volumes:
      - ./notebooks:/home/jovyan/work
      - ./data:/home/jovyan/data
    ports:
      - "8888:8888"
    environment:
      - JUPYTER_ENABLE_LAB=yes
    depends_on:
      - postgis

  webserver:
    build: ./docker/web
    volumes:
      - ./app:/app
    ports:
      - "5000:5000"
    depends_on:
      - postgis

volumes:
  postgres_data:
```

Listing 1: Estructura Docker Compose

3.2. Parte 2: Adquisición y Procesamiento de Datos (20 %)

Entregable 2: Dataset Integrado

- **Datos vectoriales**: Límites, manzanas censales, infraestructura
- **Datos raster**: DEM, imágenes satelitales (Sentinel-2/Landsat)
- **Datos tabulares**: Censo, socioeconómicos, ambientales
- **Red vial**: Desde OpenStreetMap usando OSMnx
- **Base de datos espacial**: Todo cargado en PostGIS

3.2.1. Fuentes de Datos Requeridas

Tipo de Dato	Fuente	Uso en el Proyecto
Límites administrativos	IDE Chile	Base cartográfica
Manzanas censales	INE	Unidad de análisis
DEM	ALOS PALSAR / SRTM	Análisis topográfico
Sentinel-2	Copernicus / GEE	Índices vegetacionales
Red vial	OpenStreetMap	Análisis de accesibilidad
Censo 2017	INE	Variables socioeconómicas
Uso del suelo	IDE Minvu	Planificación territorial

Tabla 1: Fuentes de datos mínimas requeridas

3.2.2. Script de Descarga Automatizada

Crear `scripts/download_data.py`:

```python
import os
import requests
import geopandas as gpd
import osmnx as ox
import ee
from pathlib import Path

class DataDownloader:
    def __init__(self, comuna_name, output_dir='../data'):
        self.comuna = comuna_name
        self.output_dir = Path(output_dir)
        self.output_dir.mkdir(exist_ok=True)

    def download_administrative_boundaries(self):
        """Descarga l mites desde IDE Chile"""
        # Implementar descarga desde WFS
        pass

    def download_osm_network(self):
        """Descarga red vial desde OpenStreetMap"""
        G = ox.graph_from_place(f"{self.comuna}, Chile",
                                network_type='all')
        ox.save_graph_geopackage(G,
                                 filepath=self.output_dir / '
                                    red_vial.gpkg')

    def download_sentinel2(self, start_date, end_date):
        """Descarga im genes Sentinel-2 desde Google Earth
           Engine"""
        ee.Initialize()
        # Implementar descarga GEE
        pass

```

```
    def download_dem(self):
        """Descarga DEM de ALOS PALSAR"""
        # Implementar descarga
        pass
```

3.3. Parte 3: Análisis Espacial Exploratorio (20 %)

Entregable 3: ESDA Completo

- **Estadísticas descriptivas espaciales** de todas las variables
- **Mapas temáticos** profesionales (mínimo 10)
- **Análisis de autocorrelación** (Moran's I global y local)
- **Hot spots y clusters** usando LISA
- **Análisis multivariado** de componentes principales espaciales

3.3.1. Notebook de Análisis Exploratorio

Crear `notebooks/01_exploratory_analysis.ipynb`:

```
# An lisis de Autocorrelaci n Espacial
import pysal
from pysal.explore import esda
import splot

# Crear matriz de pesos espaciales
w = pysal.lib.weights.Queen.from_dataframe(gdf)
w.transform = 'r'  # Row standardization

# Moran's I Global
mi = esda.Moran(gdf['variable'], w)
print(f"Moran's I: {mi.I:.4f}")
print(f"P-value: {mi.p_norm:.4f}")

# LISA - Local Moran
lisa = esda.Moran_Local(gdf['variable'], w)

# Visualizaci n
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# Moran Scatterplot
splot.esda.moran_scatterplot(mi, ax=axes[0])

# LISA Cluster Map
splot.esda.lisa_cluster(lisa, gdf, ax=axes[1])
```

3.4. Parte 4: Geoestadística y Análisis Avanzado (15 %)

Entregable 4: Análisis Geoestadístico

- **Semivariogramas** de variables continuas principales
- **Interpolación espacial** (Kriging vs IDW comparación)
- **Superficies de predicción** con medidas de incertidumbre
- **Validación cruzada** de modelos de interpolación
- **Análisis de anisotropía** si aplica

3.4.1. Análisis de Semivariogramas

