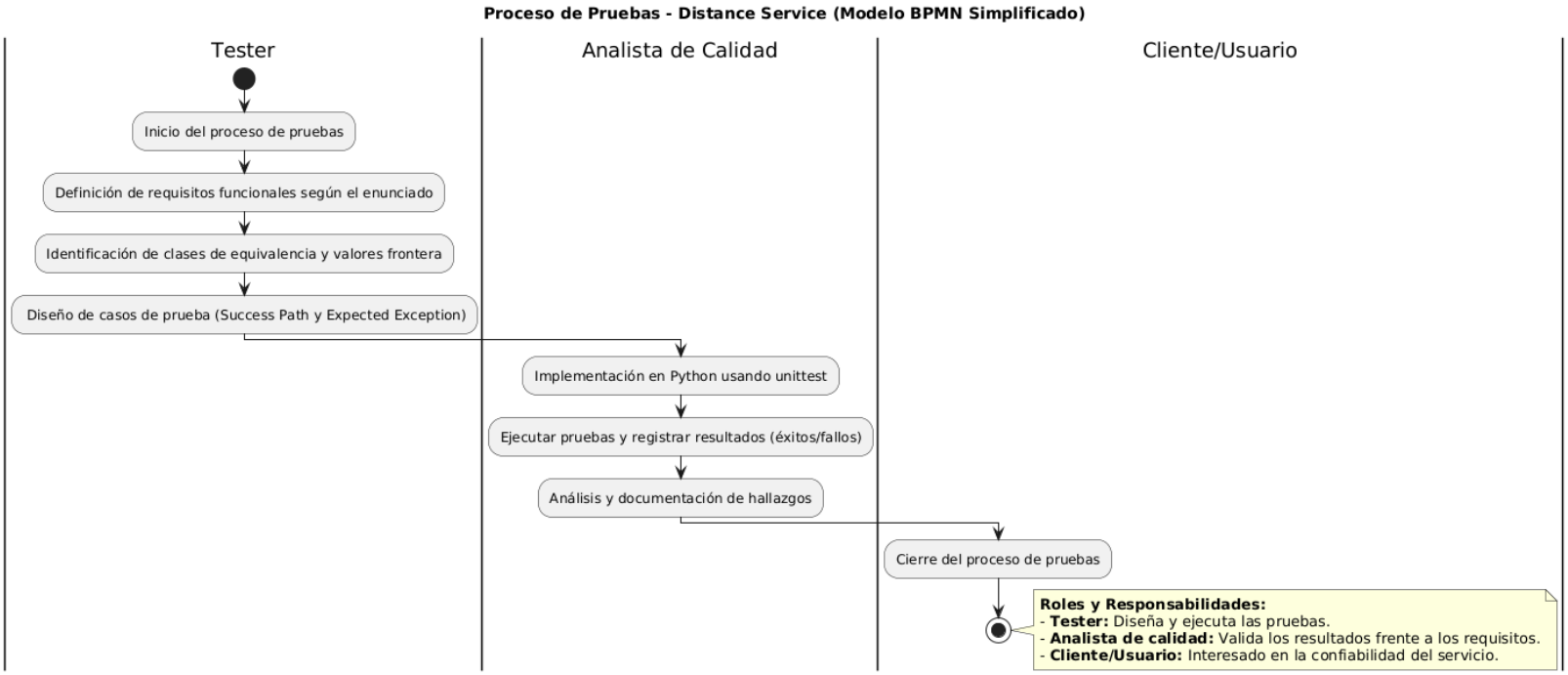
**Informe de Pruebas del Servicio *gRPC***

**Asignatura:** ICI 423 - Pruebas de Sistemas
**Evaluación:** Entrega I
**Estudiante:** Luca Carabelli Sánchez
**Fecha:** 13-10-2025

**Herramienta de IA utilizada:** ChatGPT (Ayuda en redacción e ideas en guía de pruebas)

**1. Proceso de Pruebas (ISO/IEC/IEEE 29119-2)**

El proceso seguido se basó en el estándar **ISO/IEC/IEEE 29119-2:2021**, que define las actividades fundamentales de las pruebas de software: planificación, diseño, implementación, ejecución, evaluación y cierre.




