



Actividad de Aprendizaje 2: Desarrollo de una API SOAP

Seguimiento de Paquetes en una Empresa de Logística

Docente: Ing. Geovanny Cudco

**Universidad de las Fuerzas
Armadas ESPE**

**Carrera de Ingeniería en Tecnologías de
la Información**

ESTUDIANTE: Jorge Andrés Arroyo Cedeño
Cherley Maria Caiza Chasipanta

Fecha de Entrega: 31 de mayo de 2025

Informe Técnico: Sistema de Rastreo de Paquetes vía API SOAP

Proyecto: EnvíosExpress S.A.C.

NRC

Arquitectura de Software

Introducción

En el contexto actual de transformación digital, las empresas de logística enfrentan una creciente demanda de transparencia, trazabilidad y acceso a información en tiempo real. EnvíosExpress S.A.C., empresa dedicada a la distribución de paquetes a nivel nacional, identificó la necesidad de ofrecer a sus clientes un servicio de rastreo de envíos eficiente y seguro. Con el fin de garantizar la interoperabilidad entre sistemas y mantener altos estándares de seguridad y comunicación, se decidió implementar un servicio web basado en el protocolo SOAP (Simple Object Access Protocol). El presente informe documenta el proceso de análisis, diseño, implementación y validación de una API SOAP que permite a los usuarios consultar el estado actual de sus paquetes utilizando un número único de seguimiento, obteniendo además detalles como ubicación actual, fecha estimada de entrega y el historial completo de eventos registrados.

Objetivos

Objetivo General

Evaluar el nivel de comprensión de los conceptos relacionados con la creación y consumo de servicios web SOAP mediante el diseño e implementación de una solución a un problema real.

Objetivos Específicos

Desarrollar e implementar un servicio web SOAP que permita a los clientes de EnvíosExpress S.A.C. consultar el estado de sus envíos en tiempo real, de forma segura y estandarizada.

Modelar entidades necesarias para la información de envío de paquetes y diseñar un WSDL según las buenas prácticas del protocolo SOAP.

Implemente la lógica de negocio para consultar por número de guía, gestar errores claros y estructurados, documentar el uso del servicio y validar su funcionamiento mediante herramientas como SoapUI.

Problema planteado

Una empresa de logística llamada **EnvíosExpress S.A.C.** desea ofrecer a sus clientes la posibilidad de consultar el estado actual de sus paquetes utilizando un número único de seguimiento (*tracking number*). Para garantizar interoperabilidad, seguridad y estándares, se ha decidido desarrollar una **API SOAP** que exponga esta funcionalidad.

Metodología

Requerimientos funcionales

El sistema debe permitir:

1. Consultar el estado actual de un paquete mediante su número de tracking.
2. Devolver información detallada:
 - Estado actual
 - Ubicación actual
3. Fecha estimada de entrega (opcional)
 - Historial de eventos del paquete
4. Manejar errores de forma clara y estandarizada.

Entidades del sistema

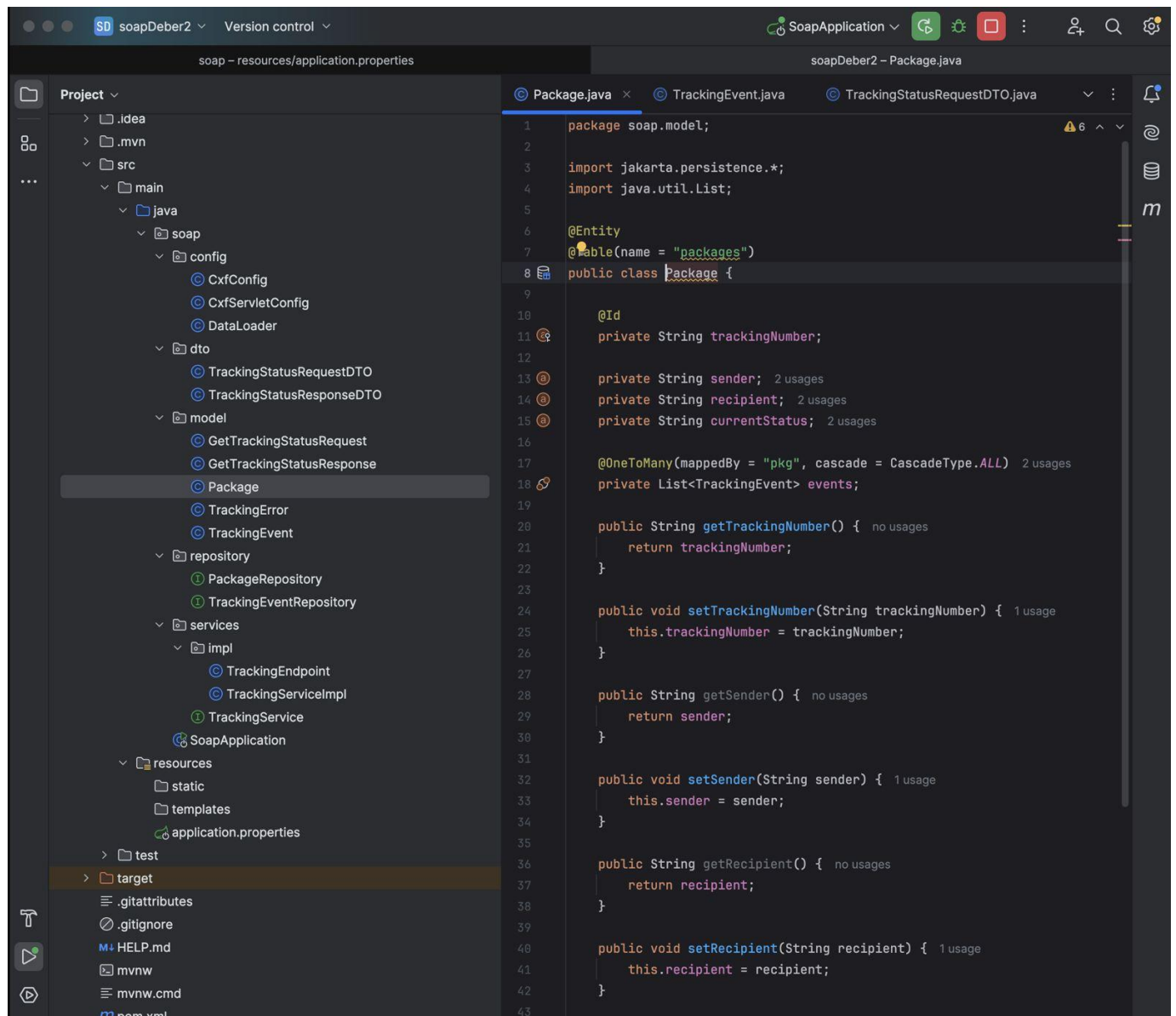
Las siguientes entidades deben modelarse para construir la API:

1. Paquete (Package)

Representa un envío individual.