```python
import skgstat as skg
from pykrige.ordinary_kriging import OrdinaryKriging

# Calcular semivariograma experimental
coords = np.column_stack([gdf.geometry.x, gdf.geometry.y])
values = gdf['variable'].values

variogram = skg.Variogram(coords, values,
                          model='exponential',
                          lag_classes=15,
                          maxlag=0.3)

# Ajustar modelo
variogram.fit()

# Par metros del modelo
nugget = variogram.nugget
sill = variogram.sill
range_ = variogram.range

# Kriging ordinario
ok = OrdinaryKriging(coords[:, 0], coords[:, 1], values,
                     variogram_model='exponential',
                     variogram_parameters={'nugget': nugget,
                                           'sill': sill,
                                           'range': range_})

# Crear grid de predicci n
grid_x = np.linspace(coords[:, 0].min(), coords[:, 0].max(), 100)
grid_y = np.linspace(coords[:, 1].min(), coords[:, 1].max(), 100)
z_pred, var_pred = ok.execute('grid', grid_x, grid_y)
```

3.5. Parte 5: Machine Learning Geoespacial (20 %)

Entregable 5: Modelo Predictivo

- **Definición clara del problema** a resolver con ML
- **Feature engineering espacial** completo
- **Comparación de algoritmos** (RF, XGBoost, SVM espacial)
- **Validación espacial** apropiada (no random split!)
- **Mapas de predicción** y medidas de incertidumbre
- **Interpretación del modelo** (SHAP values, feature importance)

3.5.1. Ejemplo: Predicción de Valores de Suelo

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GroupKFold
import shap

# Feature Engineering Espacial
def create_spatial_features(gdf):
    features = pd.DataFrame()

    # Coordenadas
    features['x'] = gdf.geometry.x
    features['y'] = gdf.geometry.y

    # Distancias a puntos de inter s
    features['dist_centro'] = gdf.geometry.distance(centro_point)
    features['dist_metro'] = gdf.geometry.apply(
        lambda x: metro_stations.distance(x).min()
    )

    # Densidades en buffer
    for radius in [500, 1000, 2000]:
        buffer = gdf.geometry.buffer(radius)
        features[f'density_{radius}m'] = buffer.apply(
            lambda x: gdf[gdf.within(x)].shape[0]
        )

    #  ndices  de vegetaci n desde Sentinel-2
    features['ndvi_mean'] = extract_zonal_stats(gdf, ndvi_raster,
        'mean')

    # Variables topogr ficas
    features['elevation'] = extract_zonal_stats(gdf, dem, 'mean')
    features['slope'] = extract_zonal_stats(gdf, slope_raster, '
       mean')

```

```
    return features

# Validaci n Espacial
spatial_cv = GroupKFold(n_splits=5)
groups = gdf['zona_id']  # Agrupar por zonas geogr ficas

# Entrenamiento
rf_model = RandomForestRegressor(n_estimators=200,
                                 max_depth=10,
                                 min_samples_leaf=5)

scores = cross_val_score(rf_model, X, y,
                         cv=spatial_cv,
                         groups=groups,
                         scoring='r2')

print(f"R2 Score (Spatial CV): {scores.mean():.3f} (+/- {scores.
    std():.3f})")

# Interpretaci n con SHAP
explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test)
```

3.6. Parte 6: Aplicación Web Interactiva (15 %)

Entregable 6: Dashboard Web

- **Mapa interactivo** con capas temáticas
- **Gráficos dinámicos** de estadísticas espaciales
- **Panel de control** para modelos predictivos
- **Descarga de resultados** en formatos estándar
- **Documentación de usuario** incluida