**Flujo resumido (modelo BPMN):**

1. **Inicio del proceso de pruebas.**
2. **Definición de requisitos funcionales** según el enunciado.
3. **Identificación de clases de equivalencia** y valores frontera.
4. **Diseño de casos de prueba (Success Path y Expected Exception).**
5. **Implementación en Python usando unittest.**
6. **Ejecución y registro de resultados (éxitos/fallos).**
7. **Análisis y documentación de hallazgos.**

8. **Cierre del proceso de pruebas.**

Roles definidos:

- **Tester:** responsable de diseñar y ejecutar las pruebas.
- **Analista de calidad:** encargado de validar los resultados frente a los requisitos.
- **Cliente/usuario:** interesado en la confiabilidad del servicio.

**2. Clases de Equivalencia y Valores Frontera**

| Requisito | Clases de equivalencia válidas | Clases inválidas | Valores frontera |
|---|---|---|---|
| Latitud | -90 ≤ lat ≤ 90 | lat < -90, lat > 90 | -90, 90 |
| Longitud | -180 ≤ lon ≤ 180 | lon < -180, lon > 180 | -180, 180 |
| Unidad | "km", "nm" | "", "random", "lightyears" | "km", "" |
| Posiciones | Ambas definidas | Una o ambas ausentes | n/a |

**Justificación:**

- Se usa **equivalencia** porque agrupa entradas que se comportan igual ante el sistema (ej. cualquier latitud > 90 genera error).
- Se usa **valores frontera** porque en coordenadas geográficas los límites son exactos y críticos.
- Se incluyeron pruebas con **datos malformados** (string, None) para evaluar robustez de entrada.

**3. Implementación y Framework**

Se utilizó el framework **unittest** de Python, aplicando los patrones:

- **Success Path:** casos donde se espera comportamiento correcto.
- **Expected Exception:** validación de excepciones ante entradas fuera de rango o malformadas.

El archivo de pruebas implementado fue:

📄 tests/test_distance_failures.py
Incluye 14 casos de prueba agrupados en 6 bloques:

1. Valores fuera de rango
2. Datos malformados
3. Entradas inválidas del servicio
4. Fallo de unidad vacía

5. Valores frontera
6. Entradas incompletas
7. Validación de millas náuticas

**4. Resultados de Ejecución**

**Resumen de salida:**

| Caso de prueba | Resultado | Observación |
|---|---|---|
| Latitud > 90 / < -90 | ❌ Falla | No lanza ValueError cuando < -90 |
| Longitud > 180 / < -180 | ❌ Falla | Falta validación para valores negativos fuera de rango |
| Latitud/Longitud válidas | ✅ OK | Correctamente instanciadas |
| Datos malformados (string, None) | ✅ OK | Lanza excepción |
| Unidad desconocida | ✅ OK | Lanza excepción |
| Unidad vacía (“”) | ❌ Falla | Calcula distancia en nm, pero reporta km |
| Falta origen o destino | ❌ Falla | No lanza error ni valida ausencia |
| Millas náuticas (“nm”) | ✅ OK | Conversión 1.852 km/nm correcta |
| Límites válidos (-90, 90, -180, 180) | ✅ OK | Correctamente procesados |