Atributos:

- trackingNumber: Identificador único
- senderName: Nombre del remitente
- receiverName: Nombre del destinatario
- origin: Ciudad/punto de origen
- destination: Ciudad/punto de destino
- weight: Peso del paquete (en kg)
- dimensions: Dimensiones del paquete (largo x ancho x alto)
- status: Estado actual del paquete (ej.: ^{En} tránsito", ^{En}tregado")
- currentLocation: Ubicación actual
- estimatedDeliveryDate: Fecha estimada de entrega
- history: Lista de eventos (TrackingEvent[])

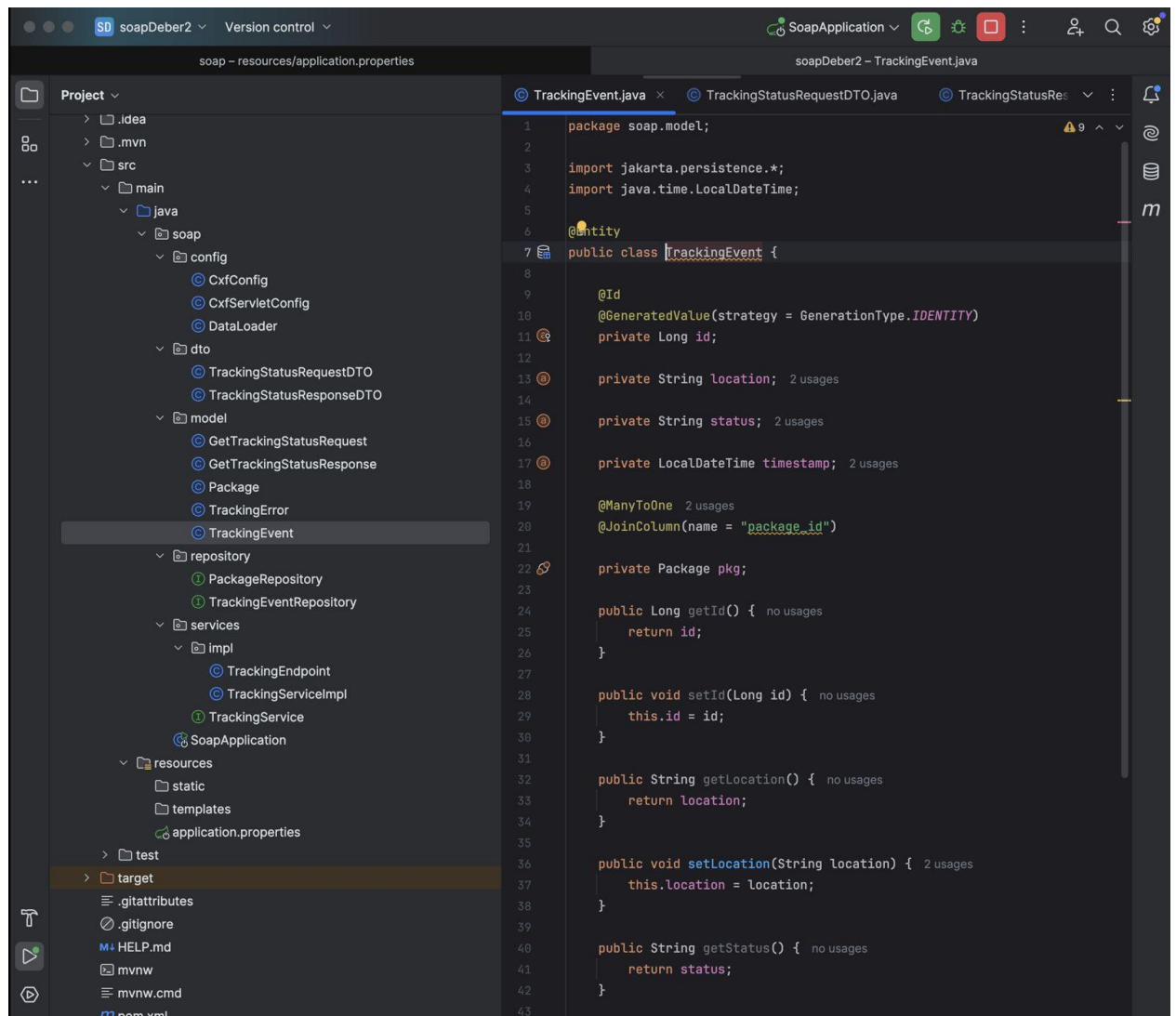


2. Evento de seguimiento (TrackingEvent)

Describe cada movimiento o cambio de estado del paquete.

Atributos:

- date: Fecha y hora del evento
- description: Descripción textual del evento
- location: Ubicación donde ocurrió el evento