3.6.1. Estructura de la Aplicación Streamlit

Crear `app/main.py`:

```
import streamlit as st
import folium
from streamlit_folium import folium_static
import plotly.express as px

st.set_page_config(page_title="An lisis Territorial Comuna",
                   layout="wide")

# Sidebar para navegaci n
st.sidebar.title("Panel de Control")
```

```
page = st.sidebar.selectbox("Seleccione una secci n:",
                            ["Inicio", "Datos", "An lisis
                               Espacial",
                             "Modelos ML", "Resultados"])

if page == "Inicio":
    st.title(f"Sistema de An lisis Territorial - {COMUNA_NAME}")
    st.markdown("""
    ## Bienvenido al Dashboard de An lisis Geoespacial

    Este sistema integra m ltiples fuentes de datos y t cnicas
    de an lisis para proporcionar insights territoriales.
    """)

    # Mapa general
    m = folium.Map(location=[lat_center, lon_center], zoom_start
       =12)

    # Agregar capas
    folium.GeoJson(comuna_boundary).add_to(m)

    # Agregar controles
    folium.LayerControl().add_to(m)

    folium_static(m)

elif page == "An lisis Espacial":
    st.header("An lisis de Autocorrelaci n Espacial")

    col1, col2 = st.columns(2)

    with col1:
        st.subheader("Moran's I Global")
        # Mostrar estad stico y p-value
        st.metric(" ndice de Moran", f"{moran_i:.4f}")
        st.metric("P-value", f"{p_value:.4f}")

    with col2:
        st.subheader("Distribuci n LISA")
        # Gr fico de distribuci n de clusters
        fig = px.pie(values=lisa_counts.values(),
                     names=lisa_counts.keys(),
                     title="Tipos de Clusters LISA")
        st.plotly_chart(fig)

elif page == "Modelos ML":
    st.header("Predicciones de Machine Learning")

    # Selector de modelo
    model_type = st.selectbox("Seleccione modelo:",
                              ["Random Forest", "XGBoost", "
```

```
                                    Neural Network"])

    # Par metros interactivos
    if st.button("Ejecutar Predicci n"):
        with st.spinner("Calculando..."):
            predictions = run_model(model_type, parameters)

        # Mostrar resultados
        st.success("Predicci n completada!")

        # Mapa de predicciones
        fig = px.choropleth_mapbox(gdf,
                                   geojson=gdf.geometry,
                                   locations=gdf.index,
                                   color='prediction',
                                   mapbox_style="open-street-map"
                                      ,
                                   zoom=11,
                                   center={"lat": lat_center,
                                           "lon": lon_center})
        st.plotly_chart(fig)
```

4. Estructura del Proyecto

4.1. Organización de Archivos

Es **obligatorio** seguir esta estructura de carpetas para facilitar la evaluación:

```
laboratorio_integrador/
          README.md                    # Documentaci n principal
          requirements.txt             # Dependencias Python
          docker-compose.yml          # Configuraci n Docker
          .env                        # Variables de entorno (no
   subir!)
          .gitignore                  # Archivos a ignorar en Git

          docker/                     # Configuraciones Docker
                jupyter/
                      Dockerfile
                postgis/
                      init.sql
                web/
                    Dockerfile

          data/                       # Datos (incluir sample
   data)
                raw/                   # Datos originales
                processed/             # Datos procesados
```

```
                README.md                # Descripci n de los
  datos

        notebooks/                       # An lisis en Jupyter
              01_data_acquisition.ipynb
              02_exploratory_analysis.ipynb
              03_geostatistics.ipynb
              04_machine_learning.ipynb
              05_results_synthesis.ipynb

        scripts/                         # Scripts Python
              download_data.py
              process_data.py
              spatial_analysis.py
              utils.py

        app/                             # Aplicaci n web
              main.py
              pages/
              components/
              static/

        outputs/                         # Resultados
              figures/                    # Gr ficos y mapas
              models/                     # Modelos entrenados
              reports/                    # Informes generados

        docs/                            # Documentaci n
            guia_usuario.md
            arquitectura.md
            api_reference.md
```

Listing 2: Estructura de carpetas requerida

5. Criterios de Evaluación

5.1. Rúbrica de Evaluación

Componente	Peso	Criterios
Configuración del entorno	10 %	[leftmargin=*,topsep=0pt,itemsep=0pt]Docker funcional (3 %) PostGIS configurado (3 %) Jupyter con librerías (2 %) Documentación clara (2 %)
Adquisición de datos	20 %	[leftmargin=*,topsep=0pt,itemsep=0pt]Variedad de fuentes (5 %) Calidad del procesamiento (5 %) Integración en PostGIS (5 %) Automatización (5 %)
Análisis espacial	20 %	[leftmargin=*,topsep=0pt,itemsep=0pt]ESDA completo (5 %) Autocorrelación espacial (5 %) Visualizaciones (5 %) Interpretación (5 %)
Geoestadística	15 %	[leftmargin=*,topsep=0pt,itemsep=0pt]Semivariogran (5 %) Interpolación (5 %) Validación (5 %)
Machine Learning	20 %	[leftmargin=*,topsep=0pt,itemsep=0pt]Feature engineering (5 %) Modelos apropiados (5 %) Validación espacial (5 %) Interpretabilidad (5 %)
Aplicación web	15 %	[leftmargin=*,topsep=0pt,itemsep=0pt]Funcionalidad (5 %) Interfaz (5 %) Interactividad (5 %)

