**Arquitectura de Software**

**Experimento 3**



**Realizado por:**

|  |  |  |
| --- | --- | --- |
| **Persona** | **Rol** | **CódigoUniandes** |
| Carlos Ernesto González Vargas | Ingeniero de Requerimientos | 200819123 |
| Sandra Milena Gómez Ríos | Ingeniero de Producción | 201110951 |
| Andrés Mauricio Erazo Benavides | Ingeniero de Soporte | 201110949 |
| David Pérez Chibuque | Ingeniero de Calidad | 201117818 |
| Willian Alejandro Idrobo Luna | Arquitecto Jefe | 201110544 |
| Erik Fernando Arcos Franco | Ingeniero de Administración | 201110856 |

 

**Contenido**

**Pag.**

[**1.** **Definición del Sistema** 3](#_Toc297581717)

[**1.1.** **Vista Funcional** 4](#_Toc297581718)

[**1.2.** **Vista Despliegue** 5](#_Toc297581719)

[**1.3.** **Identificación de patrones** 6](#_Toc297581720)

[**2.** **Experimento A – Seguridad** 7](#_Toc297581721)

[**2.1.** **A1 – Autenticación y Autorización** 7](#_Toc297581722)

[**2.2.** **A2 – Denegación de Servicio / Suplantación** 10](#_Toc297581723)

[**3.** **Experimento B – Disponibilidad** 12](#_Toc297581724)

[**3.1.** **B – Disponibilidad** 12](#_Toc297581725)

**Arquitectura de Software**

**Experimento 3**

1. **Definición del Sistema**

La empresa **Alpes Smart Home (ASH)** desea entrar a participar en el mercado de las casas inteligentes dado el considerable avance de la tecnología en dispositivos móviles, sensores inalámbricos y sistemas embebidos que ha habido durante los últimos años. Por esta razón la empresa **ASH** plantea el proyecto ***Casa Inteligente***, el cual surge para dar respuesta al diseño de arquitectura de solución de un sistema que permita la administración de viviendas inteligentes.

El sistema se debe cumplir con las necesidades iniciales del sistema, descritas a continuación:

* Analizar la información enviada por los sensores de humo, ventanas y puertas de las oficinas y viviendas monitoreadas.
* Una vez se detecta humo en una posición específica de la vivienda u oficina, se procede a notificar a los servicios de control de emergencias. Si se detecta que la vivienda está habitada, se informa a los bomberos y servicios de ambulancias, el número estimado de personas que están en ese momento en el lugar, así como las posibles ubicaciones.
* Si se trata de un sensor de ventana o puerta abierta sin autorización, se debe notificar a la policía sobre el evento.
* La comunicación con las entidades externas se realizará por medio de web services suministrados por dichas entidades.
* Generar un reporte consolidado mostrando todos los eventos registrados (ordenadas por tipo de evento) de forma cronológica (hora, día, mes, año).
* Evitar consulta y modificación de información no autorizadas. También deberá evitar suplantación de servicio y ante el fallo de un componente de sistema se debe continuar respondiendo solicitudes sin perder el estado del sistema.
* Por cada inmueble u oficina se tienen en promedio 50 tags RFID enviando información cada segundo a cuatro antenas ubicadas en diferentes posiciones de la casa u oficina. Estas antenas a su vez consolidan la información y la envían a la central.
* Cuando se detecta que un tag RFID ha violado una de las reglas definidas para su comportamiento, se debe generar una alarma indicando a todas las porterías y a los responsables de la vivienda u oficina lo ocurrido.

Para el experimento descrito a continuación se tiene el siguiente alcance:

* Definir un esquema xml para el intercambio de información con entidades externas.
* Crear un web service en una tecnología distinta a la del sistema central, para que éste sea consumido desde el sistema central.
* Definir una excepción en caso de que la notificación ( comunicación con el web service) no pueda realizarse.
* Permitir la notificación de eventos relevantes por medio de correo electrónico.
* En caso de que ocurra un evento en la casa, se debe empezar a trasmitir video usando el API de aplicaciones como Skype.
  1. **Vista Funcional**

Figura 1. Vista Funcional

* 1. **Vista Despliegue**

Figura 2. Vista Despliegue

* 1. **Identificación de patrones**

Para el desarrollo de este sistema se tuvieron en cuenta los siguientes patrones:

***Acceso a recursos – temprana***

Se aplica en la parte de obtención de la conexión a la base de datos para la generación de reportes.

***Acceso a recursos – caché***

Es aplicada para el manejo de y la información de los usuarios y las reglas que son configuradas para cada casa u oficina.

***Manejo de eventos – respuesta reactiva a eventos***

Al llegar un evento al sistema se delega la responsabilidad de manejar el evento a un hilo que se encargue de procesarlo de acuerdo a las reglas configuradas.

***Concurrencia y sincronización – coordinador***

Existe un componente receptor de todos los eventos que se encarga de crear una instancia de un manejador que procesa el evento de acuerdo a su tipo.

**Autenticación / Autorización**

La generación del reporte requiere que el usuario cuente con el rol de administrador. Es por esto que se usa tácticas de autenticación / autorización.

Se implementó parte de la estructura de PACE para manejar la seguridad del sistema.

**Mantener datos confidenciales**

El password de la cuenta del usuario se reconoce como información sensible y debe ser manejado como tal. Es por esto que se usa tácticas de cifrado de datos (hash) para conservar la confidencialidad de este campo.

**Firmas Digitales**

Para evitar la suplantación de identidad y verificar la legitimidad de fuente de información se implementó la táctica de firmas digitales.

**Redundancia Activa**

Para favorecer alta disponibilidad se hace uso de balanceo de cargas a servidores distintos usando el algoritmo round robin.

1. **Experimento A – Comunicación con Web Service**

El objetivo de este experimento definir un esquema xml y comunicar sistemas que funcionen bajo distintas plataformas. De manera más específica, desde el sistema central ( java) se consumirá un web service publicado por un sistema externo (.Net). Dicha comunicación se hará a través del esquema definido inicialmente.

Tabla 1. Descripción del Experimento A

|  |  |
| --- | --- |
| **Descripción del Experimento** | |
| ***Título*:** Interoperabilidad – Comunicación a través de web service entre sistemas. | ***ID*:** A1 |
| ***Descripción*:** Este experimento busca realizar una comunicación entre sistemas desarrollados bajo distintas plataformas ( Java y .Net) para favorecer el atributo de calidad Interoperabilidad. | ***Responsable*:** Ingenium |
| ***Propósito:***  ( ) Reparación, actualizar, clarificar  ( X ) Obtener Información técnica  ( ) Obtener información de negocio  ( ) Otros: | |
| ***Propósito:***  Definir un esquema XML para estandarizar la comunicación entre el sistema central y los sistemas externos.  Comunicar sistemas desarrollados bajos distintas tecnologías. | |
| ***Descripción del experimento:***   * Definir un esquema xml para la comunicación entre el sistema central y los sistemas externos. * Desde el sistema central consumir el servicio suministrado por el sistema externo (policía). Dicha comunicación se realizará haciendo uso del esquema xml definido inicialmente. | |
| ***Artefactos Creados:***  EnternalSystem ->Proyecto desarrollado en Microsoft.Net que simulará un sistema externo (policía). Este sistema publicará un web service que será consumido por el sistema central. | |
| ***Criterio de terminación:***  El sistema central debe poder comunicarse con el sistema externo, enviar una notificación y recibir la respuesta del sistema externo haciendo uso del esquema definido inicialmente. | |
| ***Recursos Requeridos:***  Recurso humano -> Se necesita el trabajo de 2 personas con conocimientos en java, JEE, JSF, Microsoft.Net y web services.  Recursos técnicos -> NetBeans IDE 7.0, Microsoft VS.Net 2008. | |
| ***Duración estimada:***  10 horas | |

Tabla 2. Resultados del Experimento A1

|  |
| --- |
| **Resultados del Experimento** |
| ***Definición de esquema xml:***  ***Esquema 1:*** *Este esquema xml permite al sistema central informar a un sistema externo acerca de la generación de un evento que solicita la atención por parte del sistema externo.*  *Homeid: Identificador de la casa que generó el evento.*  *EventType: Tipo de evento generado en la casa*  *Time: Fecha y hora de generación del evento*  *ContactName: Nombre del contacto asociado a la casa (dueño y/o habitante)*  *CantactPhone: Teléfono donde se puede ubicar al dueño y/o habitante de la casa.*  <?xml version="1.0" encoding="utf-8"?>  <xs:schema id="XMLNotificationSchema"  targetNamespace="http://tempuri.org/XMLNotificationSchema.xsd"  elementFormDefault="qualified"  xmlns="http://tempuri.org/XMLNotificationSchema.xsd"  xmlns:mstns="http://tempuri.org/XMLNotificationSchema.xsd"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  >  <xs:element name="notification">  <xs:complexType>  <xs:sequence>  <xs:element name="homeid" />  <xs:element name="eventtype" ></xs:element>  <xs:element name="time"></xs:element>  <xs:element name="contactname"></xs:element>  <xs:element name="contactphone"></xs:element>  <xs:element name="address"></xs:element>  <xs:element name="city"></xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:schema>  ***Esquema 2:*** *Con este esquema xml, el sistema externo genera una respuesta para el sistema central, respuesta que confirma que la notificación se realizó correctamente.*  *Homeid: Identificador de la casa de la cual se recibió la notificación.*  *EventType: Evento que fue notificado.*  *Message: Mensaje de respuesta generado por el sistema externo, donde informa el tiempo aproximado de llegada al lugar de los hechos.*  <?xml version="1.0" encoding="utf-8"?>  <xs:schema id="XMLResponseSchema"  targetNamespace="http://tempuri.org/XMLResponseSchema.xsd"  elementFormDefault="qualified"  xmlns="http://tempuri.org/XMLResponseSchema.xsd"  xmlns:mstns="http://tempuri.org/XMLResponseSchema.xsd"  xmlns:xs="http://www.w3.org/2001/XMLSchema"  >  <xs:element name="response">  <xs:complexType>  <xs:sequence>  <xs:element name="homeid" />  <xs:element name="eventtype" ></xs:element>  <xs:element name="message"></xs:element>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:schema>  ***WSDL Asociado:*** *A continuación se presenta el wsdl publicado por el sistema externo que permite la comunicación entre los dos sistemas.*  <?xml version="1.0" encoding="utf-8" ?>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:definitions xmlns:soap="**http://schemas.xmlsoap.org/wsdl/soap/**" xmlns:tm="**http://microsoft.com/wsdl/mime/textMatching/**" xmlns:soapenc="**http://schemas.xmlsoap.org/soap/encoding/**" xmlns:mime="**http://schemas.xmlsoap.org/wsdl/mime/**" xmlns:tns="**http://tempuri.org/**" xmlns:s="**http://www.w3.org/2001/XMLSchema**" xmlns:soap12="**http://schemas.xmlsoap.org/wsdl/soap12/**" xmlns:http="**http://schemas.xmlsoap.org/wsdl/http/**" targetNamespace="**http://tempuri.org/**" xmlns:wsdl="**http://schemas.xmlsoap.org/wsdl/**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:types>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:schema elementFormDefault="**qualified**" targetNamespace="**http://tempuri.org/**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:element name="**HelloWorld**">    <s:complexType />    </s:element>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:element name="**HelloWorldResponse**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:complexType>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:sequence>    <s:element minOccurs="**0**" maxOccurs="**1**" name="**HelloWorldResult**" type="**s:string**" />    </s:sequence>    </s:complexType>    </s:element>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:element name="**NotifyEvent**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:complexType>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:sequence>    <s:element minOccurs="**0**" maxOccurs="**1**" name="**xmlEventNotification**" type="**s:string**" />    </s:sequence>    </s:complexType>    </s:element>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:element name="**NotifyEventResponse**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:complexType>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <s:sequence>    <s:element minOccurs="**0**" maxOccurs="**1**" name="**NotifyEventResult**" type="**s:string**" />    </s:sequence>    </s:complexType>    </s:element>    </s:schema>    </wsdl:types>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:message name="**HelloWorldSoapIn**">    <wsdl:part name="**parameters**" element="**tns:HelloWorld**" />    </wsdl:message>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:message name="**HelloWorldSoapOut**">    <wsdl:part name="**parameters**" element="**tns:HelloWorldResponse**" />    </wsdl:message>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:message name="**NotifyEventSoapIn**">    <wsdl:part name="**parameters**" element="**tns:NotifyEvent**" />    </wsdl:message>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:message name="**NotifyEventSoapOut**">    <wsdl:part name="**parameters**" element="**tns:NotifyEventResponse**" />    </wsdl:message>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:portType name="**PoliceServiceSoap**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**HelloWorld**">    <wsdl:input message="**tns:HelloWorldSoapIn**" />    <wsdl:output message="**tns:HelloWorldSoapOut**" />    </wsdl:operation>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**NotifyEvent**">    <wsdl:input message="**tns:NotifyEventSoapIn**" />    <wsdl:output message="**tns:NotifyEventSoapOut**" />    </wsdl:operation>    </wsdl:portType>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:binding name="**PoliceServiceSoap**" type="**tns:PoliceServiceSoap**">    <soap:binding transport="**http://schemas.xmlsoap.org/soap/http**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**HelloWorld**">    <soap:operation soapAction="**http://tempuri.org/HelloWorld**" style="**document**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:input>    <soap:body use="**literal**" />    </wsdl:input>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:output>    <soap:body use="**literal**" />    </wsdl:output>    </wsdl:operation>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**NotifyEvent**">    <soap:operation soapAction="**http://tempuri.org/NotifyEvent**" style="**document**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:input>    <soap:body use="**literal**" />    </wsdl:input>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:output>    <soap:body use="**literal**" />    </wsdl:output>    </wsdl:operation>    </wsdl:binding>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:binding name="**PoliceServiceSoap12**" type="**tns:PoliceServiceSoap**">    <soap12:binding transport="**http://schemas.xmlsoap.org/soap/http**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**HelloWorld**">    <soap12:operation soapAction="**http://tempuri.org/HelloWorld**" style="**document**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:input>    <soap12:body use="**literal**" />    </wsdl:input>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:output>    <soap12:body use="**literal**" />    </wsdl:output>    </wsdl:operation>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:operation name="**NotifyEvent**">    <soap12:operation soapAction="**http://tempuri.org/NotifyEvent**" style="**document**" />  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:input>    <soap12:body use="**literal**" />    </wsdl:input>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:output>    <soap12:body use="**literal**" />    </wsdl:output>    </wsdl:operation>    </wsdl:binding>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:service name="**PoliceService**">  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:port name="**PoliceServiceSoap**" binding="**tns:PoliceServiceSoap**">    <soap:address location="**http://localhost/ExternalSystemPolice/PoliceService.asmx**" />    </wsdl:port>  [**-**](http://localhost/ExternalSystemPolice/PoliceService.asmx?wsdl) <wsdl:port name="**PoliceServiceSoap12**" binding="**tns:PoliceServiceSoap12**">    <soap12:address location="**http://localhost/ExternalSystemPolice/PoliceService.asmx**" />    </wsdl:port>    </wsdl:service>    </wsdl:definitions>  El sistema central al recibir un evento consume el servicio Web invocando a “NotifyEvent” y pasándole como parámetro una cadena en formato xml cumpliendo con el esquema 1. El sistema externo recibe dicha notificación, genera un mensaje de respuesta cumpliendo con el esquema 2 y lo retorna al invocador (sistema central). Por último el sistema central, presenta el mensaje proveniente del sistema externo.  El sistema externo simula un tiempo de respuesta lento. Aquí se hace uso de comunicación asíncrona, para que el sistema central consuma el servicio web por cada evento que se genera. De esta manera se envían notificaciones a los sistemas externos sin tener que esperar la respuesta de dicho sistema ( asíncrono). Esto favorece en gran medida el tiempo de desempeño del sistema central.  La definición de un esquema xml para la comunicación de sistemas, favorece la interoperabilidad. Con esto, sin importar la tecnología que ha implementado el sistema externo, la comunicación se hace de manera transparente. Y si llegado el caso se requiere adicionar un nuevo parámetro para la comunicación, basta con modificar el esquema de comunicación, pero el servicio no requeriría modificación alguna.  La comunicación desde el sistema central, se realiza a través de un proxy generado de acuerdo al wsdl que publica el sistema externo. En caso de que el sistema externo adicione un nuevo servicio y éste requiera ser consumido por el sistema central, el proxy debe ser modificado / actualizado. |
| ***Duración Real:***  10 horas |
| ***Recursos Reales:***  Recurso humano -> Se necesita el trabajo de 2 personas con conocimientos en java, JEE, JSF, Microsoft.Net y web services.  Recursos técnicos -> NetBeans IDE 7.0, Microsoft VS.Net 2008. |
| ***Recomendaciones:***  Desde el sistema central se debe implementar un mecanismo de generación del proxy dinámico. Esto debido a que el web service suministrado por el sistema externo puede cambiar su estructura y/o agregar nuevos servicios. |

1. **Experimento B – Comunicación Asincrona – Envío de correo electrónico**
   1. **B – Comunicación Asíncrona**

Tabla 6. Descripción del Experimento B

|  |  |
| --- | --- |
| **Descripción del Experimento** | |
| ***Título*:** Disponibilidad | ***ID*:** B |
| ***Descripción*:** Este experimento busca verificar la disponibilidad del sistema | ***Responsable*:**Ingenium |
| ***Propósito:***  ( ) Reparación, actualizar, clarificar  ( X ) Obtener Información técnica  ( ) Obtener información de negocio  ( ) Otros: | |
| ***Propósito:***  Verificar las decisiones de diseño asociadas a la disponibilidad del sistema. | |
| ***Descripción del experimento:*** | |
| ***Artefactos Creados:*** | |
| ***Criterio de terminación:*** | |
| ***Recursos Requeridos:*** | |
| ***Duración estimada:***  18 horas | |

Tabla 7. Resultados del Experimento B1

|  |
| --- |
| **Resultados del Experimento** |
| ***Resumen de los resultados:*** |
| ***Duración Real:***  14 horas |
| ***Recursos Reales:***  Recurso humano -> Se necesita el trabajo de 2 personas con conocimientos en java, sockets, estilos y tácticas de arquitectura.  Recursos técnicos ->NetBeans IDE 7.0, Microsoft Visio, JUnit, JUnitPerf |
| ***Recomendaciones:***  Para aumentar el desempeño del sistema, se puede mover el módulo de seguridad del servidor de balanceo de cargas a cada uno de los servidores centrales. Para aumentar disponibilidad, se recomienda aumentar un servidor central al balanceador de cargas. |

1. **Experimento C – Video llamada**
   1. **B – Video llamada**

Tabla 6. Descripción del Experimento B

|  |  |
| --- | --- |
| **Descripción del Experimento** | |
| ***Título*:** Disponibilidad | ***ID*:** B |
| ***Descripción*:** Este experimento busca verificar la disponibilidad del sistema | ***Responsable*:**Ingenium |
| ***Propósito:***  ( ) Reparación, actualizar, clarificar  ( X ) Obtener Información técnica  ( ) Obtener información de negocio  ( ) Otros: | |
| ***Propósito:***  Verificar las decisiones de diseño asociadas a la disponibilidad del sistema. | |
| ***Descripción del experimento:*** | |
| ***Artefactos Creados:*** | |
| ***Criterio de terminación:*** | |
| ***Recursos Requeridos:*** | |
| ***Duración estimada:***  18 horas | |

Tabla 7. Resultados del Experimento B1

|  |
| --- |
| **Resultados del Experimento** |
| ***Resumen de los resultados:*** |
| ***Duración Real:***  14 horas |
| ***Recursos Reales:***  Recurso humano -> Se necesita el trabajo de 2 personas con conocimientos en java, sockets, estilos y tácticas de arquitectura.  Recursos técnicos ->NetBeans IDE 7.0, Microsoft Visio, JUnit, JUnitPerf |
| ***Recomendaciones:***  Para aumentar el desempeño del sistema, se puede mover el módulo de seguridad del servidor de balanceo de cargas a cada uno de los servidores centrales. Para aumentar disponibilidad, se recomienda aumentar un servidor central al balanceador de cargas. |