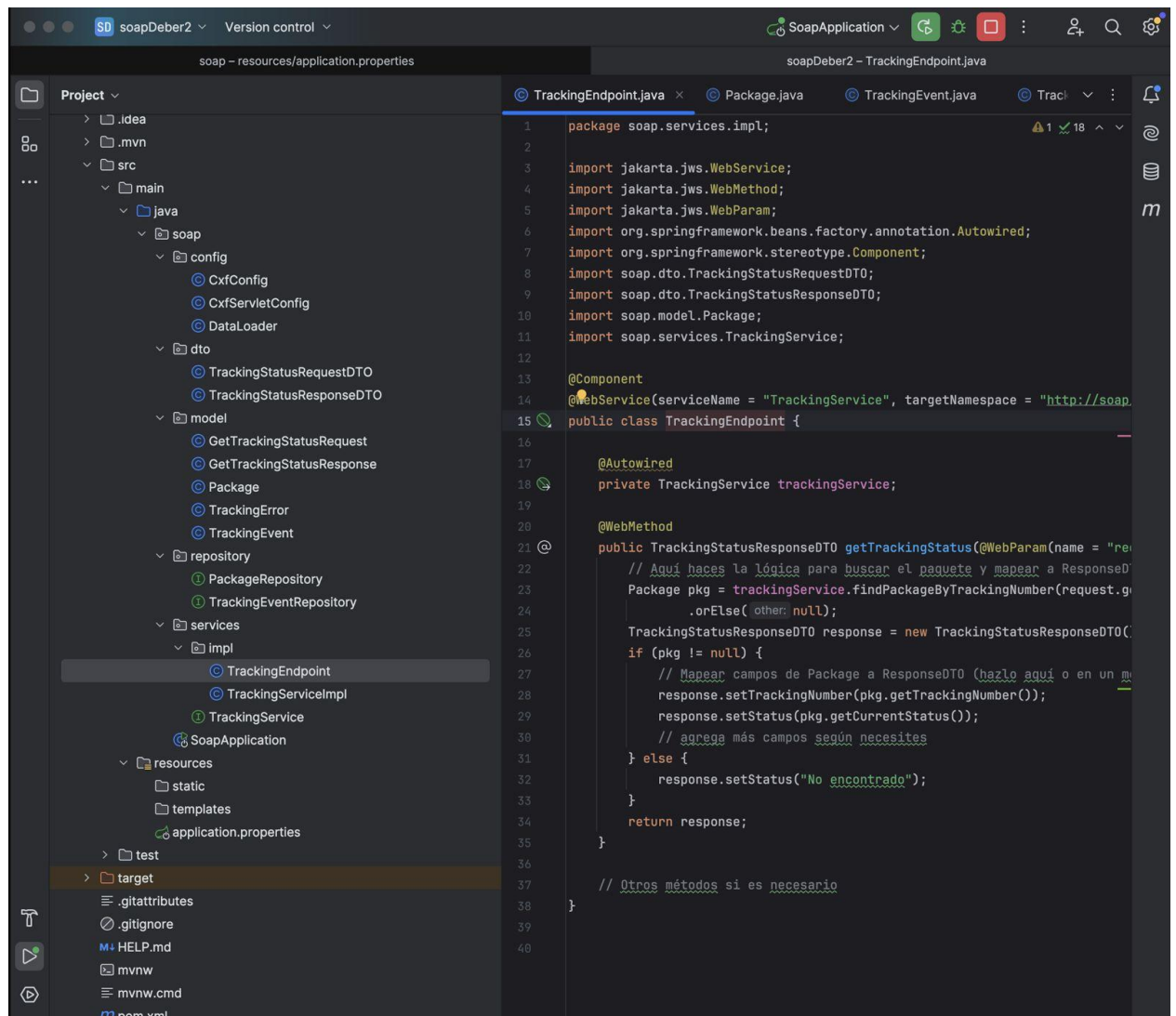


3. Solicitud de consulta (GetTrackingStatusRequest)

Objeto usado por el cliente para solicitar datos de seguimiento.

Atributos:

- trackingNumber: Número de guía del paquete a consultar

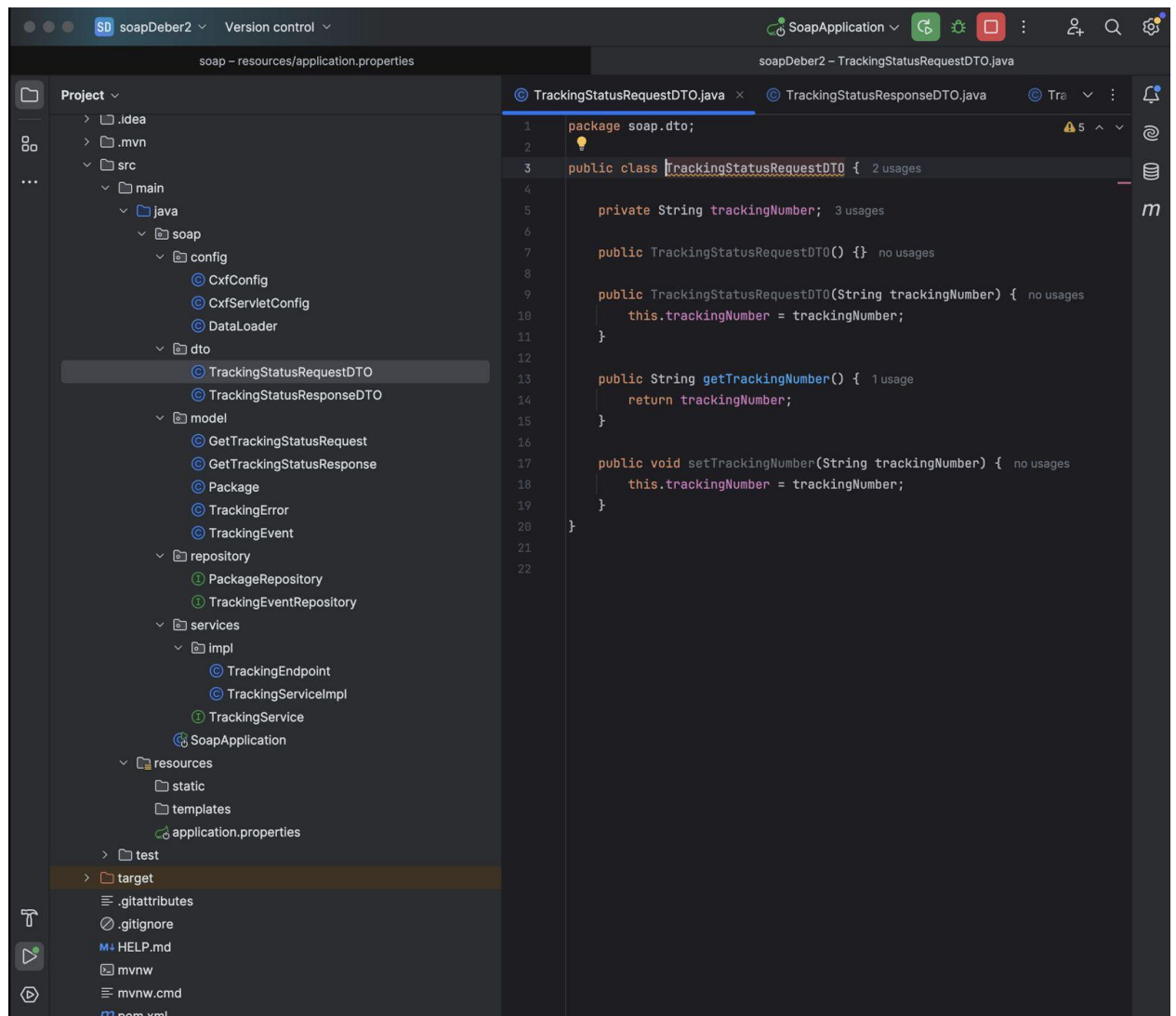


4. Respuesta de consulta (GetTrackingStatusResponse)

Objeto devuelto por el servicio con la información del paquete.