Tabla 2: Distribución de puntajes por componente

5.2. Criterios de Excelencia

Para optar a nota máxima (7.0), además de cumplir todos los requisitos, deben incluir **al menos 3** de los siguientes elementos:

1. **Deep Learning**: Implementar CNN para clasificación de imágenes satelitales
2. **Series temporales**: Análisis de cambios usando múltiples fechas de imágenes
3. **Optimización espacial**: Problema de localización óptima resuelto

4. **API REST**: Endpoints para acceder a los modelos y datos
5. **Visualización 3D**: Incorporar visualizaciones 3D del terreno
6. **Análisis de redes**: Análisis avanzado de la red vial (centralidad, accesibilidad)
7. **Validación externa**: Comparar con datos de terreno o fuentes independientes

6. Entregables y Plazos

6.1. Hitos del Proyecto

Semana	Hito	Entregable
1	Formación y Setup	[leftmargin=*,topsep=0pt,itemsep=0pt]Grupos formados Comuna seleccionada Ambiente Docker funcionando Repositorio Git creado
2	Datos y Análisis	[leftmargin=*,topsep=0pt,itemsep=0pt]Todos los datos descargados ESDA completado Primeros modelos ML
3	Finalización	[leftmargin=*,topsep=0pt,itemsep=0pt]Aplicación web funcional Documentación completa Video de presentación (5 min) Código en repositorio

6.2. Formato de Entrega

Entrega vía Moodle antes de las 23:59 del día límite:

- Link al repositorio GitHub (público o con acceso al profesor)
- ZIP con snapshot del código (backup)
- Link al video de YouTube (no listado)
- Informe PDF de máximo 10 páginas

6.3. Presentación del Proyecto

Video de presentación (5 minutos):

1. Introducción y problemática (30 seg)
2. Demo del ambiente y datos (1 min)
3. Resultados del análisis espacial (1 min)
4. Modelos de ML y predicciones (1 min)
5. Demo de la aplicación web (1 min)
6. Conclusiones y aprendizajes (30 seg)

7. Recursos y Soporte

7.1. Recursos Recomendados

7.1.1. Documentación Técnica

- GeoPandas: `https://geopandas.org`
- PySAL: `https://pysal.org`
- OSMnx: `https://osmnx.readthedocs.io`
- Rasterio: `https://rasterio.readthedocs.io`
- Streamlit: `https://docs.streamlit.io`

7.1.2. Fuentes de Datos

- IDE Chile: `https://www.ide.cl`
- INE: `https://www.ine.cl`
- Google Earth Engine: `https://earthengine.google.com`
- OpenStreetMap: `https://www.openstreetmap.org`
- Copernicus Hub: `https://scihub.copernicus.eu`

7.2. Soporte y Consultas

Canales de Comunicación

- **Horario de consultas**: Martes y Jueves 15:00-17:00
- **Foro Moodle**: Para dudas generales
- **Email**: francisco.parra.o@usach.cl (solo urgencias)
- **Discord del curso**: Canal #laboratorio-integrador

8. Anexo: Código de Inicio Rápido

8.1. Script de Configuración Inicial

Crear archivo `setup.sh`:

```
#!/bin/bash
# Script de configuraci n inicial del proyecto

echo "====================================="
echo "Configuraci n Laboratorio Integrador"
echo "====================================="

# Crear estructura de directorios
mkdir -p data/{raw,processed}
mkdir -p notebooks
mkdir -p scripts
mkdir -p app/{pages,components,static}
mkdir -p outputs/{figures,models,reports}
mkdir -p docker/{jupyter,postgis,web}

# Crear archivo de ambiente
cat > .env << EOF
POSTGRES_DB=geodatabase
POSTGRES_USER=geouser
POSTGRES_PASSWORD=geopass
JUPYTER_TOKEN=your_token_here
COMUNA_NAME=your_comuna_here
EOF

# Crear requirements.txt
cat > requirements.txt << EOF
# Geospatial
geopandas==0.14.0
shapely==2.0.2
pyproj==3.6.1
rasterio==1.3.9
fiona==1.9.5
osmnx==1.7.1