**5. Hallazgos Principales**

| Tipo de fallo | Descripción | Severidad | Recomendación |
|---|---|---|---|
| Validación incompleta de latitud y longitud | Solo valida valores positivos fuera de rango. | Alta | Agregar verificación if lat < -90 or lat > 90: y if lon < -180 or lon > 180: |
| Unidad vacía mal manejada | Calcula .nautical() pero reporta “km”. | Media | Cambiar .nautical() a .km() en el bloque if request.unit == "". |
| Entradas incompletas | Falta control para source o destination ausentes. | Media | Validar existencia de ambas posiciones antes del cálculo. |

| Tipo de fallo | Descripción | Severidad | Recomendación |
|---|---|---|---|
| **Mensajes de error inconsistentes** | **En algunos casos se devuelve resultado numérico en lugar de excepción.** | **Baja** | **Estandarizar el manejo de excepciones y mensajes.** |

**6. Prueba de Millas Náuticas**

La prueba test_service_nautical_miles_calculation validó correctamente el factor de conversión de 1 nm = 1.852 km:

**[INFO] Distancia esperada (nm): 27.314**

**[INFO] Distancia devuelta por el servicio: 27.31**

**Resultado: OK**

**7. Conclusión**

- Las pruebas cumplen con el estándar ISO/IEC/IEEE 29119-2, usando un proceso formal y documentado.
- Se aplicaron correctamente clases de equivalencia y valores frontera.
- El sistema presenta 4 fallos funcionales que deben ser corregidos para cumplir los requisitos.
- Se evidenció la robustez del cálculo de distancias en millas náuticas y la correcta validación de datos de tipo y formato.
- Los resultados fueron validados con geopy.distance.geodesic() como oráculo externo.

En síntesis: las pruebas son exitosas porque logran detectar los defectos esperados sin modificar el código fuente original, cumpliendo el objetivo pedagógico del ejercicio.

**Imagen de pruebas:**

MINGW64:/c/Users/yo/Desktop/Prueba sistemas

```
yo@DESKTOP-IDAQ548 MINGW64 ~/Desktop/Prueba sistemas (master)
$ python -m unittest tests.test_distance_service -v
test_latitude_above_maximum (tests.test_distance_service.TestDistanceFailureCases.test_latitude_above_maximum)
Latitud > 90 debera lanzar ValueError. ... ok
test_latitude_as_string (tests.test_distance_service.TestDistanceFailureCases.test_latitude_as_string)
Si latitud se pasa como string, debe lanzar TypeError o ValueError. ... ok
test_latitude_below_minimum (tests.test_distance_service.TestDistanceFailureCases.test_latitude_below_minimum)
Latitud < -90 debera lanzar ValueError (rango invlido). ... FAIL
test_latitude_exact_boundary (tests.test_distance_service.TestDistanceFailureCases.test_latitude_exact_boundary)
Latitudes exactas de frontera (-90, 90) deben ser vlidas. ... ok
test_longitude_above_maximum (tests.test_distance_service.TestDistanceFailureCases.test_longitude_above_maximum)
Longitud > 180 debera lanzar ValueError. ... ok
test_longitude_below_minimum (tests.test_distance_service.TestDistanceFailureCases.test_longitude_below_minimum)
Longitud < -180 debera lanzar ValueError. ... FAIL
test_longitude_exact_boundary (tests.test_distance_service.TestDistanceFailureCases.test_longitude_exact_boundary)
Longitudes exactas de frontera (-180, 180) deben ser vlidas. ... ok
test_longitude_none_value (tests.test_distance_service.TestDistanceFailureCases.test_longitude_none_value)
Si longitud es None, el sistema debe fallar. ... ok
test_missing_destination_position (tests.test_distance_service.TestDistanceFailureCases.test_missing_destination_position)
Falta el destino -> debera generar error. ... FAIL
test_missing_source_position (tests.test_distance_service.TestDistanceFailureCases.test_missing_source_position)
Falta el origen -> debera generar error. ... FAIL
test_service_empty_unit_should_return_kilometers (tests.test_distance_service.TestDistanceFailureCases.test_service_empty_unit_should_return_kilometers)
Si la unidad esperada est en blanco, la respuesta debe ser en kilmetros. ... FAIL
test_service_invalid_latitude_source (tests.test_distance_service.TestDistanceFailureCases.test_service_invalid_latitude_source)
Latitud de origen invlida debe devolver distance=-1.0 y unit='invalid'. ... C:\Users\yo\AppData\Local\Programs\Python\Python313\Lib\site-packages\geopy\point.py:472: UserWarning: Latitude normalization has been prohibited in the newer versions of geopy, because the norma
lized value happened to be on a different pole, which is probably not what was meant. If you pass coordinates as positional args, please make sure that the order is (latitude, longitude) or (y, x) in Cartesian terms.
  return cls(*args)
ok
test_service_invalid_longitude_destination (tests.test_distance_service.TestDistanceFailureCases.test_service_invalid_longitude_destination)
Longitud de destino invlida debe devolver distance=-1.0. ... ok
test_service_unit_unexpected_value (tests.test_distance_service.TestDistanceFailureCases.test_service_unit_unexpected_value)
Unidad desconocida debe devolver -1.0 y 'invalid' o lanzar excepcin. ... ok

======================================================================
FAIL: test_latitude_below_minimum (tests.test_distance_service.TestDistanceFailureCases.test_latitude_below_minimum)
Latitud < -90 debera lanzar ValueError (rango invlido).
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\yo\Desktop\Prueba sistemas\tests\test_distance_service.py", line 19, in test_latitude_below_minimum
    with self.assertRaises(ValueError):
         ~~~~~~~~~~~~~~~~~~^^^^^^^^^^^
AssertionError: ValueError not raised

======================================================================
FAIL: test_longitude_below_minimum (tests.test_distance_service.TestDistanceFailureCases.test_longitude_below_minimum)
Longitud < -180 debera lanzar ValueError.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\yo\Desktop\Prueba sistemas\tests\test_distance_service.py", line 29, in test_longitude_below_minimum
    with self.assertRaises(ValueError):
         ~~~~~~~~~~~~~~~~~~^^^^^^^^^^^
AssertionError: ValueError not raised

======================================================================
FAIL: test_missing_destination_position (tests.test_distance_service.TestDistanceFailureCases.test_missing_destination_position)
Falta el destino -> debera generar error.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\yo\Desktop\Prueba sistemas\tests\test_distance_service.py", line 131, in test_missing_destination_position
    with self.assertRaises(Exception):
         ~~~~~~~~~~~~~~~~~~^^^^^^^^^^^
AssertionError: Exception not raised

======================================================================
FAIL: test_missing_source_position (tests.test_distance_service.TestDistanceFailureCases.test_missing_source_position)
Falta el origen -> debera generar error.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\yo\Desktop\Prueba sistemas\tests\test_distance_service.py", line 139, in test_missing_source_position
    with self.assertRaises(Exception):
         ~~~~~~~~~~~~~~~~~~^^^^^^^^^^^
AssertionError: Exception not raised

======================================================================
FAIL: test_service_empty_unit_should_return_kilometers (tests.test_distance_service.TestDistanceFailureCases.test_service_empty_unit_should_return_kilometers)
Si la unidad esperada est en blanco, la respuesta debe ser en kilmetros.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\yo\Desktop\Prueba sistemas\tests\test_distance_service.py", line 110, in test_service_empty_unit_should_return_kilometers
    self.assertAlmostEqual(resp.distance, expected_km, delta=0.5)
    ~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 50.56411361694336 != 93.64478249445669 within 0.5 delta (43.08066887751333 difference)

----------------------------------------------------------------------
Ran 14 tests in 0.003s

FAILED (failures=5)

[INFO] Distancia esperada (km): 93.64478249445669
[INFO] Distancia devuelta por el servicio: 50.56411361694336
[INFO] Unidad reportada: km
```


```
Ran 14 tests in 0.003s

FAILED (failures=5)

[INFO] Distancia esperada (km): 93.64478249445669
[INFO] Distancia devuelta por el servicio: 50.56411361694336
[INFO] Unidad reportada: km
```