Atributos:

- status: Estado actual
- currentLocation: Ubicación actual
- estimatedDeliveryDate: Fecha estimada de entrega
- history: Lista de eventos (TrackingEvent[])

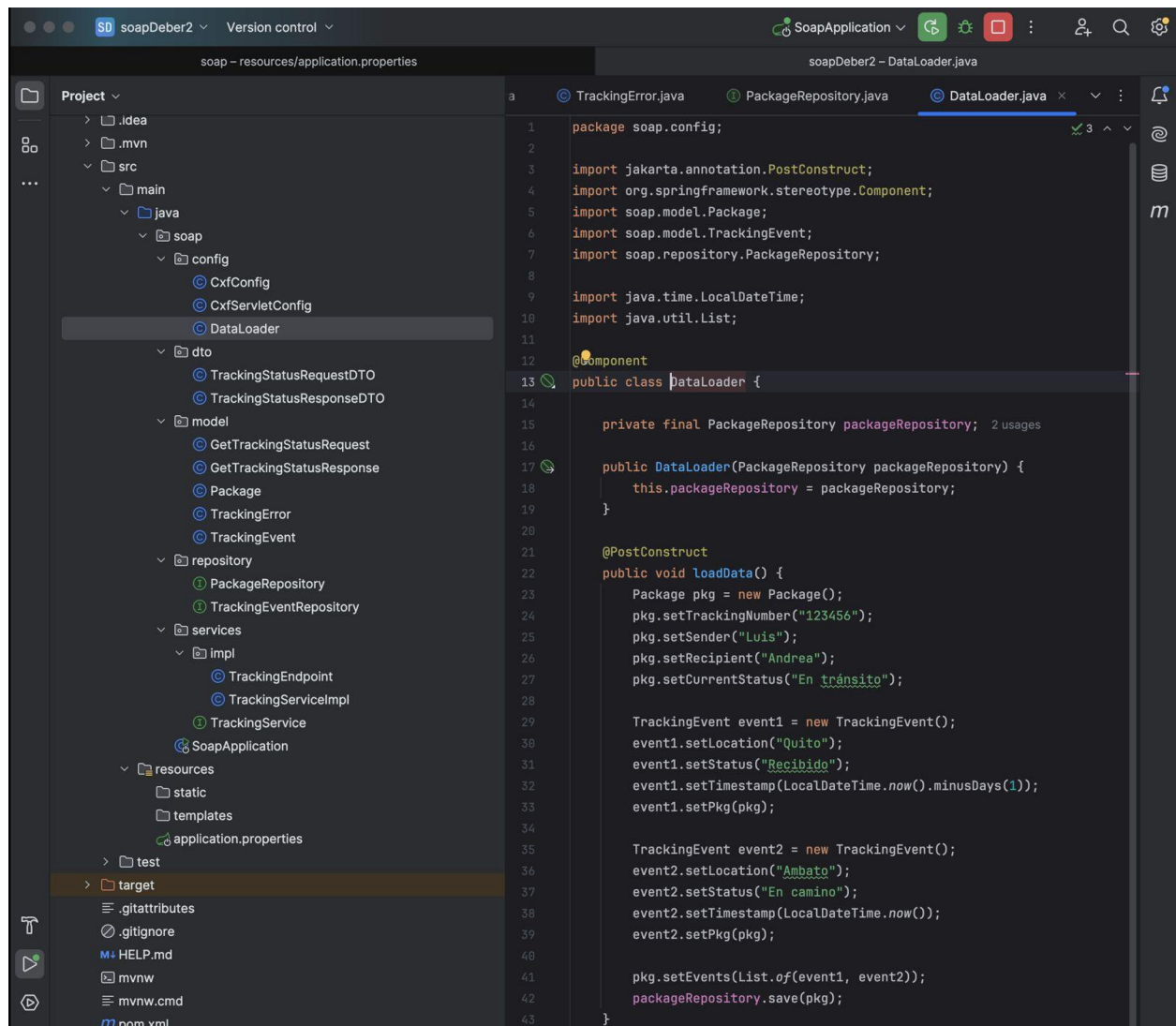


5. Error (TrackingError)

(Opcional pero recomendado)

Atributos:

- errorCode: Código numérico del error
- errorMessage: Mensaje descriptivo del error
- invalidField: Campo que causó el error