```

```
# Data Science
pandas==2.1.3
numpy==1.24.3
scikit-learn==1.3.2
xgboost==2.0.2

# Spatial Analysis
pysal==2.9.3
esda==2.5.1
splot==1.1.5
scikit-gstat==1.0.15
pykrige==1.7.0

# Visualization
matplotlib==3.8.1
seaborn==0.13.0
plotly==5.18.0
folium==0.15.0
streamlit==1.28.2
streamlit-folium==0.15.0

# Database
psycopg2-binary==2.9.9
sqlalchemy==2.0.23
geoalchemy2==0.14.2

# Web
fastapi==0.104.1
uvicorn==0.24.0

# Utils
python-dotenv==1.0.0
tqdm==4.66.1
click==8.1.7
EOF

# Crear .gitignore
cat > .gitignore << EOF
# Python
__pycache__/
*.py[cod]
*$py.class
*.so
.Python
env/
venv/
.env

# Jupyter
.ipynb_checkpoints
*/.ipynb_checkpoints/*
```

```
# Data
data/raw/*
data/processed/*
*.tif
*.shp
*.gpkg
!data/raw/sample*
!data/processed/sample*

# Models
*.pkl
*.h5
*.pt

# OS
.DS_Store
Thumbs.db

# IDE
.vscode/
.idea/
*.swp
*.swo
EOF

echo "Configuraci n completada!"
echo "Siguiente paso: docker-compose up -d"
```

8.2. Notebook de Ejemplo

Crear `notebooks/00_template.ipynb`:

```
# Celda 1: Configuraci n inicial
import warnings
warnings.filterwarnings('ignore')

import sys
sys.path.append('../scripts')

from pathlib import Path
import pandas as pd
import geopandas as gpd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Configuraci n de visualizaci n
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")

```

```
# Paths
DATA_DIR = Path('../data')
RAW_DATA = DATA_DIR / 'raw'
PROCESSED_DATA = DATA_DIR / 'processed'
OUTPUT_DIR = Path('../outputs')

print(f"Ambiente configurado correctamente!")
print(f"Comuna de an lisis: {os.getenv('COMUNA_NAME')}")

# Celda 2: Conexi n a PostGIS
from sqlalchemy import create_engine
from geoalchemy2 import Geometry

# Crear conexi n
engine = create_engine(
    f"postgresql://geouser:geopass@postgis:5432/geodatabase"
)

# Test de conexi n
with engine.connect() as conn:
    result = conn.execute("SELECT PostGIS_Version();")
    print(f"PostGIS Version: {result.fetchone()[0]}")

# Celda 3: Funciones auxiliares
def load_geodata(table_name):
    """Carga datos geoespaciales desde PostGIS"""
    return gpd.read_postgis(
        f"SELECT * FROM {table_name}",
        engine,
        geom_col='geometry'
    )

def save_map(fig, name):
    """Guarda figuras en alta resoluci n"""
    fig.savefig(OUTPUT_DIR / 'figures' / f'{name}.png',
                dpi=300, bbox_inches='tight')
    print(f"Mapa guardado: {name}.png")

# Celda 4: Carga de datos inicial
comuna_boundary = load_geodata('comuna_boundary')
print(f" rea de la comuna: {comuna_boundary.area[0] / 1e6:.2f}
    km ")
print(f"Sistema de coordenadas: {comuna_boundary.crs}")
```

9. Conclusión

Este laboratorio integrador representa una oportunidad única para aplicar todo lo aprendido en un proyecto real y complejo. El éxito dependerá de:

1. **Planificación**: Dividir tareas y gestionar tiempo

2. **Colaboración**: Trabajo efectivo en equipo
3. **Documentación**: Código y procesos claros
4. **Creatividad**: Soluciones innovadoras a problemas reales
5. **Rigurosidad**: Métodos apropiados y validación correcta

¡Éxito en su proyecto!
Recuerden que este trabajo es una excelente pieza para su portafolio profesional. Háganlo con dedicación y será una carta de presentación valiosa en su carrera.