

GAUDI SOLUTIONS  
ARQUITECTURA DE  
SOFTWARE  
10 Julio. de 2013

INTEGRANTES	CODIGOS
Néstor Cruz Hernández	201310690
Juan Pedro Mendoza	200310723
Felipe Rojas Echeverri	201315979
María Paula Forero	201310697
Julián Aguirre Domínguez	201221709




## Experimento Seguridad y disponibilidad TransAlpes

Descripción del Experimento	
Título: <b>Experimento - Interoperabilidad</b>	ID: <b>test-01</b>
Descripción: Probar que sea posible la interoperabilidad con aplicaciones externas basadas en tecnologías diferentes al sistema desarrollado,	Responsable: Gaudi Solutions
Propósito: <input type="checkbox"/> Reparación, actualizar, clarificar <input checked="" type="checkbox"/> Obtener Información técnica <input type="checkbox"/> Obtener información de negocio <input type="checkbox"/> Otros:	
Propósito: <ul style="list-style-type: none"> <li>Establecer un mecanismo de interoperabilidad que permita el intercambio de mensajes entre el sistema central y sistemas externos como: clientes, policía, bomberos, cruz roja, etc.</li> </ul>	
Descripción del experimento:  <p>A continuación se describen los pasos del experimento.</p> <ol style="list-style-type: none"> <li>Se envía una trama de información al sistema central.</li> <li>La trama ingresa al sistema y es almacenada en un módulo de persistencia.</li> <li>Una aplicación basada en .NET consume un servicio que le permite acceder a la información de la trama.</li> <li>La aplicación basada en .NET muestra la información del vehículo: estado de la carga, posición, etc.</li> </ol> <p>Para acceder a la información del vehículo se expuso un servicio web basado en Java, que permite consultar las tramas persistidas.</p>	
Artefactos Creados: <ul style="list-style-type: none"> <li>Componente de persistencia.</li> <li>Componente de integración.</li> <li>Documento de experimento.</li> </ul>	
Criterio de terminación: La prueba termina luego de que es posible persistir y consultar una trama a través del sistema central.	
Recursos Requeridos: <ul style="list-style-type: none"> <li>2 máquinas virtuales prestadas por la Universidad de los Andes</li> <li>Los 2 computadores deben tener instalada la máquina virtual de java versión 1.7</li> <li>IDE de desarrollo Eclipse.</li> <li>Servidor de aplicaciones Glassfish V4</li> </ul>	
Duración estimada : La duración estimada de la ejecución del experimento es de 2 horas.	
Resultados del Experimento	
Resumen de los resultados: De la ejecución de este experimento, se obtienen los siguientes resultados:  Como se evidencia en la siguiente pantalla, se ingresa el id del vehículo 2, al presionar el botón consultar el	

servicio retorna la información del vehículo y se muestra su posición en el mapa.

**Sistema de seguimiento y Monitoreo Vehicular (TransAlpes)**

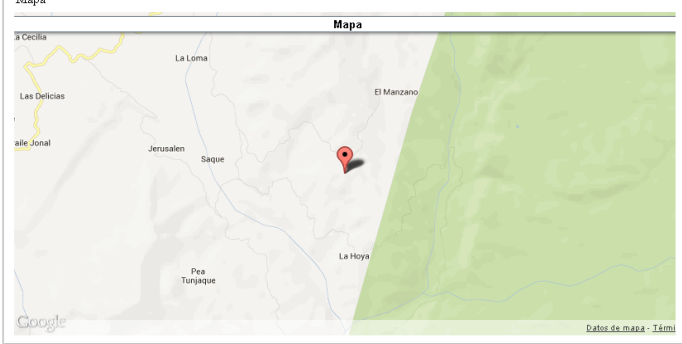
ID Vehículo:

Universidad de los Andes Colombia 

Información Vehículo

Tipo de Vehículo: 7  
Latitud: 4° 39' 58''  
Longitud: -74° 6' 38''  
Cupo Disponible: 100 %  
Tiempo Ruta: 720 Minutos  
Estado Mercancia: EXCELENTE  
Temperatura: -100 °C