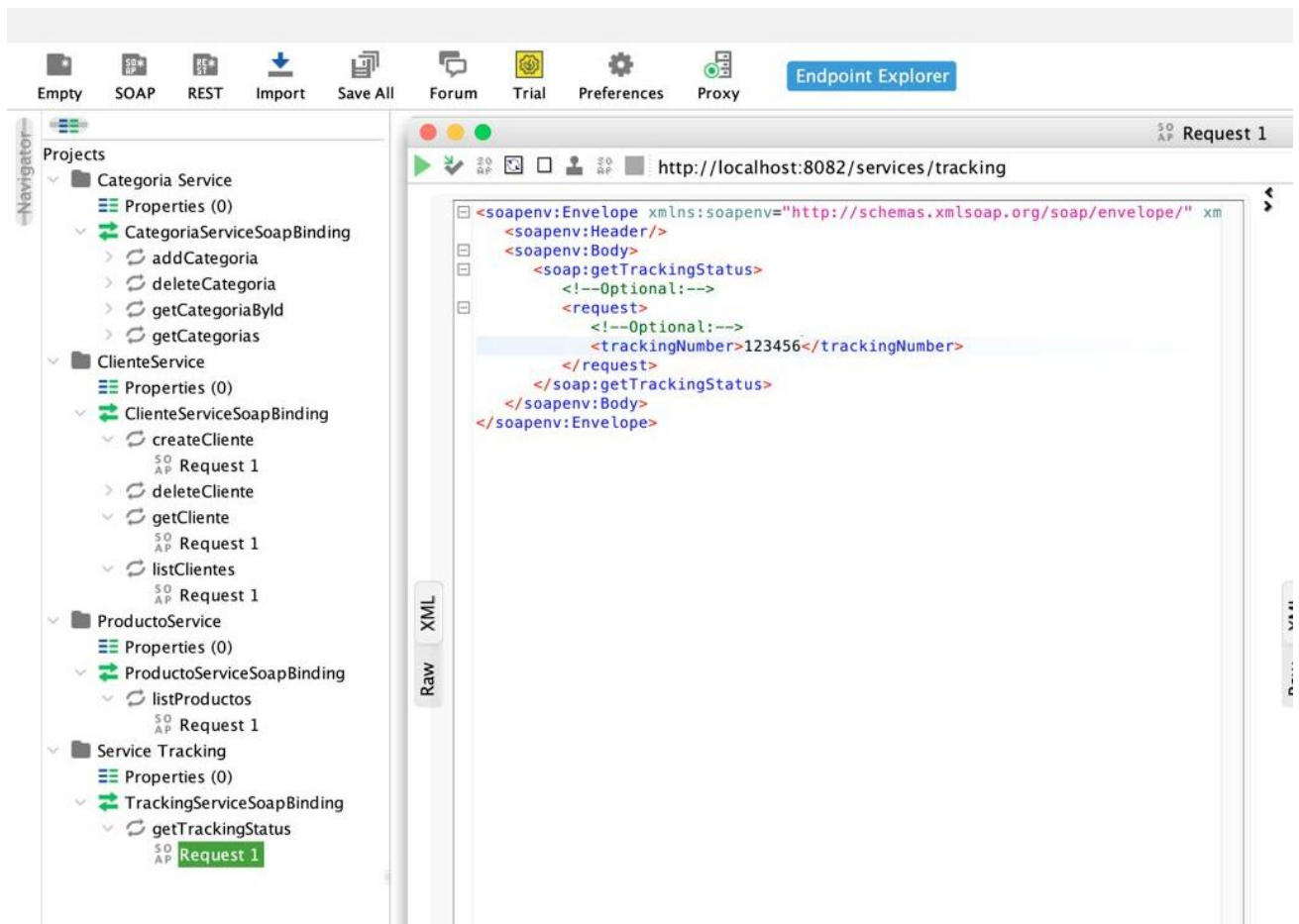


Ejemplo de Mensaje SOAP Request

```

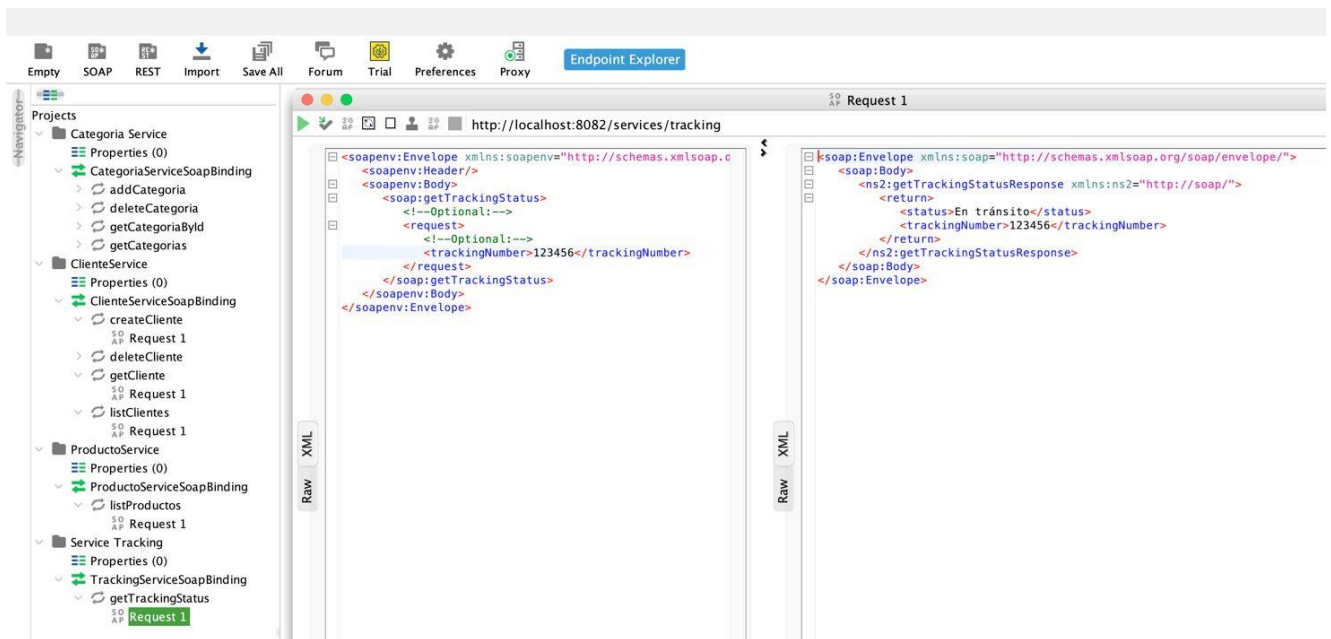
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soap="http://soap/">
  <soapenv:Header/>
  <soapenv:Body>
    <soap:getTrackingStatus>
      <!--Optional:-->
      <request>
        <!--Optional:-->
        <trackingNumber>123456</trackingNumber>
      </request>
    </soap:getTrackingStatus>
  </soapenv:Body>
</soapenv:Envelope>

```

Ejemplo de Mensaje SOAP Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns2:getTrackingStatusResponse xmlns:ns2="http://soap/">
      <return>
        <status>En tránsito</status>
        <trackingNumber>123456</trackingNumber>
      </return>
    </ns2:getTrackingStatusResponse>
  </soap:Body>
</soap:Envelope>
```

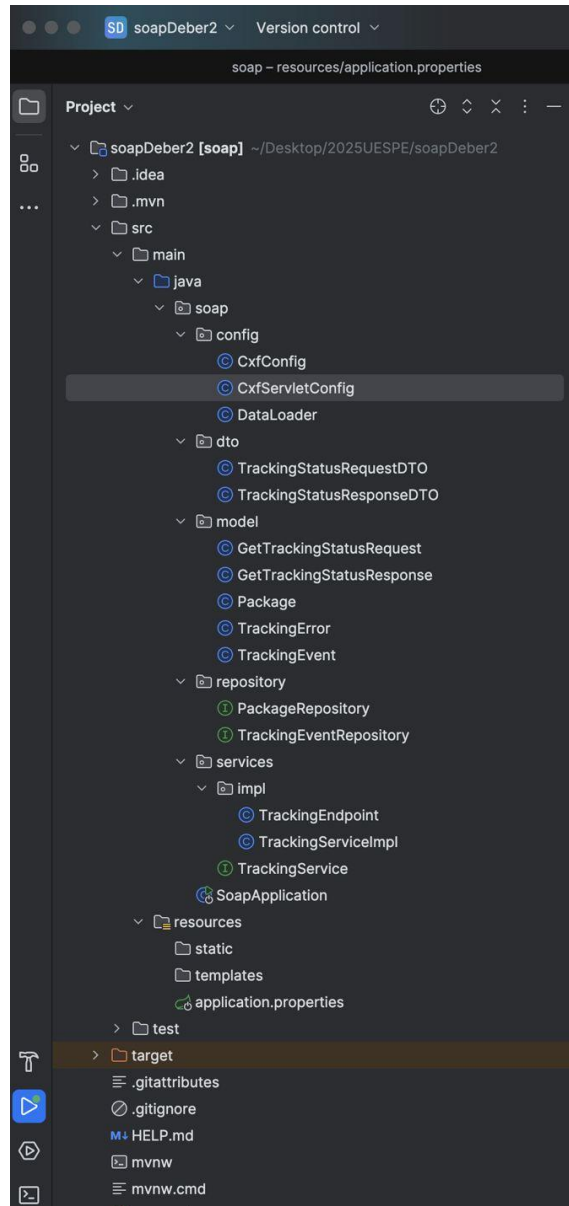


Resultados

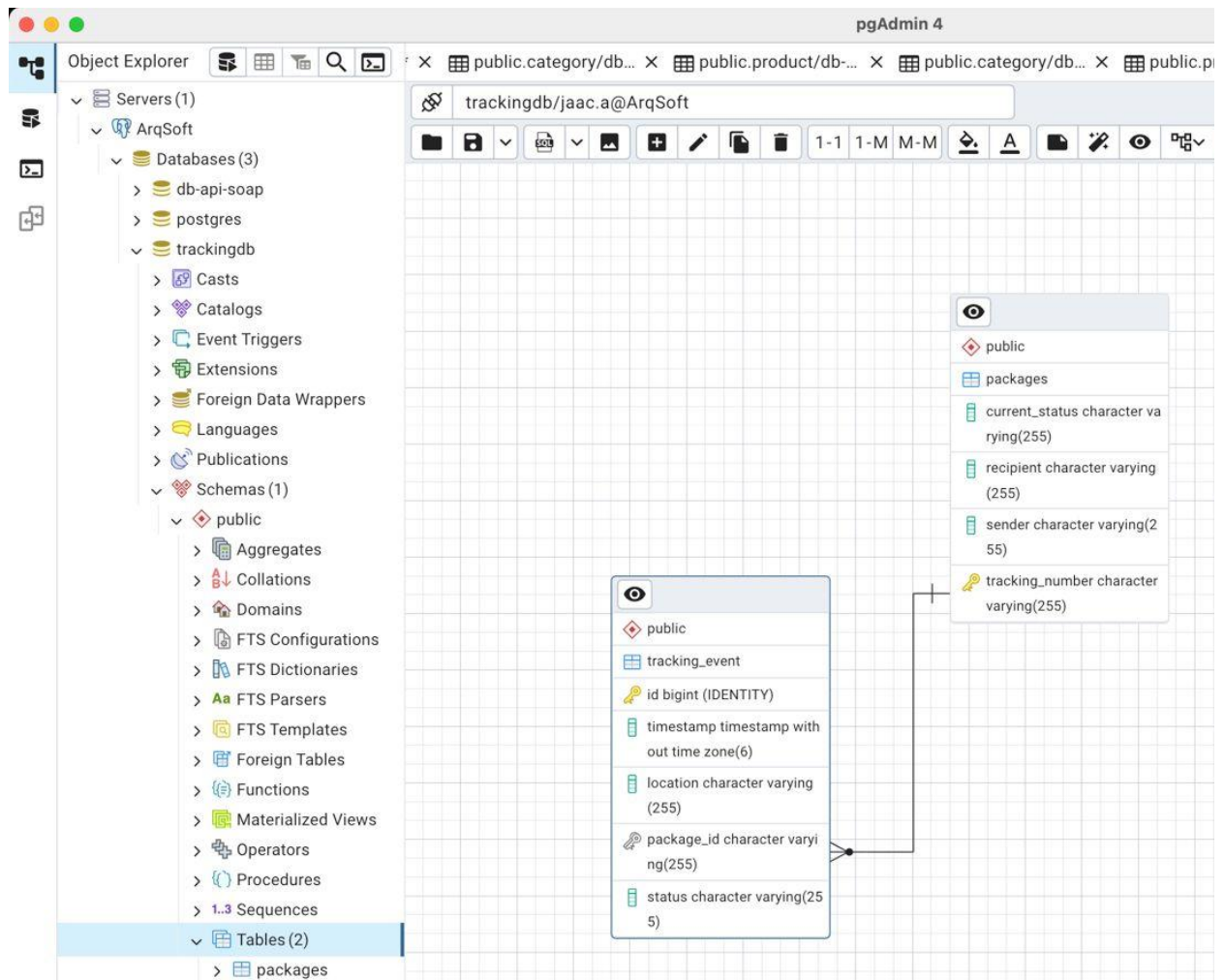
El sistema implementado cumple con los requisitos establecidos y ofrece interoperabilidad mediante SOAP y WSDL, respuesta estructurada con datos de seguimiento completos, gestión de errores mediante objetos específicos y documentación en el archivo README.md para el lanzamiento