Mapa



SOAP Request:

```
= <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://integration.test.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
= <soapenv:Body>
= <q0:getVehicleInfo>
<vehicleId>2</vehicleId>
</q0:getVehicleInfo>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP Response:

```
= <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
= <S:Body>
= <ns2:getVehicleInfoResponse xmlns:ns2="http://integration.test.com/">
<return>7</return>
<return>4;39;58</return>
<return>-74;6;38</return>
<return>100</return>
<return>720</return>
<return>1</return>
<return>-100</return>
</ns2:getVehicleInfoResponse>
</S:Body>
</S:Envelope>
```

Duración Real: 1.3 horas
Recursos Reales: <ul style="list-style-type: none"> <li>• 3 máquinas virtuales prestadas por la Universidad de los Andes</li> <li>• Los 3 computadores deben tener instalada la máquina virtual de java versión 1.6</li> <li>• IDE de desarrollo Eclipse.</li> <li>• IDE de desarrollo Eclipse.</li> <li>• Servidor de aplicaciones Glassfish V4</li> </ul>
Recomendaciones:

Descripción del Experimento	
Título: <b>Experimento - Interoperabilidad</b>	ID: <b>test-02</b>
Descripción: Probar que sea posible la interoperabilidad con aplicaciones externas basadas en tecnologías diferentes al sistema desarrollado,	Responsable: Gaudi Solutions
Propósito: ( ) Reparación, actualizar, clarificar ( X ) Obtener Información técnica ( ) Obtener información de negocio ( ) Otros:	
Propósito: <ul style="list-style-type: none"> <li>• Establecer un mecanismo de interoperabilidad que permita el intercambio de mensajes entre el sistema central y sistemas externos como: policía, bomberos, cruz roja, etc.</li> </ul>	
Descripción del experimento: <p>A continuación se describen los pasos del experimento.</p> <ol style="list-style-type: none"> <li>1. Se envía una trama de alarma al Componente de alarmas.</li> <li>2. El componente de alarmas invoca un servicio web basado en Java y éste se encarga de invocar el servicio web de la policía desarrollado en .NET.</li> <li>3. Cuando el sistema de la policía reciba una alarma, envía un correo invocando la posición del vehículo.</li> </ol>	
Artefactos Creados: <ul style="list-style-type: none"> <li>• Componente de alarmas.</li> <li>• Componente de integración.</li> <li>• Documento de experimento.</li> </ul>	
Criterio de terminación: La prueba termina una vez es enviada la posición del vehículo a través de un correo electrónico.	
Recursos Requeridos:	

- 2 máquinas virtuales prestadas por la Universidad de los Andes
- Los 2 computadores deben tener instalada la máquina virtual de java versión 1.7
- IDE de desarrollo Eclipse.
- Servidor de aplicaciones Glassfish V4
- Servidor http IIS

Duración estimada :

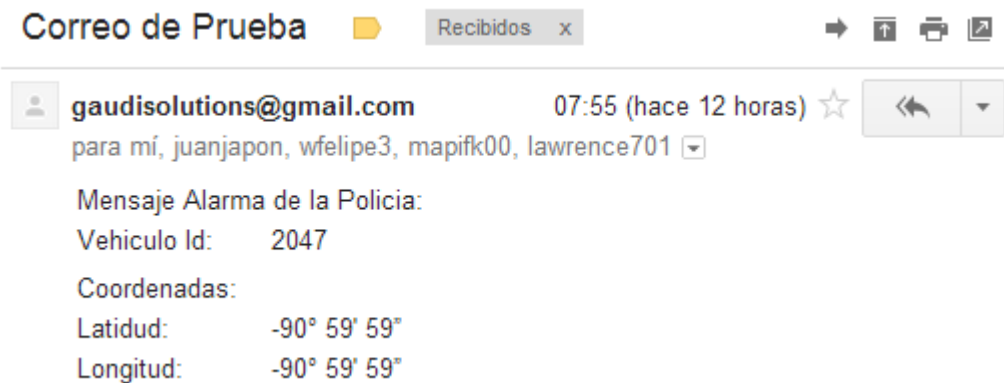
La duración estimada de la ejecución del experimento es de 2 horas.

### Resultados del Experimento

Resumen de los resultados:

De la ejecución de este experimento, se obtienen los siguientes resultados:

Se envía una alarma al sistema de la policía originada por el vehículo N° 2047



A continuación se muestra la información intercambiada desde el Componente de alarmas hacia el componente de integración:

SOAP Request:

```

=<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://integration.test.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
=<soapenv:Body>
=<q0:sendAlarm>
=<alarm>
<laGrades>90</laGrades>
<laMinutes>59</laMinutes>
<laSeconds>59</laSeconds>
<laSign>1</laSign>
<loGrades>90</loGrades>
<loMinutes>59</loMinutes>
<loSeconds>59</loSeconds>
<loSign>1</loSign>
<vehicleId>2047</vehicleId>

```

```

<driverStatus>1</driverStatus>
<emergencyType>1</emergencyType>
</alarm>
</q0:sendAlarm>
</soapenv:Body>
/soapenv:Envelope>

```

SOAP Response:

```

= <S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
= <S:Body>
= <ns2:sendAlarmResponse xmlns:ns2="http://integration.test.com/">
<return>1</return>
</ns2:sendAlarmResponse>
</S:Body>
</S:Envelope>

```

A continuación se muestra el formato de información intercambiada desde el Componente de integración hacia el sistema de la policía:

localhost/PoliciaWeb/Service1.aspx?op=AlarmaPolicia

Service1

Haga clic [aquí](#) para obtener una lista completa de operaciones.

### AlarmaPolicia

**Prueba**

El formulario de prueba sólo está disponible para métodos con tipos primitivos como parámetros.

**SOAP 1.1**

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.1. Es necesario reemplazar los **marcadores de posición** que aparecen con valores reales.

```

POST /PoliciaWeb/Service1.aspx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/AlarmaPolicia"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AlarmaPolicia xmlns="http://tempuri.org/">
      <id_vehiculo>string</id_vehiculo>
      <latitud>
        <int>int</int>
      </latitud>
      <longitud>
        <int>int</int>
      </longitud>
    </AlarmaPolicia>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AlarmaPoliciaResponse xmlns="http://tempuri.org/">
      <AlarmaPoliciaResult>string</AlarmaPoliciaResult>
    </AlarmaPoliciaResponse>
  </soap:Body>
</soap:Envelope>

```

**SOAP 1.2**

A continuación se muestra un ejemplo de solicitud y respuesta para SOAP 1.2. Es necesario reemplazar los **marcadores de posición** que aparecen con valores reales.