del proyecto y las pruebas del servicio. Además, proporciona respuestas claras a datos no válidos o no encontrados.

Estructura General del Api

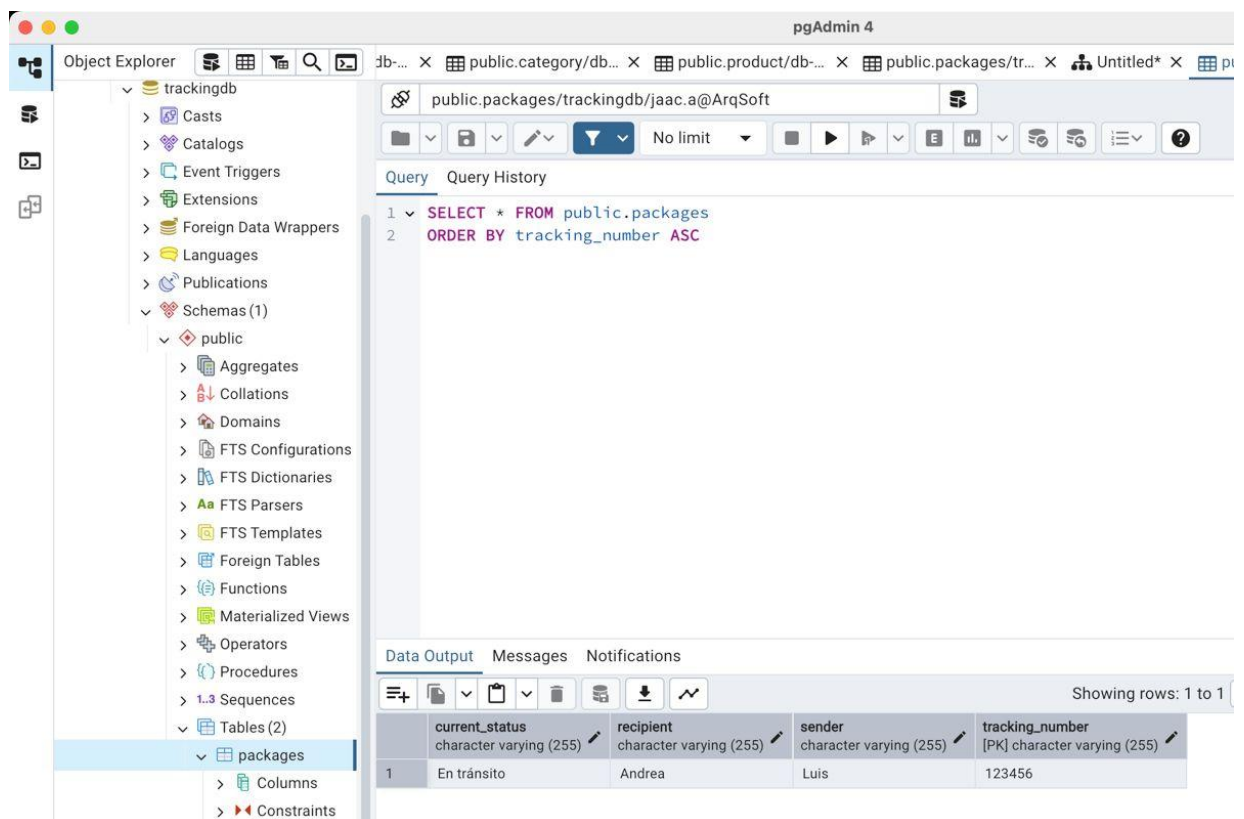
La estructura del proyecto fue organizada esto facilita el mantenimiento, escalabilidad y comprensión del código.



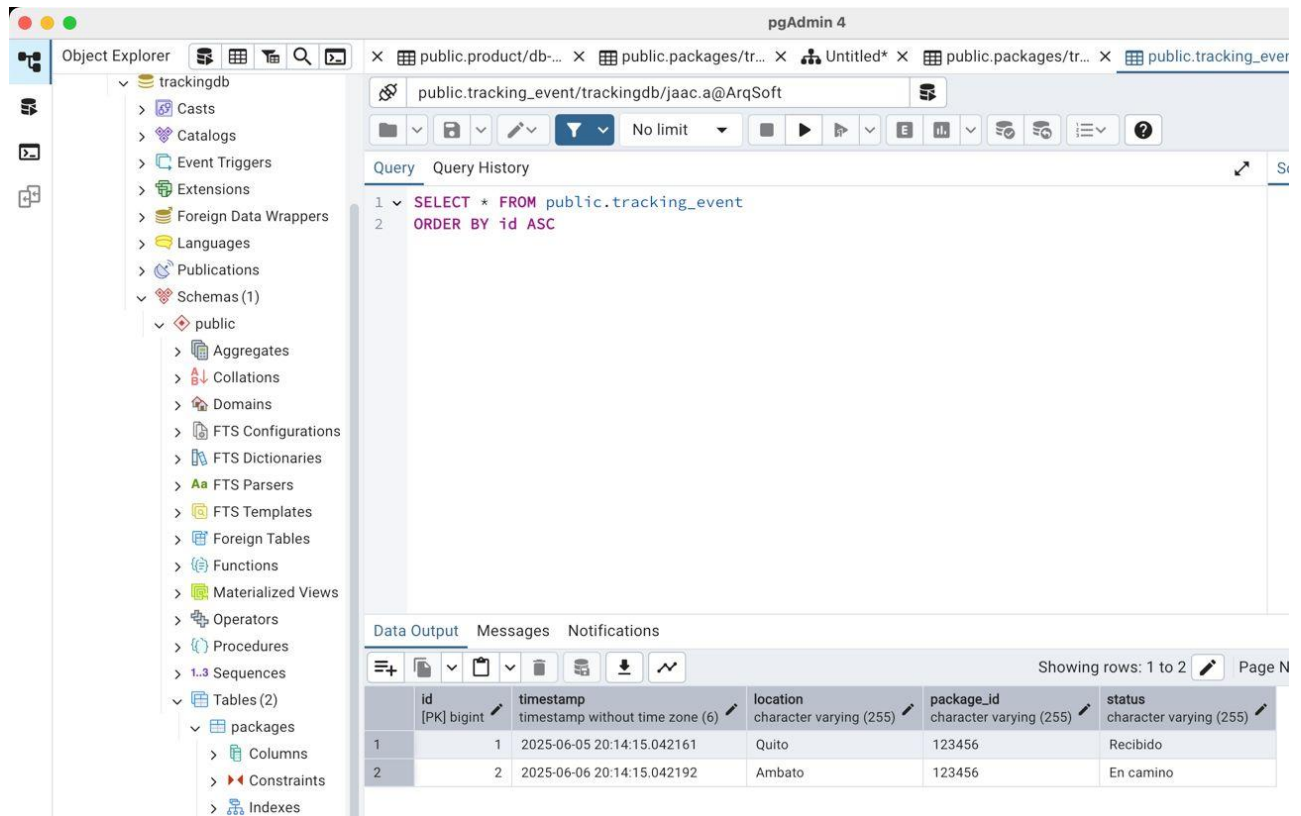
Creación de las tablas en pgAdmin 4



Comprobamos que si se guardan y muestra el registro



Comprobamos el estado en la otra tabla



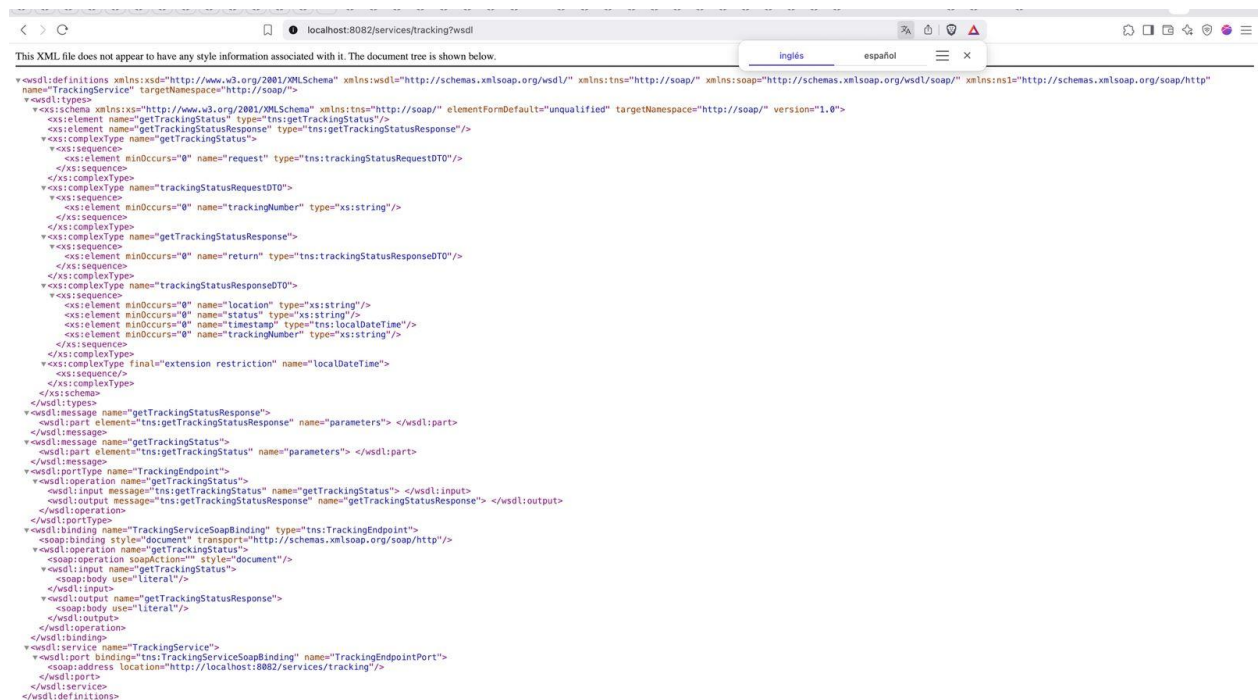
The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure for 'trackingdb', including Schemas (public), Tables (packages), and Columns. The main pane shows a SQL query:

```
1 SELECT * FROM public.tracking_event
2 ORDER BY id ASC
```

Below the query, the Data Output tab displays the results of the query in a table format. The table has 6 columns: id, timestamp, location, package_id, and status. The results show two rows of data:

id	timestamp	location	package_id	status
1	2025-06-05 20:14:15.042161	Quito	123456	Recibido
2	2025-06-06 20:14:15.042192	Ambato	123456	En camino

Archivo WSDL de servicio SOAP desplegado en <http://localhost:8082/services/tracking?wsdl>



The screenshot shows a web browser displaying the WSDL file for the tracking service. The URL is <http://localhost:8082/services/tracking?wsdl>. The page shows the XML content of the WSDL file, which defines the service and its operations. The XML content is as follows:

```
<?xml version='1.0'?>
<definitions xmlns:xsd='http://www.w3.org/2001/XMLSchema' xmlns:soap='http://schemas.xmlsoap.org/soap/' xmlns:tns='http://schemas.xmlsoap.org/soap/http' targetNamespace='http://schemas.xmlsoap.org/soap/http'>
  <types>
    <xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema' xmlns:tns='http://schemas.xmlsoap.org/soap/http' elementFormDefault='unqualified' targetNamespace='http://schemas.xmlsoap.org/soap/http' version='1.0'>
      <xsd:element name='getTrackingStatus' type='tns:getTrackingStatusResponse'/>
      <xsd:complexType name='getTrackingStatusResponse'>
        <xsd:sequence base='xsd:string' minOccurs='0' maxOccurs='1'>
          <xsd:element name='request' type='tns:trackingStatusRequestDTO'/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name='trackingStatusRequestDTO'>
        <xsd:sequence base='xsd:string' minOccurs='0' maxOccurs='1'>
          <xsd:element name='trackingNumber' type='xsd:string'/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name='getTrackingStatusResponse'>
        <xsd:sequence base='xsd:string' minOccurs='0' maxOccurs='1'>
          <xsd:element name='return' type='tns:trackingStatusResponseDTO'/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name='trackingStatusResponseDTO'>
        <xsd:sequence base='xsd:string' minOccurs='0' maxOccurs='1'>
          <xsd:element name='location' type='xsd:string'/>
          <xsd:element name='status' type='xsd:string'/>
          <xsd:element name='timestamp' type='tns:localDateTime'/>
          <xsd:element name='trackingNumber' type='xsd:string'/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType final='extension restriction' base='tns:trackingStatusResponseDTO'>
        <xsd:sequence base='xsd:string' minOccurs='0' maxOccurs='1'>
          <xsd:element name='trackingNumber' type='xsd:string'/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <message name='getTrackingStatusResponse'>
    <part element='tns:getTrackingStatusResponse' name='parameters'/>
  </message>
  <message name='getTrackingStatus'>
    <part element='tns:getTrackingStatus' name='parameters'/>
  </message>
  <portType name='TrackingEndpoint'>
    <operation name='getTrackingStatus'>
      <input message='tns:getTrackingStatus' name='getTrackingStatus'/>
      <output message='tns:getTrackingStatusResponse' name='getTrackingStatusResponse'/>
    </operation>
  </portType>
  <binding name='TrackingServiceSoapBinding' type='tns:TrackingEndpoint'>
    <soap:binding style='document' transport='http://schemas.xmlsoap.org/soap/http'/>
    <operation name='getTrackingStatus'>
      <soap:operation soapAction='http://schemas.xmlsoap.org/soap/http' style='document'/>
      <input name='getTrackingStatus'>
        <soap:body use='literal'/>
      </input>
      <output name='getTrackingStatusResponse'>
        <soap:body use='literal'/>
      </output>
    </operation>
  </binding>
  <service name='TrackingService'>
    <port binding='tns:TrackingServiceSoapBinding' name='TrackingEndpointPort'>
      <soap:address location='http://localhost:8082/services/tracking'/>
    </port>
  </service>
</definitions>
```

Bibliografía

- Oracle. (2024). *Java Web Services Tutorial*. Disponible en: <https://docs.oracle.com/javaee/>
- Spring.io. (2024). *Spring Web Services Documentation*. Recuperado de: <https://spring.io/projects/spring-ws>
- W3C. (2023). *SOAP Version 1.2 Part 1: Messaging Framework*. Disponible en: <https://www.w3.org/TR/soap12-part1/>
- Richardson, L. & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.
- Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. University of California