```

POST /PoliciaWeb/Service1.aspx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <AlarmaPolicia xmlns="http://tempuri.org/">
      <id_vehiculo>string</id_vehiculo>
      <latitud>
        <int>int</int>
      </latitud>
      <longitud>
        <int>int</int>
      </longitud>
    </AlarmaPolicia>
  </soap12:Body>
</soap12:Envelope>

```

El XSD usado para definir la estructura de la información de la alarma y , de éste componente y permitir la integración, es el siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.3-
7528; 2013-04-29T19:34:10+0000) JAXWS-RI/2.2.8 JAXWS/2.2 svn-revision#unknown. -->
<xs:schema targetNamespace="http://integration.test.com/" version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://integration.test.com/">
  <xs:element type="tns:getVehicleInfo"
name="getVehicleInfo"/>
  <xs:element type="tns:getVehicleInfoResponse"
name="getVehicleInfoResponse"/>
  <xs:element type="tns:sendAlarm"
name="sendAlarm"/>
  <xs:element type="tns:sendAlarmResponse"
name="sendAlarmResponse"/>
  <xs:complexType
name="getVehicleInfo">
    <xs:sequence>
      <xs:element type="xs:string" name="vehicleId"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType
name="getVehicleInfoResponse">
    <xs:sequence>
      <xs:element type="xs:string" name="return"
minOccurs="0" maxOccurs="unbounded"
nillable="true"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType
name="sendAlarm">
    <xs:sequence>
      <xs:element type="tns:alarmFrameDTO" name="alarm"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType
name="alarmFrameDTO">
    <xs:complexContent base="tns:frameDTO">
      <xs:sequence>
        <xs:element type="xs:int"
name="driverStatus"/>
        <xs:element type="xs:int"
name="emergencyType"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType
name="frameDTO">
    <xs:sequence>
      <xs:element type="xs:int"
name="laGrades"/>
      <xs:element type="xs:int" name="laMinutes"/>
      <xs:element type="xs:int"
name="laSeconds"/>
      <xs:element type="xs:int" name="laSign"/>
      <xs:element type="xs:int"
name="loGrades"/>
      <xs:element type="xs:int" name="loMinutes"/>
      <xs:element type="xs:int"
name="loSeconds"/>
      <xs:element type="xs:int" name="loSign"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType
name="sendAlarmResponse">
    <xs:sequence>
      <xs:element type="xs:string" name="return"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

En este XSD se define la latitud y la longitud del vehículo en grados, minutos y segundos, además del id del vehículo, el estado del conductor y el tipo de emergencia.

Duración Real:  
1 hora

Recursos Reales:

- 2 máquinas virtuales prestadas por la Universidad de los Andes
- Los 2 computadores deben tener instalada la máquina virtual de java versión 1.6
- IDE de desarrollo Eclipse.
- Servidor de aplicaciones Glassfish V4
- Servidor http IIS

Recomendaciones:

Descripción del Experimento	
Título: <b>Experimento - Modificabilidad</b>	ID: <b>test-03</b>
Descripción: Probar que sea posible modificar el procesamiento de las tramas cuando se haga un rediseño sobre la misma	Responsable: Gaudi Solutions
Propósito: <input type="checkbox"/> Reparación, actualizar, clarificar <input checked="" type="checkbox"/> Obtener Información técnica <input type="checkbox"/> Obtener información de negocio <input type="checkbox"/> Otros:	
Propósito: <ul style="list-style-type: none"> <li>Establecer un mecanismo de modificabilidad que permita cambiar el diseño de las tramas en menos de 12 horas</li> </ul>	
Descripción del experimento:  A continuación se describen los pasos del experimento.  <ol style="list-style-type: none"> <li>Se agrega un nuevo atributo a la trama</li> <li>Se implementa en menos de 12 horas</li> </ol>	
Artefactos Creados: <ul style="list-style-type: none"> <li>Modificación de la trama y el código fuente</li> </ul>	
Criterio de terminación: La prueba termina una vez se haya agregado el nuevo atributo a la trama	
Recursos Requeridos: <ul style="list-style-type: none"> <li>1 Computador</li> <li>El computador debe tener instalada la máquina virtual de java versión 1.7</li> <li>IDE de desarrollo Eclipse.</li> </ul>	
Duración estimada : La duración estimada de la ejecución del experimento es de 2 horas.	
Resultados del Experimento	
Resumen de los resultados:  De la ejecución de este experimento, se obtienen los siguientes resultados:  <ul style="list-style-type: none"> <li>Se agrega un Nuevo atributo a la trama</li> <li>El nuevo atributo es procesado correctamente</li> </ul>	
Duración Real: 1.5 horas	
Recursos Reales:  <ul style="list-style-type: none"> <li>1 Computador</li> <li>El computador debe tener instalada la máquina virtual de java versión 1.7</li> <li>IDE de desarrollo Eclipse.</li> </ul>	



Para lograr este nivel de modificabilidad en la trama, se pasó de procesar directamente los bits a realizar transformaciones por medio de String, estuvo un impacto en el desempeño agregando 0.5 milisegundos de latencia, dando una latencia promedio por trama de 12 milisegundos

Recomendaciones: